## Module: Optional Spark with Snowflake (Requires you to have an available Spark environment that is not provided)

Apache Spark is an open-source, reliable, scalable and distributed generalpurpose computing engine used for **processing** and analyzing big data files from different sources like HDFS, S3, Azure e.t.c

### Snowflake Spark Connector

Snowflake Spark connector “spark-snowflake” enables Apache Spark to read data from, and write data to Snowflake tables. When you use a connector, Spark treats Snowflake as data sources similar to HDFS, S3, JDBC, e.t.c. In fact, Snowflake spark-connector provides the data source . `"net.snowflake.spark.snowflake"` and it’s short-form `"snowflake"`

Snowflake provides a separate Spark connector for each Spark version hence, make sure you download and used the right version for your Spark.

The connector uses the JDBC driver to communicate with Snowflake and performs the following operations.

- Create a Spark DataFrame by reading a table from Snowflake.
- Write the Spark DataFrame to Snowflake table.

Data transfer between Spark RDD/DataFrame/Dataset and Snowflake happens through Snowflake internal storage (created automatically) or external storage (user provides AWS/Azure) which is used by Snowflake Spark connector to store temporary session data.

Every time when you access the Snowflake from Spark, It does the following.

- The session is created with a stage along with storage on Snowflake schema.
- It maintains the stage thorough out the session.
- Uses the stage to store intermediate data and
- Finally drops the stage when you end the connection.

### Maven dependency

```
<dependency>
     <groupId>net.snowflake</groupId>
```

```
        <artifactId>spark-snowflake_2.11</artifactId>
        <version>2.5.9-spark_2.4</version>
```

```
</dependency>
```

## Create a Snowflake table to access from Spark

Unfortunately, while working with Spark, you can't use the default database that comes with Snowflake account as spark-connector needs the privilege to create a stage on schema but we can't change the permission on default schema hence, will create a new database and table.

In order to create a Database, logon to Snowflake web console, select the Databases from the top menu and select "create a new database" option and finally enter the database name on the form and select "Finish" button.

To create a table you can use either Snowflake web console or use the below program to create.

```
val properties = new java.util.Properties()    properties.put("user",
"user")
  properties.put("password", "#########")
properties.put("account", "oea82")
properties.put("warehouse", "mywh")
properties.put("db", "EMP")
properties.put("schema", "public")
properties.put("role","ACCOUNTADMIN")
  //JDBC connection string        val   jdbcUrl   =   "jdbc:snowflake://oea82.us-east-
1.snowflakecomputing.com/"                          val         connection         =
DriverManager.getConnection(jdbcUrl,   properties)         val   statement   =
connection.createStatement      statement.executeUpdate("create or replace table
EMPLOYEE(name
VARCHAR, department VARCHAR, salary number)")
statement.close    connection.close()
```

Scala

Copy

This Spark with Snowflake example is also available at GitHub project for reference

## Spark Connection parameters

In order to read/write you need to basically provide the following options

- `sfURL` : URL of your account for e.g https://oea82.us-east-lakecomputing.com/
- `sfAccount` : You account name, you can get this from URL for e.g 1
- : Snowflake user name, typically your login user
- : user password e :
- `sfUser` `sfWarehous` Snowflake Dataware house name “oea82”
- : Snowflake Database name
- `sfPassword` : Database schema where your table belongs
- : Snowflake user role
- e
- `sfSchema`
- `sfRole`
- `sfDatabase`

and mor

## Write Spark DataFrame to Snowflake table Example

By using the `write()` method (which is `DataFrameWriter` object) of the DataFrame and providing below values, you can write the Spark DataFrame to Snowflake table.

Use `format()` to specify the data source name either `snowflake` or `net.snowflake.spark.snowflake`

Use `Option()` to specify the connection parameters like URL, account, username, password, database name, schema, role and more.

Use `dbtable` option to specify the Snowflake table name you wanted to write to

Use mode() to specify if you wanted to overwrite, append, or ignore if the file already present.

```
 package com.sparkbyexamples.spark
 import org.apache.spark.sql.{SaveMode, SparkSession}

object WriteEmpDataFrameToSnowflake extends App {

  val spark = SparkSession.builder()
    .master("local[1]")
    .appName("SparkByExamples.com")
    .getOrCreate();

  spark.sparkContext.setLogLevel("ERROR")
   import spark.implicits._
   val simpleData = Seq(("James","Sales",3000),
    ("Michael","Sales",4600),
    ("Robert","Sales",4100),
```

```
    ("Maria","Finance",3000),
    ("Raman","Finance",3000),
    ("Scott","Finance",3300),
    ("Jen","Finance",3900),
    ("Jeff","Marketing",3000),
    ("Kumar","Marketing",2000)
  )
  val df = simpleData.toDF("name","department","salary")    df.show()

  var sfOptions = Map(
    "sfURL" -> "https://oea82.us-east-1.snowflakecomputing.com/",
    "sfAccount" -> "oea82",
    "sfUser" -> "user",
    "sfPassword" -> "#####################",
    "sfDatabase" -> "EMP",
    "sfSchema" -> "PUBLIC",
    "sfRole" -> "ACCOUNTADMIN"
  )
  df.write
    .format("snowflake")
    .options(sfOptions)
    .option("dbtable", "EMPLOYEE")
    .mode(SaveMode.Overwrite)
    .save()

}
```

Scala

Copy

This Spark Snowflake connector scala example is also available at GitHub project [WriteEmpDataFrameToSnowflake.scala](WriteEmpDataFrameToSnowflake.scala) for reference

## Read Snowflake table into Spark DataFrame Example

By using the `read()` method (which is `DataFrameReader` object) of the SparkSession and providing data source name via format() method, connection options, and table name using `dbtable`

```
 package  com.sparkbyexamples.spark
 import  org.apache.spark.sql.{DataFrame, SparkSession}

object ReadEmpFromSnowflake extends App{
```

```
  val spark = SparkSession.builder()
    .master("local[1]")
    .appName("SparkByExamples.com")
    .getOrCreate();

  var sfOptions = Map(
    "sfURL" -> "https://oea82.us-east-1.snowflakecomputing.com/",
    "sfAccount" -> "oea82",
    "sfUser" -> "user",
    "sfPassword" -> "##############",
    "sfDatabase" -> "EMP",
    "sfSchema" -> "PUBLIC",
    "sfRole" -> "ACCOUNTADMIN"
  )
  val df: DataFrame = spark.read
    .format("net.snowflake.spark.snowflake")  // or just use "snowflake"
.options(sfOptions)
    .option("dbtable", "EMPLOYEE")
    .load()

  df.show(false)
}
```

Scala

Copy

This yields below output

```
+-------+----------+------+
|NAME   |DEPARTMENT|SALARY|
+-------+----------+------+
|James  |Sales     |3000  |
|Michael|Sales     |4600  |
|Robert |Sales     |4100  |
|Maria  |Finance   |3000  |
|Raman  |Finance   |3000  |
|Scott  |Finance   |3300  |
|Jen    |Finance   |3900  |
|Jeff   |Marketing |3000  |
|Kumar  |Marketing |2000  |
+-------+----------+------+
```

Scala

Copy

Above Snowflake with Spark example demonstrates reading the entire table from the Snowflake table using `dbtable` option and creating a Spark DataFrame, below example uses a `query` option to execute a group by aggregate SQL query.

```
  val df1: DataFrame = spark.read
    .format("net.snowflake.spark.snowflake")
    .options(sfOptions)
    .option("query",  "select department, sum(salary) as total_salary from EMPLOYEE group by department")
    .load() df1.show(false)
```

Scala

Copy

This yields the below output.

```
+----------+------------+
|DEPARTMENT|TOTAL_SALARY|
+----------+------------+
|Sales     |11700       |
|Finance   |13200       |
|Marketing |5000        |
+----------+------------+
```

Scala

Copy

This Spark Snowflake connector scala example is also available at GitHub project [ReadEmpFromSnowflake]

## Column Mapping

When your column names do not match between Spark DataFrame schema and Snowflake table-use `<strong>columnmap</strong>` options with a parameter as a single string literal.

```
.option("columnmap",  "Map(col_2 -> col_b, col_3 -> col_a)")
```

Scala

Copy

## Saving Modes

Spark `DataFrameWriter` also has a method mode() to specify SaveMode; the argument to this method either takes below string or a constant from `SaveMode` class.

overwrite – mode is used to overwrite the existing file, alternatively, you `SaveMode.Overwrite` can use .

append – To add the data to the existing file, alternatively, you can `SaveMode.Append` use .

ignore – Ignores write operation when the file already exists, alternatively you can use . `SaveMode.Ignore`

errorifexists or error – This is a default option when the file already exists, it returns an error, alternatively, you can use . `SaveMode.ErrorIfExists`

## Conclusion

In this article, you have learned Snowflake is a cloud-based Dataware house database and storage engine that uses traditional ANSI SQL syntax to interact with the database and learned how to read a Snowflake table to Spark DataFrame and write Spark DataFrame to Snowflake table using Snowflake connector.