배우기 SQL

#sql

1: SQL

SQL SQL (Structured Query Language). .

: **ISO / ANSI SQL** . .

1986	SQL-86	ANSI X3.135-1986, ISO 9075 : 1987	1986-01-01
1989	SQL-89	ANSI X3.135-1989, ISO / IEC 9075 : 1989	1989-01-01
1992	SQL-92	ISO / IEC 9075 : 1992	1992-01-01
1999	SQL : 1999	ISO / IEC 9075 : 1999	1999-12-16
2003	SQL : 2003	ISO / IEC 9075 : 2003	2003-12-15
2006	SQL : 2006	ISO / IEC 9075 : 2006	2006-06-01
2008	SQL : 2008	ISO / IEC 9075 : 2008	2008-07-15
2011	SQL : 2011	ISO / IEC 9075 : 2011	2011 12 15
2016	SQL : 2016	ISO / IEC 9075 : 2016	2016-12-01

Examples

SQL (Structured Query Language) (RDBMS) . SQL (RDSMS) " SQL"(NoSQL) .

SQL 3 .

1. DDL (Data Definition Language) : .
2. DML ( ) : , , .
3. DCL ( ) : .

Wikipedia SQL

DML `INSERT` , `SELECT` , `UPDATE` `DELETE` Create, Read, Update Delete (CRUD).
3 (INSERT, UPDATE, DELETE) ( ) `MERGE` .

Wikipedia CRUD

SQL / . "SQL " .
Microsoft "SQL Server" . SQL SQL Server .

SQL : https://riptutorial.com/ko/sql/topic/184/sql-

2: ALTER TABLE

SQL ALTER / .

- ALTER TABLE [ ] ADD [ ] [ ]

Examples

```
ALTER TABLE Employees
ADD StartingDate date NOT NULL DEFAULT GetDate(),
    DateOfBirth date NULL
```

`StartingDate` NULL `DateOfBirth` NULL .

```
ALTER TABLE Employees
DROP COLUMN salary;
```

( ) .

```
ALTER TABLE Employees
DROP CONSTRAINT DefaultSalary
```

Employees DefaultSalary .

: - .

```
ALTER TABLE Employees
ADD CONSTRAINT DefaultSalary DEFAULT ((100)) FOR [Salary]
```

Salary 100 DefaultSalary .

.

- **-** .
- **-** .
- **Not Null -** .
- **-** .
- **-** .
- **-** .

Oracle .

```
ALTER TABLE Employees
ALTER COLUMN StartingDate DATETIME NOT NULL DEFAULT (GETDATE())
```

`StartingDate` `date` `datetime` .

```
ALTER TABLE EMPLOYEES ADD pk_EmployeeID PRIMARY KEY (ID)
```

`ID` Employees . ID . .

```
ALTER TABLE EMPLOYEES ADD pk_EmployeeID PRIMARY KEY (ID, FName)
```

ALTER TABLE : https://riptutorial.com/ko/sql/topic/356/alter-table

3: AND OR

1. SELECT * FROM WHERE (1) AND (2);
2. SELECT * FROM WHERE (1) OR (2);

Examples

AND

.

	10	
	20	
	24	

```
select Name from table where Age>10 AND City='Prague'
```

.

```
select Name from table where Age=10 OR City='Prague'
```

.

AND OR : https://riptutorial.com/ko/sql/topic/1386/and---or-

4: CREATE FUNCTION

- function_name ([list_of_paramers]) return return_data_type AS BEGIN function_body RETURN scalar_expression END

function_name	
list_of_paramenters	
return_data_type	. SQL
function_body	
scalar_expression	

CREATE FUNCTION SELECT, INSERT, UPDATE DELETE . .

Examples

```
CREATE FUNCTION FirstWord (@input varchar(1000))
RETURNS varchar(1000)
AS
BEGIN
    DECLARE @output varchar(1000)
    SET @output = SUBSTRING(@input, 0, CASE CHARINDEX(' ', @input)
        WHEN 0 THEN LEN(@input) + 1
        ELSE CHARINDEX(' ', @input)
    END)

    RETURN @output
END
```

FirstWord varchar varchar .

CREATE FUNCTION : https://riptutorial.com/ko/sql/topic/2437/create-function

5: DROP DELETE

- MSSQL :
- [ ] { _ | database_snapshot_name} [, ... n] [;]
- MySQL :
- DROP {DATABASE | SCHEMA} [ ] db_name

`DROP DATABASE` SQL h .   h   i .

Examples

.   .

Employees Database .

```
DROP DATABASE [dbo].[Employees]
```

DROP DELETE   : https://riptutorial.com/ko/sql/topic/3974/drop--delete-

6: GROUP BY

SELECT `GROUP BY` . . . GROUP BY `HAVING` .

- GROUP BY {

  | ROLLUP (<group_by_expression> [, ... n])
  | CUBE (<group_by_expression> [, ... n])
  | GROUPING SETS ([, ... n])
  | () - .
  } [, ... n]

- <group_by_expression> :: =

  | ([, ... n])

- <grouping_set> :: =
  () - .
  | <grouping_set_item>
  | (<grouping_set_item> [, ... n])

- <grouping_set_item> :: =
  <group_by_expression>
  | ROLLUP (<group_by_expression> [, ... n])
  | CUBE (<group_by_expression> [, ... n])

Examples

GROUP BY .

.

, "Westerosians":

	GreatHouseAllegience
Myrcella	
Catelyn	

GROUP BY COUNT .

```
SELECT Count(*) Number_of_Westerosians
FROM Westerosians
```

...

6

GROUP BY COUNT Great House .

```
SELECT GreatHouseAllegience House, Count(*) Number_of_Westerosians
FROM Westerosians
GROUP BY GreatHouseAllegience
```

...

	1
	2

GROUP BY ORDER BY .

```
SELECT GreatHouseAllegience House, Count(*) Number_of_Westerosians
FROM Westerosians
GROUP BY GreatHouseAllegience
ORDER BY Number_of_Westerosians Desc
```

...

	2
	1

HAVING GROUP BY

HAVING GROUP BY . : .

:

( ).

```
SELECT
  a.Id,
  a.Name,
  COUNT(*) BooksWritten
```

```
FROM BooksAuthors ba
  INNER JOIN Authors a ON a.id = ba.authorid
GROUP BY
  a.Id,
  a.Name
HAVING COUNT(*) > 1    -- equals to HAVING BooksWritten > 1
;
```

3 ( ).

```
SELECT
  b.Id,
  b.Title,
  COUNT(*) NumberOfAuthors
FROM BooksAuthors ba
  INNER JOIN Books b ON b.id = ba.bookid
GROUP BY
  b.Id,
  b.Title
HAVING COUNT(*) > 3    -- equals to HAVING NumberOfAuthors > 3
;
```

GROUP BY

GROUP BY "for each" . :

```
SELECT EmpID, SUM (MonthlySalary)
FROM Employee
GROUP BY EmpID
```

:

"EmpID MonthlySalary "

:

```
+-----+-------------+
|EmpID|MonthlySalary|
+-----+-------------+
|1    |200          |
+-----+-------------+
|2    |300          |
+-----+-------------+
```

:

```
+-+---+
|1|200|
+-+---+
|2|300|
+-+---+
```

. :

```
+-----+-------------+
|EmpID|MonthlySalary|
+-----+-------------+
|1    |200          |
+-----+-------------+
|1    |300          |
+-----+-------------+
|2    |300          |
+-----+-------------+
```

:

```
+-+---+
|1|500|
+-+---+
|2|300|
+-+---+
```

EmpID 1 .

ROLAP ( )

SQL . "ALL" . .

- `with data cube` .
- `with roll up` .

SQL : 1999,2003,2006,2008,2011.

.

		_
	1	100
	2	250
	2	300

```
select Food,Brand,Total_amount
from Table
group by Food,Brand,Total_amount with cube
```

		_
	1	100
	2	250
		350

		_
	2	300
		300
	1	100
	2	550
		650

```
select Food,Brand,Total_amount
from Table
group by Food,Brand,Total_amount with roll up
```

		_
	1	100
	2	250
	2	300
		350
		300
		650

GROUP BY : https://riptutorial.com/ko/sql/topic/627/group-by

7: IN

Examples

IN

id

```
select *
from products
where id in (1,8,3)
```

.

```
select *
from products
where id = 1
   or id = 8
   or id = 3
```

IN

```
SELECT *
FROM customers
WHERE id IN (
    SELECT DISTINCT customer_id
    FROM orders
);
```

.

IN : https://riptutorial.com/ko/sql/topic/3169/in-

8: LIKE

- **% :** SELECT * FROM [] WHERE [ ] '% %'

  _ : SELECT * FROM [] WHERE [ ] 'V_n %'

  [charlist]   **:** SELECT * FROM [table] WHERE [column_name] 'V [abc] n %' .

WHERE LIKE   .   .

- % (Percentage Symbol) - 0   .
- _ () -   .

Examples

( ) `%`   0   .

'%'   0   .

Employees .

	FName	LName		ID	DepartmentId		Hire_date
1			2468101214	1	1	400	23-03-2005
2		Amudsen	2479100211	1	1	400	11-01-2010
			2462544026	2	1	600	06-08-2015
4			2454124602	1	1	400	23-03-2005
5			2468021911	2	1	800	01-01-2000

Employees 'on' FName   .

```
SELECT * FROM Employees WHERE FName LIKE '%on%';
```

	FName	LName		ID	DepartmentId		Hire_date
	R **on** ny		2462544026	2	1	600	06-08-2015
4	J		2454124602	1	1	400	23-03-2005

Employees '246' PhoneNumber   .

```
SELECT * FROM Employees WHERE PhoneNumber LIKE '246%';
```

	FName	LName		ID	DepartmentId		Hire_date
1			**246** 8101214	1	1	400	23-03-2005
			246 2544026	2	1	600	06-08-2015
5			**246** 8021911	2	1	800	01-01-2000

Employees '11' PhoneNumber   .

```
SELECT * FROM Employees WHERE PhoneNumber LIKE '%11'
```

	FName	LName		ID	DepartmentId		Hire_date
2		Amudsen	24791002 **11**	1	1	400	11-01-2010
5			24680219 **11**	2	1	800	01-01-2000

Fname   Employees 'n' .

```
SELECT * FROM Employees WHERE FName LIKE '__n%';
```

('n')

	FName	LName		ID	DepartmentId		Hire_date
			2462544026	2	1	600	06-08-2015
4			2454124602	1	1	400	23-03-2005

(SQL-SELECT)   (%) (_)   .

_ ()   .

Fname 'j' 'n' Fname 3   .

```
SELECT * FROM Employees WHERE FName LIKE 'j_n'
```

_ ()   .

, "jon", "jan", "jen" .

jn, john, jordan, justin, jason, julian, jillian, joann   .   Fname 3 .

, "LaSt", "LoSt", "HaLt" .

```
SELECT * FROM Employees WHERE FName LIKE '_A_T'
```

(: `[af]`) (: `[abcdef]`) .

"gary" "mary" .

```
SELECT * FROM Employees WHERE FName LIKE '[a-g]ary'
```

"mary" "gary" .

```
SELECT * FROM Employees WHERE Fname LIKE '[lmnop]ary'
```

range set ^ .

"gary" "mary" .

```
SELECT * FROM Employees WHERE FName LIKE '[^a-g]ary'
```

"mary" "gary" .

```
SELECT * FROM Employees WHERE Fname LIKE '[^lmnop]ary'
```

ANY ALL

:
. 'electronics', 'books' 'video'.

```
SELECT *
FROM   purchase_table
WHERE  product_type LIKE ANY ('electronics', 'books', 'video');
```

( )
' ' " ' '( ) .

```
SELECT *
FROM   customer_table
WHERE  full_address LIKE ALL ('%united kingdom%', '%london%', '%eastern road%');
```

:
ALL .
' ' " " .

```
SELECT *
FROM   customer_table
WHERE  product_type NOT LIKE ALL ('electronics', 'books', 'video');
```

Employees A F FName .

```
SELECT * FROM Employees WHERE FName LIKE '[A-F]%'
```

LIKE- ESCAPE

LIKE -query .

```
SELECT *
FROM T_Whatever
WHERE SomeField LIKE CONCAT('%', @in_SearchText, '%')
```

(fulltext-search LIKE LIKE) "50 %" "a_b" .

LIKE -escape .

```
SELECT *
FROM T_Whatever
WHERE SomeField LIKE CONCAT('%', @in_SearchText, '%') ESCAPE '\'
```

, \ ESCAPE . , \ % _ .

```
string stringToSearch = "abc_def 50%";
string newString = "";
foreach(char c in stringToSearch)
     newString += @"\" + c;

sqlCmd.Parameters.Add("@in_SearchText", newString);
// instead of sqlCmd.Parameters.Add("@in_SearchText", stringToSearch);
```

: . 1 (utf-8). : `string stringToSearch = "Les Mise\u0301rables";` . / / . graphemeCluster .

C # ReverseString .

SQL LIKE . SQL .

SQL %, _, [charlist], [^ charlist]

% - 0 .

```
    Eg:  //selects all customers with a City starting with "Lo"
         SELECT * FROM Customers
         WHERE City LIKE 'Lo%';

        //selects all customers with a City containing the pattern "es"
       SELECT * FROM Customers
        WHERE City LIKE '%es%';
```

_ - .

```
Eg://selects all customers with a City starting with any character, followed by "erlin"
SELECT * FROM Customers
WHERE City LIKE '_erlin';
```

[charlist] -

```
Eg://selects all customers with a City starting with "a", "d", or "l"
SELECT * FROM Customers
WHERE City LIKE '[adl]%';

//selects all customers with a City starting with "a", "d", or "l"
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

[^ charlist] - .

```
Eg://selects all customers with a City starting with a character that is not "a", "p", or "l"
SELECT * FROM Customers
WHERE City LIKE '[^apl]%';

or

SELECT * FROM Customers
WHERE City NOT LIKE '[apl]%' and city like '_%';
```

LIKE : https://riptutorial.com/ko/sql/topic/860/like-

9: SQL Group by vs Distinct

Examples

GROUP BY DISTINCT

GROUP BY . .

	userId		orderValue	
1	43	A	25	20-03-2016
2	57	B	50	22-03-2016
	43	A	30	25-03-2016
4	82	C	10	26-03-2016
5	21	A	45	29-03-2016

GROUP BY .

```
SELECT
    storeName,
    COUNT(*) AS total_nr_orders,
    COUNT(DISTINCT userId) AS nr_unique_customers,
    AVG(orderValue) AS average_order_value,
    MIN(orderDate) AS first_order,
    MAX(orderDate) AS lastOrder
FROM
    orders
GROUP BY
    storeName;
```

.

	total_nr_orders	nr_unique_customers	average_order_value		
A		2	33.3	20-03-2016	29-03-2016
B	1	1	50	22-03-2016	22-03-2016
C	1	1	10	26-03-2016	26-03-2016

DISTINCT .

```
SELECT DISTINCT
    storeName,
    userId
FROM
    orders;
```

	userId
A	43
B	57
C	82
A	21

SQL Group by vs Distinct : https://riptutorial.com/ko/sql/topic/2499/sql-group-by-vs-distinct

10: SQL

SQL SQL . SQL SQL . SQL .

Examples

SQL

.

```
https://somepage.com/ajax/login.ashx?username=admin&password=123
```

login.ashx .

```
strUserName = getHttpsRequestParameterString("username");
strPassword = getHttpsRequestParameterString("password");
```

.

SQL :

```
txtSQL = "SELECT * FROM Users WHERE username = '" + strUserName + "' AND password = '"+
strPassword +"'";
```

.

SQL .

```
-- strUserName = "d'Alambert";
txtSQL = "SELECT * FROM Users WHERE username = 'd'Alambert' AND password = '123'";
```

d `d'Alambert` SQL `d'Alambert` .

(:

```
strUserName = strUserName.Replace("'", "''");
strPassword = strPassword.Replace("'", "''");
```

.

```
cmd.CommandText = "SELECT * FROM Users WHERE username = @username AND password = @password";

cmd.Parameters.Add("@username", strUserName);
cmd.Parameters.Add("@password", strPassword);
```

( ) SQL .

,   .

```
lol'; DROP DATABASE master; --
```

SQL   :

```
"SELECT * FROM Users WHERE username = 'somebody' AND password = 'lol'; DROP DATABASE master; -
-'";
```

SQL DB !

SQL .

,   ,   ,   .

.

.   .

SQL

```
SQL = "SELECT * FROM Users WHERE username = '" + user + "' AND password ='" + pw + "'";
db.execute(SQL);
```

`pw' or '1'='1`   . SQL .

```
SELECT * FROM Users WHERE username = 'somebody' AND password ='pw' or '1'='1'
```

`'1'='1'`   `Users`   .

SQL .

```
SQL = "SELECT * FROM Users WHERE username = ? AND password = ?";
db.execute(SQL, [user, pw]);
```

SQL   : https://riptutorial.com/ko/sql/topic/3517/sql-

11: SQL

Examples

.
SQL .
.

```
DECLARE @db_name nvarchar(255)
DECLARE @sql nvarchar(MAX)

DECLARE @schema nvarchar(255)
DECLARE @table nvarchar(255)
DECLARE @column nvarchar(255)




DECLARE db_cursor CURSOR FOR
SELECT name FROM sys.databases


OPEN db_cursor
FETCH NEXT FROM db_cursor INTO @db_name

WHILE @@FETCH_STATUS = 0
BEGIN
    SET @sql = 'SELECT * FROM ' + QUOTENAME(@db_name) + '.information_schema.columns'
    PRINT ''
    PRINT ''
    PRINT ''
    PRINT @sql
    -- EXECUTE(@sql)



    -- For each database

    DECLARE @sqlstatement nvarchar(4000)
    --move declare cursor into sql to be executed
    SET @sqlstatement = 'DECLARE  columns_cursor CURSOR FOR SELECT TABLE_SCHEMA, TABLE_NAME,
COLUMN_NAME FROM ' + QUOTENAME(@db_name) + '.information_schema.columns ORDER BY TABLE_SCHEMA,
TABLE_NAME, ORDINAL_POSITION'



    EXEC sp_executesql @sqlstatement


    OPEN columns_cursor
    FETCH NEXT FROM columns_cursor
    INTO @schema, @table, @column

    WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT @schema + '.' + @table + '.' + @column
        --EXEC asp_DoSomethingStoredProc @UserId

    FETCH NEXT FROM columns_cursor --have to fetch again within loop
    INTO @schema, @table, @column

    END
    CLOSE columns_cursor
    DEALLOCATE columns_cursor

    -- End for each database





    FETCH NEXT FROM db_cursor INTO @db_name
END

CLOSE db_cursor
DEALLOCATE db_cursor
```

SQL   : https://riptutorial.com/ko/sql/topic/8895/sql-

12: SQL

SQL .

Examples

/

/ `CamelCase` `snake_case` :

```
SELECT FirstName, LastName
FROM Employees
WHERE Salary > 500;
```

```
SELECT first_name, last_name
FROM employees
WHERE salary > 500;
```

. . .

`tbl` `col` . SQL ( ) .

###

SQL . .

*

`SELECT *` .

`SELECT *` . .

I / O .

.

```
--SELECT *                                      don't
  SELECT ID, FName, LName, PhoneNumber   -- do
  FROM Emplopees;
```

.

EXISTS `SELECT *` EXISTS ( ). EXISTS `SELECT *` .

```
-- list departments where nobody was hired recently
SELECT ID,
       Name
```

```
FROM Departments
WHERE NOT EXISTS (SELECT *
                  FROM Employees
                  WHERE DepartmentID = Departments.ID
                    AND HireDate >= '2015-01-01');
```

.   .

```
SELECT d.Name, COUNT(*) AS Employees FROM Departments AS d JOIN Employees AS e ON d.ID =
e.DepartmentID WHERE d.Name != 'HR' HAVING COUNT(*) > 10 ORDER BY COUNT(*) DESC;
```

.

```
SELECT d.Name,
       COUNT(*) AS Employees
FROM Departments AS d
JOIN Employees AS e ON d.ID = e.DepartmentID
WHERE d.Name != 'HR'
HAVING COUNT(*) > 10
ORDER BY COUNT(*) DESC;
```

SQL   .

```
SELECT   d.Name,
         COUNT(*) AS Employees
FROM     Departments AS d
JOIN     Employees AS e ON d.ID = e.DepartmentID
WHERE    d.Name != 'HR'
HAVING   COUNT(*) > 10
ORDER BY COUNT(*) DESC;
```

(SQL   .)

.

```
SELECT
    d.Name,
    COUNT(*) AS Employees
FROM
    Departments AS d
JOIN
    Employees AS e
    ON d.ID = e.DepartmentID
WHERE
    d.Name != 'HR'
HAVING
    COUNT(*) > 10
ORDER BY
    COUNT(*) DESC;
```

.

```
SELECT Model,
       EmployeeID
FROM Cars
WHERE CustomerID = 42
  AND Status     = 'READY';
```

SQL . C # `@"..."` , Python `"""..."""` C ++ `R"(...)"` .

. .

- WHERE . .
- .
- SQL .

  ```
  SELECT d.Name,
         e.Fname || e.LName AS EmpName
  FROM      Departments AS d
  LEFT JOIN Employees   AS e ON d.ID = e.DepartmentID;
  ```

- USING .

  ```
  SELECT RecipeID,
         Recipes.Name,
         COUNT(*) AS NumberOfIngredients
  FROM      Recipes
  LEFT JOIN Ingredients USING (RecipeID);
  ```

  (.
  USING . , `RecipeID` .

SQL : https://riptutorial.com/ko/sql/topic/9843/sql--

13: TRUNCATE

TRUNCATE . DELETE .

- TRUNCATE TABLE table_name;

TRUNCATE DDL (Data Definition Language) DELETE (, DML,) . TRUNCATE TRUNCATE .

- TRUNCATE . TRUNCATE DML (ON DELETE) . .
- TRUNCATE , DELETE .
- ID (MS SQL Server) TRUNCATE . .
- SQL TRUNCATE

Examples

Employee

```
TRUNCATE TABLE Employee;
```

truncate table DELETE TABLE .

. . delete table .

DELETE TRUNCATE TRUNCATE . . 1 .

, .

TRUNCATE : https://riptutorial.com/ko/sql/topic/1466/truncate

14: TRY / CATCH

TRY / CATCH MS SQL Server T-SQL .

.NET T-SQL .

Examples

TRY / CATCH

datetime .

```
BEGIN TRANSACTION
BEGIN TRY
    INSERT INTO dbo.Sale(Price, SaleDate, Quantity)
    VALUES (5.2, GETDATE(), 1)
    INSERT INTO dbo.Sale(Price, SaleDate, Quantity)
    VALUES (5.2, 'not a date', 1)
    COMMIT TRANSACTION
END TRY
BEGIN CATCH
    THROW
    ROLLBACK TRANSACTION
END CATCH
```

.

```
BEGIN TRANSACTION
BEGIN TRY
    INSERT INTO dbo.Sale(Price, SaleDate, Quantity)
    VALUES (5.2, GETDATE(), 1)
    INSERT INTO dbo.Sale(Price, SaleDate, Quantity)
    VALUES (5.2, GETDATE(), 1)
    COMMIT TRANSACTION
END TRY
BEGIN CATCH
    THROW
    ROLLBACK TRANSACTION
END CATCH
```

TRY / CATCH : https://riptutorial.com/ko/sql/topic/4420/try---catch

15: UNION / UNION ALL

SQL ***UNION*** ***SELECT*** . UNION SELECT .

- SELECT column_1 [, column_2] FROM table_1 [, table_2] [WHERE]
  UNION | UNION ALL
  SELECT column_1 [, column_2] FROM table_1 [, table_2] [WHERE]

`UNION` `UNION ALL` SELECT / .

`UNION` / `UNION ALL` .

`UNION` `UNION ALL` `UNION` `UNION ALL` .

`UNION ALL` .

Examples

UNION ALL

```
CREATE TABLE HR_EMPLOYEES
(
    PersonID int,
    LastName VARCHAR(30),
    FirstName VARCHAR(30),
    Position VARCHAR(30)
);

CREATE TABLE FINANCE_EMPLOYEES
(
    PersonID INT,
    LastName VARCHAR(30),
    FirstName VARCHAR(30),
    Position VARCHAR(30)
);
```

`managers` .

`UNION` `position` (A) `manager`

```
SELECT
    FirstName, LastName
FROM
    HR_EMPLOYEES
WHERE
    Position = 'manager'
UNION ALL
SELECT
    FirstName, LastName
FROM
    FINANCE_EMPLOYEES
WHERE
```

```
    Position = 'manager'
```

`UNION` . `UNION ALL` .

select .

```
SELECT
    FirstName as 'First Name', LastName as 'Last Name'
FROM
    HR_EMPLOYEES
WHERE
    Position = 'manager'
UNION ALL
SELECT
    FirstName, LastName
FROM
    FINANCE_EMPLOYEES
WHERE
    Position = 'manager'
```

:

- `UNION` 2 .
- `UNION ALL` 2 .

  `UNION` . .

UNION

. `UNION` .

```
SELECT C1, C2, C3 FROM Table1 WHERE C1 = @Param1
UNION
SELECT C1, C2, C3 FROM Table1 WHERE C2 = @Param2
```

. .

UNION ALL

( ).

```
SELECT C1 FROM Table1
UNION ALL
SELECT C1 FROM Table2
```

(,) . .

UNION / UNION ALL : https://riptutorial.com/ko/sql/topic/349/union---union-all

16: WHERE HAVING

- SELECT column_name
  FROM table_name
  WHERE column_name
- SELECT column_name, aggregate_function (column_name)
  FROM table_name
  GROUP BY column_name
  aggregate_function (column_name)

Examples

WHERE   .

Steam   10   .   .

```
SELECT *
FROM Items
WHERE Price < 10
```

IN   .

Car Table .

```
SELECT *
FROM Cars
WHERE TotalCost IN (100, 200, 300)
```

200 Car # 2 100 Car # 3 . `OR`   . :

```
SELECT *
FROM Cars
WHERE TotalCost = 100 OR TotalCost = 200 OR TotalCost = 300
```

LIKE   .

LIKE   .

Employees   .

```
SELECT *
FROM Employees
WHERE FName LIKE 'John'
```

'John'   Employee # 1 .

```
SELECT *
FROM Employees
WHERE FName like 'John%'
```

% .

- John% - 'John' Employee .
- %John - 'John' Employee .
- %John% - 'John' Employee .

'John' Employee # 2 'Johnathon' Employee # 4 .

NULL / NOT NULL WHERE

```
SELECT *
FROM Employees
WHERE ManagerId IS NULL
```

. ManagerId NULL Employee NULL .

.

```
Id    FName    LName    PhoneNumber    ManagerId    DepartmentId
1     James    Smith    1234567890     NULL         1
```

```
SELECT *
FROM Employees
WHERE ManagerId IS NOT NULL
```

ManagerId NULL Employee NULL .

.

```
Id    FName       LName       PhoneNumber    ManagerId    DepartmentId
2     John        Johnson     2468101214     1            1
3     Michael     Williams    1357911131     1            2
4     Johnathon   Smith       1212121212     2            1
```

: WHERE `WHERE ManagerId = NULL` `WHERE ManagerId <> NULL` .

HAVING

WHERE HAVING .

(Wikipedia) .

COUNT() , SUM() , MIN() MAX() .

Car Table .

```
SELECT CustomerId, COUNT(Id) AS [Number of Cars]
FROM Cars
GROUP BY CustomerId
HAVING COUNT(Id) > 1
```

`CustomerId` `Number of Cars` . # 1.

.

ID	
1	2

BETWEEN

Item Sales Customers .

: BETWEEN .

Numbers BETWEEN :

```
SELECT * From ItemSales
WHERE Quantity BETWEEN 10 AND 17
```

10 17 `ItemSales` . .

		ItemId		
1	2013-07-01	100	10	34.5
4	2013-07-23	100	15	34.5
5	2013 7 24	145	10	34.5

BETWEEN :

```
SELECT * From ItemSales
WHERE SaleDate BETWEEN '2013-07-11' AND '2013-05-24'
```

2013 7 11 2013 5 24 `SaleDate` `ItemSales` .

		ItemId		
	2013-07-11	100	20	34.5
4	2013-07-23	100	15	34.5

		ItemId		
5	2013 7 24	145	10	34.5

datetime datetime 24 .

BETWEEN :

```
SELECT Id, FName, LName FROM Customers
WHERE LName BETWEEN 'D' AND 'L';
```

: SQL

'D' 'L' . # 1 # 3 . 'M' # 2 .

	FName	LName
1		

```
SELECT * FROM Employees
```

. Employees .

```
Id    FName      LName     PhoneNumber    ManagerId    DepartmentId     Salary  Hire_date
CreatedDate    ModifiedDate
1     James      Smith     1234567890     NULL         1                1000    01-01-2002     01-01-
2002     01-01-2002
2     John       Johnson   2468101214     1            1                400     23-03-2005     23-03-
2005     01-01-2002
3     Michael    Williams  1357911131     1            2                600     12-05-2009     12-05-
2009     NULL
4     Johnathon  Smith     1212121212     2            1                500     24-07-2016     24-07-
2016     01-01-2002
```

SELECT WHERE . = :

```
SELECT * FROM Employees WHERE DepartmentId = 1
```

DepartmentId 1 .

```
Id    FName      LName     PhoneNumber    ManagerId    DepartmentId     Salary  Hire_date
CreatedDate    ModifiedDate
1     James      Smith     1234567890     NULL         1                1000    01-01-2002     01-01-
2002     01-01-2002
2     John       Johnson   2468101214     1            1                400     23-03-2005     23-03-
2005     01-01-2002
4     Johnathon  Smith     1212121212     2            1                500     24-07-2016     24-07-
2016     01-01-2002
```

AND OR

WHERE . Employees .

```
Id    FName      LName     PhoneNumber    ManagerId    DepartmentId     Salary   Hire_date
CreatedDate    ModifiedDate
1     James      Smith     1234567890     NULL         1                1000     01-01-2002     01-01-
2002     01-01-2002
2     John       Johnson   2468101214     1            1                400      23-03-2005     23-03-
2005     01-01-2002
3     Michael    Williams  1357911131     1            2                600      12-05-2009     12-05-
2009     NULL
4     Johnathon  Smith     1212121212     2            1                500      24-07-2016     24-07-
2016     01-01-2002
```

```
SELECT * FROM Employees WHERE DepartmentId = 1 AND ManagerId = 1
```

:

```
Id    FName      LName     PhoneNumber    ManagerId    DepartmentId     Salary   Hire_date
CreatedDate    ModifiedDate
2     John       Johnson   2468101214     1            1                400      23-03-2005     23-03-
2005     01-01-2002
```

```
SELECT * FROM Employees WHERE DepartmentId = 2 OR ManagerId = 2
```

:

```
Id    FName      LName     PhoneNumber    ManagerId    DepartmentId     Salary   Hire_date
CreatedDate    ModifiedDate
3     Michael    Williams  1357911131     1            2                600      12-05-2009     12-05-
2009     NULL
4     Johnathon  Smith     1212121212     2            1                500      24-07-2016     24-07-
2016     01-01-2002
```

HAVING .

ID	ID		
1	2	5	100
1		2	200
1	4	1	500
2	1	4	50
	5	6	700

ProductID 2 3   HAVING

```
select customerId
from orders
where productID in (2,3)
group by customerId
having count(distinct productID) = 2
```

:

ID
1

productID   HAVING   productIds   HAVING .

```
select customerId
from orders
group by customerId
having sum(case when productID = 2 then 1 else 0 end) > 0
   and sum(case when productID = 3 then 1 else 0 end) > 0
```

productID 2 productID 3   .

`TableName`   `TableName1` .

```
SELECT * FROM TableName t WHERE EXISTS (
    SELECT 1 FROM TableName1 t1 where t.Id = t1.Id)
```

WHERE HAVING   : https://riptutorial.com/ko/sql/topic/636/where--having---

17: XML

Examples

XML

```
DECLARE @xmlIN XML = '<TableData>
<aaa Main="First">
  <row name="a" value="1" />
  <row name="b" value="2" />
  <row name="c" value="3" />
</aaa>
<aaa Main="Second">
  <row name="a" value="3" />
  <row name="b" value="4" />
  <row name="c" value="5" />
</aaa>
<aaa Main="Third">
  <row name="a" value="10" />
  <row name="b" value="20" />
  <row name="c" value="30" />
</aaa>
</TableData>'

SELECT t.col.value('../@Main', 'varchar(10)') [Header],
t.col.value('@name', 'VARCHAR(25)') [name],
t.col.value('@value',  'VARCHAR(25)') [Value]
FROM    @xmlIn.nodes('//TableData/aaa/row') AS t (col)
```

```
Header    name    Value
First      a        1
First      b        2
First      c        3
Second     a        3
Second     b        4
Second     c        5
Third      a        10
Third      b        20
Third      c        30
```

XML : https://riptutorial.com/ko/sql/topic/4421/xml

18:

Examples

ON

.
().
.
.

, N .

.

```
ALTER TABLE dbo.T_Room  WITH CHECK ADD  CONSTRAINT FK_T_Room_T_Client FOREIGN KEY(RM_CLI_ID)
REFERENCES dbo.T_Client (CLI_ID)
GO
```

.

```
DELETE FROM T_Client WHERE CLI_ID = x
```

.

. . . N . (:).

.
`ON DELETE CASCADE` .

```
ALTER TABLE dbo.T_Room  -- WITH CHECK -- SQL-Server can specify WITH CHECK/WITH NOCHECK
ADD  CONSTRAINT FK_T_Room_T_Client FOREIGN KEY(RM_CLI_ID)
REFERENCES dbo.T_Client (CLI_ID)
ON DELETE CASCADE
```

.

```
DELETE FROM T_Client WHERE CLI_ID = x
```

.
- .

: Microsoft SQL-Server . .

```
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_T_FMS_Navigation_T_FMS_Navigation]') AND parent_object_id =
OBJECT_ID(N'[dbo].[T_FMS_Navigation]'))
ALTER TABLE [dbo].[T_FMS_Navigation]  WITH CHECK ADD  CONSTRAINT
```

```
[FK_T_FMS_Navigation_T_FMS_Navigation] FOREIGN KEY([NA_NA_UID])
REFERENCES [dbo].[T_FMS_Navigation] ([NA_UID])
ON DELETE CASCADE
GO

IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_T_FMS_Navigation_T_FMS_Navigation]') AND parent_object_id =
OBJECT_ID(N'[dbo].[T_FMS_Navigation]'))
ALTER TABLE [dbo].[T_FMS_Navigation] CHECK CONSTRAINT [FK_T_FMS_Navigation_T_FMS_Navigation]
GO
```

Microsoft-SQL-server `ON DELETE CASCADE` . .

PostgreSQL .

.

.

.

:

. "T_Room"... ( )

: https://riptutorial.com/ko/sql/topic/3518/-

19:

SELECT SQL . FROM .

- SELECT [DISTINCT] [column1] [, [column2] ...]
  FROM [table]
  [WHERE]
  [GROUP BY [column1] [, [column2] ...]

  [HAVING [column1] [, [column2] ...]

  [ASC | DESC]

SELECT (( ).

```
SELECT Name, SerialNumber
FROM ArmyInfo
```

`Name` `Serial Number` `Rank` (:

```
SELECT *
FROM ArmyInfo
```

. `SELECT *` .

Examples

.

.

:

	FName	LName	
1			
2			4

:

1	
2	

4	

select

`*` .

, `FROM` . `JOIN` .

.

```
SELECT * FROM Employees
```

`Employees` .

	FName	LName	
1			
2			4

.

.

```
SELECT
    Employees.*,
    Departments.Name
FROM
    Employees
JOIN
    Departments
    ON Departments.Id = Employees.DeptId
```

`Employee` `Departments` `Name` .

	FName	LName		
1				
2			4	

`*` .

1. IO, , . .
2. `SELECT <columns> FROM <table>` IO .
3. (/ IO)
    -
    - /
4.

. `SELECT * FROM orders JOIN people ON people.id = orders.personid ORDER BY displayname` - `displayname` `ORDER BY` (MS SQL Server " ") .

* ,?

* .

(, `tablealias.*` *).

`SELECT A.col1, A.Col2 FROM A WHERE EXISTS (SELECT * FROM B where A.ID = B.A_ID)` `EXISTS` `SELECT A.col1, A.Col2 FROM A WHERE EXISTS (SELECT * FROM B where A.ID = B.A_ID)` . B , * . `COUNT(*)` .

WHERE SELECT .

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

[condition] >, <, =, <>,> =, <=, LIKE, NOT, IN, BETWEEN SQL .

'READY' 'Cars' .

```
SELECT * FROM Cars WHERE status = 'READY'
```

WHERE HAVING .

```
SELECT
    PhoneNumber,
    Email,
    PreferredContact
FROM Customers
```

`Customers` `PhoneNumber` , `Email` `PreferredContact` . `SELECT` .

.

		PreferredContact
3347927472	william.jones@example.com	
2137921892	dmiller@example.net	
	richard0123@example.com	

`[table_name].[column_name]` `[table_name].[column_name]` `[table_name].[column_name]` `[table_name].[column_name]`

```
SELECT
    Customers.PhoneNumber,
    Customers.Email,
    Customers.PreferredContact,
    Orders.Id AS OrderId
FROM
    Customers
LEFT JOIN
    Orders ON Orders.CustomerId = Customers.Id
```

* `AS OrderId` `Orders` `Id` `OrderId` . .

. . ( ) . `Customers c` `Customers AS c` . `c` `Customers` `Email` . `c.Email` .

```
SELECT
    c.PhoneNumber,
    c.Email,
    c.PreferredContact,
    o.Id AS OrderId
FROM
    Customers c
LEFT JOIN
    Orders o ON o.CustomerId = c.Id
```

SELECT

.

(: ID 2) . .

.

SQL

(`"`) SQL .

```
SELECT
    FName AS "First Name",
    MName AS "Middle Name",
    LName AS "Last Name"
FROM Employees
```

SQL

(`'`), (`"`) (`[]`) Microsoft SQL Server .

```
SELECT
    FName AS "First Name",
    MName AS 'Middle Name',
    LName AS [Last Name]
FROM Employees
```

:

() `FName` `LName` . `AS` . , .

```
SELECT
    FName "First Name",
    MName "Middle Name",
    LName "Last Name"
FROM Employees
```


(, `AS`) .

, .

```
SELECT
    FName AS FirstName,
    LName AS LastName
FROM Employees
```


MS SQL Server `<alias> = <column-or-calculation>` . .

```
SELECT FullName = FirstName + ' ' + LastName,
       Addr1    = FullStreetAddress,
       Addr2    = TownName
FROM CustomerDetails
```

.

```
SELECT FirstName + ' ' + LastName As FullName
       FullStreetAddress          As Addr1,
       TownName                   As Addr2
FROM CustomerDetails
```

:

FullName	Addr1	Addr2
	123 AnyStreet	
	668 MyRoad	
	999	

= As , . = .

SQL

, :

```
SELECT
    FName as "SELECT",
    MName as "FROM",
    LName as "WHERE"
FROM Employees
```

SQL

MSSQL .

```
SELECT
    FName AS "SELECT",
    MName AS 'FROM',
    LName AS [WHERE]
FROM Employees
```


ORDER BY .

```
SELECT
    FName AS FirstName,
    LName AS LastName
FROM
    Employees
ORDER BY
    LastName DESC
```

,

```
SELECT
    FName AS SELECT,
    LName AS FROM
FROM
    Employees
ORDER BY
    LastName DESC
```

(SELECT FROM)

.

```
SELECT * FROM Employees ORDER BY LName
```

Employees .

	FName	LName	
2			2468101214
1			1234567890
			1357911131

```
SELECT * FROM Employees ORDER BY LName DESC
```

```
 SELECT * FROM Employees ORDER BY LName ASC
```

.

. :

```
SELECT * FROM Employees ORDER BY LName ASC, FName ASC
```

LName LName FName . .

ORDER BY . 1 .

```
SELECT Id, FName, LName, PhoneNumber FROM Employees ORDER BY 3
```

CASE ORDER BY .

```
SELECT Id, FName, LName, PhoneNumber FROM Employees ORDER BY CASE WHEN LName='Jones` THEN 0
ELSE 1 END ASC
```

LName "Jones" .

.

SQL .

```
SELECT
    "ORDER",
    ID
FROM ORDERS
```

.

DBMS . , SQL Server .

```
SELECT
    [Order],
    ID
FROM ORDERS
```

MySQL (MariaDB) .

```
SELECT
    `Order`,
    id
FROM orders
```

SQL 2008 `FETCH FIRST` .

```
SELECT Id, ProductName, UnitPrice, Package
FROM Product
ORDER BY UnitPrice DESC
FETCH FIRST 10 ROWS ONLY
```

RDMS . . OpenEdge 11.x `FETCH FIRST <n> ROWS ONLY` .

`FETCH FIRST <n> ROWS ONLY OFFSET <m> ROWS` .

```
SELECT Id, ProductName, UnitPrice, Package
FROM Product
ORDER BY UnitPrice DESC
OFFSET 5 ROWS
FETCH FIRST 10 ROWS ONLY
```

SQL Server MS Access .

```
SELECT TOP 10 Id, ProductName, UnitPrice, Package
FROM Product
ORDER BY UnitPrice DESC
```

MySQL PostgreSQL `LIMIT` .

```
SELECT Id, ProductName, UnitPrice, Package
FROM Product
ORDER BY UnitPrice DESC
LIMIT 10
```

Oracle `ROWNUM` .

```
SELECT Id, ProductName, UnitPrice, Package
FROM Product
WHERE ROWNUM <= 10
ORDER BY UnitPrice DESC
```

: **10** .

```
Id     ProductName                 UnitPrice             Package
38     Côte de Blaye               263.50                12 - 75 cl bottles
```

```
29    Thüringer Rostbratwurst   123.79                50 bags x 30 sausgs.
9    Mishi Kobe Niku            97.00                 18 - 500 g pkgs.
20    Sir Rodney's Marmalade    81.00                 30 gift boxes
18    Carnarvon Tigers          62.50                 16 kg pkg.
59    Raclette Courdavault      55.00                 5 kg pkg.
51    Manjimup Dried Apples     53.00                 50 - 300 g pkgs.
62    Tarte au sucre            49.30                 48 pies
43    Ipoh Coffee               46.00                 16 - 500 g tins
28    Rössle Sauerkraut         45.60                 25 - 825 g cans
```

:

Microsoft SQL `TOP` `WHERE` `ROWNUM` `WHERE` `WHERE` . 0.

```
SELECT e.Fname, e.LName
FROM Employees e
```

Employees 'e' . .

```
SELECT e.Fname, e.LName, m.Fname AS ManagerFirstName
FROM Employees e
    JOIN Managers m ON e.ManagerId = m.Id
```

. ,

```
SELECT e.Fname, Employees.LName, m.Fname AS ManagerFirstName
FROM Employees e
JOIN Managers m ON e.ManagerId = m.Id
```

.

`INNER JOIN` SQL ' ' . 1992 SQL `NATURAL JOIN` (mySQL, PostgreSQL Oracle SQL Server) , . (`Id` `ManagerId`) (`LName` , `FName`) :

```
SELECT Fname, LName, ManagerFirstName
FROM Employees
     NATURAL JOIN
     ( SELECT Id AS ManagerId, Fname AS ManagerFirstName
       FROM Managers ) m;
```

dervied / (SQL .) .

```
SELECT *
FROM
    table1,
    table2
```

```
SELECT
    table1.column1,
    table1.column2,
    table2.column1
FROM
    table1,
    table2
```

SQL (cross product) ,

.

.

`AVG()` .

```
SELECT AVG(Salary) FROM Employees
```

where .

```
SELECT AVG(Salary) FROM Employees where DepartmentId = 1
```

group by .

.

```
SELECT AVG(Salary) FROM Employees GROUP BY DepartmentId
```

`MIN()` .

```
SELECT MIN(Salary) FROM Employees
```

`MAX()` .

```
SELECT MAX(Salary) FROM Employees
```

`COUNT()` .

```
SELECT Count(*) FROM Employees
```

.

```
SELECT Count(*) FROM Employees where ManagerId IS NOT NULL
```

. `NULL` .

```
Select Count(ManagerId) from Employees
```

Count distinct count distinct .

```
Select Count(DISTINCT DepartmentId) from Employees
```

`SUM()` .

```
SELECT SUM(Salary) FROM Employees
```

null

```
SELECT Name FROM Customers WHERE PhoneNumber IS NULL
```

nulls . `=` , `IS NULL` `IS NOT NULL` .

CASE

'on the fly' CASE .

```
SELECT CASE WHEN Col1 < 50 THEN 'under' ELSE 'over' END threshold
FROM TableName
```

.

```
SELECT
    CASE WHEN Col1 < 50 THEN 'under'
         WHEN Col1 > 50 AND Col1 <100 THEN 'between'
         ELSE 'over'
    END threshold
FROM TableName
```

`CASE` `CASE` .

```
SELECT
    CASE WHEN Col1 < 50 THEN 'under'
         ELSE
            CASE WHEN Col1 > 50 AND Col1 <100 THEN Col1
            ELSE 'over' END
    END threshold
FROM TableName
```

LOCK select .

SQL

```
SELECT * FROM TableName WITH (nolock)
```

MySQL

```
SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SELECT * FROM TableName;
SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;
```

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SELECT * FROM TableName;
```

DB2

```
SELECT * FROM TableName WITH UR;
```

`UR` " ".

.

( )

```
SELECT DISTINCT ContinentCode
FROM Countries;
```

`Countries` `ContinentCode` `DISTINCT` ( ).

OC
AF

SQLFiddle

.

```
SELECT * FROM Cars  WHERE status IN ( 'Waiting', 'Working' )
```

.

```
SELECT * FROM Cars  WHERE ( status = 'Waiting' OR status = 'Working' )
```

, `value IN ( <value list> )` (`OR`).

:

```
SELECT category, COUNT(*) AS item_count
FROM item
```

```
GROUP BY category;
```

:

```
SELECT department, AVG(income)
FROM employees
GROUP BY department;
```

GROUP BY .

WHERE GROUP BY , WHERE :

```
SELECT department, AVG(income)
FROM employees
WHERE department <> 'ACCOUNTING'
GROUP BY department;
```

(: 1000) HAVING .

```
SELECT department, AVG(income)
FROM employees
WHERE department <> 'ACCOUNTING'
GROUP BY department
HAVING avg(income) > 1000;
```

▪

AND .

	18	
	21	
	22	
	23	

```
SELECT name FROM persons WHERE gender = 'M' AND age > 20;
```

.

OR

```
SELECT name FROM persons WHERE gender = 'M' OR age < 20;
```

.

.

```
SELECT name
FROM persons
WHERE (gender = 'M' AND age < 20)
   OR (gender = 'F' AND age > 20);
```

.

: https://riptutorial.com/ko/sql/topic/222/

20:

- WITH QueryName [(ColumnName, ...)] AS (
  ...
)
  SELECT ... FROM QueryName ...;

- RECURSIVE QueryName [(ColumnName, ...)] AS (
  ...
  UNION []
  SELECT ... FROM QueryName ...
)
  SELECT ... FROM QueryName ...;

: WITH

. WITH . CTE Temp Table Table TempDB .

:

- , .
- .
- .
- / .
- (SQL).
- . , .
- . CTE ( ).

Examples

.

```
WITH ReadyCars AS (
  SELECT *
  FROM Cars
  WHERE Status = 'READY'
)
SELECT ID, Model, TotalCost
FROM ReadyCars
ORDER BY TotalCost;
```

1	F-150	200
2	F-150	230

```
SELECT ID, Model, TotalCost
```

```
FROM (
  SELECT *
  FROM Cars
  WHERE Status = 'READY'
) AS ReadyCars
ORDER BY TotalCost
```

.

```
WITH RECURSIVE ManagersOfJonathon AS (
    -- start with this row
    SELECT *
    FROM Employees
    WHERE ID = 4

    UNION ALL

    -- get manager(s) of all previously selected rows
    SELECT Employees.*
    FROM Employees
    JOIN ManagersOfJonathon
        ON Employees.ID = ManagersOfJonathon.ManagerID
)
SELECT * FROM ManagersOfJonathon;
```

	FName	LName		ID	DepartmentId
4			1212121212	**2**	1
2			2468101214	**1**	1
1			1234567890		1

. 2 D .

1 - 5 `i` `Numbers` .

```
--Give a table name `Numbers" and a column `i` to hold the numbers
WITH Numbers(i) AS (
    --Starting number/index
    SELECT 1
    --Top-level UNION ALL operator required for recursion
    UNION ALL
    --Iteration expression:
    SELECT i + 1
    --Table expression we first declared used as source for recursion
    FROM Numbers
    --Clause to define the end of the recursion
    WHERE i < 5
)
--Use the generated table expression like a regular table
SELECT i FROM Numbers;
```

1

2
4
5

.

```
WITH RECURSIVE ManagedByJames(Level, ID, FName, LName) AS (
    -- start with this row
    SELECT 1, ID, FName, LName
    FROM Employees
    WHERE ID = 1

    UNION ALL

    -- get employees that have any of the previously selected rows as manager
    SELECT ManagedByJames.Level + 1,
           Employees.ID,
           Employees.FName,
           Employees.LName
    FROM Employees
    JOIN ManagedByJames
        ON Employees.ManagerID = ManagedByJames.ID

    ORDER BY 1 DESC   -- depth-first search
)
SELECT * FROM ManagedByJames;
```

		FName	LName
1	1		
2	2		
	4		
2			

CTE Oracle CONNECT BY

CONNECT BY SQL CTE . SQL Server ( ). .

```
  WITH tbl AS (
       SELECT id, name, parent_id
         FROM mytable)
     , tbl_hierarchy AS (
       /* Anchor */
       SELECT 1 AS "LEVEL"
            --, 1 AS CONNECT_BY_ISROOT
            --, 0 AS CONNECT_BY_ISBRANCH
            , CASE WHEN t.id IN (SELECT parent_id FROM tbl) THEN 0 ELSE 1 END AS
```

```
CONNECT_BY_ISLEAF
            , 0 AS CONNECT_BY_ISCYCLE
            , '/' + CAST(t.id   AS VARCHAR(MAX)) + '/' AS SYS_CONNECT_BY_PATH_id
            , '/' + CAST(t.name AS VARCHAR(MAX)) + '/' AS SYS_CONNECT_BY_PATH_name
            , t.id AS root_id
            , t.*
         FROM tbl t
        WHERE t.parent_id IS NULL                            -- START WITH parent_id IS NULL
       UNION ALL
       /* Recursive */
       SELECT th."LEVEL" + 1 AS "LEVEL"
            --, 0 AS CONNECT_BY_ISROOT
            --, CASE WHEN t.id IN (SELECT parent_id FROM tbl) THEN 1 ELSE 0 END AS
CONNECT_BY_ISBRANCH
            , CASE WHEN t.id IN (SELECT parent_id FROM tbl) THEN 0 ELSE 1 END AS
CONNECT_BY_ISLEAF
            , CASE WHEN th.SYS_CONNECT_BY_PATH_id LIKE '%/' + CAST(t.id AS VARCHAR(MAX)) +
'/%' THEN 1 ELSE 0 END AS CONNECT_BY_ISCYCLE
            , th.SYS_CONNECT_BY_PATH_id   + CAST(t.id   AS VARCHAR(MAX)) + '/' AS
SYS_CONNECT_BY_PATH_id
            , th.SYS_CONNECT_BY_PATH_name + CAST(t.name AS VARCHAR(MAX)) + '/' AS
SYS_CONNECT_BY_PATH_name
            , th.root_id
            , t.*
         FROM tbl t
              JOIN tbl_hierarchy th ON (th.id = t.parent_id) -- CONNECT BY PRIOR id =
parent_id
        WHERE th.CONNECT_BY_ISCYCLE = 0)                     -- NOCYCLE
SELECT th.*
     --, REPLICATE(' ', (th."LEVEL" - 1) * 3) + th.name AS tbl_hierarchy
  FROM tbl_hierarchy th
       JOIN tbl CONNECT_BY_ROOT ON (CONNECT_BY_ROOT.id = th.root_id)
 ORDER BY th.SYS_CONNECT_BY_PATH_name;                       -- ORDER SIBLINGS BY name
```

CONNECT BY :

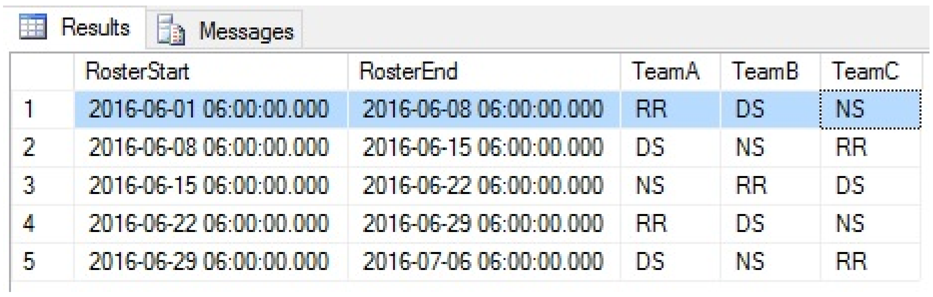
- - CONNECT BY : .
  - START WITH : .
  - ORDER SIBLINGS BY : .
- - NOCYCLE : . Directed Acyclic Graphs .
- - PRIOR : .
  - CONNECT_BY_ROOT : .
- - LEVEL : .
  - CONNECT_BY_ISLEAF : .
  - CONNECT_BY_ISCYCLE : .
- - SYS_CONNECT_BY_PATH : / .

,

```
DECLARE @DateFrom DATETIME = '2016-06-01 06:00'
DECLARE @DateTo DATETIME = '2016-07-01 06:00'
DECLARE @IntervalDays INT = 7

-- Transition Sequence = Rest & Relax into Day Shift into Night Shift
-- RR (Rest & Relax) = 1
-- DS (Day Shift) = 2
-- NS (Night Shift) = 3
```

```
;WITH roster AS
(
   SELECT @DateFrom AS RosterStart, 1 AS TeamA, 2 AS TeamB, 3 AS TeamC
   UNION ALL
   SELECT DATEADD(d, @IntervalDays, RosterStart),
          CASE TeamA WHEN 1 THEN 2 WHEN 2 THEN 3 WHEN 3 THEN 1 END AS TeamA,
          CASE TeamB WHEN 1 THEN 2 WHEN 2 THEN 3 WHEN 3 THEN 1 END AS TeamB,
          CASE TeamC WHEN 1 THEN 2 WHEN 2 THEN 3 WHEN 3 THEN 1 END AS TeamC
   FROM roster WHERE RosterStart < DATEADD(d, -@IntervalDays, @DateTo)
)

SELECT RosterStart,
       ISNULL(LEAD(RosterStart) OVER (ORDER BY RosterStart), RosterStart + @IntervalDays) AS
RosterEnd,
       CASE TeamA WHEN 1 THEN 'RR' WHEN 2 THEN 'DS' WHEN 3 THEN 'NS' END AS TeamA,
       CASE TeamB WHEN 1 THEN 'RR' WHEN 2 THEN 'DS' WHEN 3 THEN 'NS' END AS TeamB,
       CASE TeamC WHEN 1 THEN 'RR' WHEN 2 THEN 'DS' WHEN 3 THEN 'NS' END AS TeamC
FROM roster
```

1 TeamA R & R, TeamB Day Shift, TeamC Night Shift .



Results | Messages

	RosterStart	RosterEnd	TeamA	TeamB	TeamC
1	2016-06-01 06:00:00.000	2016-06-08 06:00:00.000	RR	DS	NS
2	2016-06-08 06:00:00.000	2016-06-15 06:00:00.000	DS	NS	RR
3	2016-06-15 06:00:00.000	2016-06-22 06:00:00.000	NS	RR	DS
4	2016-06-22 06:00:00.000	2016-06-29 06:00:00.000	RR	DS	NS
5	2016-06-29 06:00:00.000	2016-07-06 06:00:00.000	DS	NS	RR

20 .

.

```
SELECT category.description, sum(product.price) as total_sales
FROM sale
LEFT JOIN product on sale.product_id = product.id
LEFT JOIN category on product.category_id = category.id
GROUP BY category.id, category.description
HAVING sum(product.price) > 20
```

:

```
WITH all_sales AS (
  SELECT product.price, category.id as category_id, category.description as
category_description
  FROM sale
  LEFT JOIN product on sale.product_id = product.id
  LEFT JOIN category on product.category_id = category.id
)
, sales_by_category AS (
  SELECT category_description, sum(price) as total_sales
  FROM all_sales
  GROUP BY category_id, category_description
)
SELECT * from sales_by_category WHERE total_sales > 20
```

SQL

" " " " .

.

```
-- all_sales: just a simple SELECT with all the needed JOINS
WITH all_sales AS (
  SELECT
  product.price as product_price,
  category.id as category_id,
  category.description as category_description
  FROM sale
  LEFT JOIN product on sale.product_id = product.id
  LEFT JOIN category on product.category_id = category.id
)
-- Group by category
, sales_by_category AS (
  SELECT category_id, category_description,
  sum(product_price) as total_sales
  FROM all_sales
  GROUP BY category_id, category_description
)
-- Filtering total_sales > 20
, top_categories AS (
  SELECT * from sales_by_category WHERE total_sales > 20
)
-- all_products: just a simple SELECT with all the needed JOINS
, all_products AS (
  SELECT
  product.id as product_id,
  product.description as product_description,
  product.price as product_price,
  category.id as category_id,
  category.description as category_description
  FROM product
  LEFT JOIN category on product.category_id = category.id
)
-- Order by product price
, cheapest_products AS (
  SELECT * from all_products
  ORDER by product_price ASC
)
-- Simple inner join
, cheapest_products_from_top_categories AS (
  SELECT product_description, product_price
  FROM cheapest_products
  INNER JOIN top_categories ON cheapest_products.category_id = top_categories.category_id
)
--The main SELECT
SELECT * from cheapest_products_from_top_categories
```

: https://riptutorial.com/ko/sql/topic/747/--

21:

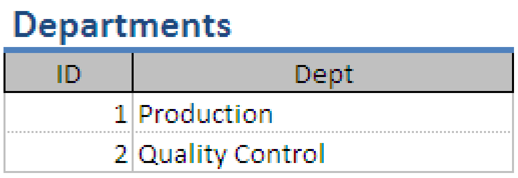
Examples

SQL . , . , . , .

:

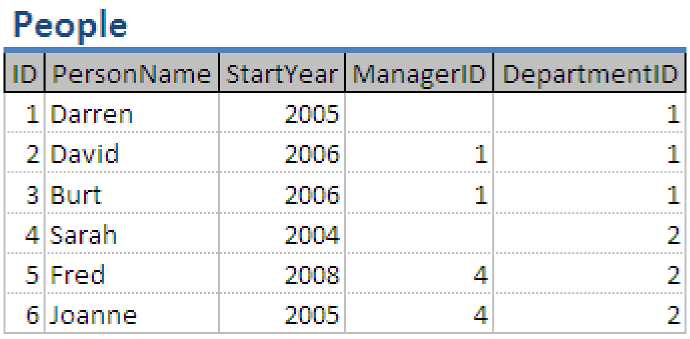
- .
- ( )

.



Departments

ID	Dept
1	Production
2	Quality Control



People

ID	PersonName	StartYear	ManagerID	DepartmentID
1	Darren	2005		1
2	David	2006	1	1
3	Burt	2006	1	1
4	Sarah	2004		2
5	Fred	2008	4	2
6	Joanne	2005	4	2

select .
<table> **.** *<condition>*

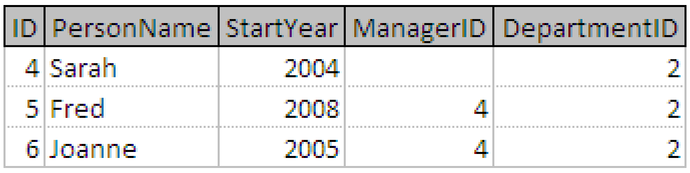
.

DepartmentID = 2

:

$$\sigma_{DepartmentID = 2}(People)$$

DepartmentID 2 *People* .

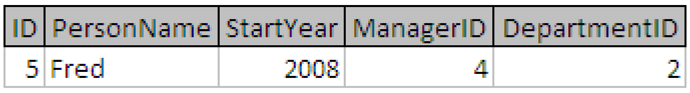


ID	PersonName	StartYear	ManagerID	DepartmentID
4	Sarah	2004		2
5	Fred	2008	4	2
6	Joanne	2005	4	2

.

StartYear> **2005** DepartmentID = 2

.



ID	PersonName	StartYear	ManagerID	DepartmentID
5	Fred	2008	4	2

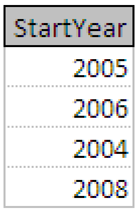
.
<table> ***over*** *<field list>*

, .
StartYear

:

$\pi_{StartYear}(People)$

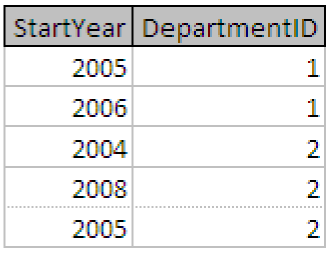
People StartYear .



StartYear
2005
2006
2004
2008

. .

.
StartUear , DepartmentID :



StartYear	DepartmentID
2005	1
2006	1
2004	2
2008	2
2005	2

2006 *StartYear* 1 *DepartmentID* .

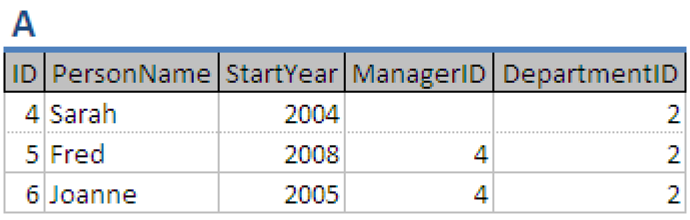
GIVING

give .

< > <>

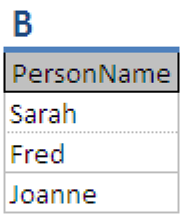
, .
DepartmentID = **2**
B PersonName A

A x B.



A

ID	PersonName	StartYear	ManagerID	DepartmentID
4	Sarah	2004		2
5	Fred	2008	4	2
6	Joanne	2005	4	2



B

PersonName
Sarah
Fred
Joanne

A .   B .

.
(DepartmentID = 2) PersonName B

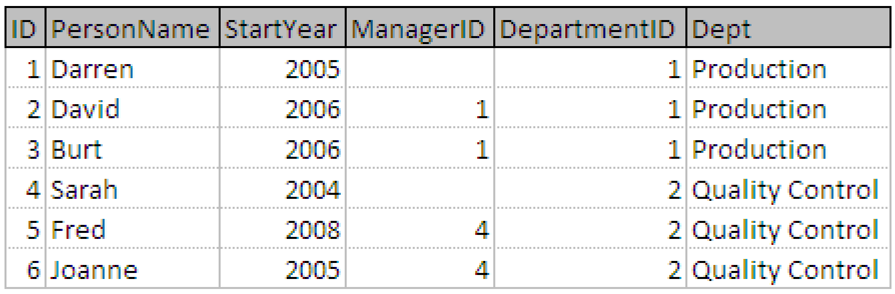
.

NATURAL JOIN

.

< ***1>*** *< 2> <1 > = < 2>*
<field 1> <table 1> <field 2> <table 2> .

DepartmentID *ID*   *People* and *Departments* .
DepartmentID = ID



ID	PersonName	StartYear	ManagerID	DepartmentID	Dept
1	Darren	2005		1	Production
2	David	2006	1	1	Production
3	Burt	2006	1	1	Production
4	Sarah	2004		2	Quality Control
5	Fred	2008	4	2	Quality Control
6	Joanne	2005	4	2	Quality Control

People *DepartmentID*   *Department* *ID* .   ,   .

. , *Name*   *PersonName* *Dept*   (, Person Name Department Name).   (*People.Name* *Departments.Name).*

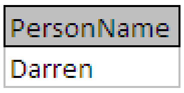
.

DepartmentID = ***ID***
= 2005 StartYear   = "B
PersonName C   B

:

((*= 2005 StartYear   = "DepartmentID = ID)) PersonName C*

.



PersonName
Darren

(: =)

: https://riptutorial.com/ko/sql/topic/7311/-

22:

. . Oracle PostgreSQL . SQL Server DB2 .

Examples

PostgreSQL

```
CREATE TABLE mytable (number INT);
INSERT INTO mytable VALUES (1);

CREATE MATERIALIZED VIEW myview AS SELECT * FROM mytable;

SELECT * FROM myview;
 number
--------
      1
(1 row)

INSERT INTO mytable VALUES(2);

SELECT * FROM myview;
 number
--------
      1
(1 row)

REFRESH MATERIALIZED VIEW myview;

SELECT * FROM myview;
 number
--------
      1
      2
(2 rows)
```

: https://riptutorial.com/ko/sql/topic/8367/--

23:

- MySQL : CREATE TABLE Employees (Id int NOT NULL, PRIMARY KEY (Id), ...);
- : CREATE TABLE Employees (ID int NOT NULL PRIMARY KEY, ...);

Examples

```
CREATE TABLE Employees (
    Id int NOT NULL,
    PRIMARY KEY (Id),
    ...
);
```

'Id'   Employees .   .   .

.   .

```
CREATE TABLE EMPLOYEE (
    e1_id INT,
    e2_id INT,
    PRIMARY KEY (e1_id, e2_id)
)
```

.   .

MySQL

```
CREATE TABLE Employees (
    Id int NOT NULL AUTO_INCREMENT,
    PRIMARY KEY (Id)
);
```

PostgreSQL

```
CREATE TABLE Employees (
    Id SERIAL PRIMARY KEY
);
```

SQL

```
CREATE TABLE Employees (
    Id int NOT NULL IDENTITY,
    PRIMARY KEY (Id)
);
```

SQLite

```
CREATE TABLE Employees (
    Id INTEGER PRIMARY KEY
);
```

: https://riptutorial.com/ko/sql/topic/505/-

24:

- INSERT INTO table_name (column1, column2, column3, ...) VALUES (1, 2, 3, ...);
- INSERT INTO table_name (column1, column2 ...)   1, 2 ... from other_table

Examples

```
INSERT INTO Customers
VALUES ('Zack', 'Smith', 'zack@example.com', '7049989942', 'EMAIL');
```

Customers   . Id   .   .

```
INSERT INTO Customers (FName, LName, Email, PreferredContact)
VALUES ('Zack', 'Smith', 'zack@example.com', 'EMAIL');
```

Customers   .   . PhoneNumber   . not null   .

SELECT   INSERT

```
INSERT INTO Customers (FName, LName, PhoneNumber)
SELECT FName, LName, PhoneNumber FROM Employees
```

Employees Customers .   .   . Id ID   .

.

```
INSERT INTO Table1
SELECT * FROM Table2
```

.

```
INSERT INTO tbl_name (field1, field2, field3)

VALUES (1,2,3), (4,5,6), (7,8,9);
```

( )   DBMS   .

MySQL - LOAD DATA INFILE

MSSQL -

: https://riptutorial.com/ko/sql/topic/465/-

25:

Examples

10

. DECIMAL NUMERIC .

:

```
DECIMAL ( precision [ , scale] )
NUMERIC ( precision [ , scale] )
```

:

```
SELECT CAST(123 AS DECIMAL(5,2)) --returns 123.00
SELECT CAST(12345.12 AS NUMERIC(10,5)) --returns 12345.12000
```

.

```
SELECT CAST( PI() AS FLOAT) --returns 3.14159265358979
SELECT CAST( PI() AS REAL) --returns 3.141593
```

.

bigint	-2 ^ 63 (-9,223,372,036,854,775,808) ~ 2 ^ 63-1 (9,223,372,036,854,775,807)	8
int	-2 ^ 31 (-2,147,483,648) ~ 2 ^ 31-1 (2,147,483,647)	4
	-2 ^ 15 (-32,768) ~ 2 ^ 15-1 (32,767)	2
	0 ~ 255	1

.

	-922,337,203,685,477.5808 ~ 922,337,203,685,477.5807	8
	-214,748.3648 214,748.3647	4

(BINARY) (VARBINARY)

2 .

:

```
BINARY [ ( n_bytes ) ]
VARBINARY [ ( n_bytes | max ) ]
```

`n_bytes` 1 - 8000 . `max` 2 ^ 31-1 .

:

```
SELECT CAST(12345 AS BINARY(10)) -- 0x00000000000000003039
SELECT CAST(12345 AS VARBINARY(10)) -- 0x00003039
```

CHAR VARCHAR

.

:

```
CHAR [ ( n_chars ) ]
VARCHAR [ ( n_chars ) ]
```

:

```
SELECT CAST('ABC' AS CHAR(10)) -- 'ABC       ' (padded with spaces on the right)
SELECT CAST('ABC' AS VARCHAR(10)) -- 'ABC' (no padding due to variable character)
SELECT CAST('ABCDEFGHIJKLMNOPQRSTUVWXYZ' AS CHAR(10))  -- 'ABCDEFGHIJ' (truncated to 10
characters)
```

NCHAR NVARCHAR

UNICODE .

:

```
NCHAR [ ( n_chars ) ]
NVARCHAR [ ( n_chars | MAX ) ]
```

8000 `MAX` .

16 GUID / UUID.

```
DECLARE @GUID UNIQUEIDENTIFIER = NEWID();
SELECT @GUID -- 'E28B3BD9-9174-41A9-8508-899A78A33540'
DECLARE @bad_GUID_string VARCHAR(100) = 'E28B3BD9-9174-41A9-8508-899A78A33540_foobarbaz'
SELECT
    @bad_GUID_string,   -- 'E28B3BD9-9174-41A9-8508-899A78A33540_foobarbaz'
    CONVERT(UNIQUEIDENTIFIER, @bad_GUID_string) -- 'E28B3BD9-9174-41A9-8508-899A78A33540'
```

: https://riptutorial.com/ko/sql/topic/1166/-

26:

- CREATE DATABASE dbname;

Examples

SQL .

```
CREATE DATABASE myDatabase;
```

myDatabase .

: https://riptutorial.com/ko/sql/topic/2744/-

27:

Examples

```
CREATE SYNONYM EmployeeData
FOR MyDatabase.dbo.Employees
```

: https://riptutorial.com/ko/sql/topic/2518/

28:

DROP TABLE , ,   .

Examples

```
Drop Table MyTable;
```

MySQL 3.19

```
DROP TABLE IF EXISTS MyTable;
```

PostgreSQL 8.x

```
DROP TABLE IF EXISTS MyTable;
```

SQL Server 2005

```
If Exists(Select * From Information_Schema.Tables
          Where Table_Schema = 'dbo'
            And Table_Name = 'MyTable')
  Drop Table dbo.MyTable
```

SQLite 3.0

```
DROP TABLE IF EXISTS MyTable;
```

: https://riptutorial.com/ko/sql/topic/1832/-

29:

.

, , .

- CONCAT (string_value1, string_value2 [, string_valueN])
- LTRIM (_)
- RTRIM (_)
- SUBSTRING (, ,)
- ASCII (_)
- (string_expression, integer_expression)
- (string_expression)
- UPPER (_)
- TRIM ([FROM])
- STRING_SPLIT (,)
- (character_expression, start, length, replaceWith_expression)
- REPLACE (string_expression, string_pattern, string_replacement)

Transact-SQL / Microsoft

MySQL

PostgreSQL

Examples

.

MSSQL `TRIM()`

```
SELECT LTRIM('  Hello  ') --returns 'Hello  '
SELECT RTRIM('  Hello  ') --returns '  Hello'
SELECT LTRIM(RTRIM('  Hello  ')) --returns 'Hello'
```

MySQL Oracle

```
SELECT TRIM('  Hello  ') --returns 'Hello'
```

(ANSI / ISO) SQL `||` . SQL Server .

```
SELECT 'Hello' || 'World' || '!'; --returns HelloWorld!
```

`CONCAT` :

```
SELECT CONCAT('Hello', 'World'); --returns 'HelloWorld'
```

CONCAT (Oracle).

```
SELECT CONCAT('Hello', 'World', '!'); --returns 'HelloWorld!'
```

.

```
SELECT CONCAT('Foo', CAST(42 AS VARCHAR(5)), 'Bar'); --returns 'Foo42Bar'
```

(: Oracle) . , CONCAT A CLOB NCLOB NCLOB . varchar2 CONCAT varchar2 .

```
SELECT CONCAT(CONCAT('Foo', 42), 'Bar') FROM dual; --returns Foo42Bar
```

+ (+).

```
SELECT 'Foo' + CAST(42 AS VARCHAR(5)) + 'Bar';
```

CONCAT SQL Server 2012 + .

```
SELECT UPPER('HelloWorld') --returns 'HELLOWORLD'
SELECT LOWER('HelloWorld') --returns 'helloworld'
```

. SUBSTRING (string_expression, start, length) . SQL 1 .

```
SELECT SUBSTRING('Hello', 1, 2) --returns 'He'
SELECT SUBSTRING('Hello', 3, 3) --returns 'llo'
```

LEN() n .

```
DECLARE @str1 VARCHAR(10) = 'Hello', @str2 VARCHAR(10) = 'FooBarBaz';
SELECT SUBSTRING(@str1, LEN(@str1) - 2, 3) --returns 'llo'
SELECT SUBSTRING(@str2, LEN(@str2) - 2, 3) --returns 'Baz'
```

. STRING_SPLIT() .

```
SELECT value FROM STRING_SPLIT('Lorem ipsum dolor sit amet.', ' ');
```

:

```
value
-----
Lorem
ipsum
dolor
sit
amet.
```

0 .

: `start` 1- (0 1).

:

```
STUFF ( character_expression , start , length , replaceWith_expression )
```

:

```
SELECT STUFF('FooBarBaz', 4, 3, 'Hello') --returns 'FooHelloBaz'
```

SQL

LEN .

```
SELECT LEN('Hello') -- returns 5

SELECT LEN('Hello '); -- returns 5
```

DATALENGTH .

```
SELECT DATALENGTH('Hello') -- returns 5

SELECT DATALENGTH('Hello '); -- returns 6
```

DATALENGTH charset .

```
DECLARE @str varchar(100) = 'Hello ' --varchar is usually an ASCII string, occupying 1 byte
per char
SELECT DATALENGTH(@str) -- returns 6

DECLARE @nstr nvarchar(100) = 'Hello ' --nvarchar is a unicode string, occupying 2 bytes per
char
SELECT DATALENGTH(@nstr) -- returns 12
```

: Length (char)

:

```
SELECT Length('Bible') FROM dual; --Returns 5
SELECT Length('righteousness') FROM dual; --Returns 13
SELECT Length(NULL) FROM dual; --Returns NULL
```

: LengthB, LengthC, Length2, Length4

:

`REPLACE( , , )`

:

```
SELECT REPLACE( 'Peter Steve Tom', 'Steve', 'Billy' ) --Return Values: Peter Billy Tom
```

.
LEFT (,)
RIGHT (,)

```
SELECT LEFT('Hello',2)  --return He
SELECT RIGHT('Hello',2) --return lo
```

Oracle SQL LEFT RIGHT . SUBSTR LENGTH   .
SUBSTR (, 1,)
SUBSTR (string-expression, length (string-expression) -integer + 1, integer)

```
SELECT SUBSTR('Hello',1,2)  --return He
SELECT SUBSTR('Hello',LENGTH('Hello')-2+1,2) --return lo
```

REVERSE (string-expression).

```
SELECT REVERSE('Hello') --returns olleH
```

`REPLICATE`   .

. REPLICATE (string-expression, integer)

```
SELECT REPLICATE ('Hello',4) --returns 'HelloHelloHelloHello'
```

REGEXP

MySQL 3.19

(   ) .

```
SELECT 'bedded' REGEXP '[a-f]' -- returns True

SELECT 'beam' REGEXP '[a-f]' -- returns False
```

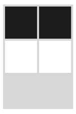
SQL

SQL   .   MySQL, Oracle SQL Server REPLACE ().

Replace   .

```
REPLACE (str, find, repl)
```

`South` Employees `Southern` .

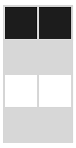


:

Replace :

```
SELECT
    FirstName,
    REPLACE (Address, 'South', 'Southern') Address
FROM Employees
ORDER BY FirstName
```

:



:

.

```
Update Employees
Set city = (Address, 'South', 'Southern');
```

WHERE .

```
Update Employees
Set Address = (Address, 'South', 'Southern')
Where Address LIKE 'South%';
```

: SQL Server

PARSENAME ( ) . , , .

MSDN : PARSENAME

```
PARSENAME('NameOfStringToParse',PartIndex)
```

1

```
SELECT PARSENAME('ServerName.DatabaseName.SchemaName.ObjectName',1)  // returns `ObjectName`
SELECT PARSENAME('[1012-1111].SchoolDatabase.school.Student',1)     // returns `Student`
```

2

```
SELECT PARSENAME('ServerName.DatabaseName.SchemaName.ObjectName',2)  // returns `SchemaName`
SELECT PARSENAME('[1012-1111].SchoolDatabase.school.Student',2)     // returns `school`
```

3

```
SELECT PARSENAME('ServerName.DatabaseName.SchemaName.ObjectName',3) // returns `DatabaseName`
SELECT PARSENAME('[1012-1111].SchoolDatabase.school.Student',3)    // returns `SchoolDatabase`
```

4

```
SELECT PARSENAME('ServerName.DatabaseName.SchemaName.ObjectName',4)  // returns `ServerName`
SELECT PARSENAME('[1012-1111].SchoolDatabase.school.Student',4)     // returns `[1012-1111]`
```

PARSENAME   null

INSTR

substring   (0).

: INSTR (,)

```
SELECT INSTR('FooBarBar', 'Bar') -- return 4
SELECT INSTR('FooBarBar', 'Xar') -- return 0
```

: https://riptutorial.com/ko/sql/topic/1120/-

30:

Examples

(MyTable)   (MyAudit)   . "inserted" INSERT UPDATE   Microsoft SQL Server   . DELETE " " .

```
CREATE TRIGGER MyTrigger
    ON MyTable
    AFTER INSERT

AS

BEGIN
    -- insert audit record to MyAudit table
    INSERT INTO MyAudit(MyTableId, User)
    (SELECT MyTableId, CURRENT_USER FROM inserted)
END
```

""

```
CREATE TRIGGER BooksDeleteTrigger
    ON MyBooksDB.Books
    AFTER DELETE
AS
  INSERT INTO BooksRecycleBin
    SELECT *
    FROM deleted;
GO
```

: https://riptutorial.com/ko/sql/topic/1432/

31:

MERGE ("update insert" UPSERT)   .   /   SQL   .

Examples

MERGE

```
MERGE INTO targetTable t
    USING sourceTable s
        ON t.PKID = s.PKID
    WHEN MATCHED AND NOT EXISTS (
            SELECT s.ColumnA, s.ColumnB, s.ColumnC
            INTERSECT
            SELECT t.ColumnA, t.ColumnB, s.ColumnC
            )
        THEN UPDATE SET
            t.ColumnA = s.ColumnA
            ,t.ColumnB = s.ColumnB
            ,t.ColumnC = s.ColumnC
    WHEN NOT MATCHED BY TARGET
        THEN INSERT (PKID, ColumnA, ColumnB, ColumnC)
        VALUES (s.PKID, s.ColumnA, s.ColumnB, s.ColumnC)
    WHEN NOT MATCHED BY SOURCE
        THEN DELETE
    ;
```

: `AND NOT EXISTS`   . `INTERSECT`   .

MySQL :

. `users` .

```
create table users(
    id int primary key auto_increment,
    name varchar(8),
    count int,
    unique key name(name)
);
```

(Joe)   .   ,   .   .

MySQL   : insert ... on duplicate key update ....   :

```
insert into users(name, count)
       values ('Joe', 1)
       on duplicate key update count=count+1;
```

PostgreSQL :

. `users` .

```
create table users(
    id serial,
    name varchar(8) unique,
    count int
);
```

(Joe)   .   ,   .   .

PostgreSQL   : insert ... conflict ... do update ....   :

```
insert into users(name, count)
    values('Joe', 1)
    on conflict (name) do update set count = users.count + 1;
```

: https://riptutorial.com/ko/sql/topic/1470/

32:

- GRANT [1] [, [2] ...] ON [] TO [ ] [, [ ] ...] [WITH GRANT OPTION]
- REVOKE [1] [, [2] ...] ON [] FROM [1] [, [2] ...]

. `WITH GRANT OPTION` , .

Examples

/

```
GRANT SELECT, UPDATE
ON Employees
TO User1, User2;
```

`Employees` `SELECT` `UPDATE` `User1` `User2` .

```
REVOKE SELECT, UPDATE
ON Employees
FROM User1, User2;
```

`User1` `User2` Employees `SELECT` `UPDATE` .

: https://riptutorial.com/ko/sql/topic/5574/---

33:

. ( ) .

. WHERE , JOIN ORDER BY .

.

. 2 .

. . INSERT . SELECT .

Examples

```
CREATE INDEX ix_cars_employee_id ON Cars (EmployeeId);
```

Cars EmployeeId . *EmployeeId* .

```
SELECT * FROM Cars WHERE EmployeeId = 1
```

.

```
CREATE INDEX ix_cars_e_c_o_ids ON Cars (EmployeeId, CarId, OwnerId);
```

. , .

.

```
SELECT * FROM Cars WHERE EmployeeId = 1 Order by CarId DESC
```

.

```
SELECT * FROM Cars WHERE OwnerId = 17 Order by CarId DESC
```

`OwnerId = 17` EmployeeId CarID .

`OwnerId` ( ).

.

```
CREATE CLUSTERED INDEX ix_clust_employee_id ON Employees(EmployeeId, Email);
```

SQL Employees . . . , . . (.)

```
CREATE UNIQUE INDEX uq_customers_email ON Customers(Email);
```

Customers *Email* . . *Email* .

```
CREATE UNIQUE INDEX ix_eid_desc ON Customers(EmployeeID);
```

EmployeeID Customers . (. ID .)

```
CREATE INDEX ix_eid_desc ON Customers(EmployeeID Desc);
```

. MSSQL .

```
UPDATE Customers SET Email = "richard0123@example.com" WHERE id = 1;
```

Customers *Email* . .

```
UPDATE Customers SET Email = "richard0123@example.com" WHERE id = 1 ON DUPLICATE KEY;
```

SAP ASE :

. `SAP ASE` .

:

```
DROP INDEX [table name].[index name]
```

:

```
DROP INDEX Cars.index_1
```

`SELECT` .

```
CREATE INDEX ix_scoreboard_score ON scoreboard (score DESC);
```

```
SELECT * FROM scoreboard ORDER BY score DESC;
```

.

```
DROP INDEX ix_cars_employee_id ON Cars;
```

`DROP` . `DROP` *ix_cars_employee_id* .

. . .

```
ALTER INDEX ix_cars_employee_id ON Cars DISABLE;
```

.

.   .

```
ALTER INDEX ix_cars_employee_id ON Cars REBUILD;
```

NULLS

```
CREATE UNIQUE INDEX idx_license_id
   ON Person(DrivingLicenseID) WHERE DrivingLicenseID IS NOT NULL
GO
```

0..1 . 0 1

B-Tree   / /   . SQLServer   (   )   (   )   .   .

.

```
ALTER INDEX index_name REBUILD;
```

DML   RDBMS   .   DB   `REORGANIZE` (SQLServer)   `COALESCE` / `SHRINK SPACE` (Oracle)   .

.   .

.   () .

Employee_Surname Employees :

```
CREATE CLUSTERED INDEX ix_employees_name ON Employees(Employee_Surname);
```

.   .

.   . ,   .   .

Employees Column Employee_Surname   .

```
CREATE NONCLUSTERED INDEX ix_employees_name ON Employees(Employee_Surname);
```

.   .   .

SQL Server SQLite   .

`order_state_id` finished (2) order   `order_state_id equal` started (1)   `order_state_id equal` .

:

```
SELECT id, comment
  FROM orders
 WHERE order_state_id =  1
   AND product_id = @some_value;
```

.

```
CREATE INDEX Started_Orders
          ON orders(product_id)
       WHERE order_state_id = 1;
```

.

: https://riptutorial.com/ko/sql/topic/344/

34:

Examples

DESCRIBE tablename;

DESCRIBE EXPLAIN . tablename DESCRIBE .

```
DESCRIBE tablename;
```

:

```
COLUMN_NAME     COLUMN_TYPE     IS_NULLABLE     COLUMN_KEY     COLUMN_DEFAULT     EXTRA
id              int(11)         NO              PRI            0
auto_increment
test            varchar(255)    YES                            (null)
```

. null . auto_increment

EXPLAIN

Explain select . .

:

```
explain select * from user join data on user.test = data.fk_user;
```

:

```
id  select_type  table   type    possible_keys  key     key_len ref       rows  Extra
1   SIMPLE       user    index   test           test    5       (null)    1     Using where;
Using index
1   SIMPLE       data    ref     fk_user        fk_user 5       user.test 1     (null)
```

type . possible_keys . key acutal used index . key_len () . . rows rows , .

: https://riptutorial.com/ko/sql/topic/2928/--

35:

- WHERE "RowCnt = 1" .
- Sum () Rank () WHERE Rank () = 1

Examples

```
WITH CTE (StudentId, Fname, LName, DOB, RowCnt)
as (
SELECT StudentId, FirstName, LastName, DateOfBirth as DOB, SUM(1) OVER (Partition By
FirstName, LastName, DateOfBirth) as RowCnt
FROM tblStudent
)
SELECT * from CTE where RowCnt > 1
ORDER BY DOB, LName
```

( ) .

: https://riptutorial.com/ko/sql/topic/1585/------

36:

Examples

```
CREATE SEQUENCE orders_seq
START WITH     1000
INCREMENT BY   1;
```

1000 1 .

seq_name .NEXTVAL . . NEXTVAL .

INSERT NEXTVAL .

```
INSERT INTO Orders (Order_UID, Customer)
        VALUES (orders_seq.NEXTVAL, 1032);
```

UPDATES .

```
UPDATE Orders
SET Order_UID = orders_seq.NEXTVAL
WHERE Customer = 581;
```

SELECTS .

```
SELECT Order_seq.NEXTVAL FROM dual;
```

: https://riptutorial.com/ko/sql/topic/1586/

37: ( )

Examples

ISO / ANSI SQL :

```
SELECT Id, Col1
FROM TableName
ORDER BY Id
OFFSET 20 ROWS
```

MySQL :

```
SELECT * FROM TableName LIMIT 20, 42424242424242;
-- skips 20 for take use very large number that is more than rows in table
```

:

```
SELECT Id,
   Col1
FROM (SELECT Id,
           Col1,
           row_number() over (order by Id) RowNumber
      FROM TableName)
WHERE RowNumber > 20
```

PostgreSQL :

```
SELECT * FROM TableName OFFSET 20;
```

SQLite :

```
SELECT * FROM TableName LIMIT -1 OFFSET 20;
```

ISO / ANSI SQL :

```
SELECT * FROM TableName FETCH FIRST 20 ROWS ONLY;
```

MySQL; PostgreSQL; SQLite :

```
SELECT * FROM TableName LIMIT 20;
```

:

```
SELECT Id,
   Col1
FROM (SELECT Id,
           Col1,
            row_number() over (order by Id) RowNumber
      FROM TableName)
WHERE RowNumber <= 20
```

SQL Server :

```
SELECT TOP 20 *
FROM dbo.[Sale]
```

( )

ISO / ANSI SQL :

```
SELECT Id, Col1
FROM TableName
ORDER BY Id
OFFSET 20 ROWS FETCH NEXT 20 ROWS ONLY;
```

MySQL :

```
SELECT * FROM TableName LIMIT 20, 20; -- offset, limit
```

; SQL Server :

```
SELECT Id,
   Col1
 FROM (SELECT Id,
            Col1,
            row_number() over (order by Id) RowNumber
      FROM TableName)
WHERE RowNumber BETWEEN 21 AND 40
```

PostgreSQL; SQLite :

```
SELECT * FROM TableName LIMIT 20 OFFSET 20;
```

( ) : https://riptutorial.com/ko/sql/topic/2927/-----

38:

, , .

, SQL SQL .

Examples

(`a` - `z`), (`0` - `9`) (`_`) .

SQL / (:

- MS SQL : `@` , `$` , `#` *( )*
- MySQL : `$` *( )*
- Oracle : `$` , `#` *( )*
- PostgreSQL : `$` *( )*

. SQL .

- MS SQL : / .
- MySQL : .
- : .
- PostgreSQL : .
- SQLite : ; / ASCII .

: https://riptutorial.com/ko/sql/topic/9677/

39:

Examples

...

```
BEGIN
  UPDATE Employees SET PhoneNumber = '5551234567' WHERE Id = 1;
  UPDATE Employees SET Salary = 650 WHERE Id = 3;
END
```

: https://riptutorial.com/ko/sql/topic/1632/-

40: ,

Examples

CROSS APPLY OUTER APPLY

.

.   .   Employee   .

CROSS APPLY   Employee .   Department Employee   .

```
SELECT *
FROM Department D
CROSS APPLY (
    SELECT *
    FROM Employee E
    WHERE E.DepartmentID = D.DepartmentID
) A
GO
SELECT *
FROM Department D
INNER JOIN Employee E
  ON D.DepartmentID = E.DepartmentID
```

. JOIN ,   ?

# 2   OUTER APPLY   Employee . Employee   5 6   NULL   .   Employee   LEFT OUTER JOIN .   . Employee   .

```
SELECT *
FROM Department D
OUTER APPLY (
    SELECT *
    FROM Employee E
    WHERE E.DepartmentID = D.DepartmentID
) A
GO
SELECT *
FROM Department D
LEFT OUTER JOIN Employee E
  ON D.DepartmentID = E.DepartmentID
GO
```

.   .

APPLY   . Script # 3 DepartmentID   .   Department   CROSS APPLY   .   (   ) DepartmentID   . u CROSS APPLY OUTER APPLY   G   CROSS APPLY OUTER APPLY   C NULL .

```
CREATE FUNCTION dbo.fn_GetAllEmployeeOfADepartment (@DeptID AS int)
```

```
RETURNS TABLE
AS
  RETURN
  (
  SELECT
    *
  FROM Employee E
  WHERE E.DepartmentID = @DeptID
  )
GO
SELECT
  *
FROM Department D
CROSS APPLY dbo.fn_GetAllEmployeeOfADepartment(D.DepartmentID)
GO
SELECT
  *
FROM Department D
OUTER APPLY dbo.fn_GetAllEmployeeOfADepartment(D.DepartmentID)
GO
```

?   CROSS / OUTER APPLY INNER JOIN / LEFT OUTER JOIN ON (1 = 1)   " " . D.DepartmentID "   ." . JOIN   ( )   /   .   APPLY .

,   : https://riptutorial.com/ko/sql/topic/2516/----

41:

JOIN () . (INNER / OUTER / CROSS LEFT / RIGHT / FULL) ( ) .

. FROM .

- `[ { INNER | { { LEFT | RIGHT | FULL } [ OUTER ] } } ] JOIN`

.

Examples

(" ") `join` .

Employees (FName) Departments (Name) .

```
SELECT Employees.FName, Departments.Name
FROM   Employees
JOIN   Departments
ON Employees.DepartmentId = Departments.Id
```

.

Employees.FName	Departments.Name

`from` , `where` . (`join`).

RDBMS . .

- .
- (CROSS JOIN) .

.

```
SELECT e.FName, d.Name
FROM   Employee e, Departments d
WHERE  e.DeptartmentId = d.Id
```

.

e.FName	d.

(   )   .   NULL .

.   NULL .

```
SELECT          Departments.Name, Employees.FName
FROM            Departments
LEFT OUTER JOIN Employees
ON              Departments.Id = Employees.DepartmentId
```

.

Departments.Name	Employees.FName

?

FROM   .

	FName	LName		ID	DepartmentId		HireDate
1			1234567890		1	1000	01-01-2002
2			2468101214	1	1	400	23-03-2005
			1357911131	1	2	600	12-05-2009
4			1212121212	2	1	500	24-07-2016

1	
2	

.
(*Departments.Id = Employees.DepartmentId*)   .   .

LEFT OUTER JOIN   (Departments)   RIGHT   NULL . NULL **Tech** NULL

			FName	LName		ID	DepartmentId		HireDate
1		**1**			**1234567890**		**1**	**1000**	**01-01-2002**

			FName	LName		ID	DepartmentId		HireDate
1		**2**			**2468101214**	**1**	**1**	**400**	**23-03-2005**
1					1357911131	1	2	600	12-05-2009
1		**4**			**1212121212**	**2**	**1**	**500**	**24-07-2016**
2		1			1234567890		1	1000	01-01-2002
2		2			2468101214	1	1	400	23-03-2005
2					**1357911131**	**1**	**2**	**600**	**12-05-2009**
2		4			1212121212	2	1	500	24-07-2016
		1			1234567890		1	1000	01-01-2002
		2			2468101214	1	1	400	23-03-2005
					1357911131	1	2	600	12-05-2009
		4			1212121212	2	1	500	24-07-2016

SELECT .

Departments.Name	Employees.FName

. .

Employees Employee . .

```
SELECT
    e.FName AS "Employee",
    m.FName AS "Manager"
FROM
    Employees e
JOIN
    Employees m
    ON e.ManagerId = m.Id
```

.

?

.

	FName	LName		ID	DepartmentId		HireDate
1			1234567890		1	1000	01-01-2002
2			2468101214	1	1	400	23-03-2005
			1357911131	1	2	600	12-05-2009
4			1212121212	2	1	500	24-07-2016

FROM . Employees ( ).

e.Id	e.FName	e.ManagerId	m.Id	m.FName	m.ManagerId
1			1		
1			2		1
1					1
1			4		2
2		1	1		
2		1	2		1
2		1			1
2		1	4		2
		1	1		
		1	2		1
		1			1
		1	4		2
4		2	1		
4		2	2		1
4		2			1
4		2	4		2

JOIN `e` `ManagerId` `m` `Id` :

e.Id	e.FName	e.ManagerId	m.Id	m.FName	m.ManagerId
2		1	1		
		1	1		
4		2	2		1

SELECT .

e.FName	m.FName

`e.FName` `m.FName` AS .

. . `TABLEA` 20 `TABLEB` 20 `20*20 = 400` .

```
SELECT d.Name, e.FName
FROM   Departments d
CROSS JOIN Employees e;
```

:

d.	e.FName

(Cartesian Join) CROSS JOIN .

u / / . , , . . . Buy Orders PurchaseOrderLineItems .

```
SELECT po.Id, po.PODate, po.VendorName, po.Status, item.ItemNo,
  item.Description, item.Cost, item.Price
FROM PurchaseOrders po
LEFT JOIN
     (
       SELECT l.PurchaseOrderId, l.ItemNo, l.Description, l.Cost, l.Price, Min(l.id) as Id
       FROM PurchaseOrderLineItems l
       GROUP BY l.PurchaseOrderId, l.ItemNo, l.Description, l.Cost, l.Price
     ) AS item ON item.PurchaseOrderId = po.Id
```

& (CROSS APPLY & LATERAL JOIN)

JOIN LATERAL JOIN (PostgreSQL 9.3).
SQL-Server & Oracle CROSS APPLY / OUTER APPLY .

( ) .

.
"" .

:

PostgreSQL 9.3

| | JOIN **LATERAL**

SQL :

|

`INNER JOIN LATERAL` `CROSS APPLY` .

`LEFT JOIN LATERAL` `OUTER APPLY` .

(PostgreSQL 9.3) :

```
SELECT * FROM T_Contacts

--LEFT JOIN T_MAP_Contacts_Ref_OrganisationalUnit ON MAP_CTCOU_CT_UID = T_Contacts.CT_UID AND
MAP_CTCOU_SoftDeleteStatus = 1
--WHERE T_MAP_Contacts_Ref_OrganisationalUnit.MAP_CTCOU_UID IS NULL -- 989


LEFT JOIN LATERAL
(
    SELECT
         --MAP_CTCOU_UID
         MAP_CTCOU_CT_UID
        ,MAP_CTCOU_COU_UID
        ,MAP_CTCOU_DateFrom
        ,MAP_CTCOU_DateTo
   FROM T_MAP_Contacts_Ref_OrganisationalUnit
   WHERE MAP_CTCOU_SoftDeleteStatus = 1
   AND MAP_CTCOU_CT_UID = T_Contacts.CT_UID

    /*
    AND
    (
        (__in_DateFrom <= T_MAP_Contacts_Ref_OrganisationalUnit.MAP_KTKOE_DateTo)
        AND
        (__in_DateTo >= T_MAP_Contacts_Ref_OrganisationalUnit.MAP_KTKOE_DateFrom)
    )
    */
   ORDER BY MAP_CTCOU_DateFrom
   LIMIT 1
) AS FirstOE
```

SQL Server

```
SELECT * FROM T_Contacts

--LEFT JOIN T_MAP_Contacts_Ref_OrganisationalUnit ON MAP_CTCOU_CT_UID = T_Contacts.CT_UID AND
MAP_CTCOU_SoftDeleteStatus = 1
--WHERE T_MAP_Contacts_Ref_OrganisationalUnit.MAP_CTCOU_UID IS NULL -- 989

-- CROSS APPLY -- = INNER JOIN
OUTER APPLY    -- = LEFT JOIN
(
    SELECT TOP 1
         --MAP_CTCOU_UID
         MAP_CTCOU_CT_UID
        ,MAP_CTCOU_COU_UID
        ,MAP_CTCOU_DateFrom
        ,MAP_CTCOU_DateTo
   FROM T_MAP_Contacts_Ref_OrganisationalUnit
   WHERE MAP_CTCOU_SoftDeleteStatus = 1
   AND MAP_CTCOU_CT_UID = T_Contacts.CT_UID

    /*
    AND
    (
        (@in_DateFrom <= T_MAP_Contacts_Ref_OrganisationalUnit.MAP_KTKOE_DateTo)
        AND
        (@in_DateTo >= T_MAP_Contacts_Ref_OrganisationalUnit.MAP_KTKOE_DateFrom)
    )
    */
   ORDER BY MAP_CTCOU_DateFrom
) AS FirstOE
```

JOIN   FULL JOIN.
(: FULL JOIN MySQL 2016   )

FULL OUTER JOIN   .

.

1 :

```
SELECT * FROM Table1

FULL JOIN Table2
     ON 1 = 2
```

2 :

```
SELECT
     COALESCE(T_Budget.Year, tYear.Year) AS RPT_BudgetInYear
    ,COALESCE(T_Budget.Value, 0.0) AS RPT_Value
FROM T_Budget

FULL JOIN tfu_RPT_All_CreateYearInterval(@budget_year_from, @budget_year_to) AS tYear
      ON tYear.Year = T_Budget.Year
```

WHERE (FULL JOIN UNION).
. AP_SoftDeleteStatus = 1 join .

FULL JOIN WHERE NULL . NULL INNER . FULL JOIN .

:

```
SELECT
     T_AccountPlan.AP_UID
    ,T_AccountPlan.AP_Code
    ,T_AccountPlan.AP_Lang_EN
    ,T_BudgetPositions.BUP_Budget
    ,T_BudgetPositions.BUP_UID
    ,T_BudgetPositions.BUP_Jahr
FROM T_BudgetPositions

FULL JOIN T_AccountPlan
    ON T_AccountPlan.AP_UID = T_BudgetPositions.BUP_AP_UID
    AND T_AccountPlan.AP_SoftDeleteStatus = 1

WHERE (1=1)
AND (T_BudgetPositions.BUP_SoftDeleteStatus = 1 OR T_BudgetPositions.BUP_SoftDeleteStatus IS
NULL)
AND (T_AccountPlan.AP_SoftDeleteStatus = 1 OR T_AccountPlan.AP_SoftDeleteStatus IS NULL)
```

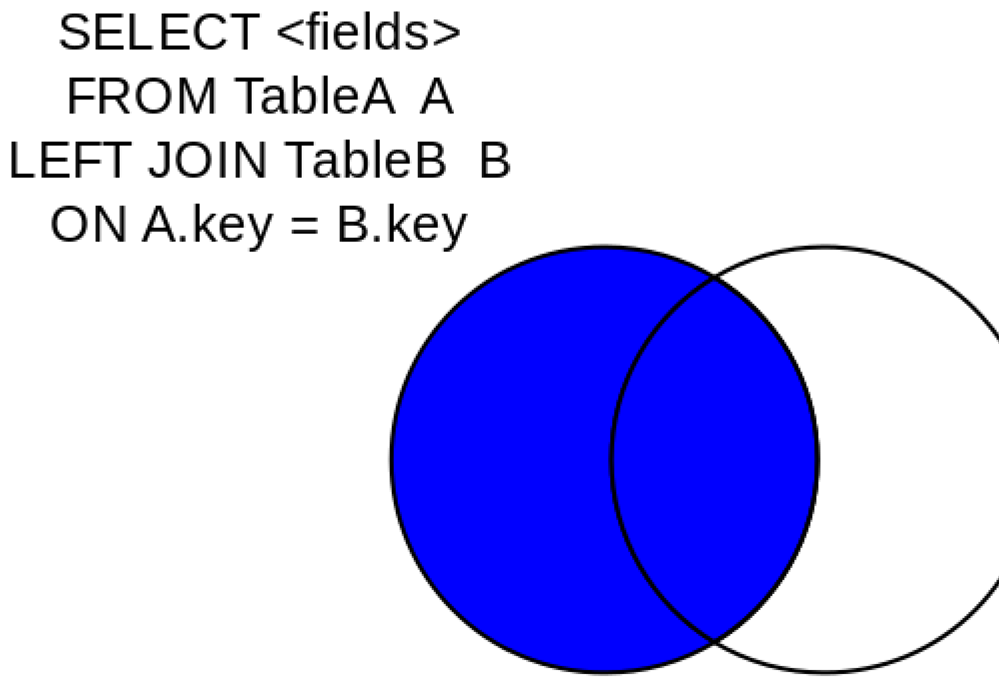
JOIN

- . SQL . .

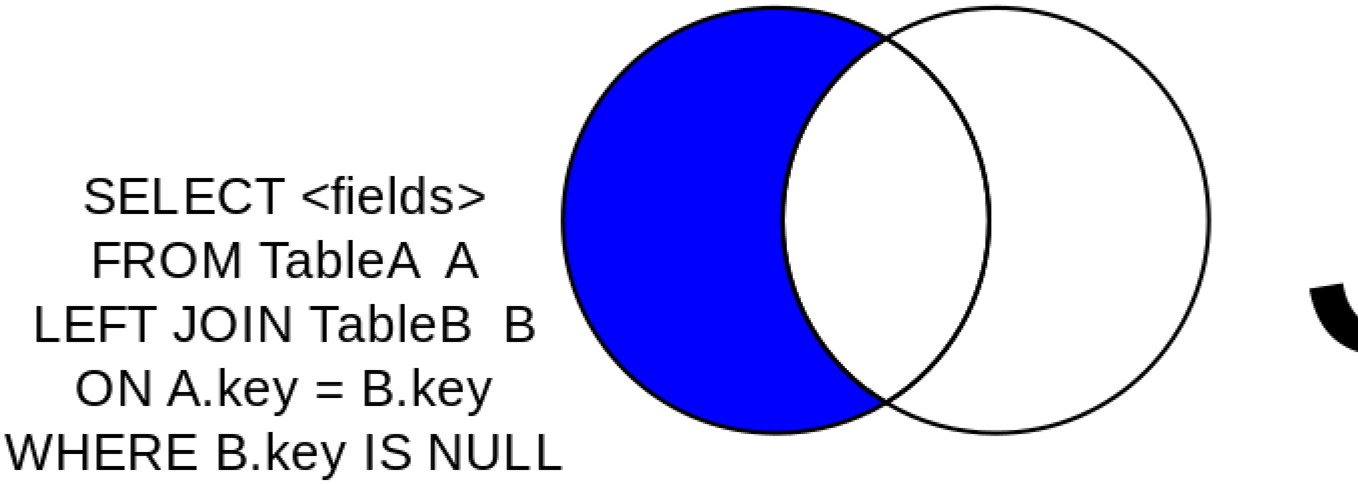
```
WITH RECURSIVE MyDescendants AS (
    SELECT Name
    FROM People
    WHERE Name = 'John Doe'

    UNION ALL

    SELECT People.Name
    FROM People
    JOIN MyDescendants ON People.Name = MyDescendants.Parent
)
SELECT * FROM MyDescendants;
```

/

SQL (`INNER JOIN` , `LEFT OUTER JOIN` , `RIGHT OUTER JOIN` `FULL OUTER JOIN`) (`INNER` `OUTER`).
. .



```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```



```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```

42:

, . .

Examples

```
BEGIN TRANSACTION
    INSERT INTO DeletedEmployees(EmployeeID, DateDeleted, User)
        (SELECT 123, GetDate(), CURRENT_USER);
    DELETE FROM Employees WHERE EmployeeID = 123;
COMMIT TRANSACTION
```

.

```
BEGIN TRY
    BEGIN TRANSACTION
        INSERT INTO Users(ID, Name, Age)
        VALUES(1, 'Bob', 24)

        DELETE FROM Users WHERE Name = 'Todd'
   COMMIT TRANSACTION
END TRY
BEGIN CATCH
   ROLLBACK TRANSACTION
END CATCH
```

: https://riptutorial.com/ko/sql/topic/2424/

43:

SQL NULL "" . SQL ' ' .

'' 0 NULL . NULL .

'NULL' NULL . 'NULL' NULL .

Examples

NULL

WHERE NULL (,) .

```
SELECT * FROM Employees WHERE ManagerId IS NULL ;
SELECT * FROM Employees WHERE ManagerId IS NOT NULL ;
```

NULL = NULL <> NULL (!= NULL) UNKNOWN WHERE .

WHERE FALSE UKNOWN TRUE TRUE .

Nullable

nullable nullable .

```
CREATE TABLE MyTable
(
    MyCol1 INT NOT NULL, -- non-nullable
    MyCol2 INT NULL      -- nullable
) ;
```

NOT NULL NOT NULL ( ) Null.

Null NULL .

```
INSERT INTO MyTable (MyCol1, MyCol2) VALUES (1, NULL) ;  -- works fine

INSERT INTO MyTable (MyCol1, MyCol2) VALUES (NULL, 2) ;
        -- cannot insert
        -- the value NULL into column 'MyCol1', table 'MyTable';
        -- column does not allow nulls. INSERT fails.
```

NULL

NULL .

```
UPDATE Employees
SET ManagerId = NULL
```

```
WHERE Id = 4
```

NULL

Employees :

```
INSERT INTO Employees
    (Id, FName, LName, PhoneNumber, ManagerId, DepartmentId, Salary, HireDate)
VALUES
    (5, 'Jane', 'Doe', NULL, NULL, 2, 800, '2016-07-22') ;
```

: https://riptutorial.com/ko/sql/topic/3421/

44:

Examples

, ,   .   .

: SQL

- 0   .
- 0 1 .
- 0   .

1	
2	

SQL   .

```
CREATE TABLE Departments (
    Id INT NOT NULL AUTO_INCREMENT,
    Name VARCHAR(25) NOT NULL,
    PRIMARY KEY(Id)
);

INSERT INTO Departments
    ([Id], [Name])
VALUES
    (1, 'HR'),
    (2, 'Sales'),
    (3, 'Tech')
;
```

	FName	LName		ID	DepartmentId		HireDate
1			1234567890		1	1000	01-01-2002
2			2468101214	1	1	400	23-03-2005
			1357911131	1	2	600	12-05-2009
4			1212121212	2	1	500	24-07-2016

SQL   .

```
CREATE TABLE Employees (
    Id INT NOT NULL AUTO_INCREMENT,
    FName VARCHAR(35) NOT NULL,
    LName VARCHAR(35) NOT NULL,
    PhoneNumber VARCHAR(11),
    ManagerId INT,
    DepartmentId INT NOT NULL,
    Salary INT NOT NULL,
    HireDate DATETIME NOT NULL,
    PRIMARY KEY(Id),
    FOREIGN KEY (ManagerId) REFERENCES Employees(Id),
    FOREIGN KEY (DepartmentId) REFERENCES Departments(Id)
);

INSERT INTO Employees
    ([Id], [FName], [LName], [PhoneNumber], [ManagerId], [DepartmentId], [Salary], [HireDate])
VALUES
    (1, 'James', 'Smith', 1234567890, NULL, 1, 1000, '01-01-2002'),
    (2, 'John', 'Johnson', 2468101214, '1', 1, 400, '23-03-2005'),
    (3, 'Michael', 'Williams', 1357911131, '1', 2, 600, '12-05-2009'),
    (4, 'Johnathon', 'Smith', 1212121212, '2', 1, 500, '24-07-2016')
;
```

	FName	LName			PreferredContact
1			william.jones@example.com	3347927472	
2			dmiller@example.net	2137921892	
			richard0123@example.com		

SQL .

```
CREATE TABLE Customers (
    Id INT NOT NULL AUTO_INCREMENT,
    FName VARCHAR(35) NOT NULL,
    LName VARCHAR(35) NOT NULL,
    Email varchar(100) NOT NULL,
    PhoneNumber VARCHAR(11),
    PreferredContact VARCHAR(5) NOT NULL,
    PRIMARY KEY(Id)
);

INSERT INTO Customers
    ([Id], [FName], [LName], [Email], [PhoneNumber], [PreferredContact])
VALUES
    (1, 'William', 'Jones', 'william.jones@example.com', '3347927472', 'PHONE'),
    (2, 'David', 'Miller', 'dmiller@example.net', '2137921892', 'EMAIL'),
    (3, 'Richard', 'Davis', 'richard0123@example.com', NULL, 'EMAIL')
;
```

	ID	EmployeeId			
1	1	2	F-150		230

	ID	EmployeeId			
2	1	2	F-150		200
	2	1			100
4			Toyota Prius		1254

SQL .

```
CREATE TABLE Cars (
    Id INT NOT NULL AUTO_INCREMENT,
    CustomerId INT NOT NULL,
    EmployeeId INT NOT NULL,
    Model varchar(50) NOT NULL,
    Status varchar(25) NOT NULL,
    TotalCost INT NOT NULL,
    PRIMARY KEY(Id),
    FOREIGN KEY (CustomerId) REFERENCES Customers(Id),
    FOREIGN KEY (EmployeeId) REFERENCES Employees(Id)
);

INSERT INTO Cars
    ([Id], [CustomerId], [EmployeeId], [Model], [Status], [TotalCost])
VALUES
    ('1', '1', '2', 'Ford F-150', 'READY', '230'),
    ('2', '1', '2', 'Ford F-150', 'READY', '200'),
    ('3', '2', '1', 'Ford Mustang', 'WAITING', '100'),
    ('4', '3', '3', 'Toyota Prius', 'WORKING', '1254')
;
```

Authors , *Books* *BooksAuthors* .

: SQL

. *Books* *Authors* *BooksAuthors* .

- .
- 1 .

( )

1	JD	
2	.	
4		
5	Jason N. Gaylord	

6		
7		
8	Wenz	

SQL :

```
CREATE TABLE Authors (
    Id INT NOT NULL AUTO_INCREMENT,
    Name VARCHAR(70) NOT NULL,
    Country VARCHAR(100) NOT NULL,
    PRIMARY KEY(Id)
);

INSERT INTO Authors
    (Name, Country)
VALUES
    ('J.D. Salinger', 'USA'),
    ('F. Scott. Fitzgerald', 'USA'),
    ('Jane Austen', 'UK'),
    ('Scott Hanselman', 'USA'),
    ('Jason N. Gaylord', 'USA'),
    ('Pranav Rastogi', 'India'),
    ('Todd Miranda', 'USA'),
    ('Christian Wenz', 'USA')
;
```

( )

1	
2	
4	
5	
6	
7	Professional ASP.NET 4.5 in C # VB

SQL :

```
CREATE TABLE Books (
    Id INT NOT NULL AUTO_INCREMENT,
    Title VARCHAR(50) NOT NULL,
    PRIMARY KEY(Id)
);
```

```
INSERT INTO Books
    (Id, Title)
VALUES
    (1, 'The Catcher in the Rye'),
    (2, 'Nine Stories'),
    (3, 'Franny and Zooey'),
    (4, 'The Great Gatsby'),
    (5, 'Tender id the Night'),
    (6, 'Pride and Prejudice'),
    (7, 'Professional ASP.NET 4.5 in C# and VB')
;
```

BooksAuthors

( )

1	1
2	1
	1
4	2
5	2
6	
7	4
7	5
7	6
7	7
7	8

SQL :

```
CREATE TABLE BooksAuthors (
    AuthorId INT NOT NULL,
    BookId  INT NOT NULL,
    FOREIGN KEY (AuthorId) REFERENCES Authors(Id),
    FOREIGN KEY (BookId) REFERENCES Books(Id)
);

INSERT INTO BooksAuthors
    (BookId, AuthorId)
VALUES
    (1, 1),
    (2, 1),
    (3, 1),
    (4, 2),
    (5, 2),
    (6, 3),
    (7, 4),
    (7, 5),
    (7, 6),
    (7, 7),
    (7, 8)
;
```

( ) :

```
SELECT * FROM Authors;
```

( ) :

```
SELECT * FROM Books;
```

( ) :

```
SELECT
  ba.AuthorId,
  a.Name AuthorName,
  ba.BookId,
  b.Title BookTitle
FROM BooksAuthors ba
  INNER JOIN Authors a ON a.id = ba.authorid
  INNER JOIN Books b ON b.id = ba.bookid
;
```

Countries .   .

: SQL

Bloomberg Reuters API 2 3 . 2 `ISO` 3 `ISO3` .

( )

	ISO	ISO3	ISONumeric				
1		AUS	36			OC	AUD
2	DE	DEU	276				EUR
2			356				INR
			418				LAK
4			840				

	ISO	ISO3	ISONumeric				
5	ZW	ZWE	716			AF	ZWL

SQL :

```
CREATE TABLE Countries (
    Id INT NOT NULL AUTO_INCREMENT,
    ISO VARCHAR(2) NOT NULL,
    ISO3 VARCHAR(3) NOT NULL,
    ISONumeric INT NOT NULL,
    CountryName VARCHAR(64) NOT NULL,
    Capital VARCHAR(64) NOT NULL,
    ContinentCode VARCHAR(2) NOT NULL,
    CurrencyCode VARCHAR(3) NOT NULL,
    PRIMARY KEY(Id)
)
;

INSERT INTO Countries
    (ISO, ISO3, ISONumeric, CountryName, Capital, ContinentCode, CurrencyCode)
VALUES
    ('AU', 'AUS', 36, 'Australia', 'Canberra', 'OC', 'AUD'),
    ('DE', 'DEU', 276, 'Germany', 'Berlin', 'EU', 'EUR'),
    ('IN', 'IND', 356, 'India', 'New Delhi', 'AS', 'INR'),
    ('LA', 'LAO', 418, 'Laos', 'Vientiane', 'AS', 'LAK'),
    ('US', 'USA', 840, 'United States', 'Washington', 'NA', 'USD'),
    ('ZW', 'ZWE', 716, 'Zimbabwe', 'Harare', 'AF', 'ZWL')
;
```

: https://riptutorial.com/ko/sql/topic/280/---

45:

EXCEPT EXCEPT .

Examples

```
--dataset schemas must be identical
SELECT 'Data1' as 'Column' UNION ALL
SELECT 'Data2' as 'Column' UNION ALL
SELECT 'Data3' as 'Column' UNION ALL
SELECT 'Data4' as 'Column' UNION ALL
SELECT 'Data5' as 'Column'
EXCEPT
SELECT 'Data3' as 'Column'
--Returns Data1, Data2, Data4, and Data5
```

: https://riptutorial.com/ko/sql/topic/4082/

46:

Examples

SuperHeros .

ID .

.

```
CREATE TABLE HeroPowers
(
    ID int NOT NULL PRIMARY KEY,
    Name nvarchar(MAX) NOT NULL,
    HeroId int REFERENCES SuperHeros(ID)
)
```

HeroId SuperHeros .

.

. . .

```
CREATE TABLE Department (
    Dept_Code        CHAR (5)     PRIMARY KEY,
    Dept_Name        VARCHAR (20) UNIQUE
);
```

.

```
INSERT INTO Department VALUES ('CS205', 'Computer Science');
```

.

```
CREATE TABLE Programming_Courses (
    Dept_Code       CHAR(5),
    Prg_Code        CHAR(9) PRIMARY KEY,
    Prg_Name        VARCHAR (50) UNIQUE,
    FOREIGN KEY (Dept_Code) References Department(Dept_Code)
);
```

.

Dept_Code Department ., :

```
INSERT INTO Programming_Courses Values ('CS300', 'FDB-DB001', 'Database Systems');
```

CS300 Department . :

```
INSERT INTO Programming_Courses VALUES ('CS205', 'FDB-DB001', 'Database Systems');
INSERT INTO Programming_Courses VALUES ('CS205', 'DB2-DB002', 'Database Systems II');
```

.

- ( ) .
- NULL .
- .
- ( ).

: https://riptutorial.com/ko/sql/topic/1533/-

47:

SQL .   .   .

SQL   Wikipeida   .

Examples

GUI (SQL Server)   SQL   .

```
-- Define a name and parameters
CREATE PROCEDURE Northwind.getEmployee
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS

-- Define the query to be run
SELECT FirstName, LastName, Department
FROM Northwind.vEmployeeDepartment
WHERE FirstName = @FirstName AND LastName = @LastName
AND EndDate IS NULL;
```

:

```
EXECUTE Northwind.getEmployee N'Ackerman', N'Pilar';

-- Or
EXEC Northwind.getEmployee @LastName = N'Ackerman', @FirstName = N'Pilar';
GO

-- Or
EXECUTE Northwind.getEmployee @FirstName = N'Pilar', @LastName = N'Ackerman';
GO
```

: https://riptutorial.com/ko/sql/topic/1701/--

48:

Examples

RDBMS .

.

T-SQL .

```
SELECT *
FROM INFORMATION_SCHEMA.COLUMNS
WHERE COLUMN_NAME LIKE '%Institution%'
```

, .

: https://riptutorial.com/ko/sql/topic/3151/-

49:

Examples

.

```
CREATE VIEW new_employees_details AS
SELECT E.id, Fname, Salary, Hire_date
FROM Employees E
WHERE hire_date > date '2015-01-01';
```

:

```
select * from new_employees_details
```

	FName		Hire_date
4		500	24-07-2016

(, ,) . .

```
Create VIEW dept_income AS
SELECT d.Name as DepartmentName, sum(e.salary) as TotalSalary
FROM Employees e
JOIN Departments d on e.DepartmentId = d.id
GROUP BY d.Name;
```

.

```
SELECT *
FROM dept_income;
```

	TotalSalary
	1900
	600

: https://riptutorial.com/ko/sql/topic/766/

50:

Examples

1		
2		

	ID	
1	2	123.50
2		14.80

```
SELECT * FROM Customer WHERE EXISTS (
    SELECT * FROM Order WHERE Order.CustomerId=Customer.Id
)
```

2		

```
SELECT * FROM Customer WHERE NOT EXISTS (
    SELECT * FROM Order WHERE Order.CustomerId = Customer.Id
)
```

1		

`EXISTS` , `IN` `JOIN` .

- `EXISTS` .
- `IN` .
- `JOIN` .

: https://riptutorial.com/ko/sql/topic/7933/-

51:

Examples

ORDER BY TOP x .

, BY GROUP , TOP , .

Q & A 5 .

ORDER BY

5 . "Id".

```
SELECT TOP 5 DisplayName, Reputation
FROM Users
```

...

DisplayName	
	1
	12567
	11739
	37628
	25784

ORDER BY

```
SELECT TOP 5 DisplayName, Reputation
FROM Users
ORDER BY Reputation desc
```

...

DisplayName	
	865023
	661741
C	**650237**
	625870

DisplayName	
	601636

SQL (: MySQL) `TOP` `SELECT` `LIMIT` . .

```
SELECT DisplayName, Reputation
FROM Users
ORDER BY Reputation DESC
LIMIT 5
```

```
SELECT DisplayName, JoinDate, Reputation
FROM Users
ORDER BY JoinDate, Reputation
```

DisplayName		
	2008-09-15	**1**
	2008-09-16	**25784**
	2008-09-16	**37628**
	2008-10-03	**11739**
	2008-10-03	**12567**

( )

('1') .

Pro : .

: ('ORDER BY 14' 'ORDER BY ').

`Reputation` select `3` .

```
SELECT DisplayName, JoinDate, Reputation
FROM Users
ORDER BY 3
```

DisplayName		
	2008-09-15	**1**
	2008-10-03	**11739**
	2008-10-03	**12567**
	2008-09-16	**25784**

DisplayName		
	2008-09-16	**37628**

.

```
SELECT DisplayName, JoinDate as jd, Reputation as rep
FROM Users
ORDER BY jd, rep
```

select . display name . Jd 1, Jd 2 .

```
SELECT DisplayName, JoinDate as jd, Reputation as rep
FROM Users
ORDER BY 2, 3
```

`Employee` `ORDER BY Department` . `Department` . CASE .


```
SELECT *
FROM Employee
ORDER BY CASE Department
         WHEN 'HR'         THEN 1
         WHEN 'Accountant' THEN 2
         ELSE                   3
         END;
```


: https://riptutorial.com/ko/sql/topic/620/

52:

DELETE .

1. DELETE FROM *TableName* [WHERE] [LIMIT]

Examples

WHERE .

WHERE .

```
DELETE FROM Employees
WHERE FName = 'John'
```

.

WHERE .

```
DELETE FROM Employees
```

TRUNCATE TRUNCATE TRUNCATE .

TRUNCATE

. . .

```
TRUNCATE TABLE Employees
```

.

DELETE .

Target DELETE .

```
DELETE FROM Source
WHERE  EXISTS ( SELECT 1 -- specific value in SELECT doesn't matter
               FROM Target
               Where Source.ID = Target.ID )
```

RDBMS (: MySQL, Oracle, PostgresSQL, Teradata) DELETE .

, Aggregate Target ID . Source .

MySQL, Oracle Teradata .

```
DELETE FROM Source
WHERE  Source.ID = TargetSchema.Target.ID
       AND TargetSchema.Target.Date = AggregateSchema.Aggregate.Date
```

PostgreSQL :

```
DELETE FROM Source
USING  TargetSchema.Target, AggregateSchema.Aggregate
WHERE  Source.ID = TargetSchema.Target.ID
       AND TargetSchema.Target.DataDate = AggregateSchema.Aggregate.AggDate
```

, INNER JOIN . ID Target Target ID Aggregate Source .

(MySQL, Oracle, Teradata) :

```
DELETE Source
FROM   Source, TargetSchema.Target, AggregateSchema.Aggregate
WHERE  Source.ID = TargetSchema.Target.ID
       AND TargetSchema.Target.DataDate = AggregateSchema.Aggregate.AggDate
```

RDBMS (: Oracle, MySQL) `Delete` (: Teradata)

(`NOT EXISTS`)

```
DELETE FROM Source
WHERE NOT EXISTS ( SELECT 1 -- specific value in SELECT doesn't matter
                   FROM Target
                   Where Source.ID = Target.ID )
```

: https://riptutorial.com/ko/sql/topic/1105/

53:

Examples

SQL

```
/*(8)*/  SELECT /*9*/ DISTINCT /*11*/ TOP
/*(1)*/  FROM
/*(3)*/        JOIN
/*(2)*/       ON
/*(4)*/  WHERE
/*(5)*/  GROUP BY
/*(6)*/  WITH {CUBE | ROLLUP}
/*(7)*/  HAVING
/*(10)*/ ORDER BY
/*(11)*/ LIMIT
```

.

VT ' '

1. FROM : FROM ( ) VT1 .
2. ON : ON VT1 . TRUE VT2 .
3. OUTER () : OUTER JOIN (CROSS JOIN INNER JOIN), VT2 , VT3. FROM FROM 1 - 3 .
4. WHERE VT3 . TRUE VT4 .
5. GROUP BY : VT4 GROUP BY . VT5 .
6. | ROLLUP : ( ) VT5 VT6 .
7. HAVING : HAVING VT6 . TRUE VT7 .
8. : SELECT VT8 .
9. DISTINCT : VT8 . VT9 .
10. ORDER BY : VT9 ORDER BY . (VC10).
11. TOP : VC10 . VT11 . LIMIT Postgres Netezza SQL TOP .

: https://riptutorial.com/ko/sql/topic/3671/-

54:

Examples

```
SELECT your_columns, COUNT(*) OVER() as Ttl_Rows FROM your_data_set
```

		Ttl_Rows
1		5
2		5
		5
4		5
5	quux	5

.

.

.

1		unique_tag
2		
42		
51	quux	

.

```
SELECT id, name, tag, COUNT(*) OVER (PARTITION BY tag) > 1 AS flag FROM items
```

.

1		unique_tag	
2			
42			

51	quux		

OVER PARTITION .

```
SELECT id, name, tag, (SELECT COUNT(tag) FROM items B WHERE tag = A.tag) > 1 AS flag FROM
items A
```

:

2016-03-12	200
2016-03-11	-50
2016-03-14	100
2016-03-15	100
2016-03-10	-250

```
SELECT date, amount, SUM(amount) OVER (ORDER BY date ASC) AS running
FROM operations
ORDER BY date ASC
```

.

2016-03-10	-250	-250
2016-03-11	-50	-300
2016-03-12	200	-100
2016-03-14	100	0
2016-03-15	100	-100

N

User_ID	
1	2016-07-20
1	2016-07-21
2	2016-07-20
2	2016-07-21

User_ID	
2	2016-07-22

```
;with CTE as
(SELECT *,
        ROW_NUMBER() OVER (PARTITION BY User_ID
                           ORDER BY Completion_Date DESC) Row_Num
FROM    Data)
SELECT * FORM CTE WHERE Row_Num <= n
```

n = 1 `user_id` .

User_ID		Row_Num
1	2016-07-21	1
2	2016-07-22	1

LAG () "out-of-sequence"

.

		STATUS_TIME	STATUS_BY
1		2016-09-28-19.47.52.501398	USER_1
		2016-09-28-19.47.52.501511	USER_2
1		2016-09-28-19.47.52.501517	USER_3
		2016-09-28-19.47.52.501521	USER_2
		2016-09-28-19.47.52.501524	USER_4

`ID` `STATUS` `STATUS` 'ONE' 'TWO' 'THREE' . (`STATUS_BY`) 'ONE' 'THREE' .

`LAG()` .

```
SELECT * FROM (
 SELECT
  t.*,
  LAG(status) OVER (PARTITION BY id ORDER BY status_time) AS prev_status
  FROM test t
) t1 WHERE status = 'THREE' AND prev_status != 'TWO'
```

LAG () .

```
SELECT A.id, A.status, B.status as prev_status, A.status_time, B.status_time as
prev_status_time
```

```
FROM Data A, Data B
WHERE A.id = B.id
AND   B.status_time = (SELECT MAX(status_time) FROM Data where status_time < A.status_time and
id = A.id)
AND   A.status = 'THREE' AND NOT B.status = 'TWO'
```

: https://riptutorial.com/ko/sql/topic/647/-

55:

- UPDATE
  SET *column_name* = *value* , *column_name2* = *value_2* , ..., *column_name_n* = *value_n*
  WHERE (condition_n)

Examples

Cars Table .

```
UPDATE Cars
SET Status = 'READY'
```

`WHERE` 'Cars' 'status' 'READY' .

Cars Table .

```
UPDATE
    Cars
SET
    Status = 'READY'
WHERE
    Id = 4
```

ID 4 'Cars' 'READY' .

`WHERE` . . .

Cars Table .

```
UPDATE Cars
SET TotalCost = TotalCost + 100
WHERE Id = 3 or Id = 4
```

. `TotalCost` 100 .

- 3 100 200 .
- # 4 1254 1354 .

.

`Customer` `Employees` `PhoneNumber` .

(Employees Customers .)

SQL

:

```
UPDATE
    Employees
SET PhoneNumber =
    (SELECT
         c.PhoneNumber
     FROM
         Customers c
     WHERE
         c.FName = Employees.FName
         AND c.LName = Employees.LName)
WHERE Employees.PhoneNumber IS NULL
```

SQL : 2003

MERGE :

```
MERGE INTO
    Employees e
USING
    Customers c
ON
    e.FName = c.Fname
    AND e.LName = c.LName
    AND e.PhoneNumber IS NULL
WHEN MATCHED THEN
   UPDATE
      SET PhoneNumber = c.PhoneNumber
```

SQL

INNER JOIN :

```
UPDATE
    Employees
SET
    PhoneNumber = c.PhoneNumber
FROM
    Employees e
INNER JOIN Customers c
        ON e.FName = c.FName
        AND e.LName = c.LName
WHERE
    PhoneNumber IS NULL
```

.

```
CREATE TABLE #TempUpdated(ID INT)

Update TableName SET Col1 = 42
    OUTPUT inserted.ID INTO #TempUpdated
    WHERE Id > 50
```

: https://riptutorial.com/ko/sql/topic/321/-

56:

CASE if-then   .

- _
  1 1
  [2 2] ...
  [ELSE resultX]
- 1 1
  [2 2] ...
  [ELSE resultX]

CASE `compareX` `input_expression`   .

CASE `conditionX` true   .

Examples

SELECT CASE (   ).

CASE   TRUE   .

(.)

```
SELECT Id, ItemId, Price,
  CASE WHEN Price < 10 THEN 'CHEAP'
       WHEN Price < 20 THEN 'AFFORDABLE'
       ELSE 'EXPENSIVE'
  END AS PriceRating
FROM ItemSales
```

	ItemId		PriceRating
1	100	34.5	
2	145	2.3	
	100	34.5	
4	100	34.5	
5	145	10	

CASE .

`CASE` `SUM`   . Excel `COUNTIF` .

"1" .

`ItemSales` "" .

	ItemId		PriceRating
1	100	34.5	
2	145	2.3	
	100	34.5	
4	100	34.5	
5	145	10	

```
SELECT
    COUNT(Id) AS ItemsCount,
    SUM ( CASE
            WHEN PriceRating = 'Expensive' THEN 1
            ELSE 0
          END
        ) AS ExpensiveItemsCount
FROM ItemSales
```

:

ItemsCount	ExpensiveItemsCount
5	

:

```
SELECT
    COUNT(Id) as ItemsCount,
    SUM (
        CASE PriceRating
            WHEN 'Expensive' THEN 1
            ELSE 0
        END
       ) AS ExpensiveItemsCount
FROM ItemSales
```

CASE

`CASE` ( ) . . `ELSE` .

```
SELECT Id, ItemId, Price,
  CASE Price WHEN 5  THEN 'CHEAP'
             WHEN 15 THEN 'AFFORDABLE'
             ELSE         'EXPENSIVE'
  END as PriceRating
FROM ItemSales
```

.   WHEN   . :

```
SELECT
    CASE ABS(CHECKSUM(NEWID())) % 4
        WHEN 0 THEN 'Dr'
        WHEN 1 THEN 'Master'
        WHEN 2 THEN 'Mr'
        WHEN 3 THEN 'Mrs'
    END
```

NULL   . WHEN NEWID()   . .

```
SELECT
    CASE
        WHEN ABS(CHECKSUM(NEWID())) % 4 = 0 THEN 'Dr'
        WHEN ABS(CHECKSUM(NEWID())) % 4 = 1 THEN 'Master'
        WHEN ABS(CHECKSUM(NEWID())) % 4 = 2 THEN 'Mr'
        WHEN ABS(CHECKSUM(NEWID())) % 4 = 3 THEN 'Mrs'
    END
```

WHEN   NULL .

ORDER BY CASE

1,2,3 ..   .

```
SELECT * FROM DEPT
ORDER BY
CASE DEPARTMENT
      WHEN 'MARKETING' THEN  1
      WHEN 'SALES' THEN 2
      WHEN 'RESEARCH' THEN 3
      WHEN 'INNOVATION' THEN 4
      ELSE        5
      END,
      CITY
```

				EMPLOYEES_NUMBER
12				9
15				12
9				8
14				12
5				11
10				13

				EMPLOYEES_NUMBER
4				11
2				9

UPDATE CASE

:

```
UPDATE ItemPrice
SET Price = Price *
  CASE ItemId
    WHEN 1 THEN 1.05
    WHEN 2 THEN 1.10
    WHEN 3 THEN 1.15
    ELSE 1.00
  END
```

NULL   CASE

'0' , NULL   '1' .

```
SELECT ID
      ,REGION
      ,CITY
      ,DEPARTMENT
      ,EMPLOYEES_NUMBER
  FROM DEPT
  ORDER BY
  CASE WHEN REGION IS NULL THEN 1
  ELSE 0
  END,
  REGION
```

				EMPLOYEES_NUMBER
10				13
14				12
9				8
12				9
5				11
15				12
4				11
2				9

ORDER BY CASE 2

. (... `ORDER BY MIN(Date1, Date2)`) `MIN()` `LEAST()` SQL `CASE` .

`CASE` `Date1` `Date2` .

	Date1	Date2
1	2017-01-01	2017-01-31
2	2017-01-31	2017-01-03
	2017-01-31	2017-01-02
4	2017-01-06	2017-01-31
5	2017-01-31	2017-01-05
6	2017-01-04	2017-01-31

```
SELECT Id, Date1, Date2
FROM YourTable
ORDER BY CASE
           WHEN COALESCE(Date1, '1753-01-01') < COALESCE(Date2, '1753-01-01') THEN Date1
           ELSE Date2
         END
```

	Date1	Date2
1	**2017-01-01**	2017-01-31
	2017-01-31	**2017-01-02**
2	2017-01-31	**2017-01-03**
6	**2017-01-04**	2017-01-31
5	2017-01-31	**2017-01-05**
4	**2017-01-06**	2017-01-31

`Id = 1` , `Date1` `2017-01-01` `Id = 3` . `Date2` `2017-01-02` ..

`2017-01-01` `2017-01-06` `Date1` `Date2` .

: https://riptutorial.com/ko/sql/topic/456/

57:

Examples

`--` .

```
SELECT *
FROM Employees -- this is a comment
WHERE FName = 'John'
```

`/* ... */` .

```
/* This query
   returns all employees */
SELECT *
FROM Employees
```

:

```
SELECT /* all columns: */ *
FROM Employees
```

: https://riptutorial.com/ko/sql/topic/1597/

58:

(1999) : 2 , , .

Examples

.

SQL .
.

. . ".

.

1. . .
2. .
3. .
4. .
5. .

5 :

		(DOB)	
1		1971/11/02	
2		1971/11/02	
		1975 8 7	2

- 1 : . `Id` , `Name` , `DOB` `Manager` .
- 2 : `Id` , `Name` (4), `DOB` `Manager` (`Manager`).).
- 3 : `Id` , `Name` , `DOB` `Manager` .
- 4 : `Id` .

:

		(DOB)	
1		1971/11/02	
1		1971/11/02	
		1975 7 18	2, 1

-

1 :   2 1   .

- 2 : DOB   .
- 3 : "   .
- 4 :   .
- 5 :   .

: https://riptutorial.com/ko/sql/topic/2515/-

59:

CREATE TABLE . C , W + .

- tableName ([ColumnName1] [datatype1] [, [ColumnName2] [datatype2] ...])

tableName	
	" . .

.

Examples

ID, `Employees` .

```
CREATE TABLE Employees(
    Id int identity(1,1) primary key not null,
    FName varchar(20) not null,
    LName varchar(20) not null,
    PhoneNumber varchar(10) not null
);
```

Transact-SQL .

`CREATE TABLE` `Employees`

ID

```
Id int identity(1,1) not null
```

`Id`	
`int`	.
`identity(1,1)`	1 1 .
`primary key`	.
`not null`	null .

.

```
CREATE TABLE ClonedEmployees AS SELECT * FROM Employees;
```

SELECT.   .   .

```
CREATE TABLE ModifiedEmployees AS
SELECT Id, CONCAT(FName," ",LName) AS FullName FROM Employees
WHERE Id > 10;
```

.

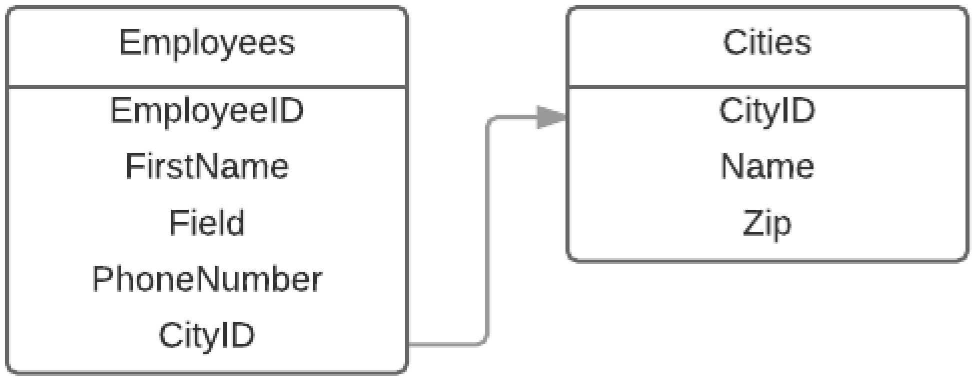
```
CREATE TABLE newtable LIKE oldtable;
INSERT newtable SELECT * FROM oldtable;
```

FOREIGN KEY CREATE TABLE

`Cities` `Employees` .

```
CREATE TABLE Cities(
    CityID INT IDENTITY(1,1) NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Zip VARCHAR(10) NOT NULL
);

CREATE TABLE Employees(
    EmployeeID INT IDENTITY (1,1) NOT NULL,
    FirstName VARCHAR(20) NOT NULL,
    LastName VARCHAR(20) NOT NULL,
    PhoneNumber VARCHAR(10) NOT NULL,
    CityID INT FOREIGN KEY REFERENCES Cities(CityID)
);
```

.



`CityID` `Employees` `CityID` `Cities` .   .

```
CityID INT FOREIGN KEY REFERENCES Cities(CityID)
```

`CityID`	
`int`	

`FOREIGN KEY`	*( ).*
`REFERENCES Cities(CityID)`	. `Cities` `CityID`

gW : . `Cities` `Employees` . .

PostgreSQL SQLite

.

```
CREATE TEMP TABLE MyTable(...);
```

SQL

.

```
CREATE TABLE #TempPhysical(...);
```

.

```
CREATE TABLE ##TempPhysicalVisibleToEveryone(...);
```

:

```
DECLARE @TempMemory TABLE(...);
```

: https://riptutorial.com/ko/sql/topic/348/-

60:

.

() . . FROM .

Examples

WHERE

. .

```
SELECT *
FROM Employees
WHERE Salary = (SELECT MAX(Salary) FROM Employees)
```

FROM

FROM .

```
SELECT Managers.Id, Employees.Salary
FROM (
  SELECT Id
  FROM Employees
  WHERE ManagerId IS NULL
) AS Managers
JOIN Employees ON Managers.Id = Employees.Id
```

SELECT

```
SELECT
  Id,
  FName,
  LName,
  (SELECT COUNT(*) FROM Cars WHERE Cars.CustomerId = Customers.Id) AS NumberOfCars
FROM Customers
```

FROM

"" FROM .

```
SELECT * FROM (SELECT city, temp_hi - temp_lo AS temp_var FROM weather) AS w
WHERE temp_var > 20;
```

20 . .

	temp_var
	21
	31
	23
	31
	27
	28
	28
	32

.

WHERE

(qquery) ( ) .

```
SELECT name, pop2000 FROM cities
WHERE pop2000 < (SELECT avg(pop2000)  FROM cities);
```

: (SELECT avg (pop2000) FROM cities) WHERE . .

	pop2000
	776733
	348189
	146866

SELECT

`SELECT` . cities .

```
SELECT w.*,  (SELECT c.state FROM cities AS c WHERE c.name = w.city ) AS state
FROM weather AS w;
```

.

```
SELECT *
FROM Employees
WHERE EmployeeID not in (SELECT EmployeeID
                                    FROM Supervisors)
```

LEFT JOIN .

```
SELECT *
FROM Employees AS e
LEFT JOIN Supervisors AS s ON s.EmployeeID=e.EmployeeID
WHERE s.EmployeeID is NULL
```

(Synchronized Coordinated) .

```
SELECT EmployeeId
    FROM Employee AS eOuter
    WHERE Salary > (
       SELECT AVG(Salary)
       FROM Employee eInner
       WHERE eInner.DepartmentId = eOuter.DepartmentId
    )
```

`SELECT AVG(Salary) ...` `Employee eOuter` .

: https://riptutorial.com/ko/sql/topic/1606/-

61: ()

. ,   ,   .

1. FIRST_VALUE (scalar_expression) OVER ([partition_by_clause] order_by_clause [_])
2. LAST_VALUE (scalar_expression) OVER ([partition_by_clause] order_by_clause [_])
3. LAG (scalar_expression [, offset] [, default]) OVER ([partition_by_clause] order_by_clause)
4. LEAD (scalar_expression [, offset], [default]) OVER ([partition_by_clause] order_by_clause)
5. PERCENT_RANK () OVER ([partition_by_clause] order_by_clause)
6. CUME_DIST () OVER ([partition_by_clause] order_by_clause)
7. PERCENTILE_DISC (numeric_literal) WITHIN GROUP (ORDER BY order_by_expression [ASC | DESC]) OVER ([<partition_by_clause>])
8. PERCENTILE_CONT (numeric_literal) WITHIN GROUP (ORDER BY order_by_expression [ASC | DESC]) OVER ([<partition_by_clause>])

Examples

FIRST_VALUE

FIRST_VALUE   .

```
SELECT StateProvinceID, Name, TaxRate,
       FIRST_VALUE(StateProvinceID)
        OVER(ORDER BY TaxRate ASC) AS FirstValue
FROM SalesTaxRate;
```

FIRST_VALUE   ID   . OVER   .

StateProvinceID			FirstValue
74		5.00	74
36		6.75	74
30		7.00	74
1	GST	7.00	74
57	GST	7.00	74
63	GST	7.00	74

LAST_VALUE

LAST_VALUE   .

```
SELECT TerritoryID, StartDate, BusinessentityID,
       LAST_VALUE(BusinessentityID)
        OVER(ORDER BY TerritoryID) AS LastValue
FROM SalesTerritoryHistory;
```

`LAST_VALUE` .

ID		ID	LastValue
1	2005-07-01 00.00.00.000	280	283
1	2006-11-01 00.00.00.000	284	283
1	2005-07-01 00.00.00.000	283	283
2	2007-01-01 00.00.00.000	277	275
2	2005-07-01 00.00.00.000	275	275
	2007-01-01 00.00.00.000	275	277

LAG LEAD

`LAG` . `SELECT` .

. offset . .

default offset `NULL` . `NULL` .

`LEAD` . `SELECT` .

. offset .

offset `NULL` . `NULL` .

```
SELECT BusinessEntityID, SalesYTD,
       LEAD(SalesYTD, 1, 0) OVER(ORDER BY BusinessEntityID) AS "Lead value",
       LAG(SalesYTD, 1, 0) OVER(ORDER BY BusinessEntityID) AS "Lag value"
FROM SalesPerson;
```

LEAD LAG BusinessEntityID .

BusinessEntityID	SalesYTD		
274	559697.5639	3763178.1787	0.0000
275	3763178.1787	4251368.5497	559697.5639
276	4251368.5497	3189418.3662	3763178.1787

BusinessEntityID	SalesYTD		
277	3189418.3662	1453719.4653	4251368.5497
278	1453719.4653	2315185.6110	3189418.3662
279	2315185.6110	1352577.1325	1453719.4653

PERCENT_RANK CUME_DIST

`PERCENT_RANK` . .

0. .

`CUME_DIST` . .

```
SELECT BusinessEntityID, JobTitle, SickLeaveHours,
PERCENT_RANK() OVER(PARTITION BY JobTitle ORDER BY SickLeaveHours DESC)
       AS "Percent Rank",
CUME_DIST() OVER(PARTITION BY JobTitle ORDER BY SickLeaveHours DESC)
       AS "Cumulative Distribution"
FROM Employee;
```

`ORDER` `SELECT` .

BusinessEntityID	JobTitle	SickLeaveHours		
267		57	0	0.25
268		56	0.333333333333333	0.75
269		56	0.333333333333333	0.75
272		55	1	1
262	Cheif	48	0	1
239		45	0	1
252		50	0	0.111111111111111
251		49	0.125	0.333333333333333
256		49	0.125	0.333333333333333
253		48	0.375	0.555555555555555
254		48	0.375	0.555555555555555

`PERCENT_RANK` . .

`CUME_DIST` .

PERCENTILE_DISC PERCENTILE_CONT

`PERCENTILE_DISC` `numeric_literal` .

`WITHIN GROUP` .

`PERCENTILE_CONT` `PERCENTILE_DISC` , .

```
SELECT BusinessEntityID, JobTitle, SickLeaveHours,
       CUME_DIST() OVER(PARTITION BY JobTitle ORDER BY SickLeaveHours ASC)
       AS "Cumulative Distribution",
       PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY SickLeaveHours)
          OVER(PARTITION BY JobTitle) AS "Percentile Discreet"
FROM Employee;
```

0.5 `PERCENTILE_DISC` . Percentile Discreet .

BusinessEntityID	JobTitle	SickLeaveHours		
272		55	0.25	**56**
268		56	0.75	**56**
269		56	0.75	**56**
267		57	1	**56**

`PERCENTILE_CONT` . "Percentile Continuous" .

```
SELECT BusinessEntityID, JobTitle, SickLeaveHours,
       CUME_DIST() OVER(PARTITION BY JobTitle ORDER BY SickLeaveHours ASC)
       AS "Cumulative Distribution",
       PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY SickLeaveHours)
          OVER(PARTITION BY JobTitle) AS "Percentile Discreet",
       PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY SickLeaveHours)
          OVER(PARTITION BY JobTitle) AS "Percentile Continuous"
FROM Employee;
```

BusinessEntityID	JobTitle	SickLeaveHours			
272		55	0.25	56	**56**
268		56	0.75	56	**56**
269		56	0.75	56	**56**
267		57	1	56	**56**

() : https://riptutorial.com/ko/sql/topic/8811/---

62: (/)

SQL . . . .

T-SQL .

- CAST (expression AS data_type [(length)])
- (data_type [()], [,])
- PARSE (string_value AS data_type [USING culture])
- DATENAME (,)
- GETDATE ()
- DATEDIFF (datepart, startdate, enddate)
- DATEADD (, ,)
- (, val_1, val_2 [, val_n])
- IIF (boolean_expression, true_value, false_value)
- (numeric_expression)
- POWER (float_expression, y)

.

10 .

1. SQL .
2. . .
3. . , . , .

SQL .

4. . .
5. . .
6. .
7. .
8. .

, , .

9. SQL , .
10. SQL (:).

Examples

, , .

`lower(char)` .

```
SELECT customer_id, lower(customer_last_name) FROM customer;
```

"SMITH" "smith" .

SQL . , , smalldatetime, datetime, datetime2 datetimeoffset . .

	hh : mm : ss [.nnnnnnn]
	YYYY-MM-DD
smalldatetime	YYYY-MM-DDh : mm : ss
	YYYY-MM-DD hh : mm : ss [.nnn]
datetime2	YYYY-MM-DD hh : mm : ss [.nnnnnnn]
datetimeoffset	YYYY-MM-DD hh : mm : ss [.nnnnnnn] [+/-] hh : mm

`DATENAME` .

```
SELECT DATENAME (weekday,'2017-01-14') as Datename
```


`GETDATE` SQL . .

```
SELECT GETDATE() as Systemdate
```

2017-01-14 11 : 11 : 47.7230728

`DATEDIFF` .

datepart . datepart , , , ,, , . startdate enddate .

```
SELECT SalesOrderID, DATEDIFF(day, OrderDate, ShipDate)
AS 'Processing time'
FROM Sales.SalesOrderHeader
```

SalesOrderID	
43659	7
43660	7
43661	7
43662	7

DATEADD .

```
SELECT DATEADD (day, 20, '2017-01-14') AS Added20MoreDays
```

Added 20MoreDays
2017-02-03 00 : 00 : 00.000

SQL @@SERVERNAME . SQL .

```
SELECT @@SERVERNAME AS 'Server'
```

SQL064

SQL .

CAST CONVERT .

CAST CONVERT .

CAST CONVERT datetime varchar .

CAST . YYYY-MM-DD .

CONVERT . 3 dd / mm / yy .

```
USE AdventureWorks2012
GO
SELECT FirstName + ' ' + LastName + ' was hired on ' +
       CAST(HireDate AS varchar(20)) AS 'Cast',
       FirstName + ' ' + LastName + ' was hired on ' +
       CONVERT(varchar, HireDate, 3) AS 'Convert'
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
ON p.BusinessEntityID = e.BusinessEntityID
GO
```

David Hamiltion 2003 2 4 .	David Hamiltion 04/02/03 .

PARSE . .

, AS . . .

, . CAST CONVERT .

```
SELECT PARSE('Monday, 13 August 2012' AS datetime2 USING 'en-US') AS 'Date in English'
```

2012 8 13 00 : 00 : 00.0000000

SQL `CHOOSE` `IIF` .

`CHOOSE` . .

index . val_1 ... val_n .

```
SELECT CHOOSE(2, 'Human Resources', 'Sales', 'Admin', 'Marketing' ) AS Result;
```


`CHOOSE` .

`IIF` . true true . .

boolean_expression . true_value boolean_expression true false_value boolean_expression false .

```
SELECT BusinessEntityID, SalesYTD,
       IIF(SalesYTD > 200000, 'Bonus', 'No Bonus') AS 'Bonus?'
FROM Sales.SalesPerson
GO
```

BusinessEntityID	SalesYTD	?
274	559697.5639	
275	3763178.1787	
285	172524.4512	

IIF . 20 . 200,000 .

SQL .

`SIGN` . . -1 +1 0 0 .

```
SELECT SIGN(-20) AS 'Sign'
```

-1

-1.

`POWER` .   .

float_expression   y   .

```
SELECT POWER(50, 3) AS Result
```

125000

(/) : https://riptutorial.com/ko/sql/topic/6898/-------

63: ()

- ([*DISTINCT*] expression) -DISTINCT
- AVG ([ALL | DISTINCT])
- COUNT ({[ALL | DISTINCT]] | *})
- GROUPING (<column_expression>)
- MAX ([ALL | DISTINCT])
- MIN ([ALL | DISTINCT])
- SUM ([ALL | DISTINCT])
- VAR ([ALL | DISTINCT])
  OVER ([partition_by_clause] order_by_clause)
- VARP ([ALL | DISTINCT])
  OVER ([partition_by_clause] order_by_clause
- STDEV ([ALL | DISTINCT])
  OVER ([partition_by_clause] order_by_clause)
- STDEVP ([ALL | DISTINCT])
  OVER ([partition_by_clause] order_by_clause)

, .

```
MIN        returns the smallest value in a given column
MAX        returns the largest value in a given column
SUM        returns the sum of the numeric values in a given column
AVG        returns the average value of a given column
COUNT      returns the total number of values in a given column
COUNT(*)   returns the number of rows in a table
GROUPING   Is a column or an expression that contains a column in a GROUP BY clause.
STDEV      returns the statistical standard deviation of all values in the specified
expression.
STDEVP     returns the statistical standard deviation for the population for all values in the
specified expression.
VAR        returns the statistical variance of all values in the specified expression. may be
followed by the OVER clause.
VARP       returns the statistical variance for the population for all values in the specified
expression.
```

`SELECT` " " . . - SQLCourse2.com

NULL .

Examples

`Sum` . group by .

```
select sum(salary) TotalSalary
from employees;
```

TotalSalary
2500

```
select DepartmentId, sum(salary) TotalSalary
from employees
group by DepartmentId;
```

DepartmentId	TotalSalary
1	2000
2	500

	Payment_type	
		100
		300
		1000
		500

```
select customer,
       sum(case when payment_type = 'credit' then amount else 0 end) as credit,
       sum(case when payment_type = 'debit' then amount else 0 end) as debit
from payments
group by customer
```

:

	400	0
	1000	500

```
select customer,
       sum(case when payment_type = 'credit' then 1 else 0 end) as credit_transaction_count,
       sum(case when payment_type = 'debit' then 1 else 0 end) as debit_transaction_count
from payments
group by customer
```

:

	credit_transaction_count	debit_transaction_count
	2	0
	1	1

AVG ()

AVG () ( ) . . .

	8,550,405	2015
	...	...
	8,000,906	2005

10 , , .

```
select city_name, AVG(population) avg_population
from city_population
where city_name = 'NEW YORK CITY';
```

.

	8,250,754

: AVG () . .

SO .

List Concatenation . ( ) . SQL .

MySQL

```
SELECT ColumnA
     , GROUP_CONCAT(ColumnB ORDER BY ColumnB SEPARATOR ',') AS ColumnBs
  FROM TableName
 GROUP BY ColumnA
 ORDER BY ColumnA;
```

DB2

```
SELECT ColumnA
     , LISTAGG(ColumnB, ',') WITHIN GROUP (ORDER BY ColumnB) AS ColumnBs
  FROM TableName
 GROUP BY ColumnA
 ORDER BY ColumnA;
```

PostgreSQL

```
SELECT ColumnA
     , STRING_AGG(ColumnB, ',' ORDER BY ColumnB) AS ColumnBs
  FROM TableName
 GROUP BY ColumnA
 ORDER BY ColumnA;
```

SQL

SQL Server 2016

(CTE DRY   )

```
  WITH CTE_TableName AS (
       SELECT ColumnA, ColumnB
         FROM TableName)
SELECT t0.ColumnA
     , STUFF((
       SELECT ',' + t1.ColumnB
         FROM CTE_TableName t1
        WHERE t1.ColumnA = t0.ColumnA
        ORDER BY t1.ColumnB
          FOR XML PATH('')), 1, 1, '') AS ColumnBs
  FROM CTE_TableName t0
 GROUP BY t0.ColumnA
 ORDER BY ColumnA;
```

SQL Server 2017 SQL Azure

```
SELECT ColumnA
     , STRING_AGG(ColumnB, ',') WITHIN GROUP (ORDER BY ColumnB) AS ColumnBs
  FROM TableName
 GROUP BY ColumnA
 ORDER BY ColumnA;
```

SQLite

:

```
SELECT ColumnA
     , GROUP_CONCAT(ColumnB, ',') AS ColumnBs
  FROM TableName
 GROUP BY ColumnA
 ORDER BY ColumnA;
```

CTE .

```
  WITH CTE_TableName AS (
       SELECT ColumnA, ColumnB
         FROM TableName
        ORDER BY ColumnA, ColumnB)
SELECT ColumnA
     , GROUP_CONCAT(ColumnB, ',') AS ColumnBs
  FROM CTE_TableName
 GROUP BY ColumnA
 ORDER BY ColumnA;
```

.

```
SELECT count(*) TotalRows
FROM employees;
```

TotalRows
4

.

```
SELECT DepartmentId, count(*) NumEmployees
FROM employees
GROUP BY DepartmentId;
```

DepartmentId	
1	
2	1

`NULL` / .

```
SELECT count(ManagerId) mgr
FROM EMPLOYEES;
```

mgr

(null managerID .)

COUNT DISTINCT DISTINCT .

:

```
 SELECT COUNT(ContinentCode) AllCount
 ,      COUNT(DISTINCT ContinentCode) SingleCount
 FROM Countries;
```

. *SingleCount* *AllCount* .

OC
AF
AF

AllCount : 7 SingleCount : 5

.

```
select max(age) from employee;
```

`employee` `age` .

:

```
SELECT MAX(column_name) FROM table_name;
```

:

```
 select min(age) from employee;
```

`employee` `age` .

:

```
 SELECT MIN(column_name) FROM table_name;
```

() : https://riptutorial.com/ko/sql/topic/1002/---

64:

- ROW_NUMBER ()
- OVER ([PARTITION BY value_expression, ... [n]] order_by_clause)

Examples

.

```
SELECT
  ROW_NUMBER() OVER(ORDER BY Fname ASC) AS RowNumber,
  Fname,
  LName
FROM Employees
```

.

```
SELECT
  ROW_NUMBER() OVER(PARTITION BY DepartmentId ORDER BY DepartmentId ASC) AS RowNumber,
  DepartmentId, Fname, LName
FROM Employees
```

(1)

```
WITH cte AS (
  SELECT ProjectID,
         ROW_NUMBER() OVER (PARTITION BY ProjectID ORDER BY InsertDate DESC) AS rn
  FROM ProjectNotes
)
DELETE FROM cte WHERE rn > 1;
```

: https://riptutorial.com/ko/sql/topic/1977/-

S. No		Contributors
1	SQL	Arjan Einbu, brichins, Burkhard, cale_b, CL., Community, Devmati Wadikar, Epodax, geeksal, H. Pauwelyn, Hari, Joey, JohnLBevan, Jon Ericson, Lankymart, Laurel, Mureinik, Nathan, omini data, PeterRing, Phrancis, Prateek, RamenChef, Ray, Simone Carletti, SZenC, t1gor, ypercube
2	ALTER TABLE	Aidan, blackbishop, bluefeet, CL., Florin Ghita, Francis Lord, guiguiblitz, Joe W, KIRAN KUMAR MATAM, Lexi, mithra chintha , Ozair Kafray, Simon Foster, Siva Rama Krishna
3	AND OR	guiguiblitz
4	CREATE FUNCTION	John Odom, Ricardo Pontual
5	DROP DELETE	Abhilash R Vankayala, John Odom
6	GROUP BY	3N1GM4, Abe Miessler, Bostjan, Devmati Wadikar, Filipe Manuel, Frank, Gidil, Jaydles, juergen d, Nathaniel Ford, Peter Gordon, Simone - Ali One, WesleyJohnson, Zahiro Mor, Zoyd
7	IN	CL., juergen d, walid, Zaga
8	LIKE	Abhilash R Vankayala, Aidan, ashja99, Bart Schuijt, CL., Cristian Abelleira, guiguiblitz, Harish Gyanani, hellyale, Jenism, Lohitha Palagiri, Mark Perera, Mr. Developer, Ojen, Phrancis, RamenChef, Redithion, Stefan Steiger, Tot Zam, Vikrant, vmaroli
9	SQL Group by vs Distinct	carlosb
10	SQL	120196, CL., Clomp, Community, Epodax, Knickerless-Noggins, Stefan Steiger
11	SQL	Stefan Steiger
12	SQL	CL., Stivan
13	TRUNCATE	Abhilash R Vankayala, CL., Cristian Abelleira, DalmTo, Hynek Bernard, inquisitive_mind, KIRAN KUMAR MATAM, Paul Bambury, ss005
14	TRY / CATCH	Uberzen1

15	UNION / UNION ALL	Andrea, Athafoud, Daniel Langemann, Jason W, Jim, Joe Taras , KIRAN KUMAR MATAM, Lankymart, Mihai-Daniel Virna, sunkuet02
16	WHERE HAVING	Arulkumar, Bostjan, CL., Community, Franck Dernoncourt, H. Pauwelyn, Jon Chan, Jon Ericson, juergen d, Matas Vaitkevicius, Mureinik, Phrancis, Tot Zam
17	XML	Steven
18		Stefan Steiger
19		Abhilash R Vankayala, aholmes, Alok Singh, Amnon, Andrii Abramov, apomene, Arpit Solanki, Arulkumar, AstraSerg, Brent Oliver, Charlie West, Chris, Christian Sagmüller, Christos, CL., controller, dariru, Daryl, David Pine, David Spillett, day_dreamer , Dean Parker, DeepSpace, Dipesh Poudel, Dror, Durgpal Singh , Epodax, Eric VB, FH-Inway, Florin Ghita, FlyingPiMonster, Franck Dernoncourt, geeksal, George Bailey, Hari K M, HoangHieu, iliketocode, Imran Ali Khan, Inca, Jared Hooper, Jaydles, John Odom, John Slegers, Jojodmo, JonH, Kapep, KartikKannapur, Lankymart, Mark Iannucci, Mark Perera, Mark Wojciechowicz, Matas Vaitkevicius, Matt, Matt S, Mattew Whitt, Matthew Moisen, MegaTom, Mihai-Daniel Virna, Mureinik, mustaccio, mxmissile, Oded, Ojen, onedaywhen, Paul Bambury, penderi, Peter Gordon, Prateek, Praveen Tiwari, Přemysl Šťastný, Preuk, Racil Hilan, Robert Columbia, Ronnie Wang, Ryan, Saroj Sasmal, Shiva, SommerEngineering, sqluser, stark, sunkuet02, ThisIsImpossible, Timothy, user1336087, user1605665, waqasahmed, wintersolider, WMios, xQbert, Yury Fedorov, Zahiro Mor, zedfoxus
20		CL., Daniel, dd4711, fuzzy_logic, Gidil, Luis Lema, ninesided, Peter K, Phrancis, Sibeesh Venu
21		CL., Darren Bartrup-Cook, Martin Smith
22		dmfay
23		Andrea Montanari, CL., FlyingPiMonster, KjetilNordin
24		Ameya Deshpande, CL., Daniel Langemann, Dipesh Poudel, inquisitive_mind, KIRAN KUMAR MATAM, rajarshig, Tot Zam, zplizzi
25		bluefeet, Jared Hooper, John Odom, Jon Chan, JonMark Perry, Phrancis
26		Emil Rowland

27		Daryl
28		CL., Joel, KIRAN KUMAR MATAM, Stu
29		ɐləx, Allan S. Hansen, Arthur D, Arulkumar, Batsu, Chris, CL., Damon Smithies, Franck Dernoncourt, Golden Gate, hatchet, Imran Ali Khan, IncrediApp, Jaydip Jadhav, Jones Joseph, Kewin Björk Nielsen, Leigh Riffel, Matas Vaitkevicius, Mateusz Piotrowski, Neria Nachum, Phrancis, RamenChef, Robert Columbia, vmaroli, ypercube
30		Daryl, IncrediApp
31		Abhilash R Vankayala, CL., Kyle Hale, SQLFox, Zoyd
32		RamenChef, user2314737
33		a1ex07, Almir Vuk, carlosb, CL., David Manheim, FlyingPiMonster, forsvarir, Franck Dernoncourt, Horaciux, Jenism, KIRAN KUMAR MATAM, mauris, Parado, Paulo Freitas , Ryan
34		Simulant
35		Darrel Lee, mnoronha
36		John Smith
37	( )	CL., Karl Blacquiere, Matas Vaitkevicius, RamenChef
38		Andreas, CL.
39		Phrancis
40	,	Karthikeyan, RamenChef
41		A_Arnold, Akshay Anand, Andy G, bignose, Branko Dimitrijevic, Casper Spruit, CL., Daniel Langemann, Darren Bartrup-Cook, Dipesh Poudel, enrico.bacis, Florin Ghita, forsvarir, Franck Dernoncourt, hairboat, Hari K M, HK1, HLGEM, inquisitive_mind , John C, John Odom, John Slegers, Mark Iannucci, Marvin, Mureinik, Phrancis, raholling, Raidri, Saroj Sasmal, Stefan Steiger, sunkuet02, Tot Zam, xenodevil, ypercube, Рахул Маквана
42		Amir Pourmand, CL., Daryl, John Odom
43		Bart Schuijt, CL., dd4711, Devmati Wadikar, Phrancis, Saroj Sasmal, StanislavL, walid, ypercube

44		Abhilash R Vankayala, Arulkumar, Athafoud, bignose, Bostjan, Brad Larson, Christian, CL., Dariusz, Dr. J. Testington, enrico.bacis, Florin Ghita, FlyingPiMonster, forsvarir, Franck Dernoncourt, hairboat, JavaHopper, Jaydles, Jon Ericson, Magisch, Matt, Mureinik, Mzzzzzz, Prateek, rdans, Shiva, tinlyx, Tot Zam, WesleyJohnson
45		LCIII
46		CL., Harjot, Yehuda Shapira
47		brichins, John Odom, Lamak, Ryan
48		Hack-R
49		Amir978, CL., Florin Ghita
50		Blag, Özgür Öztürk
51		Andi Mohr, CL., Cristian Abelleira, Jaydles, mithra chintha, nazark, Özgür Öztürk, Parado, Phrancis, Wolfgang
52		Batsu, Chip, CL., Dylan Vander Berg, fredden, Joel, KIRAN KUMAR MATAM, Phrancis, Umesh, xenodevil, Zoyd
53		a1ex07, Gallus, Ryan Rockey, ypercube
54		Arkh, beercohol, bhs, Gidil, Jerry Jeremiah, Mureinik, mustaccio
55		Akshay Anand, CL., Daniel Vérité, Dariusz, Dipesh Poudel, FlyingPiMonster, Gidil, H. Pauwelyn, Jon Chan, KIRAN KUMAR MATAM, Matas Vaitkevicius, Matt, Phrancis, Sanjay Bharwani, sunkuet02, Tot Zam, TriskalJM, vmaroli, WesleyJohnson
56		ɐləx, Christos, CL., Dariusz, Fenton, Infinity, Jaydles, Matt, MotKohn, Mureinik, Peter Lang, Stanislovas Kalašnikovas
57		CL., Phrancis
58		Darren Bartrup-Cook
59		Aidan, alex9311, Almir Vuk, Ares, CL., drunken_monkey, Dylan Vander Berg, Franck Dernoncourt, H. Pauwelyn, Jojodmo, KIRAN KUMAR MATAM, Matas Vaitkevicius, Prateek
60		CL., dasblinkenlight, KIRAN KUMAR MATAM, Nunie123, Phrancis, RamenChef, tinlyx
61	()	CL., omini data

62	(/)	CL., Kewin Björk Nielsen, Mark Stewart
63	()	ashja99, CL., Florin Ghita, Ian Kenney, Imran Ali Khan, Jon Chan, juergen d, KIRAN KUMAR MATAM, Mark Stewart, Maverick, Nathan, omini data, Peter K, Reboot, Tot Zam, William Ledbetter, winseybash, Алексей Неудачин
64		CL., Phrancis, user1221533