**HIVE AND SPARK SQL EXAMPLE :**

**EXAMPLE**

Use the dataset present at the location:

(hdfs:///bigdatapgp/common\_folder/assigment6/airline\_data/)

Dataset Description:

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| *ID* | *Unique flight ID* |
| *Year* | *2004-2008* |
| *Month* | *1-12* |
| *DayofMonth* | *1-31* |
| *DayOfWeek* | *1 (Monday) - 7 (Sunday)* |
| *DepTime* | *Actual departure time (local, hhmm)* |
| *CRSDepTime* | *Scheduled departure time (local, hhmm)* |
| *ArrTime* | *Actual arrival time (local, hhmm)* |
| *CRSArrTime* | *Scheduled arrival time (local, hhmm)* |
| *UniqueCarrier* | *Unique carrier code* |
| *FlightNum* | *Flight number* |
| *TailNum* | *Plane tail number* |
| *ActualElapsedTime* | *Time in minutes* |
| *CRSElapsedTime* | *Time in minutes* |
| *AirTime* | *Time in minutes* |
| *ArrDelay* | *Arrival delay, in minutes* |
| *DepDelay* | *Departure delay, in minutes* |
| *Origin* | *Origin IATA airport code (International Air Transport Association)* |
| *Dest* | *Destination IATA airport code* |
| *Distance* | *In miles* |
| *Taxiin* | *Time taken from runway to the terminal after landing, excluding deceleration, in minutes* |
| *TaxiOut* | *Taxi taken from terminal to the runway before take-off, excluding acceleration, in minutes* |
| *Cancelled* | *Whether the flight was cancelled* |
| *Cancellation Code* | *Reason for cancellation (A = carrier, B = weather, C = NAS, D = security)* |
| *Diverted* | *1 = yes, 0 = no* |
| *Carrier Delay* | *Time in minutes* |
| *WeatherDelay* | *Time in minutes* |
| *NASDelay* | *Time in minutes* |
| *Security Delay* | *Time in minutes* |
| *LateAircraftDelay* | *Time in minutes* |

Write the Hive queries for the following:

1. Create partitioned tables based on the year.
2. Insert data into the table created by you previously.
3. Compare the average arrival delay for each year.
4. Compare the average arrival delay for each month.
5. Compare the average taxi-in time for each year.
6. Compare the average taxi-in time for each month.
7. Compare the average taxi-out time for each month.
8. Compare the average departure delay for each month.
9. Compare the time taken for partitioned and non-partitioned tables to run a SELECT \* query on the complete database.

**ANSWERS**

**# StackOverFlow with dataset files as comments,posts,posttypes,users**

**# Create Database and External table**

**-- Create database ADIDAS;**

**-- USE ADIDAS;**

Create external table

comments(id int,

userid int)

row format delimited

fields terminated by ',' location '/tmp/comments';

create external table posts (

id int,

post\_type smallint,

creationdate timestamp,

score int,

viewcount int,

owneruserid smallint,

title string,

answercount int,

commentcount smallint)

row format delimited

fields terminated by ',' location '/tmp/comments';

create external table posttypes (

id int,

name string)

row format delimited

fields terminated by ',' location '/tmp/posttypes';

create external table users (

id int,

reputation int,

displayname string,

loc string,

age tinyint

)

row format delimited

fields terminated by ',' location '/tmp/users';

**#A. Find the display name and no. of posts created by the user who has got maximum reputation.**

select displayname,reputation from users group by displayname,reputation order by reputation desc;

**#B. Find the average age of users on the Stack Overflow site.**

select avg(age) from users;

**#C. Find the display name of user who posted the oldest post on Stack Overflow (in terms of date).**

select u.displayname,p.creationdate from users u join posts p on (u.id = p.owneruserid) order by p.creationdate;

**#D. Find the display name and no. of comments done by the user who has got maximum reputation.**

select u.displayname,p.commentcount,max(reputation) as rep from users u join posts p on u.id = p.owneruserid join comments c on c.userid = p.owneruserid group by u.displayname,p.commentcount order by rep desc;

**#E(i). Find the display name of user who has created maximum no. of posts on Stack Overflow.**

select u.displayname,p.owneruserid,count(\*) as cid from users u join posts p on p.owneruserid = u.id group by u.displayname,p.owneruserid order by cid desc;

**(ii) Find the display name of user who has commented maximum no. of posts on Stack Overflow.**

select u.displayname,p.commentcount from users u join posts p on p.owneruserid = u.id group by u.displayname,p.commentcount order by p.commentcount desc;

**#F. Find the owner name and id of user whose post has got maximum no. of view counts so far.**

select u.displayname,u.id,p.viewcount from users u join posts p on u.id = p.owneruserid order by p.viewcount desc ;

#G(i). Find the title and owner name of the post which has maximum no. of Answer Count.

select u.displayname,p.title,p.answercount from users u join posts p on u.id = p.owneruserid order by p.answercount desc ;

**(ii) Find the title and owner name of post who has got maximum no. of Comment count.**

select u.displayname,p.title,p.commentcount from users u join posts p on u.id = p.owneruserid where p.title != '' order by p.commentcount desc ;

#H. Find the location which has maximum no of Stack Overflow users

select loc,count(\*) as c from users group by loc order by c desc;

**#I. Find the total no. of answers, posts, comments created by Indian users.**

select count(p.post\_type) from posts p join users u on u.id = p.owneruserid where p.posttype = 2 and u.loc == 'India';

select count(p.id) from posts p join users u on u.id = p.owneruserid where u.loc == 'India';

select count(\*) from comments c join users u on u.id = c.userid where u.loc == 'India';

**EXAMPLE 2:**

Use the dataset present at the location:

(hdfs:///bigdatapgp/common\_folder/assigment6/airline\_data/)

Dataset Description:

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| *ID* | *Unique flight ID* |
| *Year* | *2004-2008* |
| *Month* | *1-12* |
| *DayofMonth* | *1-31* |
| *DayOfWeek* | *1 (Monday) - 7 (Sunday)* |
| *DepTime* | *Actual departure time (local, hhmm)* |
| *CRSDepTime* | *Scheduled departure time (local, hhmm)* |
| *ArrTime* | *Actual arrival time (local, hhmm)* |
| *CRSArrTime* | *Scheduled arrival time (local, hhmm)* |
| *UniqueCarrier* | *Unique carrier code* |
| *FlightNum* | *Flight number* |
| *TailNum* | *Plane tail number* |
| *ActualElapsedTime* | *Time in minutes* |
| *CRSElapsedTime* | *Time in minutes* |
| *AirTime* | *Time in minutes* |
| *ArrDelay* | *Arrival delay, in minutes* |
| *DepDelay* | *Departure delay, in minutes* |
| *Origin* | *Origin IATA airport code (International Air Transport Association)* |
| *Dest* | *Destination IATA airport code* |
| *Distance* | *In miles* |
| *Taxiin* | *Time taken from runway to the terminal after landing, excluding deceleration, in minutes* |
| *TaxiOut* | *Taxi taken from terminal to the runway before take-off, excluding acceleration, in minutes* |
| *Cancelled* | *Whether the flight was cancelled* |
| *Cancellation Code* | *Reason for cancellation (A = carrier, B = weather, C = NAS, D = security)* |
| *Diverted* | *1 = yes, 0 = no* |
| *Carrier Delay* | *Time in minutes* |
| *WeatherDelay* | *Time in minutes* |
| *NASDelay* | *Time in minutes* |
| *Security Delay* | *Time in minutes* |
| *LateAircraftDelay* | *Time in minutes* |

Solve the following using Spark SQL:

1. Count the number of flights which have the unique ordered pair of source and destination, i.e., for two flights, either the origin differs or destination differs.
2. For each year fetch the airport code that has the maximum number of outgoing flights and order the results in descending order w.r.t the number of outgoing flights.
3. Year-wise count the total number of Cancelled flights due to bad Weather, order the results w.r.t the count.

**Expected Solution:** You need to paste the Spark SQL commands and your findings for each step.

**ANSWERS :**



1. Count the number of flights which have the unique ordered pair of source and destination, i.e., for two flights, either the origin differs or destination differs.

2. For each year fetch the airport code that has the maximum number of outgoing flights and order the results in descending order w.r.t the number of outgoing flights.

3. Year-wise count the total number of Cancelled flights due to bad Weather, order the results w.r.t the count.

Use the dataset present at the location:

(hdfs:///bigdatapgp/common\_folder/assigment6/airline\_data/)

###################################################################

// In Scala

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.\_

import org.apache.spark.sql.functions.\_

val schema = StructType(Array(

StructField("\_id", IntegerType, true),

StructField("Year", StringType, true),

StructField("Month", StringType, true),

StructField("DayofMonth", StringType, true),

StructField("DayOfWeek", StringType, true),

StructField("DepTime", StringType, true),

StructField("CRSDepTime", StringType, true),

StructField("ArrTime", StringType, true),

StructField("CRSArrTime", StringType, true),

StructField("UniqueCarrier", StringType, true),

StructField("FlightNum", StringType, true),

StructField("TailNum", StringType, true),

StructField("ActualElapsedTime", StringType, true),

StructField("CRSElapsedTime", StringType, true),

StructField("AirTime", StringType, true),

StructField("ArrDelay", StringType, true),

StructField("DepDelay", StringType, true),

StructField("Origin", StringType, true),

StructField("Dest", StringType, true),

StructField("Distance", StringType, true),

StructField("Taxiin", StringType, true),

StructField("TaxiOut", StringType, true),

StructField("Cancelled", StringType, true),

StructField("Cancellation\_Code", StringType, true),

StructField("Diverted", StringType, true),

StructField("Carrier\_Delay", StringType, true),

StructField("WeatherDelay", StringType, true),

StructField("NASDelay", StringType, true),

StructField("Security\_Delay", StringType, true),

StructField("LateAircraftDelay", StringType, true)

))

var file = "maprfs:///data/flights.json"

val df = spark.read.format("csv").option("inferSchema", "false").schema(schema).load(file)

case class Flight(\_id: Integer, Year: String, Month: String, DayofMonth: String, DayOfWeek: String, DepTime: String, CRSDepTime: String, ArrTime: String,CRSArrTime: String,

UniqueCarrier: String, FlightNum: String, TailNum: String, ActualElapsedTime: String, CRSElapsedTime: String, AirTime: String, ArrDelay: String, DepDelay: String, Origin: String, Dest: String, Distance: String,

Taxiin: String, TaxiOut: String, Cancelled: String, Cancellation\_Code: String, Diverted: String, Carrier\_Delay: String, WeatherDelay: String, NASDelay: String, Security\_Delay: String, LateAircraftDelay: String) extends Serializable

val df = spark.read.format("csv").option("inferSchema", "false").schema(schema).load(file).as[Flight]

// val schema = "date STRING, delay INT, distance INT, origin STRING, destination STRING"

// cache DataFrame in columnar format in memory

df.cache

// create Table view of DataFrame for Spark SQL

df.createOrReplaceTempView("flights")

1. Count the number of flights which have the unique ordered pair of source and destination, i.e., for two flights, either the origin differs or destination differs.

=================================================================================================================================================================================

df.filter($"Origin" != $"Dest").groupBy($"FlightNum").count.orderBy(desc( "count")).show

2. For each year fetch the airport code that has the maximum number of outgoing flights and order the results in descending order w.r.t the number of outgoing flights.

===================================================================================================================================================================================

Here Origin = Origin IATA airport code (International Air Transport Association)

%sql

flights\_delta = spark.read.format("delta").load("/mnt/home/edureka\_918210/airline\_data/Airline.csv")

display(flights\_delta.groupBy("Year","Origin").max(count("Origin").alias("MaxFlights")).orderBy("MaxFlights", ascending=True))

3. Year-wise count the total number of Cancelled flights due to bad Weather, order the results w.r.t the count.

=======================================================================================================================

// Spark SQL

spark.sql("select Year, count(Cancelled) from flights group by Year").show

df.select($"Year",$"Count").orderby(desc("Year")).show

**EXAMPLE** :

Use the dataset present at the location:

(hdfs:///bigdatapgp/common\_folder/assigment6/airportdataset/)

Dataset Description:

|  |  |
| --- | --- |
| ***Fields*** | ***Description*** |
| *Airport ID* | *Unique OpenFlights identifier for this airport.* |
| *Name* | *Name of airport. May or may not contain the City name.* |
| *City* | *Main city served by airport. May be spelled differently from Name.* |
| *Country* | *Country or territory where airport is located.* |
| *IATA/FAA* | *3-letter FAA code, for airports located in Country "United States of America". 3-letter IATA code, for all other airports. NULL if not Assigned/Unknown.* |
| *ICAO* | *4-letter ICAO code. Null if not assigned.* |
| *Latitude* | *Decimal degrees, usually to six significant digits. Negative is South, positive is North.* |
| *Longitude* | *Decimal degrees, usually to six significant digits. Negative is West, positive is East.* |
| *Altitude* | *In feet.* |
| *Timezone* | *Hours offset from UTC. Fractional hours are expressed as decimals, eg. India is 5.5.* |
| *DST* | *Daylight savings time. One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U (Unknown).* |
| *Tz database*  *timezone* | *Timezone in "tz" (Olson) format, eg. "America/Los\_Angeles".* |

Solve the following using Spark SQL:

1. Load the dataset into a DataFrame and print its schema
2. Find out how many airports are there in the Southeast part in the dataset?
3. Find out how many unique cities have airports in each country
4. What is the average Altitude (in feet) of airports in each Country?
5. Find out in each timezone how many airports are operating?
6. Calculate average latitude and longitude for these airports in each country
7. Count how many different DSTs are there?

**ANSWERS :**



val sqlContext = new org.apache.spark.sql.SQLContext(sc)

import sqlContext.implicits.\_

1. Load the dataset into a DataFrame and print its schema

=================================================================

val airportDF = sqlContext.load("com.databricks.spark.csv", Map("path" -> "/mnt/home/edureka\_918210/airline\_data/Airline.csv", "header" -> "true"))

airportDF.show

airportDF.registerTempTable("airports")

2. Find out how many airports are there in the Southeast part in the dataset?

===================================================================================

sqlContext.sql("select AirportID, Name, Latitude, Longitude from airports where Latitude<0 and Longitude>0").collect.foreach(println)

3. Find out how many unique cities have airports in each country

======================================================================

sqlContext.sql("select Country, count(distinct(City)) from airports group by Country").collect.foreach(println)

4. What is the average Altitude (in feet) of airports in each Country?

==============================================================================

sqlContext.sql("select Country , avg(Altitude) from airports group by Country").collect

5. Find out in each timezone how many airports are operating?

==================================================================

sqlContext.sql("select Tz , count(Tz) from airports group by Tz").collect.foreach(println)

6. Calculate average latitude and longitude for these airports in each country

========================================================================================

sqlContext.sql("select Country, avg(Latitude), avg(Longitude) from airports group by Country").collect.foreach(println)

7. Count how many different DSTs are there?

================================================

sqlContext.sql("select count(distinct(DST)) from airports").collect.foreach(println)

**EXAMPLE :**

import sqlContext.implicits.\_

val airportDF = sqlContext.load("com.databricks.spark.csv", Map("path" -> "/mnt/home/edureka\_918210/airline\_data/Airline.csv", "header" -> "true"))

airportDF.show

airportDF.registerTempTable("airports")

sqlContext.sql("select AirportID, Name, Latitude, Longitude from airports where Latitude<0 and Longitude>0").collect.foreach(println)

hdfs:///bigdatapgp/common\_folder/assigment6/airline\_data/Airline.csv

https://github.com/poonamvligade/Apache-Spark-Projects/blob/master/AirlinesProject/SparkAirlineAanalytics.md

1. Create partitioned tables based on the year.

==============================================================

CREATE TABLE IF NOT EXISTS AirlineAnalytics(

ID int,

Year STRING,

Month STRING,

DayofMonth STRING,

DayOfWeek STRING,

DepTime STRING,

CRSDepTime STRING,

ArrTime STRING,

CRSArrTime STRING,

UniqueCarrier STRING,

FlightNum STRING,

TailNum STRING,

ActualElapsedTime STRING,

CRSElapsedTime STRING,

AirTime STRING,

ArrDelay STRING,

DepDelay STRING,

Origin STRING,

Dest STRING,

Distance STRING,

Taxiin STRING,

TaxiOut STRING,

Cancelled STRING,

Cancellation\_Code STRING,

Diverted STRING,

Carrier\_Delay STRING,

WeatherDelay STRING,

NASDelay STRING,

Security\_Delay STRING,

LateAircraftDelay STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

PARTITIONED BY (Year STRING) STORED AS SEQUENCEFILE;

2. Insert data into the table created by you previously.

================================================================

LOAD DATA LOCAL INPATH '/mnt/home/edureka\_918210/airline\_data/Airline.csv' INTO TABLE AirlineAnalytics;

-- PATH TO BE PROVIDED DURING TESTING

3. Compare the average arrival delay for each year.

===============================================================

select avg(ArrDelay),Year from AirlineAnalytics group by Year;

4. Compare the average arrival delay for each month

============================================================

select avg(ArrDelay),Month from AirlineAnalytics where YEAR = '2004' group by Month;

5. Compare the average taxi-in time for each year.

========================================================

select avg(Taxiin),Year from AirlineAnalytics group by Year;

6. Compare the average taxi-in time for each month.

=========================================================

select avg(Taxiin),Month from AirlineAnalytics where YEAR = '2004' group by Month;

7. Compare the average taxi-out time for each month.

========================================================

select avg(TaxiOut),Month from AirlineAnalytics where YEAR = '2004' group by Month;

8. Compare the average departure delay for each month.

===============================================================

select avg(DepDelay),Month from AirlineAnalytics where YEAR = '2004' group by Month;

9. Compare the time taken for partitioned and non-partitioned tables to run a SELECT \* query on the complete database.

=============================================================================================================================

TIME calculation:

Parititioned :

-------------------------

CREATE TABLE IF NOT EXISTS AirlineAnalytics(

ID int,

Year STRING,

Month STRING,

DayofMonth STRING,

DayOfWeek STRING,

DepTime STRING,

CRSDepTime STRING,

ArrTime STRING,

CRSArrTime STRING,

UniqueCarrier STRING,

FlightNum STRING,

TailNum STRING,

ActualElapsedTime STRING,

CRSElapsedTime STRING,

AirTime STRING,

ArrDelay STRING,

DepDelay STRING,

Origin STRING,

Dest STRING,

Distance STRING,

Taxiin STRING,

TaxiOut STRING,

Cancelled STRING,

Cancellation\_Code STRING,

Diverted STRING,

Carrier\_Delay STRING,

WeatherDelay STRING,

NASDelay STRING,

Security\_Delay STRING,

LateAircraftDelay STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

PARTITIONED BY (Year STRING) STORED AS SEQUENCEFILE;

LOAD DATA LOCAL INPATH '/mnt/home/edureka\_918210/airline\_data/Airline.csv' INTO TABLE AirlineAnalytics;

select \* from AirlineAnalytics;

TIME calculation:

Non-Parititioned :

----------------------

CREATE TABLE IF NOT EXISTS AirlineAnalytics(

ID int,

Year STRING,

Month STRING,

DayofMonth STRING,

DayOfWeek STRING,

DepTime STRING,

CRSDepTime STRING,

ArrTime STRING,

CRSArrTime STRING,

UniqueCarrier STRING,

FlightNum STRING,

TailNum STRING,

ActualElapsedTime STRING,

CRSElapsedTime STRING,

AirTime STRING,

ArrDelay STRING,

DepDelay STRING,

Origin STRING,

Dest STRING,

Distance STRING,

Taxiin STRING,

TaxiOut STRING,

Cancelled STRING,

Cancellation\_Code STRING,

Diverted STRING,

Carrier\_Delay STRING,

WeatherDelay STRING,

NASDelay STRING,

Security\_Delay STRING,

LateAircraftDelay STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

LOAD DATA LOCAL INPATH '/mnt/home/edureka\_918210/airline\_data/Airline.csv' INTO TABLE AirlineAnalytics;

select \* from AirlineAnalytics;