FREE eBook

LEARNING
# amazon-redshift

Free unaffiliated eBook created from
Stack Overflow contributors.

#amazon-redshift

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: amazon-redshift

It is an unofficial and free amazon-redshift ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official amazon-redshift.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with amazon-redshift

Remarks

This section provides an overview of what amazon-redshift is, and why a developer might want to use it.

It should also mention any large subjects within amazon-redshift, and link out to the related topics. Since the Documentation for amazon-redshift is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

As shown in the step by step guide (getting started with Amazon Redshift),it involves :

- Step 1: Set Up Prerequisites
- Step 2: Launch a Sample Amazon Redshift Cluster
- Step 3: Authorize Access to the Cluster
- Step 4: Connect to the Sample Cluster

Setting Up Prerequisites

> Setting up Prerequisites involves Signing Up for AWS account and installing SQL Client Drivers and Tools like SQL Workbench/J
>
> To install SQL Workbench J and configure it:
>
> - Use this link to download generic package for all systems (sql-workbench) . Assuming you have all its prerequisites installed .
> - Get the appropriate JDBC/ODBC Driver.
> - Using the script in the installation directory launch SQL-Workbench .
> - Open Connection Window > Manage Drivers > Browse and select the driver file.
> - In the Select Connection Profile Window. Select the created driver in the Driver box. Paste the URL from the redshift cluster created (JDBC URL from the Amazon Redshift console) and the master user Username , Password.

Launch a Sample Amazon Redshift Cluster

- Click on Launch Cluster from Amazon Redshift Dashboard
- On cluster details page choose any Cluster Identifier, Database Name, Database Port (Choose an open port in your firewall since you cannot change the port number once the cluster is created.), Master User Name, Master User Password. On the Node Configuration page , select the Version (Redshift Version) , Node Type , Cluster Type and Number of Compute Nodes.
- Based on the EC2-VPC or EC2 Classic platform you select, the security steps vary for authorizing your cluster. For the rest of the pages you can use the default settings for now.

Authorize Cluster

For EC2-VPC Platform, click on the created cluster name after opening Redshift cluster tab from navigation pane and go to the configuration tab. In cluster properties choose the security group. Edit the inbound and outbound rules(Protocol, Port Range ,Source) as per your requirements from the Inbound and Outbound Tab.

For EC2-Classic Platform ,click on the created cluster name after opening Redshift cluster tab from navigation pane and go to the configuration tab. Choose default under Cluster Properties , for Cluster Security Groups. Then Choose the cluster security group from the Security Groups tab, in the cluster security group list. Select CIDR/IP from the connection type in the security group connections tab and authorize it with an IP/Port.

Connect to the Sample Cluster

Follow the last step in setting up prerequisites.

Connect Amazon redshift database and fetch data into Array using Node.js

Best way to connect amazon redshift using JDBC , Use proper driver as per version http://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html

Step-1: npm install jdbc

Step-2:

```
var JDBC = require('jdbc');
var jinst = require('jdbc/lib/jinst');
// isJvmCreated will be true after the first java call.  When this happens, the
// options and classpath cannot be adjusted.
if (!jinst.isJvmCreated()) {
  // Add all java options required by your project here.  You get one chance to
  // setup the options before the first java call.
  jinst.addOption("-Xrs");
  // Add all jar files required by your project here.  You get one chance to
  // setup the classpath before the first java call.
  jinst.setupClasspath(['./drivers/hsqldb.jar',
                        './drivers/derby.jar',
                        './drivers/derbyclient.jar',
                        './drivers/derbytools.jar',
                        './lib/drivers/RedshiftJDBC41-1.1.10.1010.jar'
                        ]);
}

var config = {
  url: 'jdbc:redshift://test-redshift.czac2vcs84ci.us-east-
.redshift.amazonaws.com:5439/testredshift?user=redshift&password=W9P3GC42GJYFpGxBitxPszAc8iZFW',

  drivername: 'com.amazon.redshift.jdbc41.Driver',
  user : 'username',
  password: 'password',
  minpoolsize: 10,
  maxpoolsize: 100
};
var hsqldbInit = false;
GLOBAL.hsqldb = new JDBC(config);`
```

Step-3: npm install async (Use async module to query your code) (Optional)

Step-4: Manually create one database name **test** and table **sample_data** , find amazon redshift database command here

Step-5:

```
var asyncjs = require('async');
hsqldb.reserve(function(err, connObj) {
    if (connObj) {
        console.log("Connection: " + connObj.uuid);
        var conn = connObj.conn;
        asyncjs.series([
            function(callback) {
                conn.createStatement(function(err, statement) {
                    if (err) {
                        callback(err);
                    } else {
                        statement.setFetchSize(100, function(err) {
                            if (err) {
                                callback(err);
                            } else {
                              statement.executeQuery("SELECT * FROM test.sample_data",
function(err, resultset) {
                                resultset.toObjArray(function(err,sresults){
                                  console.log(sresults);
                                });
                              });
                            }
                        })
                      }
                    })
                 }
              ])
         }
      })
```

Read Getting started with amazon-redshift online: https://riptutorial.com/amazon-redshift/topic/5489/getting-started-with-amazon-redshift

Chapter 2: Reading JSON array in Redshift

Introduction

Currently, reading all elements in JSON array is not possible in Redshift. For e.g. if you want to read Manufacturer and model as columns from following JSON

Remarks

dim_idnumberinfomanufacturermodel 2001
Nissan~Sentra^Nissan~Maxima^Ford~Taurus^Ford~Escort^NissanSentra 2002
Nissan~Sentra^Nissan~Maxima^Ford~Taurus^Ford~Escort^NissanMaxima 2003
Nissan~Sentra^Nissan~Maxima^Ford~Taurus^Ford~Escort^FordTaurus 2004
Nissan~Sentra^Nissan~Maxima^Ford~Taurus^Ford~Escort^FordEscort

Examples

Reading array elements in JSON

```
-- Create a sample JSON with ARRAY
```

```
create table car_sample(dim_id integer, info varchar(2000)); insert into car_sample values (200,
'{"cars": [ { "Manufacturer": "Nissan", "Models": [{"Name":"Sentra", "doors":4}, {"Name":"Maxima",
"doors":4} ]}, {"Manufacturer": "Ford", "Models": [{"Name":"Taurus", "doors":4}, {"Name":"Escort",
"doors":4} ]} ] }')
```

```
-- Create a supporting table for CROSS JOIN
```

```
create table series1_10 (number integer );
insert into series1_10 values (1);
insert into series1_10 values (2);
insert into series1_10 values (3);
insert into series1_10 values (4);
insert into series1_10 values (5);
insert into series1_10 values (6);
insert into series1_10 values (7);
insert into series1_10 values (8);
insert into series1_10 values (9);
insert into series1_10 values (10);
```

```
-- UDF for extracting JSON array into one ^ delimited string
CREATE OR REPLACE FUNCTION f_extractJson (jsonVar varchar) RETURNS varchar
IMMUTABLE as $$
 def myfunc(myParm):
  import json
  cars=json.loads(jsonVar)
  parsedString=''
  for car in cars["cars"]:
   for model in car["Models"]:
    parsedString=parsedString+car["Manufacturer"]+'~'+model["Name"]+'^'
  return parsedString

 return myfunc(jsonVar)
$$ LANGUAGE plpythonu;

-- Check the data

select dim_id, f_extractJson(info) from car_sample;

-- Pivot rows

WITH w1 AS (select dim_id, f_extractJson(info) info from car_sample)
select dim_id,number, info, split_part(split_part(info,'^',number),'~', 1)
Manufacturer, split_part(split_part(info,'^',number),'~', 2) Model
from w1 cross join series1_10
where number <= regexp_count(info,'[=^=]') ;
```

Read Reading JSON array in Redshift online: https://riptutorial.com/amazon-redshift/topic/8769/reading-json-array-in-redshift

Credits

S. No	Chapters	Contributors
1	Getting started with amazon-redshift	Akshay Hazari, Community, Hardik Raval
2	Reading JSON array in Redshift	Dinesh Pohane