Project Gladiator

Retail Project
Group 7

Prepared By:

Sr. No.	Name	PS No.
1	Pratiksha Shetty	10676654
2	Aman Sharma	10677201
3	Gautam Nair	10676940

INDEX

==========

Problem Statement

==========

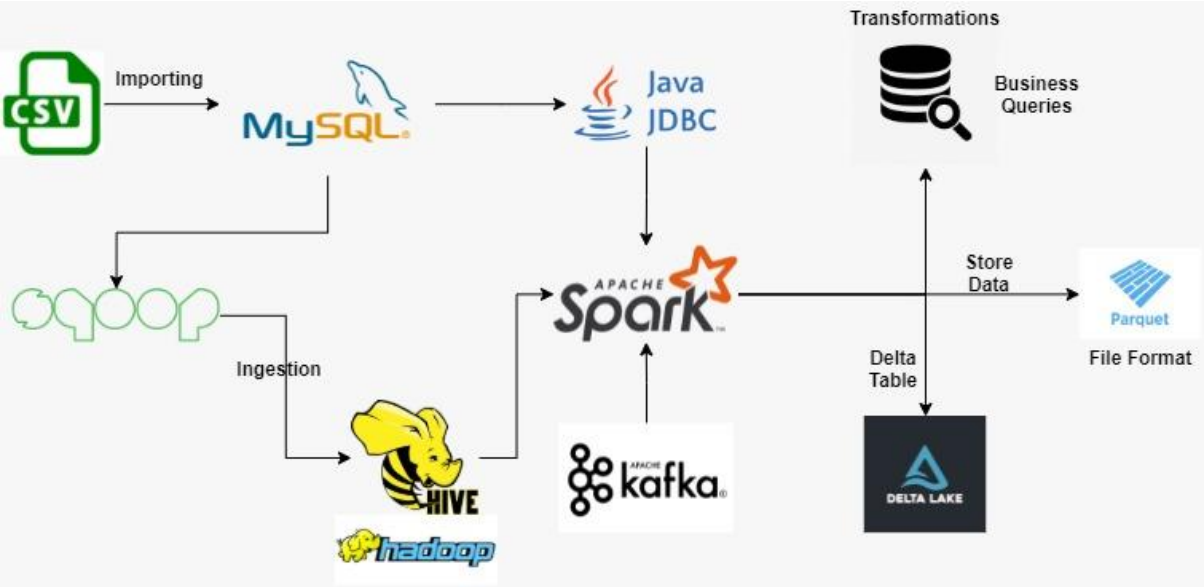
The E-commerce industry is growing like never before. In its process of growth, it is rapidly creating tons of data. Without processing this data, it becomes challenging to obtain insights from it. The dataset is a relational set of files describing customer orders over time. The dataset is anonymized and contains a sample of over millions of orders.

As a big data engineer you should help the data science team to deal with heterogeneous data by analysing the concrete information which is obtained by transforming raw data thus providing Business Insights to Business Users.

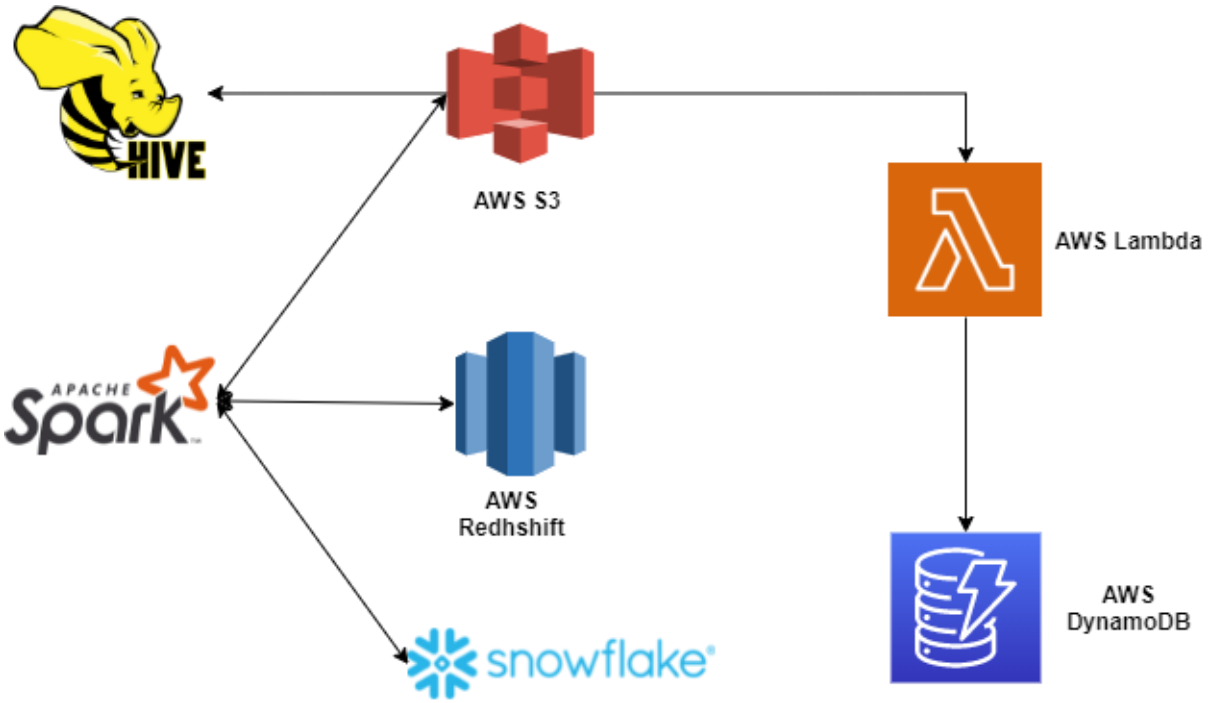
DESIGN AND ALGORITHM

==========

On Premises:

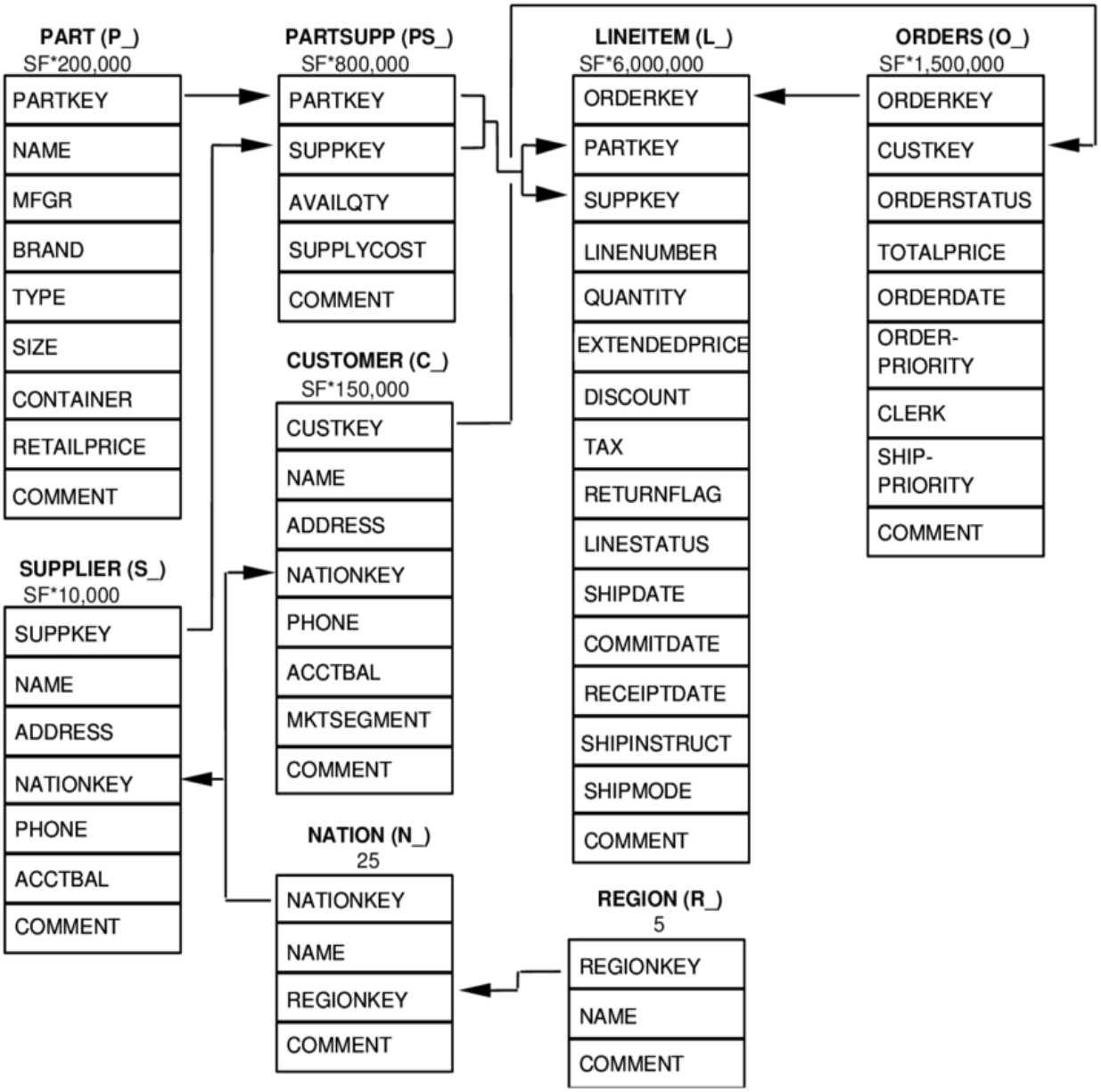


On AWS:



SCHEMA OF TABLES

==========



Task 1: Data Loading (Data Ingestion) to different sources

==========

1.1 Loading data from .csv files to MySQL

Since most of the enterprises store their Transactional data on MySQL or Oracle, it would be best to mimic the real time scenarios and ingest the data through MySQL.

Step1: Create Database

create database retail_pg;

```
mysql> create database retail_pg;
Query OK, 1 row affected (0.01 sec)
```

Step2: Create Tables

1) Region Table

```
create table Region(R_RegionKey int(38) not null,R_Name
varchar(25) not null,R_Comment varchar(152) not null,
primary key(R_RegionKey));
```

2) Nation Table

```
create table Nation(N_NationKey int(38) not null,N_Name
varchar(25) not null,N_RegionKey int(38) not null,
N_Comment varchar(152) not null, primary
key(N_NationKey));
```

3) Supplier Table

```
create table Supplier(S_SuppKey int(38) not null,S_Name
varchar(25) not null,S_Address varchar(40) not
null,S_NationKey int(38) not null,S_Phone varchar(15) not
null,s_AcctBal float(12,2) not null,S_Comment
varchar(101) not null, primary key(S_SuppKey), foreign
key(S_NationKey) references Nation(N_NationKey));
```

4) Part Table

```
create table Part(P_PartKey int(38) not null,P_Name
varchar(55) not null, P_MFGR varchar(25) not null
,P_Brand varchar(10) not null,P_Type varchar(25) not
null,P_Size int(38) not null,P_Container varchar(10) not
null,P_RetailPrice float(12,2) not null,P_Comment
varchar(23) not null, primary key(P_PartKey));
```

5) Partsupp Table

```
create table Partsupp(PS_Partkey int(38) not null,
PS_Suppkey int(38) not null, PS_Availability int(38) not
null, PS_Supplycost double(12,2) not null, PS_Comment
varchar(199) not null, foreign key(PS_PartKey) references
Part(P_PartKey), foreign key(PS_SuppKey) references
Supplier(S_SuppKey));
```

6) Customer Table

```
create table Customer(C_CustKey int(38) not null,C_Name
varchar(25) not null,C_Address varchar(40) not
null,C_NationKey int(38) not null,C_Phone varchar(15) not
null,C_AcctBal float (12,2) not null,C_MKTSegment varchar
(10) not null,C_Comment varchar(117) not null, primary
key(C_CustKey), foreign key(C_NationKey) references
Nation(N_NationKey));
```

7) Orders Table

```
create table Orders(O_OrderKey int(38) not null,O_CustKey
int(38) not null,O_OrderStatus varchar(1) not
null,O_TotalPrice float(12,2) not null,O_OrderDate date
not null,O_OrderPriority varchar(15) not null,O_Clerk
varchar(15) not null,O_ShipPriority int(38) not
null,O_Comment varchar(79) not null , primary
key(O_OrderKey), foreign key(O_CustKey) references
Customer(C_CustKey));
```

8) LineItem Table

```
create table LineItem(L_OrderKey int(38) not
null,L_PartKey int(38) not null,L_SuppKey int(38) not
null,L_LineNumber int(38) not null,L_Quantity float(12,2)
not null,L_ExtendedPrice float(12,2) not null ,L_Discount
float(12,2) not null,L_Tax double(12,2) not
null,L_ReturnFlag varchar(1) not null,L_LineStatus
varchar(1) not null,L_ShipDate date not null,L_CommitDate
date not null,L_ReceiptDate date not null,L_ShipInstruct
varchar(25) not null,L_ShipMode varchar(10) not
null,L_Comment varchar(44) not null);
```

```
mysql> use retail_pg;
Database changed
mysql> create table Region(R_RegionKey int(38) not null,R_Name varchar(25) not null,R_Comment varchar(152) not null, p
rimary key(R_RegionKey));
Query OK, 0 rows affected (0.06 sec)

mysql> create table Nation(N_NationKey int(38) not null,N_Name varchar(25) not null,N_RegionKey int(38) not null, N_Co
mment varchar(152) not null, primary key(N_NationKey));
Query OK, 0 rows affected (0.02 sec)

mysql> create table Supplier(S_SuppKey int(38) not null,S_Name varchar(25) not null,S_Address varchar(40) not null,S_N
ationKey int(38) not null,S_Phone varchar(15) not null,s_AcctBal float(12,2) not null,S_Comment varchar(101) not null,
 primary key(S_SuppKey), foreign key(S_NationKey) references Nation(N_NationKey));
Query OK, 0 rows affected (0.02 sec)

mysql> create table Customer(C_CustKey int(38) not null,C_Name varchar(25) not null,C_Address varchar(40) not null,C_N
ationKey int(38) not null,C_Phone varchar(15) not null,C_AcctBal float(12,2) not null,C_MKTSegment varchar(10) not nul
l,C_Comment varchar(117) not null, primary key(C_CustKey), foreign key(C_NationKey) references Nation(N_NationKey));
Query OK, 0 rows affected (0.02 sec)

mysql> create table Part(P_PartKey int(38) not null,P_Name varchar(55) not null, P_MFGR varchar(25) not null ,P_Brand
varchar(10) not null,P_Type varchar(25) not null,P_Size int(38) not null,P_Container varchar(10) not null,P_RetailPric
e float(12,2) not null,P_Comment varchar(23) not null, primary key(P_PartKey));
Query OK, 0 rows affected (0.01 sec)

mysql> create table Orders(O_OrderKey int(38) not null,O_CustKey int(38) not null,O_OrderStatus varchar(1) not null,O_
TotalPrice float(12,2) not null,O_OrderDate date not null,O_OrderPriority varchar(15) not null,O_Clerk varchar(15) not
 null,O_ShipPriority int(38) not null,O_Comment varchar(79) not null , primary key(O_OrderKey), foreign key(O_CustKey)
 references Customer(C_CustKey));
Query OK, 0 rows affected (0.03 sec)

mysql> create table LineItem(L_OrderKey int(38) not null,L_PartKey int(38) not null,L_SuppKey int(38) not null,L_LineN
umber int(38) not null,L_Quantity float(12,2) not null,L_ExtendedPrice float(12,2) not null ,L_Discount float(12,2) no
t null,L_Tax double(12,2) not null,L_ReturnFlag varchar(1) not null,L_LineStatus varchar(1) not null,L_ShipDate date n
ot null,L_CommitDate date not null,L_ReceiptDate date not null,L_ShipInstruct varchar(25) not null,L_ShipMode varchar(
10) not null,L_Comment varchar(44) not null, primary key(L_LineNumber), foreign key(L_OrderKey) references Orders(O_Or
derKey), foreign key(L_PartKey) references Part(P_PartKey), foreign key(L_SuppKey) references Supplier(S_SuppKey));
Query OK, 0 rows affected (0.04 sec)

mysql> create table Partsupp(PS_Partkey int(38) not null, PS_Suppkey int(38) not null, PS_Availability int(38) not nul
l, PS_Supplycost double(12,2) not null, PS_Comment varchar(199) not null, foreign key(PS_PartKey) references Part(P_Pa
rtKey), foreign key(PS_SuppKey) references Supplier(S_SuppKey));
Query OK, 0 rows affected (0.02 sec)
```

Step3: Load Data from files to the created tables

1) Region Table

```
load data local infile
'/home/ak/PG/semi/csv/region_0_0_0.csv' into table Region
fields terminated by ';' enclosed by '"' lines terminated
by '\n' ignore 1 lines;
```

2) Nation Table

```
load data local infile
'/home/ak/PG/semi/csv/nation_0_0_0.csv' into table Nation
fields terminated by ';' enclosed by '"' lines terminated
by '\n' ignore 1 lines;
```

3) Supplier Table

```
load data local infile
'/home/ak/PG/semi/csv/supplier_0_0_0.csv' into table
Supplier fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
```

4) Part Table

```
load data local infile
'/home/ak/PG/semi/csv/part_0_0_0.csv' into table Part
fields terminated by ';' enclosed by '"' lines terminated
by '\n' ignore 1 lines;
```

5) Partsupp Table

```
load data local infile
'/home/ak/PG/semi/csv/partsupp_0_0_0.csv' into table
Partsupp fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
```

6) Customer Table

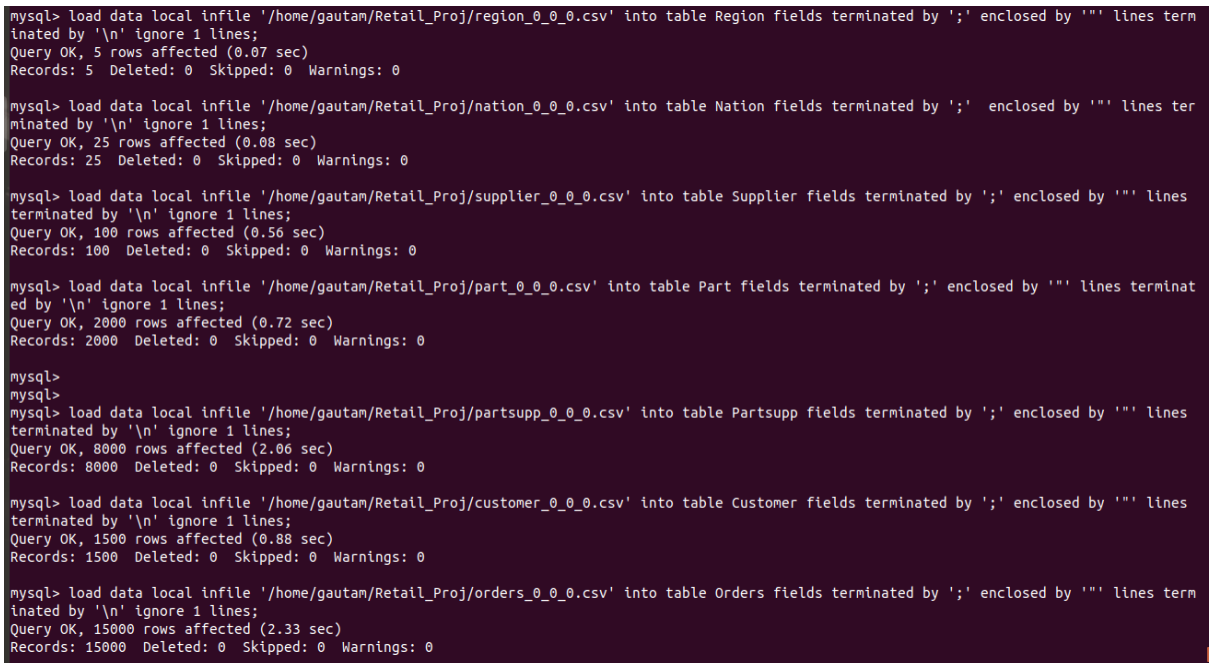
```
load data local infile
'/home/ak/PG/semi/csv/customer_0_0_0.csv' into table
Customer fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
```

7) Orders Table

```
load data local infile
'/home/ak/PG/semi/csv/orders_0_0_0.csv' into table Orders
fields terminated by ';' enclosed by '"' lines terminated
by '\n' ignore 1 lines;
```

8) LineItem Table

```
load data local infile
'/home/ak/PG/semi/csv/lineitem_0_0_0.csv' into table
LineItem fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
```



```
mysql> load data local infile '/home/gautam/Retail_Proj/region_0_0_0.csv' into table Region fields terminated by ';' enclosed by '"' lines term
inated by '\n' ignore 1 lines;
Query OK, 5 rows affected (0.07 sec)
Records: 5  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/home/gautam/Retail_Proj/nation_0_0_0.csv' into table Nation fields terminated by ';'  enclosed by '"' lines ter
minated by '\n' ignore 1 lines;
Query OK, 25 rows affected (0.08 sec)
Records: 25  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/home/gautam/Retail_Proj/supplier_0_0_0.csv' into table Supplier fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
Query OK, 100 rows affected (0.56 sec)
Records: 100  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/home/gautam/Retail_Proj/part_0_0_0.csv' into table Part fields terminated by ';' enclosed by '"' lines terminat
ed by '\n' ignore 1 lines;
Query OK, 2000 rows affected (0.72 sec)
Records: 2000  Deleted: 0  Skipped: 0  Warnings: 0

mysql>
mysql>
mysql> load data local infile '/home/gautam/Retail_Proj/partsupp_0_0_0.csv' into table Partsupp fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
Query OK, 8000 rows affected (2.06 sec)
Records: 8000  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/home/gautam/Retail_Proj/customer_0_0_0.csv' into table Customer fields terminated by ';' enclosed by '"' lines
terminated by '\n' ignore 1 lines;
Query OK, 1500 rows affected (0.88 sec)
Records: 1500  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile '/home/gautam/Retail_Proj/orders_0_0_0.csv' into table Orders fields terminated by ';' enclosed by '"' lines term
inated by '\n' ignore 1 lines;
Query OK, 15000 rows affected (2.33 sec)
Records: 15000  Deleted: 0  Skipped: 0  Warnings: 0
```

1.2 Loading data from .csv files to HDFS

Storing 2 files in HDFS to create an external table on top of the data.

1.2.1: Create a folder in HDFS

```
hdfs dfs -mkdir /retail_datasets
hdfs dfs -mkdir /retail_datasets/nation
hdfs dfs -mkdir /retail_datasets/supplier
```

1.2.2: Load Files to the created folder

1) Nation File

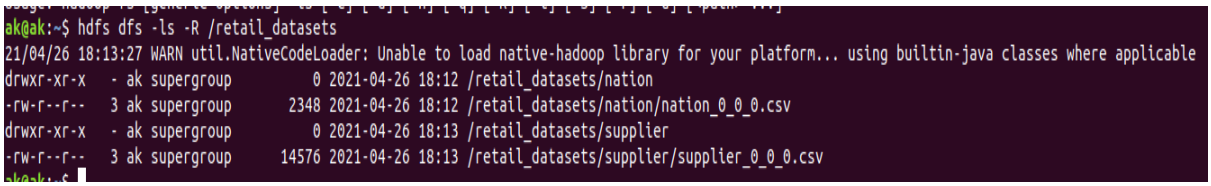
```
hdfs dfs -copyFromLocal
/home/ak/project_gladiator/nation_0_0_0.csv
/retail_datasets/nation
```

2) Supplier File

```
hdfs dfs -put
/home/ak/project_gladiator/supplier_0_0_0.csv
/retail_datasets/supplier
```

1.2.3: Verify that files are successfully loaded to the folder in HDFS

```
hdfs dfs -ls -R /retail_datasets
```



1.3 Loading data from MySQL to Hive using Sqoop

Creating managed tables and external tables on hive in order to store data.

Managed table: Managed tables are Hive owned tables where the entire lifecycle of the tables data are managed and controlled by Hive. To create Managed tables, we just use the simple CREATE Statement. When we load a data into a Managed table, it is moved into HIVE warehouse directory. If the table is later dropped, the table including its metadata and its data is deleted, which means the data no longer exists anywhere.

External table: External tables are tables where Hive has loose coupling with the data. The location of the external data is specified at table creation time and also uses the key word EXTERNAL in CREATE STATEMENT. The data is not moved to HIVE warehouse directory, and it is actually saved in an external location, therefore when you drop the external_table, HIVE will leave the data untouched and only delete the metadata.

1.3.1: Create database in hive

create database retail_pg;

```
hive> create database retail_pg;
OK
Time taken: 9.102 seconds
```

1.3.2: Create managed tables in hive

#Note: The preferred option while loading data would be to enforce a custom schema, this ensures that the data types are consistent and avoids any unexpected behavior.

1) Region Table

```
create table Region(R_RegionKey int,R_Name
string,R_Comment string);
```

2) Customer Table

```
create table Customer(C_CustKey int,C_Name
string,C_Address string,C_NationKey int,C_Phone
string,C_AcctBal double,C_MKTSegment string,C_Comment
string);
```

3) Part Table

```
create table Part(P_PartKey int,P_Name string, P_MFGR
string,P_Brand string,P_Type string,P_Size
int,P_Container string,P_RetailPrice double,P_Comment
string);
```

4) Orders Table

```
create table Orders(O_OrderKey int,O_CustKey
int,O_OrderStatus string,O_TotalPrice double,O_OrderDate
string,O_OrderPriority string,O_Clerk
string,O_ShipPriority int,O_Comment string);
```

5) LineItem Table

```
create table LineItem(L_OrderKey int,L_PartKey
int,L_SuppKey int,L_LineNumber int,L_Quantity
double,L_ExtendedPrice double,L_Discount double,L_Tax
double,L_ReturnFlag string,L_LineStatus string,L_ShipDate
string,L_CommitDate string,L_ReceiptDate
string,L_ShipInstruct string,L_ShipMode string,L_Comment
string);
```

6) Partsupp Table

```
create table Partsupp(PS_Partkey int, PS_Suppkey int,
PS_Availability int, PS_Supplycost double, PS_Comment
string);
```

```
hive> create table Region(R_RegionKey int,R_Name string,R_Comment string);
OK
Time taken: 0.884 seconds
hive> create table Customer(C_CustKey int,C_Name string,C_Address string,C_NationKey int,C_Phone stri
ng,C_AcctBal double,C_MKTSegment string,C_Comment string);
OK
Time taken: 0.115 seconds
hive> create table Part(P_PartKey int,P_Name string, P_MFGR string,P_Brand string,P_Type string,P_Siz
e int,P_Container string,P_RetailPrice double,P_Comment string);
OK
Time taken: 0.148 seconds
hive> create table Orders(O_OrderKey int,O_CustKey int,O_OrderStatus string,O_TotalPrice double,O_Ord
erDate string,O_OrderPriority string,O_Clerk string,O_ShipPriority int,O_Comment string);
OK
Time taken: 0.158 seconds
hive> create table LineItem(L_OrderKey int,L_PartKey int,L_SuppKey int,L_LineNumber int,L_Quantity do
uble,L_ExtendedPrice double,L_Discount double,L_Tax double,L_ReturnFlag string,L_LineStatus string,L_
ShipDate string,L_CommitDate string,L_ReceiptDate string,L_ShipInstruct string,L_ShipMode string,L_Co
mment string);
OK
Time taken: 0.163 seconds
hive> create table Partsupp(PS_Partkey int, PS_Suppkey int, PS_Availability int, PS_Supplycost double
, PS_Comment string);
OK
Time taken: 0.267 seconds
```

1.3.3: Import from MySQL table to Hive managed table via Sqoop

1) Customer Table

```
sqoop import --connect
jdbc:mysql://localhost:3306/retail_db?useSSL=false -
username hiveuser -password hivepassword --table CUSTOMER
--hive-import --hive-table retail.CUSTOMER  --fields-
terminated-by "\001" -m 2;
```

2) LineItem Table

```
sqoop import --connect
jdbc:mysql://localhost:3306/retail_db?useSSL=false -
username hiveuser -password hivepassword --table LINEITEM
--hive-import --hive-table retail.LINEITEM  --fields-
terminated-by "\001" --split-by L_LINENUMBER;
```

3) Orders Table

```
sqoop-import --connect jdbc:mysql://localhost/retail_db -
username hiveuser  -password hivepassword  --table Orders
-hive-table retail_proj.ORDERS  -hive-import --fields-
terminated-by  '\001';
```

4) Part Table

```
sqoop-import --connect jdbc:mysql://localhost/retail_db -
username hiveuser  -password hivepassword  --table Part -
hive-table retail_proj.PART  -hive-import --fields-
terminated-by  '\001' --split-by P_PARTKEY -m 3;
```

5) Partsupp Table

```
sqoop-import --connect jdbc:mysql://localhost/retail_db -
username hiveuser  -password hivepassword  --table
Partsupp -hive-table retail_proj.PARTSUPP  -hive-import -
-fields-terminated-by  '\001' --split-by PS_PARTKEY -m 3;
```

6) Region Table

```
sqoop-import --connect jdbc:mysql://localhost/retail_db -
username hiveuser  -password hivepassword  --table region
-hive-table retail_proj.REGION  -hive-import --fields-
terminated-by  '\001' -m 1;
```

1.3.4: Create external tables in hive on top of data stored in HDFS

1) Nation Table

```
CREATE EXTERNAL TABLE Nation(N_NationKey int, N_Name string, N_RegionKey int, N_Comment string) row format delimited fields terminated BY ';' stored AS textfile location '/retail_datasets/nation/' tblproperties ("skip.header.line.count"="1");
```

```
hive> CREATE EXTERNAL TABLE Nation(N_NationKey int, N_Name string, N_RegionKey int, N_Comment string)
 row format delimited fields terminated BY ';' stored AS textfile location '/retail_datasets/nation/'
 tblproperties ("skip.header.line.count"="1");
OK
Time taken: 0.097 seconds
hive> select * from nation;
OK
0       "ALGERIA"       0       " haggle. carefully final deposits detect slyly agai"
1       "ARGENTINA"     1       "al foxes promise slyly according to the regular accounts. bold reque
sts alon"
2       "BRAZIL"        1       "y alongside of the pending deposits. carefully special packages are
about the ironic forges. slyly special "
3       "CANADA"        1       "eas hang ironic, silent packages. slyly regular packages are furious
ly over the tithes. fluffily bold"
```

2) Supplier Table

```
CREATE EXTERNAL TABLE SUPPLIER(S_SUPPKEY int, S_NAME string, S_ADDRESS string, S_NATIONKEY int, S_PHONE string, S_ACCTBAL double, S_COMMENT string)   ROW FORMAT DELIMITED FIELDS TERMINATED BY ';' STORED AS TEXTFILE LOCATION '/aman/retail_proj/SUPPLIER/' tblproperties ("skip.header.line.count"="1");
```

```
hive> CREATE EXTERNAL TABLE SUPPLIER(S_SUPPKEY int, S_NAME string, S_ADDRESS string, S_NATIONKEY int,
 S_PHONE string, S_ACCTBAL double, S_COMMENT string)   ROW FORMAT DELIMITED FIELDS TERMINATED BY ';'
STORED AS TEXTFILE   LOCATION '/retail_datasets/supplier/' tblproperties ("skip.header.line.count"="1
");
OK
Time taken: 0.102 seconds
hive> select * from supplier;
OK


1       "Supplier#000000001"    " N kD4on9OM Ipw3,gf0JBoQDd7tgrzrddZ"   17      "27-918-335-1736"
       5755.94 "each slyly above the careful"
2       "Supplier#000000002"    "89eJ5ksX3ImxJQBvxObC," 5       "15-679-861-2259"       4032.68 " sly
ly bold instructions. idle dependen"
3       "Supplier#000000003"    "q1,G3Pj6OjIuUYfUoH18BFTKP5aU9bEV3"     1       "11-383-516-1199"
       4192.4  "blithely silent requests after the express dependencies are sl"
```

Task 2: Checking for Bad Records

==========

Bad or Dirty Data refers to information that can be erroneous, misleading, and without general formatting.
Bad records can be:

- Missing Data: Empty fields that should contain data.
- Wrong or inaccurate data: Information that has not been entered correctly or maintained.
- Inappropriate data: Data that's been entered in the wrong field.
- Non-conforming data: Data that hasn't been normalized as per the system of records.
- Duplicate data: A single Account, Contact, Lead, etc. that occupies more than one record in the database.
- Poor data entry: Misspells, typos, transpositions, and variations in spelling, naming or formatting.

Missing Data is checked below. Other parameters are checked by verifying distinct rows of each column of a table.

2.1 In CSV files through PySpark

Using count of rows in the dataframe before dropping null records and after dropping null records ensured that there are no null records in our data file.

```
>>> orderDf = spark.read.option("sep",";").option("header","true").csv("file:///
/home/ak/project_gladiator/orders_0_0_0.csv", sep=";", schema=orderschema)
>>> orderDf.count()
15000
>>> orderDf.na.drop().count()
15000
```

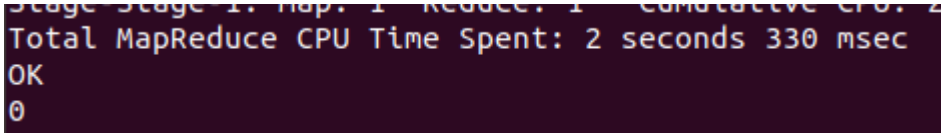
2.2 In MySQL through MySQL CLI

Using 'IS NULL' functionality of MySQL and checking for every string datatype column that there are no empty strings ensured that there are no null records in our MySQL tables

```
mysql> select count(*) from Nation where N_NationKey is NULL or N_Name is NULL o
r N_Name = ' ' or N_RegionKey is NULL or N_Comment is NULL or N_Comment = ' ';
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.01 sec)
```

2.3 In Hive through Hive CLI

Using ‘IS NULL’ functionality of Hive and checking for every string datatype column that there are no empty strings ensured that there are no null records in our Hive tables.



Validation table:

Tables	CSV files		MySQL		Hive	
	Rows	Columns	Rows	Columns	Rows	Columns
Region	5	3	5	3	5	3
Nation	25	4	25	4	25	4
Supplier	100	7	100	7	100	7
Customer	1500	8	1500	8	1500	8
Part	2000	9	2000	9	2000	9
Orders	15000	9	15000	9	15000	9
Lineitem	60175	16	60175	16	60175	16
Partsupp	8000	5	8000	5	8000	5

Task 3: Pyspark dataframe Read/Write Operations
==========

3.1 Using Hive Tables

3.1.1 Through Spark SQL

```
partDf = spark.sql("select * from retail_db.part")

partsuppDf = spark.sql("select * from
retail_db.partsupp")
```

3.1.2 Through DSL

```
nationDf=spark.read.table("retail_proj.nation")

regionDf=spark.read.table("retail_proj.region")
```

3.2 Using MySQL Tables

Read Modes: Often while reading data from external sources we encounter corrupt data, read modes instruct Spark to handle corrupt data in a specific way.
There are 3 typical read modes and the default read mode is permissive.

- permissive — All fields are set to null and corrupted records are placed in a string column called _corrupt_record
- dropMalformed — Drops all rows containing corrupt records.
- failFast — Fails when corrupt records are encountered.

a)Customer Dataframe

```
customerDf =
spark.read.format("jdbc").option("url",
"jdbc:mysql://localhost:3306/retail_db?useSSL=false").opt
ion("driver", "com.mysql.jdbc.Driver").option("dbtable",
"Customer").option("mode","DROPMALFORMED").option("user",
"hiveuser").option("password", "hivepassword").load()
```

b)Lineitem Dataframe

```
lineitemDf =
spspark.read.format("jdbc").option("url",
"jdbc:mysql://localhost:3306/retail_db?useSSL=false").opt
ion("driver", "com.mysql.jdbc.Driver").option("dbtable",
"Lineitem").option("mode","DROPMALFORMED").option("user",
"hiveuser").option("password", "hivepassword").load()
```

3.3 Using CSV files and imposing user defined schema

There are various parameters that are specified along with spark.read.

This includes:

- **Format** — specifies the file format as in CSV, JSON, or parquet. The default is parquet.
- **Option** — a set of key-value configurations to parameterize how to read data.
- **Schema** — optional one used to specify if you would like to infer the schema from the data source.

a) Order Dataframe

```
from pyspark.sql.types import StructType,StructField,
StringType,IntegerType,TimestampType,DoubleType
orderschema = StructType([
    StructField("O_ORDERKEY", IntegerType()),
    StructField("O_CUSTKEY", IntegerType()),
    StructField("O_ORDERSTATUS", StringType()),
    StructField("O_TOTALPRICE", DoubleType()),
    StructField("O_ORDERDATE", TimestampType()),
    StructField("O_ORDERPRIORITY", StringType()),
    StructField("O_CLERK", StringType()),
    StructField("O_SHIPPRIORITY", IntegerType()),
    StructField("O_COMMENT", StringType())
])
orderDf =
spark.read.option("sep",";").option("header","true").sche
ma(orderschema).csv("file:////home/ak/PG/semi/csv/ORDERS_
0_0_0.csv", sep=";")
```

b) Supplier Dataframe

```
suppschema = StructType([
    StructField("S_SUPPKEY", IntegerType()),
    StructField("S_NAME", StringType()),
    StructField("S_ADDRESS", StringType()),
    StructField("S_NATIONKEY", IntegerType()),
    StructField("S_PHONE", StringType()),
    StructField("S_ACCTBAL", DoubleType()),
    StructField("S_COMMENT", StringType())
])
suppDf =
spark.read.option("header","true").csv("file:///home/ak/P
G/semi/csv/SUPPLIER_0_0_0.csv", sep=";",
schema=suppschema
```

3.4 Typecasting and saving dataframe as a Table in Hive

In order to write dataframe to either table in hive or file, there are different save modes which can simplify our task. Save modes specify what will happen if Spark finds data already at the destination.
There are 4 typical save modes and the default mode is errorIfExists

- Append → appends output data to files that already exist.
- Overwrite → completely overwrites any data present at the destination.
- errorIfExists → throws an error if data already exists at the destination
- Ignore → if data exists do nothing with the dataFrame.

⇒ Typecasting with save mode as “append”

new_orderDf = orderDf.withColumn("O_ORDERDATE",orderDf.O_ORDERDATE.cast("date"))

new_orderDf.write.mode("append").saveAsTable("retail_pg.orders")

```
>>> orderDf.printSchema()
root
 |-- O_ORDERKEY: integer (nullable = true)
 |-- O_CUSTKEY: integer (nullable = true)
 |-- O_ORDERSTATUS: string (nullable = true)
 |-- O_TOTALPRICE: double (nullable = true)
 |-- O_ORDERDATE: timestamp (nullable = true)
 |-- O_ORDERPRIORITY: string (nullable = true)
 |-- O_CLERK: string (nullable = true)
 |-- O_SHIPPRIORITY: integer (nullable = true)
 |-- O_COMMENT: string (nullable = true)
```

```
>>> new_orderDf = ordersDf.withColumn("O_ORDERDATE",ordersDf.O_ORDERDATE.cast("date"))
>>> new_orderDf.write.mode("append").format("hive").saveAsTable("retail_pgl.orders")
21/05/06 18:51:05 WARN metastore.HiveMetaStore: Location: file:/home/ak/spark-warehouse/retail_pgl.db
/orders specified for non-external table:orders
>>> spark.sql("desc retail_pgl.orders").show()
+---------------+---------+-------+
|       col_name|data_type|comment|
+---------------+---------+-------+
|     O_ORDERKEY|      int|   null|
|      O_CUSTKEY|      int|   null|
|  O_ORDERSTATUS|   string|   null|
|   O_TOTALPRICE|   double|   null|
|    O_ORDERDATE|     date|   null|
|O_ORDERPRIORITY|   string|   null|
|        O_CLERK|   string|   null|
| O_SHIPPRIORITY|      int|   null|
|      O_COMMENT|   string|   null|
+---------------+---------+-------+
```

Task 4: Partitioning and Bucketing along with different file formats and codecs

==========

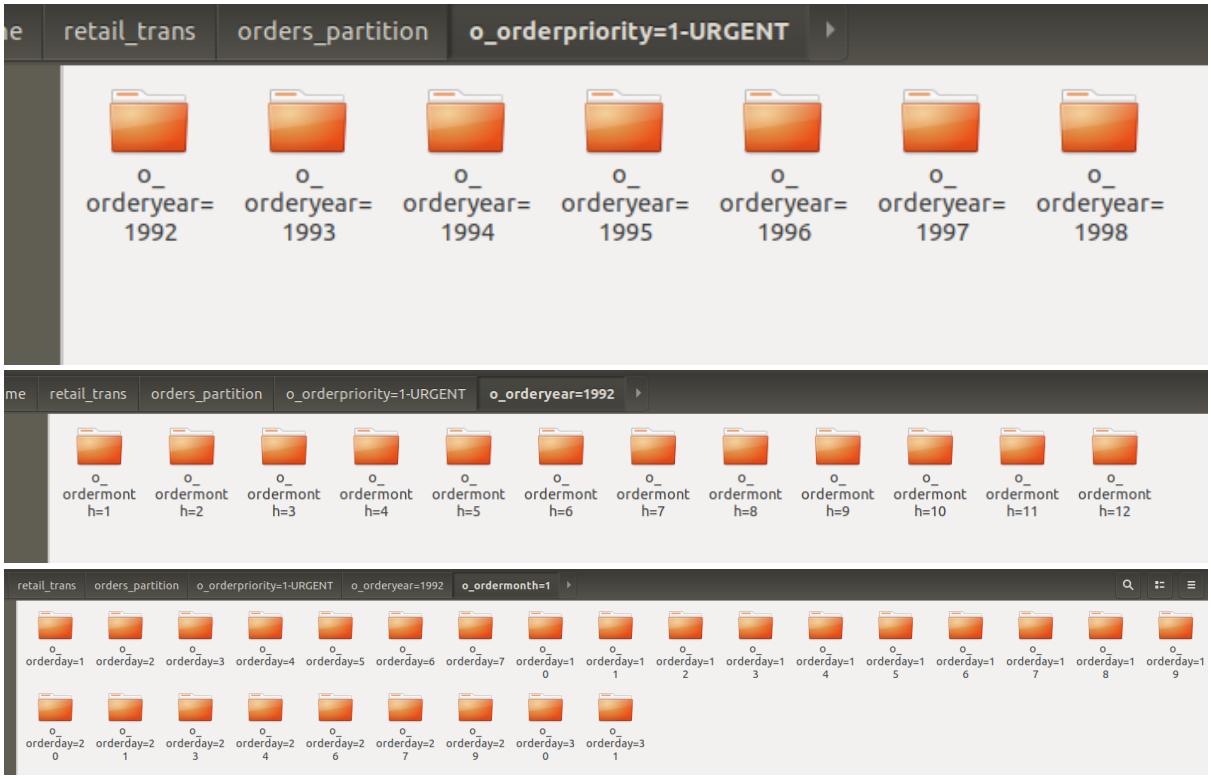
Partitioning: With partition, we can group the same kind of data together. It is used for distributing the load horizontally.

Bucketing: In bucketing, the partitions can be subdivided into buckets based on the hash function of a column. It gives extra structure to the data which can be used for more efficient queries.

File formats and Codec: Parquet is chosen as our file format to store data. Even though the ORC file format has the best compression rate, the decompression time it takes is much higher than the parquet file format. Also, the codec supported for parquet is brotli, uncompressed, lz4, gzip, lzo, snappy, none and zstd. We tried for the native libraries, out of which gzip gave us better compression. So, parquet as a file format and gzip as the codec was chosen as the best solution to store our data.

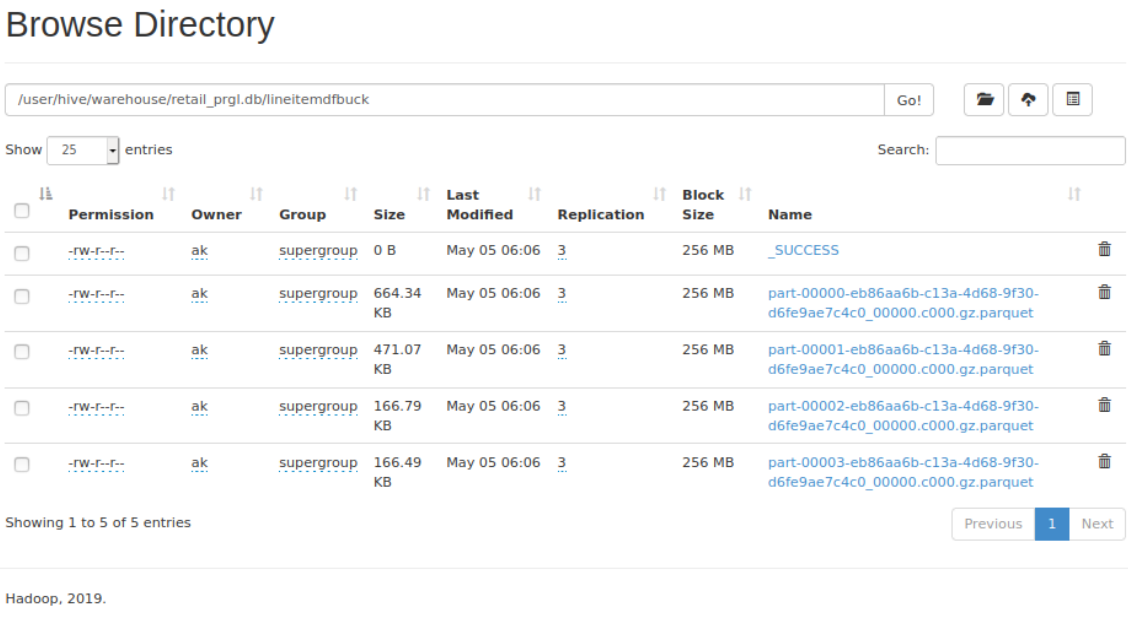
4.1 Partitioning using PySpark

```
spark.sql("SELECT O_ORDERKEY, O_CUSTKEY, O_ORDERSTATUS,
O_TOTALPRICE, O_CLERK, O_SHIPPRIORITY, O_COMMENT,
O_ORDERPRIORITY, year(O_ORDERDATE) as O_ORDERYEAR,
month(O_ORDERDATE) as O_ORDERMONTH, day(O_ORDERDATE) as
O_ORDERDAY FROM
RETAIL_PG.ORDERS").write.partitionBy("o_orderpriority","o_order
year","o_ordermonth","o_orderday").mode("overwrite").option("co
dec","gzip").parquet("file:///home/ak/retail_trans/orders_parti
tion")
```



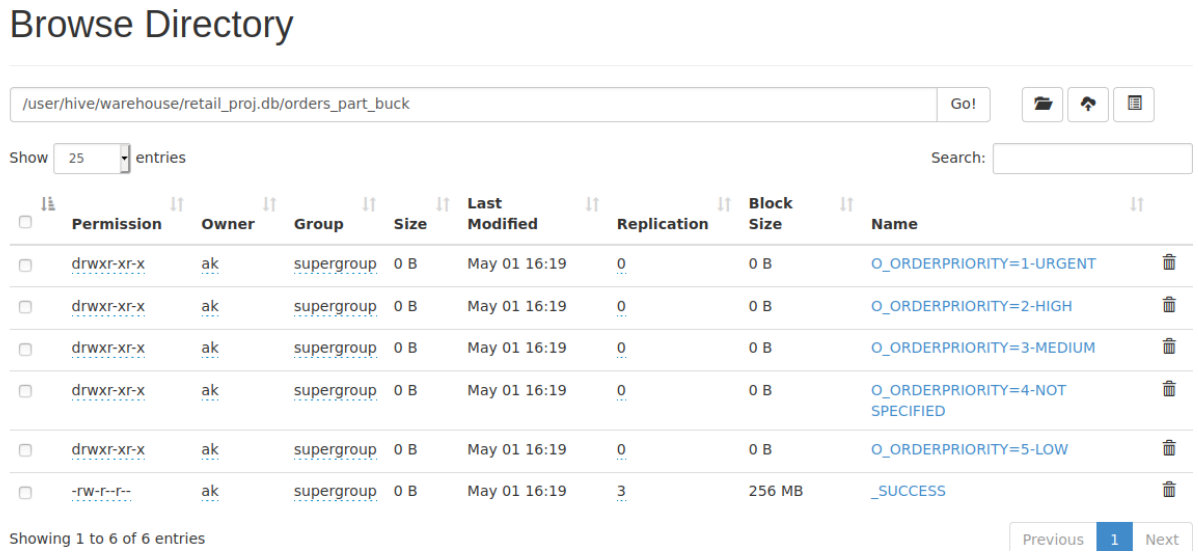
4.2 Bucketing using PySpark

```
lineitemDf.write.bucketBy(4,"l_shipmode").format("parquet").option("codec","gzip").mode("append").saveAsTable("retail_prgl.lineitemDfbuck")
```



4.3 Partitioning and Bucketing using PySpark

```
ordersDf.write.partitionBy("o_orderpriority").bucketBy(4,"o_orderstatus").format("parquet").option("codec","gzip").mode("append").saveAsTable("retail_proj.orders_part_buck3")
```



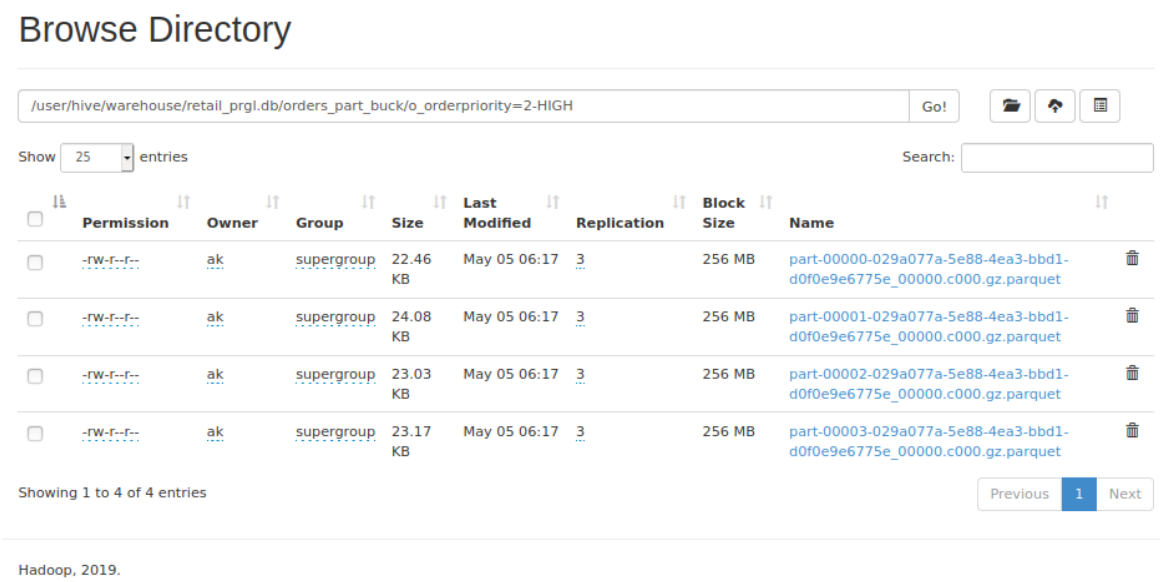
Browse Directory

/user/hive/warehouse/retail_proj.db/orders_part_buck Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	ak	supergroup	0 B	May 01 16:19	0	0 B	O_ORDERPRIORITY=1-URGENT
drwxr-xr-x	ak	supergroup	0 B	May 01 16:19	0	0 B	O_ORDERPRIORITY=2-HIGH
drwxr-xr-x	ak	supergroup	0 B	May 01 16:19	0	0 B	O_ORDERPRIORITY=3-MEDIUM
drwxr-xr-x	ak	supergroup	0 B	May 01 16:19	0	0 B	O_ORDERPRIORITY=4-NOT SPECIFIED
drwxr-xr-x	ak	supergroup	0 B	May 01 16:19	0	0 B	O_ORDERPRIORITY=5-LOW
-rw-r--r--	ak	supergroup	0 B	May 01 16:19	3	256 MB	_SUCCESS

Showing 1 to 6 of 6 entries Previous 1 Next



Browse Directory

/user/hive/warehouse/retail_prgl.db/orders_part_buck/o_orderpriority=2-HIGH Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	ak	supergroup	22.46 KB	May 05 06:17	3	256 MB	part-00000-029a077a-5e88-4ea3-bbd1-d0f0e9e6775e_00000.c000.gz.parquet
-rw-r--r--	ak	supergroup	24.08 KB	May 05 06:17	3	256 MB	part-00001-029a077a-5e88-4ea3-bbd1-d0f0e9e6775e_00000.c000.gz.parquet
-rw-r--r--	ak	supergroup	23.03 KB	May 05 06:17	3	256 MB	part-00002-029a077a-5e88-4ea3-bbd1-d0f0e9e6775e_00000.c000.gz.parquet
-rw-r--r--	ak	supergroup	23.17 KB	May 05 06:17	3	256 MB	part-00003-029a077a-5e88-4ea3-bbd1-d0f0e9e6775e_00000.c000.gz.parquet

Showing 1 to 4 of 4 entries Previous 1 Next

Hadoop, 2019.

Task 5: DML operations using Delta Lake

==========

Delta Lake brings support for ACID transactions to data lakes, scalable metadata handling, schema evolution, data versioning, updates and deletes, for example. All the data is stored in the Apache Parquet format and users can enforce schemas (and change them with relative ease if necessary).

```
pyspark --packages io.delta:delta-core_2.11:0.4.0
```

5.1 Write to Delta Location

```
customerDf =
spark.read.option("sep",";").option("header","true").opti
on("inferSchema","true").csv("file:////home/ak/PG/comma/C
USTOMER_0_0_0.csv")

customerDf.write.format("delta").save("file:///home/ak/PG
/delta/customer/")
```

5.2 Creating Delta Table

```
from pyspark.sql.functions import *

from delta.tables import *

custDeltaTable=DeltaTable.forPath(spark,"file:///home/ak/
PG/delta/customer/")

custDeltaTable.toDF.show()
```

```
>>>
>>> custDeltaTable=DeltaTable.forPath(spark,"file:///home/ak/PG/delta/customer/")
>>> custDeltaTable.toDF().show()
+---------+------------------+--------------------+-----------+--------------+---------+------------+--------------------+
|C_CUSTKEY|            C_NAME|           C_ADDRESS|C_NATIONKEY|       C_PHONE|C_ACCTBAL|C_MKTSEGMENT|           C_COMMENT|
+---------+------------------+--------------------+-----------+--------------+---------+------------+--------------------+
|        1|Customer#000000001|    IVhzIApeRb ot,c,E|         15|25-989-741-2988|   711.56|    BUILDING|to the even, regu...|
|        2|Customer#000000002|XSTf4,NCwDVaWNe6t...|         13|23-768-687-3665|   121.65|  AUTOMOBILE|l accounts. blith...|
|        3|Customer#000000003|         MG9kdTD2WBHm|          1|11-719-748-3364|  7498.12|  AUTOMOBILE| deposits eat sly...|
|        4|Customer#000000004|          XxVSJsLAGtn|          4|14-128-190-5944|  2866.83|   MACHINERY| requests. final,...|
|        5|Customer#000000005|KvpyuHCplrB84WgAi...|          3|13-750-942-6364|   794.47|   HOUSEHOLD|n accounts will h...|
|        6|Customer#000000006|sKZz0CsnMD7mp4Xd0...|         20|30-114-968-4951|  7638.57|  AUTOMOBILE|tions. even depos...|
|        7|Customer#000000007|TcGe5gaZNgVePxU5k...|         18|28-190-982-9759|  9561.95|  AUTOMOBILE|ainst the ironic,...|
|        8|Customer#000000008|I0B10bB0AymmC, 0P...|         17|27-147-574-9335|  6819.74|    BUILDING|among the slyly r...|
|        9|Customer#000000009|xKiAFTjUsCuxfeleN...|          8|18-338-906-3675|  8324.07|   FURNITURE|r theodolites acc...|
|       10|Customer#000000010|6LrEaV6KR6PLVcgl2...|          5|15-741-346-9870|  2753.54|   HOUSEHOLD|es regular deposi...|
|       11|Customer#000000011|PkWS 3HlXqwTuzrKg...|         23|33-464-151-3439|  -272.6|    BUILDING|ckages. requests ...|
|       12|Customer#000000012|       9PWKuhzT4Zr1Q|         13|23-791-276-1263|  3396.49|   HOUSEHOLD| to the carefully...|
|       13|Customer#000000013|nsXQu0oVjD7PM659u...|          3|13-761-547-5974|  3857.34|    BUILDING|ounts sleep caref...|
|       14|Customer#000000014|     KXkletMlL2JQEA |          1|11-845-129-3851|   5266.3|   FURNITURE|, ironic packages...|
|       15|Customer#000000015|YtWggXoOLdwdo7b0y...|         23|33-687-542-7601|  2788.52|   HOUSEHOLD| platelets. regul...|
|       16|Customer#000000016| cYiaeMLZSMAOQ2 d0W,|         10|20-781-609-3107|  4681.03|   FURNITURE|kly silent courts...|
|       17|Customer#000000017|izrh 6jdqtp2eqdtb...|          2|12-970-682-3487|     6.34|  AUTOMOBILE|packages wake! bl...|
|       18|Customer#000000018|3txGO AiuFux3zT0Z...|          6|16-155-215-1315|  5494.43|    BUILDING|s sleep. carefull...|
|       19|Customer#000000019|uc,3bHIx84H,wdrmL...|         18|28-396-526-5053|  8914.71|   HOUSEHOLD| nag. furiously c...|
|       20|Customer#000000020|       JrPk8Pqplj4Ne|         22|32-957-234-8742|   7603.4|   FURNITURE|g alongside of th...|
+---------+------------------+--------------------+-----------+--------------+---------+------------+--------------------+
only showing top 20 rows
```

5.3 Performing Delete operation

```
custDeltaTable.delete("C_CUSTKEY<=20")


custDeltaTable.toDF.show()
```

```
>>>
>>> custDeltaTable.delete("C_CUSTKEY<=20")
>>> custDeltaTable.toDF().show()
+---------+------------------+--------------------+-----------+--------------+---------+------------+--------------------+
|C_CUSTKEY|            C_NAME|           C_ADDRESS|C_NATIONKEY|       C_PHONE|C_ACCTBAL|C_MKTSEGMENT|           C_COMMENT|
+---------+------------------+--------------------+-----------+--------------+---------+------------+--------------------+
|       21|Customer#000000021|       XYmVpr9yAHDEn|          8|18-902-614-8344|  1428.25|   MACHINERY| quickly final ac...|
|       22|Customer#000000022|QI6p41,FNs5k7RZoC...|          3|13-806-545-9701|   591.98|   MACHINERY|s nod furiously a...|
|       23|Customer#000000023|OdY W13N7Be3OC5Mp...|          3|13-312-472-8245|  3332.02|   HOUSEHOLD|deposits. special...|
|       24|Customer#000000024|HXAFgIAyjxtdqwimt...|         13|23-127-851-8031|  9255.67|   MACHINERY|into beans. fluff...|
|       25|Customer#000000025|Hp8GyFQgGHFYSilH5...|         12|22-603-468-3533|   7133.7|   FURNITURE|y. accounts sleep...|
|       26|Customer#000000026|     8ljrc5ZeMl7UciP|         22|32-363-455-4837|  5182.05|  AUTOMOBILE|c requests use fu...|
|       27|Customer#000000027|   IS8GIyxpBrLpMT0u7|          3|13-137-193-2709|  5679.84|    BUILDING| about the carefu...|
|       28|Customer#000000028|iVyg0daQ,Tha8x2WP...|          8|18-774-241-1462|  1007.18|   FURNITURE| along the regula...|
|       29|Customer#000000029|sJ5adtfyAkCK63df2...|          0|10-773-203-7342|  7618.27|   FURNITURE|its after the car...|
|       30|Customer#000000030|nJDsELGAavU63Jl0c...|          1|11-764-165-5076|  9321.01|    BUILDING|lithely final req...|
|       31|Customer#000000031|LUACbO0viaAv6eXOA...|         23|33-197-837-7094|  5236.89|   HOUSEHOLD|s use among the b...|
|       32|Customer#000000032|jD2xZzi UmId,DCtN...|         15|25-430-914-2194|  3471.53|    BUILDING|cial ideas. final...|
|       33|Customer#000000033|qFSlMuLucBmx9xnn5...|         17|27-375-391-1280|   -78.56|  AUTOMOBILE|s. slyly regular ...|
|       34|Customer#000000034|Q6G9wZ6dnczmtOx50...|         15|25-344-968-5422|   8589.7|   HOUSEHOLD|nder against the ...|
|       35|Customer#000000035|       TEjWGE4nBzJL2|         17|27-566-888-7431|  1228.24|   HOUSEHOLD|requests. special...|
|       36|Customer#000000036|3TvCzjuPzpJ0,DdJ8...|         21|31-704-669-5769|  4987.27|    BUILDING|haggle. enticing,...|
|       37|Customer#000000037|   7EV4Pwh,3SboctTWt|          8|18-385-235-7162|  -917.75|   FURNITURE|ilent packages ar...|
|       38|Customer#000000038|a5Ee5e9568R8RLP 2ap7|         12|22-306-880-7212|  6345.11|   HOUSEHOLD|lar excuses. clos...|
|       39|Customer#000000039|nnbRg,Pvy33dfkorY...|          2|12-387-467-6509|  6264.31|  AUTOMOBILE|tions. slyly sile...|
|       40|Customer#000000040|      gOnGWAyhSV1ofv|          3|13-652-915-8939|   1335.3|    BUILDING|rges impress afte...|
+---------+------------------+--------------------+-----------+--------------+---------+------------+--------------------+
only showing top 20 rows
```

5.4 Performing Update operation

```
custDeltaTable.update(col("C_ADDRESS") ==
"XYmVpr9yAHDEn", { "C_ADDRESS": lit("B. H. Street, Pune")
} )

custDeltaTable.toDF.show()
```

```
>>>
>>> custDeltaTable.update(col("C_ADDRESS") == "XYmVpr9yAHDEn", { "C_ADDRESS": lit("B. H. Street, Pune") } )
>>> custDeltaTable.toDF().show()
+---------+----------------+------------------+----------+---------------+---------+-----------+--------------------+
|C_CUSTKEY|          C_NAME|         C_ADDRESS|C_NATIONKEY|        C_PHONE|C_ACCTBAL|C_MKTSEGMENT|           C_COMMENT|
+---------+----------------+------------------+----------+---------------+---------+-----------+--------------------+
|       21|Customer#000000021|  B. H. Street, Pune|         8|18-902-614-8344|  1428.25|   MACHINERY| quickly final ac...|
|       22|Customer#000000022|QI6p41,FNs5k7RZoC...|         3|13-806-545-9701|   591.98|   MACHINERY|s nod furiously a...|
|       23|Customer#000000023|OdY W13N7Be3OC5Mp...|         3|13-312-472-8245|  3332.02|   HOUSEHOLD|deposits. special...|
|       24|Customer#000000024|HXAFgIAyjxtdqwimt...|        13|23-127-851-8031|  9255.67|   MACHINERY|into beans. fluff...|
|       25|Customer#000000025|Hp8GyFQgGHFYSilH5...|        12|22-603-468-3533|   7133.7|   FURNITURE|y. accounts sleep...|
|       26|Customer#000000026|     8ljrc5ZeMl7UciP|        22|32-363-455-4837|  5182.05|  AUTOMOBILE|c requests use fu...|
|       27|Customer#000000027|   IS8GIyxpBrLpMT0u7|         3|13-137-193-2709|  5679.84|    BUILDING| about the carefu...|
|       28|Customer#000000028|iVyg0daQ,Tha8x2WP...|         8|18-774-241-1462|  1007.18|   FURNITURE| along the regula...|
|       29|Customer#000000029|sJ5adtfyAkCK63df2...|         0|10-773-203-7342|  7618.27|   FURNITURE|its after the car...|
|       30|Customer#000000030|nJDsELGAavU63Jl0c...|         1|11-764-165-5076|  9321.01|    BUILDING|lithely final req...|
|       31|Customer#000000031|LUACbO0viaAv6eXOA...|        23|33-197-837-7094|  5236.89|   HOUSEHOLD|s use among the b...|
|       32|Customer#000000032|jD2xZzi UmId,DCtN...|        15|25-430-914-2194|  3471.53|    BUILDING|cial ideas. final...|
|       33|Customer#000000033|qFSlMuLucBmx9xnn5...|        17|27-375-391-1280|   -78.56|  AUTOMOBILE|s. slyly regular ...|
|       34|Customer#000000034|Q6G9wZ6dnczmtOx50...|        15|25-344-968-5422|   8589.7|   HOUSEHOLD|nder against the ...|
|       35|Customer#000000035|       TEjWGE4nBzJL2|        17|27-566-888-7431|  1228.24|   HOUSEHOLD|requests. special...|
|       36|Customer#000000036|3TvCzjuPzpJ0,DdJ8...|        21|31-704-669-5769|  4987.27|    BUILDING|haggle. enticing,...|
|       37|Customer#000000037|   7EV4Pwh,3SboctTWt|         8|18-385-235-7162|  -917.75|   FURNITURE|ilent packages ar...|
|       38|Customer#000000038|a5Ee5e9568R8RLP 2ap7|        12|22-306-880-7212|  6345.11|   HOUSEHOLD|lar excuses. clos...|
|       39|Customer#000000039|nnbRg,Pvy33dfkorY...|         2|12-387-467-6509|  6264.31|  AUTOMOBILE|tions. slyly sile...|
|       40|Customer#000000040|      gOnGWAyhSV1ofv|         3|13-652-915-8939|   1335.3|    BUILDING|rges impress afte...|
+---------+----------------+------------------+----------+---------------+---------+-----------+--------------------+
only showing top 20 rows
```

Task 6 ⇒ Kafka and Spark Structured Streaming

==========

6.1 Start Zookeeper Daemons

```
./zookeeper-server-start.sh../config/zookeeper.PROPERTIES
```

6.2 Start Kafka Server

```
./kafka-server-start.sh ../config/server.PROPERTIES
```

6.3 Create Kafka Topic

```
./kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 2 --partitions 1 --topic customer
```

1) Kafka:

6.4 Copy content of the file to topic via Kafka Producer

Creating and running Customerstream.py file

```
from kafka import KafkaProducer
import json
import logging
import csv
from json import dumps
from time import sleep
from kafka.errors import KafkaError
logging.basicConfig(level=logging.INFO)
```

```
producer =
KafkaProducer(bootstrap_servers=['localhost:9092'],
                        value_serializer=lambda x:
                        dumps(x).encode('utf-8'))

with open('/home/gautam/Proj/customer_0_0_0.csv') as
file:
    reader=csv.reader(file, delimiter=';')
    for messages in reader:
     producer.send('customer',messages)
     producer.flush()
     sleep(10)
```

```
gautam@gautam-HP-Laptop-15-bs0xx:~$ cd Desktop/
gautam@gautam-HP-Laptop-15-bs0xx:~/Desktop$ python Customerstream.py
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv4 ('127.0.0.1', 9092)]>: connecting to localhost:909
2 [('127.0.0.1', 9092) IPv4]
INFO:kafka.conn:Probing node bootstrap-0 broker version
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv4 ('127.0.0.1', 9092)]>: Connection complete.
INFO:kafka.conn:Broker version identified as 2.4.0
INFO:kafka.conn:Set configuration api_version=(2, 4, 0) to skip auto check_version requests on startup
INFO:kafka.conn:<BrokerConnection node_id=0 host=gautam-HP-Laptop-15-bs0xx:9092 <connecting> [IPv4 ('127.0.1.1', 9092)]>: connecting to gautam-
HP-Laptop-15-bs0xx:9092 [('127.0.1.1', 9092) IPv4]
INFO:kafka.conn:<BrokerConnection node_id=0 host=gautam-HP-Laptop-15-bs0xx:9092 <connecting> [IPv4 ('127.0.1.1', 9092)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connected> [IPv4 ('127.0.0.1', 9092)]>: Closing connection.
```

6.5 Start Consumer

```
./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic customer --from-beginning
```

```
gautam@gautam-HP-Laptop-15-bs0xx:~/kafka_2.11-2.4.1/bin$ ./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic customer --from-
beginning
["C_CUSTKEY", "C_NAME", "C_ADDRESS", "C_NATIONKEY", "C_PHONE", "C_ACCTBAL", "C_MKTSEGMENT", "C_COMMENT"]
["1", "Customer#000000001", "IVhzIApeRb ot,c,E", "15", "25-989-741-2988", "711.56", "BUILDING", "to the even, regular platelets. regular, ironi
c epitaphs nag e"]
["2", "Customer#000000002", "XSTf4,NCwDVaWNe6tEgvwfmRchLXak", "13", "23-768-687-3665", "121.65", "AUTOMOBILE", "l accounts. blithely ironic the
odolites integrate boldly: caref"]
["3", "Customer#000000003", "MG9kdTD2WBHm", "1", "11-719-748-3364", "7498.12", "AUTOMOBILE", " deposits eat slyly ironic, even instructions. ex
press foxes detect slyly. blithely even accounts abov"]
["4", "Customer#000000004", "XxVSJsLAGtn", "4", "14-128-190-5944", "2866.83", "MACHINERY", " requests. final, regular ideas sleep final accou"]
["5", "Customer#000000005", "KvpyuHCplrB84WgAiGV6sYpZq7Tj", "3", "13-750-942-6364", "794.47", "HOUSEHOLD", "n accounts will have to unwind. fox
es cajole accor"]
["6", "Customer#000000006", "sKZz0CsnMD7mp4Xd0YrBvx,LREYKUWAh yVn", "20", "30-114-968-4951", "7638.57", "AUTOMOBILE", "tions. even deposits boo
st according to the slyly bold packages. final accounts cajole requests. furious"]
["7", "Customer#000000007", "TcGe5gaZNgVePxU5kRrvXBfkasDTea", "18", "28-190-982-9759", "9561.95", "AUTOMOBILE", "ainst the ironic, express theo
dolites. express, even pinto beans among the exp"]
["8", "Customer#000000008", "I0B10bB0AymmC, 0PrRYBCP1yGJ8xcBPmWhl5", "17", "27-147-574-9335", "6819.74", "BUILDING", "among the slyly regular t
heodolites kindle blithely courts. carefully even theodolites haggle slyly along the ide"]
["9", "Customer#000000009", "xKiAFTjUsCuxfeleNqefumTrjS", "8", "18-338-906-3675", "8324.07", "FURNITURE", "r theodolites according to the reque
sts wake thinly excuses: pending requests haggle furiousl"]
["10", "Customer#000000010", "6LrEaV6KR6PLVcgl2ArL Q3rqzLzcT1 v2", "5", "15-741-346-9870", "2753.54", "HOUSEHOLD", "es regular deposits haggle.
 fur"]
["11", "Customer#000000011", "PkWS 3HlXqwTuzrKg633BEi", "23", "33-464-151-3439", "-272.60", "BUILDING", "ckages. requests sleep slyly. quickly
even pinto beans promise above the slyly regular pinto beans. "]
["12", "Customer#000000012", "9PWKuhzT4Zr1Q", "13", "23-791-276-1263", "3396.49", "HOUSEHOLD", " to the carefully final braids. blithely regula
r requests nag. ironic theodolites boost quickly along"]
["13", "Customer#000000013", "nsXQu0oVjD7PM659uC3SRSp", "3", "13-761-547-5974", "3857.34", "BUILDING", "ounts sleep carefully after the close f
rays. carefully bold notornis use ironic requests. blithely"]
["14", "Customer#000000014", "KXkletMlL2JQEA ", "1", "11-845-129-3851", "5266.30", "FURNITURE", ", ironic packages across the unus"]
["15", "Customer#000000015", "YtWggXoOLdwdo7b0y,BZaGUQMLJMX1Y,EC,6Dn", "23", "33-687-542-7601", "2788.52", "HOUSEHOLD", " platelets. regular de
posits detect asymptotes. blithely unusual packages nag slyly at the fluf"]
["16", "Customer#000000016", "cYiaeMLZSMAOQ2 d0W,", "10", "20-781-609-3107", "4681.03", "FURNITURE", "kly silent courts. thinly regular theodol
ites sleep fluffily after "]
["17", "Customer#000000017", "izrh 6jdqtp2eqdtbkswDD8SG4SzXruMfIXyR7", "2", "12-970-682-3487", "6.34", "AUTOMOBILE", "packages wake! blithely e
ven pint"]
["18", "Customer#000000018", "3txGO AiuFux3zT0Z9NYaFRnZt", "6", "16-155-215-1315", "5494.43", "BUILDING", "s sleep. carefully even instructions
 nag furiously alongside of t"]
```

2) Kafka Spark Structured Streaming:

6.6 Start PySpark

```
pyspark  --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.0
```

6.7 ReadStream

```
from pyspark.sql.functions import *
```

```
data= spark.readStream.format('kafka') \
        .option('kafka.bootstrap.servers', 'localhost:9092') \
        .option('subscribe','customer') \
        .option('startingOffsets','earliest') \
        .load()
```

```
customerdf =
data.select(split(col('value'),';').alias('splitted_value
')).withColumn("CustomerKey",col("splitted_value").getIte
m(0)).withColumn("Name",col("splitted_value").getItem(1))
.withColumn("Address",col("splitted_value").getItem(2))
.withColumn("NationKey",col("splitted_value").getItem(3))
.withColumn("Phone No",col("splitted_value").getItem(4))
.withColumn("AccountBalance",col("splitted_value").getIte
m(5)).withColumn("MKTSegment",col("splitted_value").getIt
em(6).withColumn("Comment",col("splitted_value").getItem(
7)).select('Customer Key','Name','Address','Nation
Key','Phone No','Account Balance','MKTSegment','Comment')
```

6.8 WriteStream

Outputmode types:

Append mode (default) - This is the default mode,where only the new rows added to the Result Table since the last trigger will be outputted to the sink. This is supported for only those queries where rows added to the Result Table is never going to change. Hence, this mode guarantees that each row will be output only once (assuming fault-tolerant sink). For example, queries with only select, where, map, flatMap, filter, join, etc. will support Append mode.
Complete mode - The whole Result Table will be outputted to the sink after every trigger. This is supported for aggregation queries.
Update mode - Only the rows in the Result Table that were updated since the last trigger will be outputted to the sink. More information to be added in future releases. Queries with aggregation: Aggregation on event-time with watermark: Append, Update,Complete

6.8.1: To Console

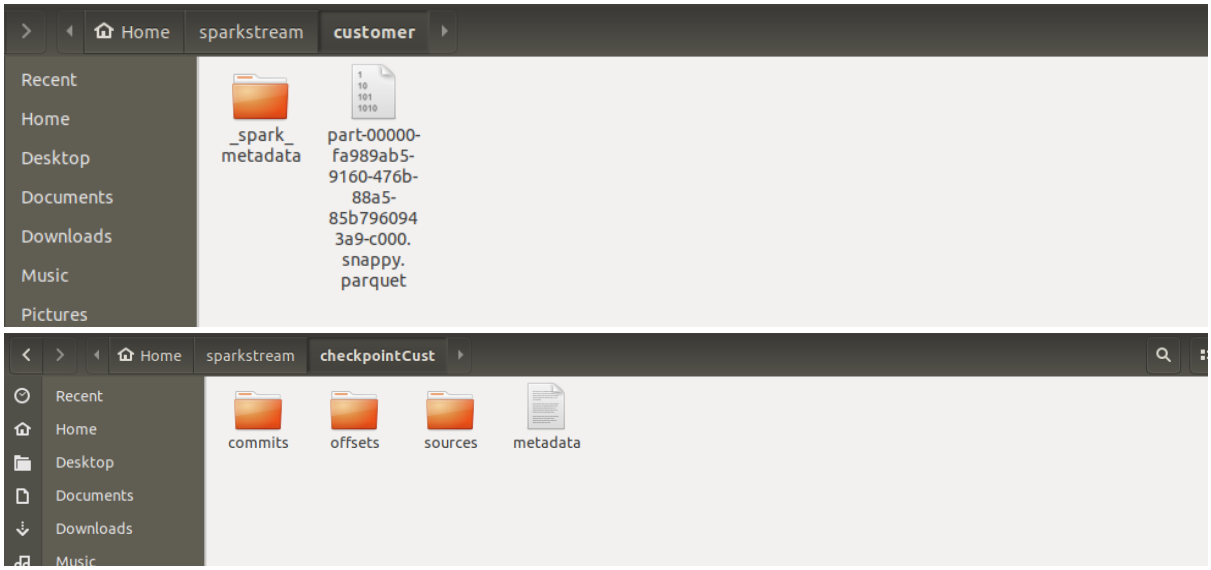
```
customer =
customerdf.writeStream.outputMode("append").format("conso
le").start()
```

```
>>> customer = customerdf.writeStream.outputMode("append").format("console").start()
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+------------+--------------------+--------------------+----------+----------------+--------------+------------+--------------------+
|Customer Key|                Name|             Address|Nation Key|        Phone No|Account Balance|  MKTSegment|             Comment|
+------------+--------------------+--------------------+----------+----------------+--------------+------------+--------------------+
|           1|"Customer#000000001"| "IVhzIApeRb ot,c,E"|        15|"25-989-741-2988"|         711.56|  "BUILDING"|"to the even, reg...|
|           2|"Customer#000000002"|"XSTf4,NCwDVaWNe6...|        13|"23-768-687-3665"|         121.65|"AUTOMOBILE"|"l accounts. blit...|
|           3|"Customer#000000003"|      "MG9kdTD2WBHm"|         1|"11-719-748-3364"|        7498.12|"AUTOMOBILE"|" deposits eat sl...|
|           4|"Customer#000000004"|       "XxVSJsLAGtn"|         4|"14-128-190-5944"|        2866.83| "MACHINERY"|" requests. final...|
|           5|"Customer#000000005"|"KvpyuHCplrB84WgA...|         3|"13-750-942-6364"|         794.47| "HOUSEHOLD"|"n accounts will ...|
|           6|"Customer#000000006"|"sKZz0CsnMD7mp4Xd...|        20|"30-114-968-4951"|        7638.57|"AUTOMOBILE"|"tions. even depo...|
|           7|"Customer#000000007"|"TcGe5gaZNgVePxU5...|        18|"28-190-982-9759"|        9561.95|"AUTOMOBILE"|"ainst the ironic...|
|           8|"Customer#000000008"|"I0B10bB0AymmC, 0...|        17|"27-147-574-9335"|        6819.74|  "BUILDING"|"among the slyly ...|
|           9|"Customer#000000009"|"xKiAFTjUsCuxfele...|         8|"18-338-906-3675"|        8324.07| "FURNITURE"|"r theodolites ac...|
|          10|"Customer#000000010"|"6LrEaV6KR6PLVcgl...|         5|"15-741-346-9870"|        2753.54| "HOUSEHOLD"|"es regular depos...|
|          11|"Customer#000000011"|"PkWS 3HlXqwTuzrK...|        23|"33-464-151-3439"|        -272.60|  "BUILDING"|"ckages. requests...|
|          12|"Customer#000000012"|     "9PWKuhzT4Zr1Q"|        13|"23-791-276-1263"|        3396.49| "HOUSEHOLD"|" to the carefull...|
|          13|"Customer#000000013"|"nsXQu0oVjD7PM659...|         3|"13-761-547-5974"|        3857.34|  "BUILDING"|"ounts sleep care...|
|          14|"Customer#000000014"|   "KXkletMlL2JQEA "|         1|"11-845-129-3851"|        5266.30| "FURNITURE"|", ironic package...|
|          15|"Customer#000000015"|"YtWggXoOLdwdo7b0...|        23|"33-687-542-7601"|        2788.52| "HOUSEHOLD"|" platelets. regu...|
|          16|"Customer#000000016"|"cYiaeMLZSMAOQ2 d...|        10|"20-781-609-3107"|        4681.03| "FURNITURE"|"kly silent court...|
|          17|"Customer#000000017"|"izrh 6jdqtp2eqdt...|         2|"12-970-682-3487"|          6.34|"AUTOMOBILE"|"packages wake! b...|
|          18|"Customer#000000018"|"3txGO AiuFux3zT0...|         6|"16-155-215-1315"|        5494.43|  "BUILDING"|"s sleep. careful...|
|          19|"Customer#000000019"|"uc,3bHIx84H,wdrm...|        18|"28-396-526-5053"|        8914.71| "HOUSEHOLD"|" nag. furiously ...|
|          20|"Customer#000000020"|     "JrPk8Pqplj4Ne"|        22|"32-957-234-8742"|        7603.40| "FURNITURE"|"g alongside of t...|
+------------+--------------------+--------------------+----------+----------------+--------------+------------+--------------------+
only showing top 20 rows
```

6.8.2: To File

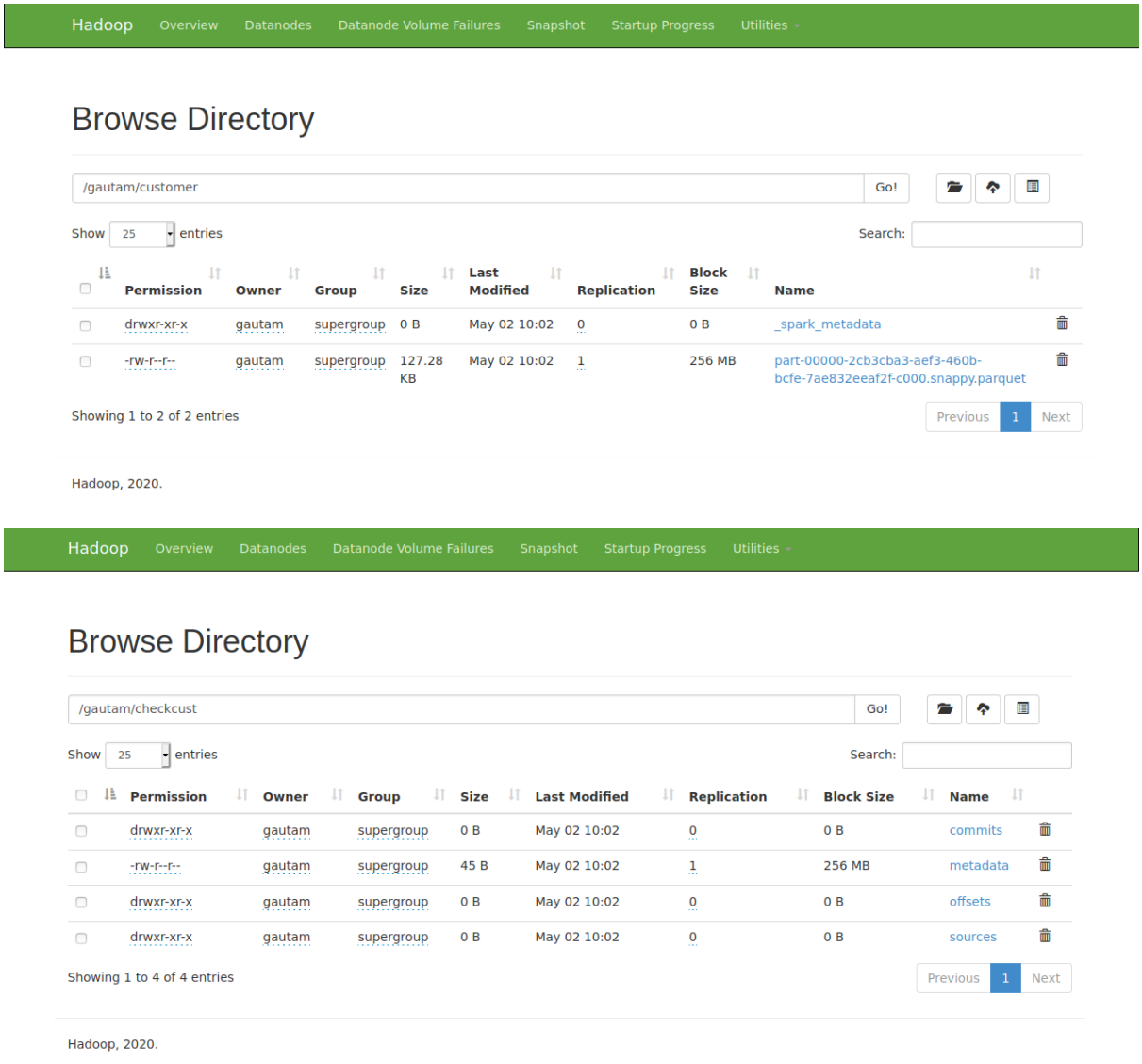
a)Local

customer=customerdf.writeStream.outputMode('append').format("parquet").option('checkpointLocation','file:///home/gautam/sparkstream/checkpoint').option("path",'file:///home/gautam/sparkstream/customer').start()



b)HDFS

customer=customerdf.writeStream.outputMode('append').format("parquet").option("checkpointLocation", "hdfs:///gautam/checkcust").start("/gautam/customer")



Task 7: Scripting

==========

7.1 Hive Script

Open ▾ | tables_hive.q ~/ | Save

```
use retail_pg_sc;

create table Region(R_RegionKey int,R_Name string,R_Comment string);
create table Customer(C_CustKey int,C_Name string,C_Address string,C_NationKey int,C_Phone
string,C_AcctBal double,C_MKTSegment string,C_Comment string);

create table Part(P_PartKey int,P_Name string, P_MFGR string,P_Brand string,P_Type string,P_Size
int,P_Container string,P_RetailPrice double,P_Comment string);

create table Orders(O_OrderKey int,O_CustKey int,O_OrderStatus string,O_TotalPrice double,O_OrderDate
string,O_OrderPriority string,O_Clerk string,O_ShipPriority int,O_Comment string);

create table LineItem(L_OrderKey int,L_PartKey int,L_SuppKey int,L_LineNumber int,L_Quantity
double,L_ExtendedPrice double,L_Discount double,L_Tax double,L_ReturnFlag string,L_LineStatus
string,L_ShipDate string,L_CommitDate string,L_ReceiptDate string,L_ShipInstruct string,L_ShipMode
string,L_Comment string);

create table Partsupp(PS_Partkey int, PS_Suppkey int, PS_Availability int, PS_Supplycost double,
PS_Comment string);

CREATE EXTERNAL TABLE Nation(N_NationKey int, N_Name string, N_RegionKey int, N_Comment string) row
format delimited fields terminated BY ';' stored AS textfile location '/retail_datasets/nation/'
tblproperties ("skip.header.line.count"="1");

CREATE EXTERNAL TABLE SUPPLIER(S_SUPPKEY int, S_NAME string, S_ADDRESS string, S_NATIONKEY int,
S_PHONE string, S_ACCTBAL double, S_COMMENT string)   ROW FORMAT DELIMITED FIELDS TERMINATED BY ';'
STORED AS TEXTFILE   LOCATION '/aman/retail_proj/SUPPLIER/' tblproperties
("skip.header.line.count"="1");

show tables;
```

Before running script:

```
hive> use retail_pg_sc;
OK
Time taken: 0.034 seconds
hive> show tables;
OK
Time taken: 0.057 seconds
```

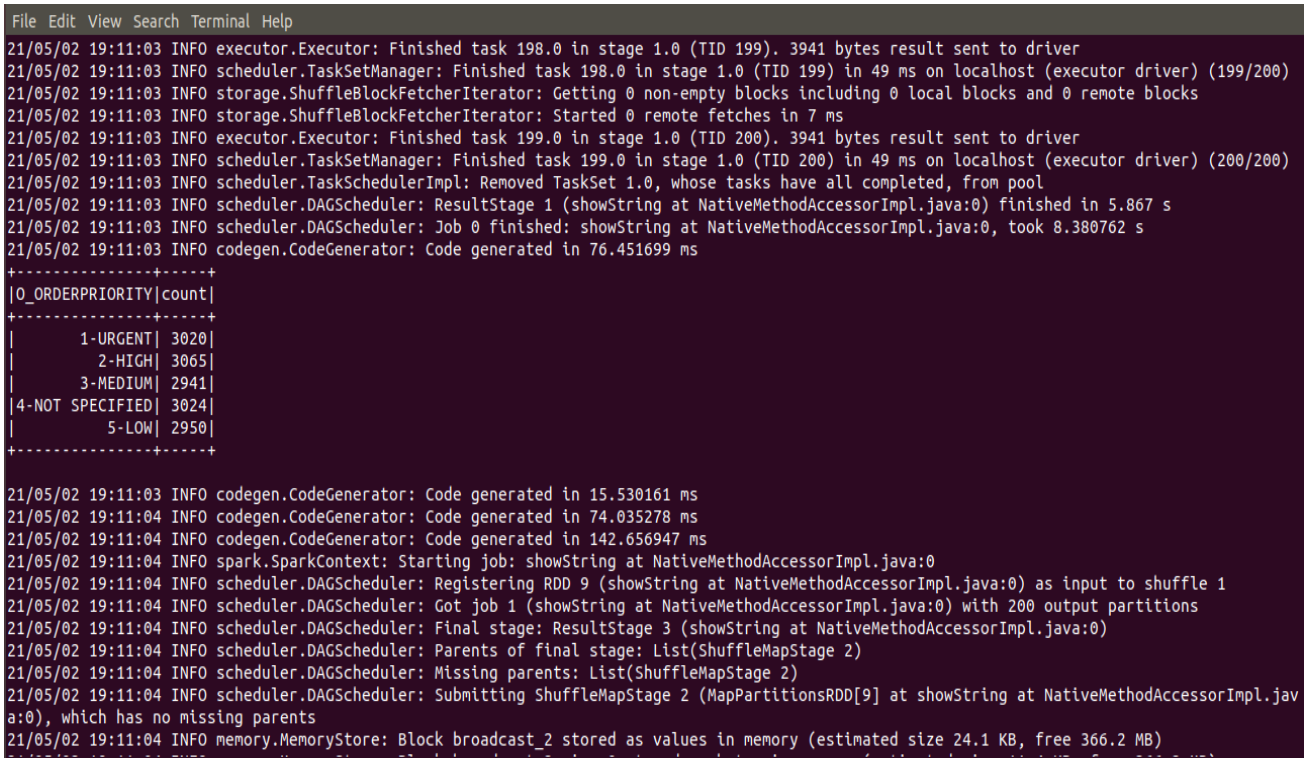
After running script:

```
ak@ak:~$ hive -f tables_hive.q
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/ak/apache-hive-2.3.7-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ak/hadoop-2.10.0/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/ak/apache-hive-2.3.7-bin/lib/hive-common-2.3.7.jar!/hive-log4j2.properties Async: true
OK
Time taken: 3.712 seconds
OK
Time taken: 0.662 seconds
OK
Time taken: 0.079 seconds
OK
Time taken: 0.105 seconds
OK
Time taken: 0.102 seconds
OK
Time taken: 0.061 seconds
OK
Time taken: 0.103 seconds
OK
Time taken: 0.082 seconds
OK
Time taken: 0.064 seconds
OK
customer
lineitem
nation
orders
part
partsupp
region
supplier
Time taken: 0.098 seconds, Fetched: 8 row(s)
```

7.2 Python Script

Open ▾ | script.py ~/PG | Save

```
1 from pyspark.sql import SparkSession
2 from pyspark.sql.functions import *
3 from pyspark.sql.types import *
4 from pyspark.sql import Window
5
6 if __name__ == "__main__":
7
8         spark = SparkSession\
9                 .builder\
10                .appName("PythonWordCount")\
11                .getOrCreate()
12
13        orderDf=spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/retail_db?useSSL=false").option("driver", "com.mysql.jdbc.Driver").option("dbtable", "ORDERS").option("mode","DROPMALFORMED").option("user", "hiveuser").option("password", "hivepassword").load()
14
15        customerDf=spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/retail_db?useSSL=false").option("driver", "com.mysql.jdbc.Driver").option("dbtable", "CUSTOMER").option("mode","DROPMALFORMED").option("user", "hiveuser").option("password", "hivepassword").load()
16
17        nationDf=spark.read.format("jdbc").option("url", "jdbc:mysql://localhost:3306/retail_db?useSSL=false").option("driver", "com.mysql.jdbc.Driver").option("dbtable", "NATION").option("mode","DROPMALFORMED").option("user", "hiveuser").option("password", "hivepassword").load()
18
19        orderDf.groupBy("O_ORDERPRIORITY").count().orderBy("O_ORDERPRIORITY").show()
20
21        orderDf.withColumn("year",year(orderDf.O_ORDERDATE)).groupBy('year').count().orderBy('year').show()
```



7.3 Sqoop Script

sqoop_incr.sh

```
lastupdatedvalue_customer=`hive -e 'select max(c_custkey) from retaill_pg.customer'`

lastupdatedvalue_lineitem=`hive -e 'select max(l_orderkey) from retaill_pg.lineitem'`

lastupdatedvalue_nation=`hive -e 'select max(n_nationkey) from retaill_pg.nation'`

lastupdatedvalue_orders=`hive -e 'select max(o_orderkey) from retaill_pg.orders'`

lastupdatedvalue_part=`hive -e 'select max(p_partkey) from retaill_pg.part'`

lastupdatedvalue_partsupp=`hive -e 'select max(ps_partkey) from retaill_pg.partsupp'`
```

```
lastupdatedvalue_region=`hive -e 'select max(r_regionkey) from
retaill_pg.region'`

lastupdatedvalue_supplier=`hive -e 'select max(s_suppkey) from
retaill_pg.supplier'`
```

```
sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Customer --target-dir
/user/hive/warehouse/retaill_pg.db/customer/ --incremental
append --check-column C_CustKey --last-value
${lastupdatedvalue_customer} --fields-terminated-by "\001" -m
2;

sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table LineItem --target-dir
/user/hive/warehouse/retaill_pg.db/lineitem/ --incremental
append --check-column L_OrderKey --last-value
${lastupdatedvalue_lineitem} --fields-terminated-by "\001" --
split-by L_LineNumber;

sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Nation --target-dir
/user/hive/warehouse/retaill_pg.db/nation/ --incremental append
--check-column N_NationKey --last-value
${lastupdatedvalue_nation} --fields-terminated-by "\001" -m 1;

sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Orders --target-dir
/user/hive/warehouse/retaill_pg.db/orders/ --incremental append
--check-column O_OrderKey --last-value
${lastupdatedvalue_orders} --fields-terminated-by "\001";

sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Part --target-dir
/user/hive/warehouse/retaill_pg.db/part/ --incremental append -
-check-column P_PartKey --last-value ${lastupdatedvalue_part} -
-fields-terminated-by "\001" -m 3;
```

```
sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Partsupp --target-dir
/user/hive/warehouse/retaill_pg.db/partsupp/ --incremental
append --check-column PS_PartKey --last-value
${lastupdatedvalue_partsupp} --fields-terminated-by "\001" -m 3
--split-by PS_PartKey;

sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Region --target-dir
/user/hive/warehouse/retaill_pg.db/region/ --incremental append
--check-column R_RegionKey --last-value
${lastupdatedvalue_region} --fields-terminated-by "\001" -m 1;

sqoop-import --connect
jdbc:mysql://localhost:3306/retaill_pg?useSSL=false --username
hiveuser --password hivepassword --table Supplier --target-dir
/user/hive/warehouse/retaill_pg.db/supplier/ --incremental
append --check-column S_SuppKey --last-value
${lastupdatedvalue_supplier} --fields-terminated-by "\001" -m
2;
```

Count of data present in hive for Customer Table:

```
>>> spark.sql("select count(*) from retaill_pg.customer").show()
+--------+
|count(1)|
+--------+
|    1500|
+--------+
```

We insert data to mysql:

```
mysql> select count(*) from retaill_pg.Customer;
+----------+
| count(*) |
+----------+
|     1500 |
+----------+
1 row in set (0.00 sec)

mysql> insert into retaill_pg.Customer values(1501,"Customer#000001501","XToT5oF
i7oIsRG2mg",19,"13-273-527-9609",5810.56,"AUTOMOBILE","ackages are slyly unusual
 req");
Query OK, 1 row affected (0.01 sec)

mysql> select count(*) from retaill_pg.Customer;
+----------+
| count(*) |
+----------+
|     1501 |
+----------+
1 row in set (0.00 sec)
```

We run the sqoop_incr.sh file to load the updated data to hive:

```
ak@ak:~$ ./sqoop_incr.sh
Warning: /home/ak/sqoop-1.4.7.bin__hadoop-2.6.0/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /home/ak/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /home/ak/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/05/02 18:55:18 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
21/05/02 18:55:18 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consi
21/05/02 18:55:18 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
21/05/02 18:55:18 INFO tool.CodeGenTool: Beginning code generation
21/05/02 18:55:18 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Customer` AS t LIM
21/05/02 18:55:18 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Customer` AS t LIM
21/05/02 18:55:18 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /home/ak/hadoop-2.10.0
Note: /tmp/sqoop-ak/compile/bee54948d932b42073e1720dc4d4955e/Customer.java uses or overrides a deprecat
Note: Recompile with -Xlint:deprecation for details.
21/05/02 18:55:23 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-ak/compile/bee54948d932b420
21/05/02 18:55:23 INFO tool.ImportTool: Maximal id query for free form incremental import: SELECT MAX(`
21/05/02 18:55:23 INFO tool.ImportTool: Incremental import based on column `C_CustKey`
21/05/02 18:55:23 INFO tool.ImportTool: Lower bound value: 0
21/05/02 18:55:23 INFO tool.ImportTool: Upper bound value: 1501
21/05/02 18:55:23 WARN manager.MySQLManager: It looks like you are importing from mysql.
21/05/02 18:55:23 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
21/05/02 18:55:23 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
21/05/02 18:55:23 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
21/05/02 18:55:23 INFO mapreduce.ImportJobBase: Beginning import of Customer
21/05/02 18:55:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
21/05/02 18:55:23 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job
```

We check whether the data is successfully inserted:

```
>>> spark.sql("select count(*) from retaill_pg.customer").show()
+--------+
|count(1)|
+--------+
|    1501|
+--------+
```

Task 8 ⇒ AWS/Snowflake Integration

==========

8.1 Hive and AWS S3 Integration

➢ **hive-site.xml**

```
<property>
    <name>fs.s3a.access.key</name>
    <value>YOUR_ACCESS_KEY</value>
</property>
<property>
    <name>fs.s3a.secret.key</name>
    <value>YOUR_SECRET_KEY</value>
</property>
```

➢ **Core-site.xml**

```
<property>
   <name>hadoop.tmp.dir</name>
   <value>/Users/nikki/hadoop/hdfs/tmp</value>
   <description>A base for other temporary
directories.</description>
</property>
<property>
   <name>fs.default.name</name>
   <value>hdfs://localhost:9000</value>
</property>
```

```
<property>
     <name>fs.s3a.impl</name>
   <value>org.apache.hadoop.fs.s3a.S3AFileSystem</value>
</property>
<property>
   <name>fs.s3a.access.key</name>
   <value>YOUR_ACCESS_KEY</value>
</property>
<property>
   <name>fs.s3a.secret.key</name>
   <value>YOUR_SECRET_KEY</value>
</property>
```

➢ **mapred-site.xml**

```
<configuration>
    <property>
        <name>mapred.job.tracker</name>
        <value>localhost:9010</value>
    </property>
    <property>
        <name>fs.s3a.impl</name>
        <value>org.apache.hadoop.fs.s3a.S3AFileSystem</value>
    </property>
<configuration>
```

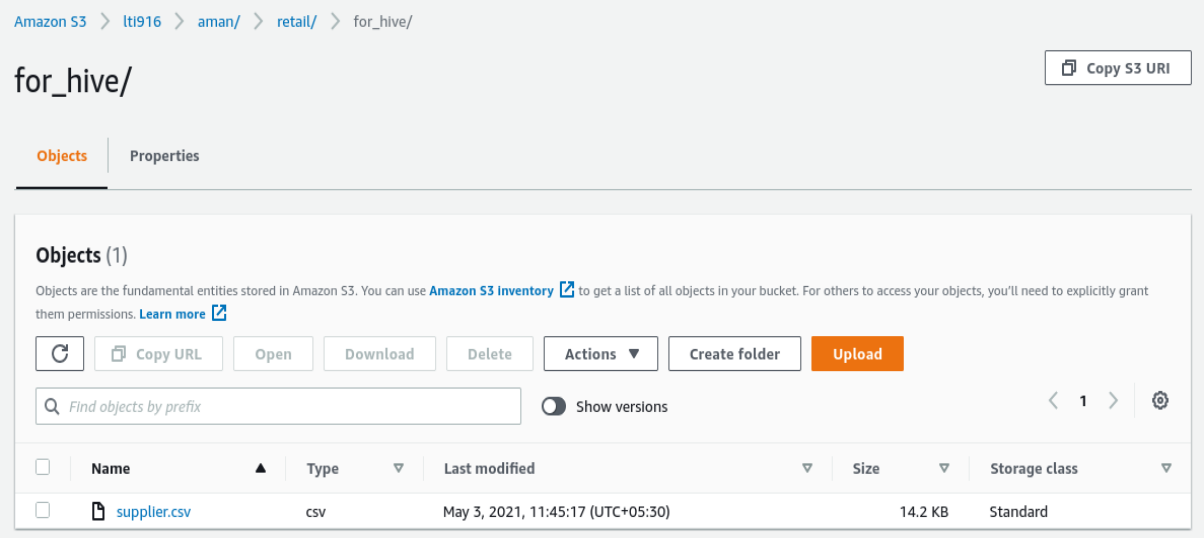
➢ **Hdfs-site.xml**

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value></value>
    </property>
    <property>
        <name>fs.s3a.access.key</name>
        <value>YOUR_ACCESS_KEY</value>
    </property>
    <property>
        <name>fs.s3a.secret.key</name>
        <value>YOUR_SECRET_KEY</value>
    </property>
</configuration>
```

➢ **EXPORT PATHS**

```
/home/ak/hadoop-2.10.0/share/hadoop/tools/lib

export JAVA_HOME=$(/usr/libexec/java_home)

export
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/usr/local/Cellar/hadoop/3
.1.1/libexec/share/hadoop/tools/lib/*

export HIVE_AUX_JARS_PATH=/home/ak/hadoop-
2.10.0/share/hadoop/tools/lib/
```

```
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/home/ak/hadoop-2.10.0/share/hadoop/tools/lib/*

hstart="/usr/local/Cellar/hadoop/3.1.1/sbin/start-dfs.sh;/usr/local/Cellar/hadoop/3.1.1/sbin/start-yarn.sh"

alias hstop= "/usr/local/Cellar/hadoop/3.1.1/sbin/stop-yarn.sh;/usr/local/Cellar/hadoop/3.1.1/sbin/stop-dfs.sh"
```



➢ **HIVE COMMAND**

```
CREATE EXTERNAL TABLE supplier_s3a(
 s_suppkey  int , s_name string, s_address string,
s_nationkey  int,
 s_phone string, s_accbal double, s_comment string)
ROW FORMAT
DELIMITED FIELDS
TERMINATED BY ';'
STORED AS TEXTFILE
```

```
LOCATION 's3a://lti916/aman/retail/for_hive/';
```

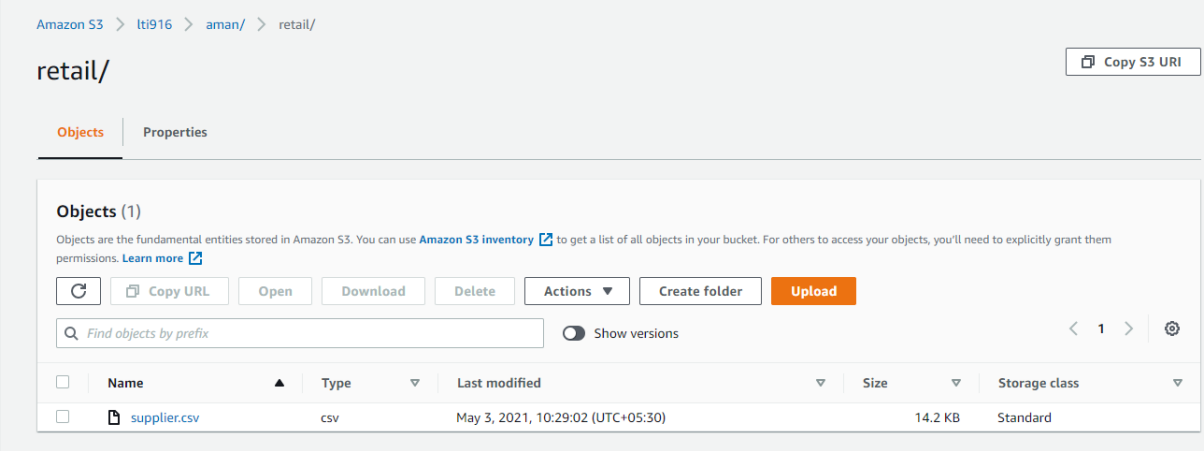
```
hive> CREATE EXTERNAL TABLE supplier_s3a(s_suppkey int, s_name string, s_address string, s_nationkey int, s_phone string, s_acctbal double, s_
comment string)
    > ROW FORMAT DELIMITED FIELDS
    > TERMINATED BY ";"
    > STORED AS TEXTFILE
    > LOCATION "s3a://lti916/aman/retail/for_hive/"
    > tblproperties ("skip.header.line.count"="1");
OK
Time taken: 5.029 seconds
hive> show table extended like supplier_s3a;
OK
tableName:supplier_s3a
owner:ak
location:s3a://lti916/aman/retail/for_hive
inputformat:org.apache.hadoop.mapred.TextInputFormat
outputformat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
columns:struct columns { i32 s_suppkey, string s_name, string s_address, i32 s_nationkey, string s_phone, double s_acctbal, string s_comment}
partitioned:false
partitionColumns:
totalNumberFiles:1
totalFileSize:14576
maxFileSize:14576
minFileSize:14576
lastAccessTime:0
lastUpdateTime:1620024177966
```

8.2 Spark and AWS S3 Integration

Download the following jar and paste it in spark/jar folder

https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/2.7.3/hadoop-aws-2.7.3.jar

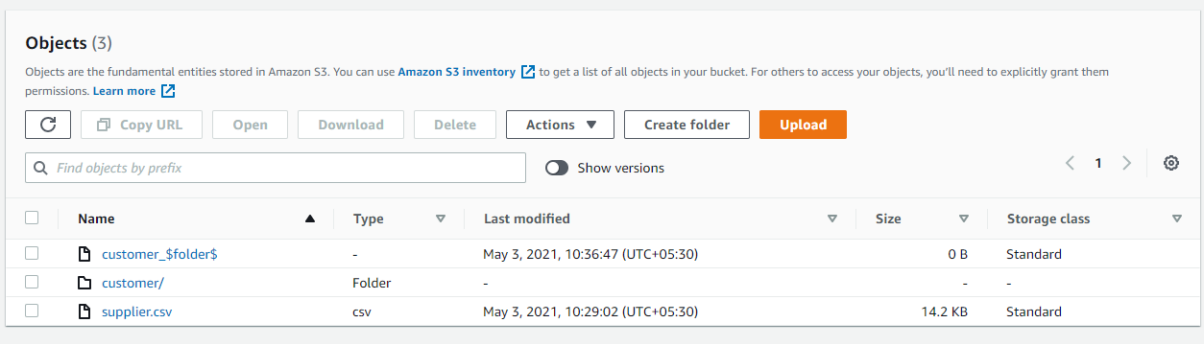
```
hadoop_conf = spark._jsc.hadoopConfiguration()
hadoop_conf.set("fs.s3n.impl",
"org.apache.hadoop.fs.s3native.NativeS3FileSystem")
hadoop_conf.set("fs.s3n.awsAccessKeyId",
'YOUR_ACCESS_KEY_ID')
hadoop_conf.set("fs.s3n.awsSecretAccessKey",
'YOUR_SECRET_ACCESS_KEY')
```

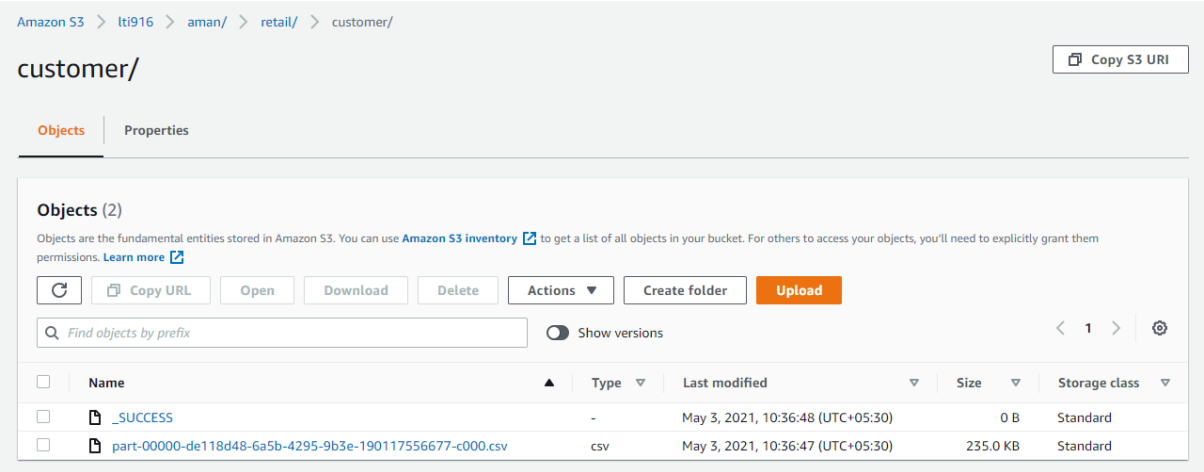



```
supplierDf = spark.read.option("header",
"false").csv("s3n://lti916/aman/retail/supplier.csv")

supplierDf.show()
```

```
+---------+------------------+--------------------+-----------+---------------+---------+--------------------+
|S_SUPPKEY|            S_NAME|           S_ADDRESS|S_NATIONKEY|        S_PHONE|S_ACCTBAL|           S_COMMENT|
+---------+------------------+--------------------+-----------+---------------+---------+--------------------+
|        1|Supplier#000000001| N kD4on9OM Ipw3,...|         17|27-918-335-1736|  5755.94|each slyly above ...|
|        2|Supplier#000000002|89eJ5ksX3ImxJQBvx...|          5|15-679-861-2259|  4032.68| slyly bold instr...|
|        3|Supplier#000000003|q1,G3Pj6OjIuUYfUo...|          1|11-383-516-1199|  4192.40|blithely silent r...|
|        4|Supplier#000000004|Bk7ah4CK8SYQTepEm...|         15|25-843-787-7479|  4641.08|riously even requ...|
|        5|Supplier#000000005|   Gcdm2rJRzl5qlTVzc|         11|21-151-690-3663|  -283.84|. slyly regular p...|
|        6|Supplier#000000006|        tQxuVm7s7CnK|         14|24-696-997-4969|  1365.79|final accounts. r...|
|        7|Supplier#000000007|s,4TicNGB4uO6PaSq...|         23|33-990-965-2201|  6820.35|s unwind silently...|
|        8|Supplier#000000008|9Sq4bBH2FQEmaFOoc...|         17|27-498-742-3860|  7627.85|al pinto beans. a...|
|        9|Supplier#000000009|1KhUgZegwM3ua7dsY...|         10|20-403-398-8662|  5302.37|s. unusual, even ...|
|       10|Supplier#000000010|  Saygah3gYWMp72i PY|         24|34-852-489-8585|  3891.91|ing waters. regul...|
|       11|Supplier#000000011|    JfwTs,LZrV, M,9C|         18|28-613-996-1505|  3393.08|y ironic packages...|
|       12|Supplier#000000012|         aLIW  q0HYd|          8|18-179-925-7181|  1432.69|al packages nag a...|
|       13|Supplier#000000013|HK71HQyWoqRWOX8GI...|          3|13-727-620-7813|  9107.22|requests engage r...|
|       14|Supplier#000000014|     EXsnO5pTNj4iZRm|         15|25-656-247-5058|  9189.82|l accounts boost....|
|       15|Supplier#000000015|olXVbNBfVzRqgokr1...|          8|18-453-357-6394|   308.56| across the furio...|
|       16|Supplier#000000016|YjP5C55zHDXL7LalK...|         22|32-822-502-4215|  2972.26|ously express ide...|
|       17|Supplier#000000017|c2d,ESHRSkK3WYnxp...|         19|29-601-884-9219|  1687.81|eep against the f...|
|       18|Supplier#000000018|    PGGVE5PWAMwKDZw |         16|26-729-551-1115|  7040.82|accounts snooze s...|
|       19|Supplier#000000019|edZT3es,nBFD8lBXT...|         24|34-278-310-2731|  6150.38|refully final fox...|
|       20|Supplier#000000020|iybAE,RmTymrZVYaF...|          3|13-715-945-6730|   530.82|n, ironic ideas w...|
+---------+------------------+--------------------+-----------+---------------+---------+--------------------+
only showing top 20 rows
```

```
customerDf = spark.read.option("header",
"false").csv("file:///home/ak/PG/data/CUSTOMER.csv")

customerDf.write.format('csv').option('header','true').sa
ve('s3n://lti916/aman/my.csv',mode='overwrite')
```





8.3 Spark and AWS Redshift Integration

Paste this jar in /home/ak/spark/jars/

https://repository.mulesoft.org/nexus/content/repositories/public/com/amazon/redshift/redshift-jdbc42/1.2.43.1067/redshift-jdbc42-1.2.43.1067.jar

```
spark._jsc.hadoopConfiguration().set("fs.s3.awsAccessKeyId", "YOUR_ACCESS_KEY_ID")
spark._jsc.hadoopConfiguration().set("fs.s3.awsSecretAccessKey", "YOUR_SECRET_ACCESS_KEY")
```

```
customerDf=spark.read.option("header","true").option("inferSchema","true").csv("file:///home/ak/PG/data/CUSTOMER.csv")

customerDf.write.format('jdbc') \
```

```
.options(url='jdbc:redshift://ltirs916.cpm2lxdgi5l8.us-
east-1.redshift.amazonaws.com:5439/am',
     driver='com.amazon.redshift.jdbc.Driver',
     dbtable='lti_schema.customer',
     user='awsuser',
     password='Pa55word')
 .mode('overwrite') \
 .save()
```

```
supplierDf=spark.read. \

options(url='jdbc:redshift://ltirs916.cpm2lxdgi5l8.us-
east-1.redshift.amazonaws.com:5439/am',
             driver =
'com.amazon.redshift.jdbc42.Driver',
             user = 'awsuser',
             password = 'Pa55word',
             dbtable = 'lti_schema.supplier')
         .format('jdbc').
         load()

supplierDf.show()
```

8.4 AWS Lambda script for loading data from S3 to Dynamodb

```
import json
import urllib.parse
import boto3
import csv

print('Loading Function')

s3 = boto3.client('s3')
```

```
dynamodb=boto3.client('dynamodb')

def lambda_handler(event, context):
 bucket = event['Records'][0]['s3']['bucket']['name']
 key = event['Records'][0]['s3']'object']['key']
  response = s3.get_object(Bucket=bucket, Key=key)
 data = response['Body'].read().dcode('utf-8')
  rows = data.split("\n")
 nationr = list(csv.reader(rows))
  print(nationr)
  arr = dynamodb.list_tables()
  if 'Nation' not in arr['TableNames']:

   dynamodb.create_table(AttributeDefinitions =
[{'AttributeName': nationr[0][0], 'AttributeType':'N'}],
   TableName = 'Nation',
   KeySchema = [{'AttributeName':nationr[0][0],
'KeyType':'HASH'}],
   BillingMode='PROVISIONED',
   ProvisionedThroughput={
     'ReadCapacityUnits':5,
     'WriteCapacityUnits':5
   })
   cols = [nationr[0][0], nationr[0][1], nationr[0][2],
nationr[0][3]]
 else:
   cols = [i['AttributeName'] for i in
dynamodb.describe_table(TableName='Nation')['Table']['Att
ributeDefinitions']]

 for row in nationr[1:]:
   try:
     dynamodb.put_item(TableName='Nation',
           Item={
             cols[0]:{'N':row[0]},
```

```
            nationr[0][1]:{'S':row[1]},
            nationr[0][2]:{'S':row[2]},
            nationr[0][3]:{'S':row[3]}
          }
        )
  except IndexError:
    break
 return {
  'statusCode': 200
}
```

```
{
 "Records" : [
   {
     "eventVersion" :  "2.0" ,
     "eventSource" :  "aws:s3" ,
     "awsRegion" :  "us-east-1" ,
     "eventTime" :  "1970-01-01T00:00:00.000Z" ,
     "eventName" :  "ObjectCreated:Put" ,
     "userIdentity" : {
     },
     "principalId" :  "EXAMPLE"
     "requestParameters" : {
   "sourceIPAddress" :  "127.0.0.1"
     },
     "responseElements" : {
   "x-amz-request-id" :  "EXAMPLE123456789" ,
"x-amz-id-2"
:"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxy
zABCDEFGH"
     },
     "s3" : {
   "s3SchemaVersion" :  "1.0" ,
   "configurationId" :  "testConfigRule" ,
```

```
   "bucket" : {
   "name" :  "lti916" ,
   "ownerIdentity" : {
     "principalId" :  "EXAMPLE"
   },
   "arn" :  "arn:aws:s3:::lti916"
   },
   "object" : {
     "key" :  "retail/dynamodbwrite/nation.csv" ,
     "size" :  1024 ,
     "eTag" :  "0123456789abcdef0123456789abcdef" ,
     "sequencer" :  "0A1B2C3D4E5F678901"}}}]}
```

8.5 Spark and Snowflake Integration

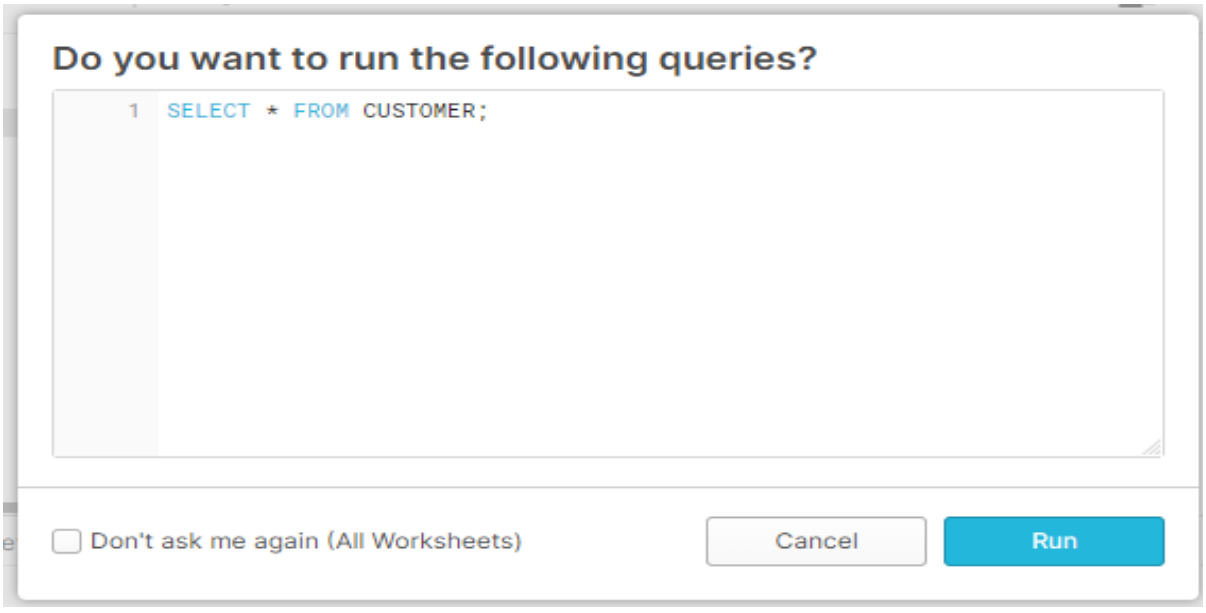
https://repo1.maven.org/maven2/net/snowflake/spark-snowflake_2.11/2.4.14-spark_2.4/spark-snowflake_2.11-2.4.14-spark_2.4.jar

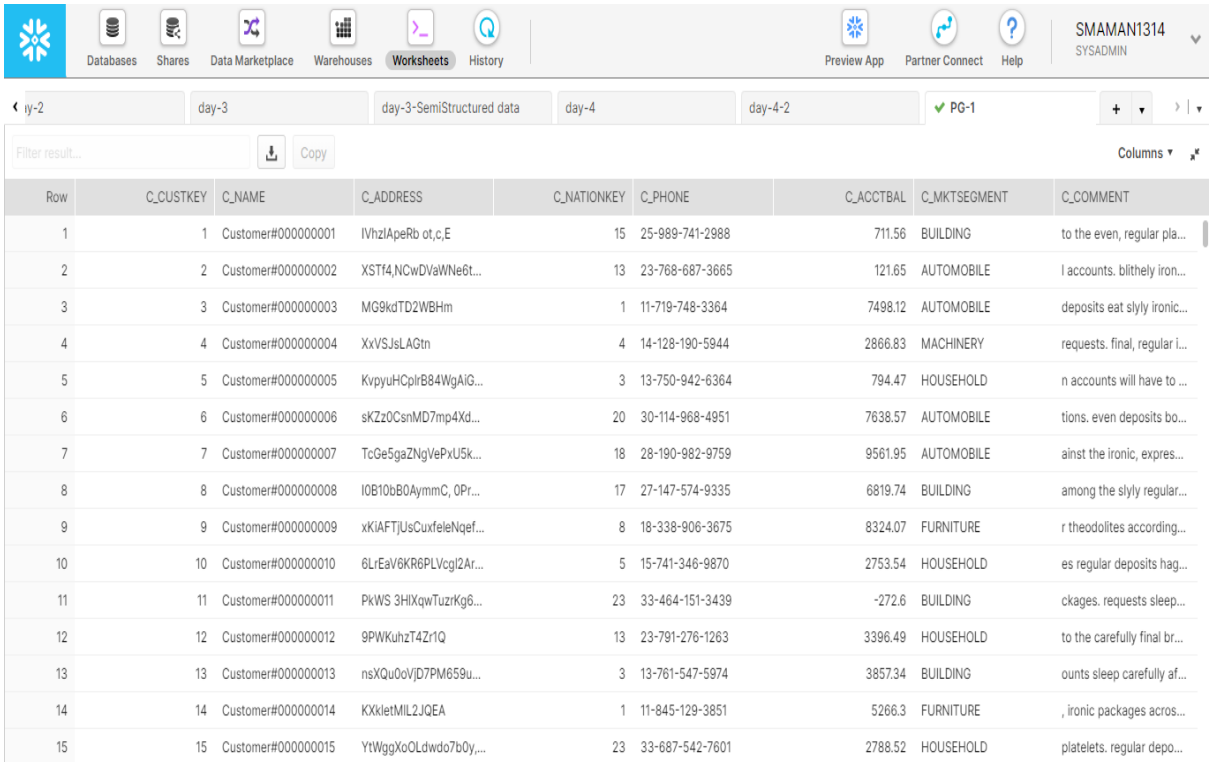
https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.8.0/snowflake-jdbc-3.8.0.jar

```
sc._jsc.hadoopConfiguration().set("fs.s3n.awsAccessKeyId"
, "YOUR_ACCESS_KEY_ID ")
sc._jsc.hadoopConfiguration().set("fs.s3n.awsSecretAccess
Key", "YOUR_SECRET_ACCESS_KEY ")

spark._jvm.net.snowflake.spark \
 .snowflake \
 .SnowflakeConnectorUtils \
.enablePushdownSession(spark._jvm.org.apache.spark.sql.Sp
arkSession.builder().getOrCreate())
```

```
SNOWFLAKE_SOURCE_NAME = "net.snowflake.spark.snowflake"

sfOptions = {
 "sfURL" : "your_url.us-east-1.snowflakecomputing.com",
 "sfUser" : "username",
 "sfPassword" : "password",
 "sfDatabase" : "LTI_DB",
 "sfSchema" : "LTISCHEMA",
 "sfWarehouse" : "AMANWH"
}
```

```
customerDf =
spark.read.option("header","true").option("inferSchema","
true").csv("/home/ak/PG/data/CUSTOMER.csv")

customerDf.write\
   .format("snowflake")\
   .options(**sfOptions)\
   .option("dbtable", "customer")\
   .mode("overwrite")\
   .save()
```





Row	C_CUSTKEY	C_NAME	C_ADDRESS	C_NATIONKEY	C_PHONE	C_ACCTBAL	C_MKTSEGMENT	C_COMMENT
1	1	Customer#000000001	IVhzIApeRb ot,c,E	15	25-989-741-2988	711.56	BUILDING	to the even, regular pla...
2	2	Customer#000000002	XSTf4,NCwDVaWNe6t...	13	23-768-687-3665	121.65	AUTOMOBILE	l accounts. blithely iron...
3	3	Customer#000000003	MG9kdTD2WBHm	1	11-719-748-3364	7498.12	AUTOMOBILE	deposits eat slyly ironic...
4	4	Customer#000000004	XxVSJsLAGtn	4	14-128-190-5944	2866.83	MACHINERY	requests. final, regular i...
5	5	Customer#000000005	KvpyuHCplrB84WgAiG...	3	13-750-942-6364	794.47	HOUSEHOLD	n accounts will have to ...
6	6	Customer#000000006	sKZz0CsnMD7mp4Xd...	20	30-114-968-4951	7638.57	AUTOMOBILE	tions. even deposits bo...
7	7	Customer#000000007	TcGe5gaZNgVePxU5k...	18	28-190-982-9759	9561.95	AUTOMOBILE	ainst the ironic, expres...
8	8	Customer#000000008	I0B10bB0AymmC, 0Pr...	17	27-147-574-9335	6819.74	BUILDING	among the slyly regular...
9	9	Customer#000000009	xKiAFTjUsCuxfeleNqef...	8	18-338-906-3675	8324.07	FURNITURE	r theodolites according...
10	10	Customer#000000010	6LrEaV6KR6PLVcgl2Ar...	5	15-741-346-9870	2753.54	HOUSEHOLD	es regular deposits hag...
11	11	Customer#000000011	PkWS 3HlXqwTuzrKg6...	23	33-464-151-3439	-272.6	BUILDING	ckages. requests sleep...
12	12	Customer#000000012	9PWKuhzT4Zr1Q	13	23-791-276-1263	3396.49	HOUSEHOLD	to the carefully final br...
13	13	Customer#000000013	nsXQu0oVjD7PM659u...	3	13-761-547-5974	3857.34	BUILDING	ounts sleep carefully af...
14	14	Customer#000000014	KXkletMlL2JQEA	1	11-845-129-3851	5266.3	FURNITURE	, ironic packages acros...
15	15	Customer#000000015	YtWggXoOLdwdo7b0y,...	23	33-687-542-7601	2788.52	HOUSEHOLD	platelets. regular depo...

```
read_customerDf =
spark.read.format(SNOWFLAKE_SOURCE_NAME) \
 .options(**sfOptions) \
 .option("query",  "select * from customer") \
 .load()
read_customerDf.show()
```

```
+---------+------------------+--------------------+-----------+---------------+---------+------------+--------------------+
|C_CUSTKEY|            C_NAME|           C_ADDRESS|C_NATIONKEY|        C_PHONE|C_ACCTBAL|C_MKTSEGMENT|           C_COMMENT|
+---------+------------------+--------------------+-----------+---------------+---------+------------+--------------------+
|        1|Customer#000000001|    IVhzIApeRb ot,c,E|         15|25-989-741-2988|   711.56|    BUILDING|to the even, regu...|
|        2|Customer#000000002|XSTf4,NCwDVaWNe6t...|         13|23-768-687-3665|   121.65|  AUTOMOBILE|l accounts. blith...|
|        3|Customer#000000003|         MG9kdTD2WBHm|          1|11-719-748-3364|  7498.12|  AUTOMOBILE| deposits eat sly...|
|        4|Customer#000000004|          XxVSJsLAGtn|          4|14-128-190-5944|  2866.83|   MACHINERY| requests. final,...|
|        5|Customer#000000005|KvpyuHCplrB84WgAi...|          3|13-750-942-6364|   794.47|   HOUSEHOLD|n accounts will h...|
|        6|Customer#000000006|sKZz0CsnMD7mp4Xd0...|         20|30-114-968-4951|  7638.57|  AUTOMOBILE|tions. even depos...|
|        7|Customer#000000007|TcGe5gaZNgVePxU5k...|         18|28-190-982-9759|  9561.95|  AUTOMOBILE|ainst the ironic,...|
|        8|Customer#000000008|I0B10bB0AymmC, 0P...|         17|27-147-574-9335|  6819.74|    BUILDING|among the slyly r...|
|        9|Customer#000000009|xKiAFTjUsCuxfeleN...|          8|18-338-906-3675|  8324.07|   FURNITURE|r theodolites acc...|
|       10|Customer#000000010|6LrEaV6KR6PLVcgl2...|          5|15-741-346-9870|  2753.54|   HOUSEHOLD|es regular deposi...|
|       11|Customer#000000011|PkWS 3HlXqwTuzrKg...|         23|33-464-151-3439|   -272.6|    BUILDING|ckages. requests ...|
|       12|Customer#000000012|       9PWKuhzT4Zr1Q|         13|23-791-276-1263|  3396.49|   HOUSEHOLD| to the carefully...|
|       13|Customer#000000013|nsXQu0oVjD7PM659u...|          3|13-761-547-5974|  3857.34|    BUILDING|ounts sleep caref...|
|       14|Customer#000000014|     KXkletMlL2JQEA |          1|11-845-129-3851|   5266.3|   FURNITURE|, ironic packages...|
|       15|Customer#000000015|YtWggXoOLdwdo7b0y...|         23|33-687-542-7601|  2788.52|   HOUSEHOLD| platelets. regul...|
|       16|Customer#000000016| cYiaeMLZSMAOQ2 d0W,|         10|20-781-609-3107|  4681.03|   FURNITURE|kly silent courts...|
|       17|Customer#000000017|izrh 6jdqtp2eqdtb...|          2|12-970-682-3487|     6.34|  AUTOMOBILE|packages wake! bl...|
|       18|Customer#000000018|3txGO AiuFux3zT0Z...|          6|16-155-215-1315|  5494.43|    BUILDING|s sleep. carefull...|
|       19|Customer#000000019|uc,3bHIx84H,wdrmL...|         18|28-396-526-5053|  8914.71|   HOUSEHOLD| nag. furiously c...|
|       20|Customer#000000020|       JrPk8Pqplj4Ne|         22|32-957-234-8742|   7603.4|   FURNITURE|g alongside of th...|
+---------+------------------+--------------------+-----------+---------------+---------+------------+--------------------+
only showing top 20 rows
```

Task 9: Business Queries (Pyspark Transformations and Visualizations)

==========

9.1 Queries

Import Essential Functions from pyspark

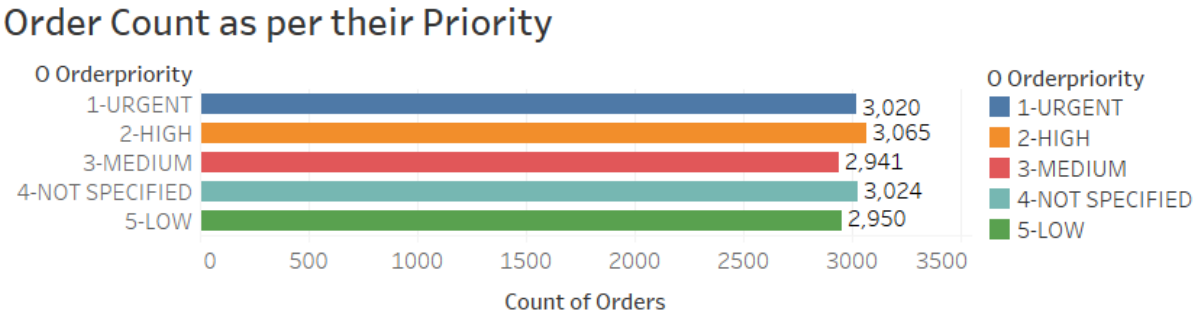
```
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

9.1.1 Using basic PySpark functions

Query 1: Company wants to know order count as per their priority

```
orderDf.groupBy("O_ORDERPRIORITY").count().orderBY("O_ORD
ERPRIORITY").show()
```

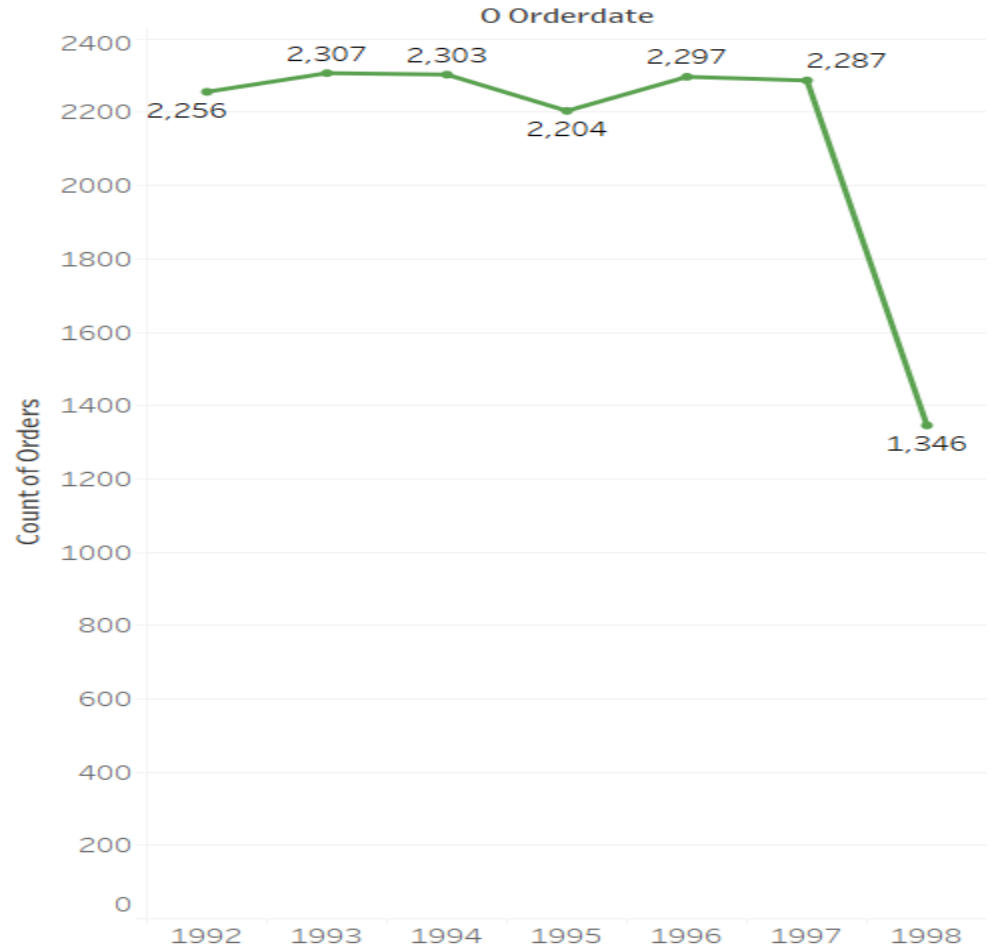
```
+---------------+-----+
|O_ORDERPRIORITY|count|
+---------------+-----+
|        1-URGENT| 3020|
|          2-HIGH| 3065|
|        3-MEDIUM| 2941|
|4-NOT SPECIFIED| 3024|
|          5-LOW| 2950|
+---------------+-----+
```



Query 2: Company wants to know number of orders placed in each year

```
orderDf.withColumn("year",year(orderDf.O_OrderDate)).groupBy('year').count().orderBy('year').show()
```

```
+----+------+
|year|count |
+----+------+
|1992|  2256|
|1993|  2307|
|1994|  2303|
|1995|  2204|
|1996|  2297|
|1997|  2287|
|1998|  1346|
+----+------+
```



9.1.2 Using Join

Different Join Optimizations are:

1) **Broadcast Join:** Broadcast join is a famous join for joining a small table (dimension table) with a big table (fact table) by avoiding costly data shuffling. We have used broadcast joins for joining our small tables like Nation, Region, Supplier and Customer.

2) **Shuffle-Hash Join:** Shuffle hash join shuffles the data based on join keys and then perform the join. While this approach always works, it can be more expensive than necessary because it requires a shuffle. So, we have not opted for this join.

3) **Sort-Merge Join:** Sort merge join perform the Sort operation first and then merges the datasets. Spark performs this join when you are joining two Big tables. It minimizes data movement in the cluster, is a highly scalable approach and performs better when compared to Shuffle Hash Joins. We have used sort merge join for join of our big tables LineItem and Orders.

------Broadcast Join------

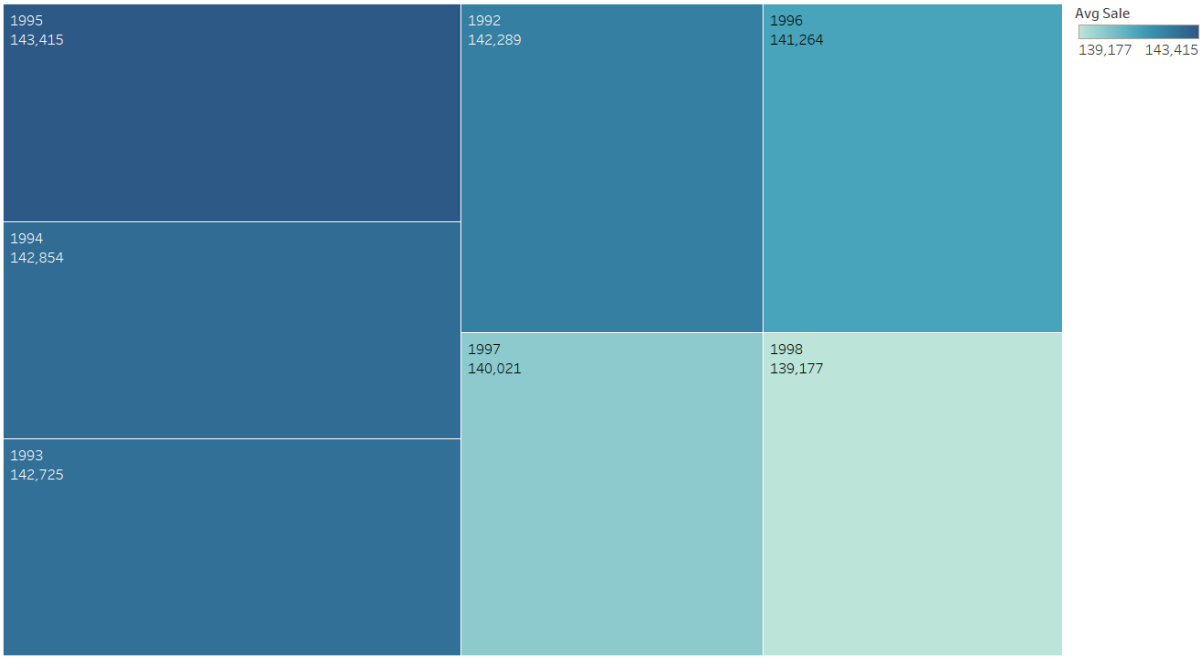
Query 3: Company wants to know the average sales for each year/nation/region?

➢ **Year**

```
ordersDf.withColumn("year",year(to_timestamp('o_orderdate','yyyy-MM-dd'))).groupBy("year").agg(round(avg("o_totalprice"),3).alias("Avg Sale")).show()
```

```
+----+----------+
|year|  Avg Sale|
+----+----------+
|1997|140021.307|
|1994|142853.583|
|1996|141264.363|
|1998|139177.195|
|1995|  143415.5|
|1992|142289.204|
|1993|142725.335|
+----+----------+
```

Average Sale for each year

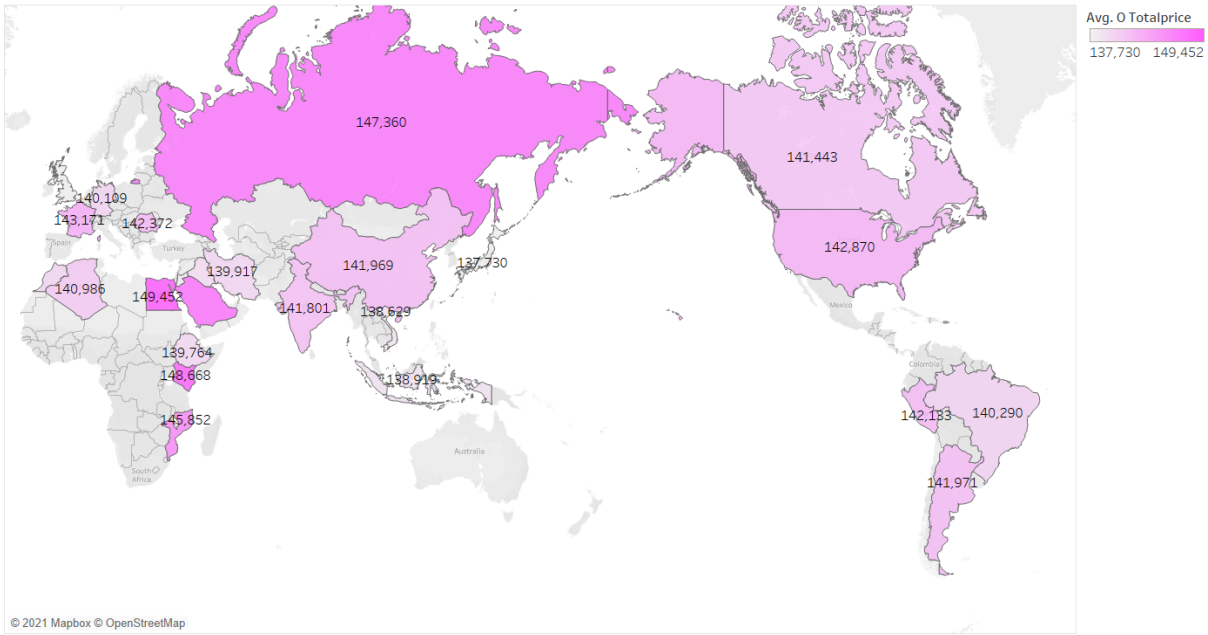


➢ Nation

```
ordersDf.join(broadcast(customerDf),ordersDf.o_custkey==customerDf.c_custkey,"inner").join(broadcast(nationDf),customerDf.c_nationkey==nationDf.n_nationkey,"inner").groupBy("n_name").agg(round(avg("o_totalprice"),3).alias("Avg Sale")).show()
```

```
+--------------+----------+
|        n_name|  Avg Sale|
+--------------+----------+
|         JAPAN|137730.116|
|        JORDAN|138091.701|
|UNITED KINGDOM|137834.018|
|        CANADA|141442.632|
|       ALGERIA|140985.926|
|       MOROCCO|139954.001|
|        FRANCE|143170.511|
|    MOZAMBIQUE|145852.065|
|         CHINA|141969.163|
| SAUDI ARABIA|147556.269|
|     ARGENTINA|141971.163|
|          IRAN|139916.708|
|        RUSSIA|147359.929|
|         INDIA|141800.826|
|         KENYA|148668.246|
|          IRAQ|139382.066|
|       VIETNAM|138629.312|
|      ETHIOPIA|139764.093|
|     INDONESIA|138919.188|
| UNITED STATES|142870.194|
+--------------+----------+
only showing top 20 rows
```

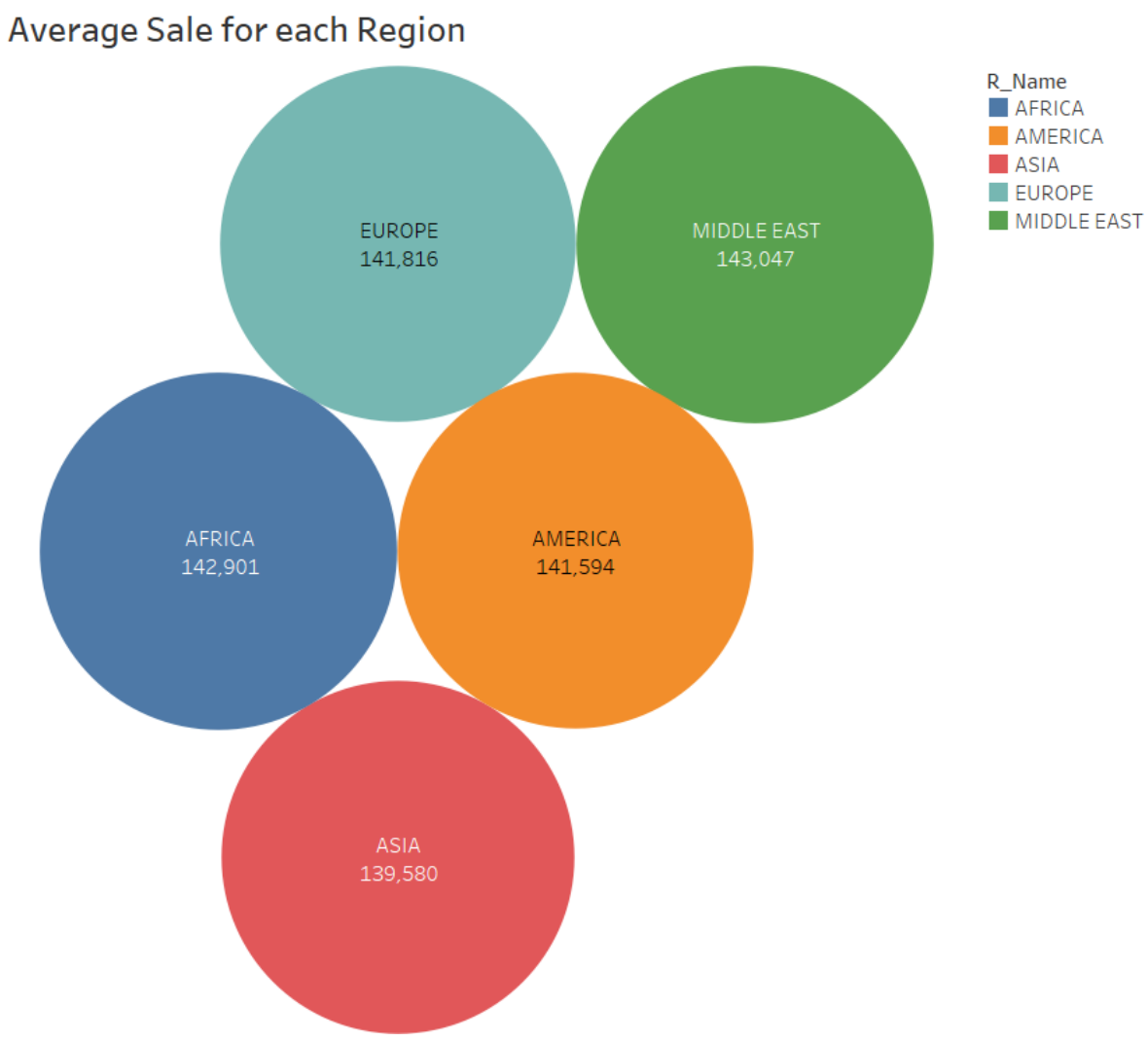
Average Sale for each nation



➢ **Region**

```
ordersDf.join(broadcast(customerDf),ordersDf.o_custkey==customerDf.c_custkey,"inner").join(broadcast(nationDf),customerDf.c_nationkey==nationDf.n_nationkey,"inner").join(broadcast(regionDf),nationDf.n_regionkey==regionDf.r_regionkey,"inner").groupBy("r_name").agg(round(avg("o_totalprice"),3).alias("Avg Sale")).show()
```

```
+-----------+----------+
|     r_name|  Avg Sale|
+-----------+----------+
|       ASIA| 139580.15|
|     EUROPE| 141816.46|
|    AMERICA|141594.129|
|     AFRICA|142901.018|
|MIDDLE EAST|143047.311|
+-----------+----------+
```



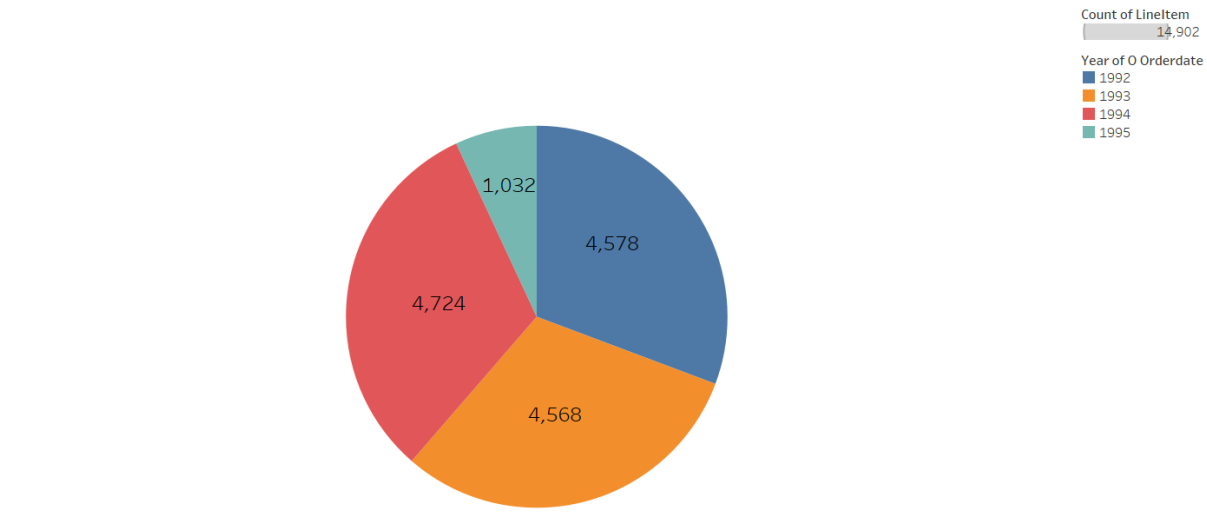
------Sort Merge Join------

Query 4: Company wants to know the count of items that were returned by the customers

```
TotItem = lineitemDf.join(orderDf,lineitemDf.L_OrderKey
==
orderDf.O_ORDERKEY,"inner").select("O_OrderDate","L_Retur
nFlag").filter(lineitemDf["L_RETURNFLAG"].like("R"))

TotItem.withColumn('Year',year(TotItem.O_OrderDate)).grou
pBy('YEAR').count().orderBy('Year').show()
```

```
+----+-----+
|YEAR|count|
+----+-----+
|1992| 4578|
|1993| 4568|
|1994| 4724|
|1995| 1032|
+----+-----+
```

Count of items returned by customers



9.1.3 Using UDF

UDF is a User Defined Function which is used to create a reusable function. Once UDF created, that can be reused on multiple DataFrames and SQL (after registering). The default type of the udf () is StringType. One needs to handle null explicitly, otherwise there would be adverse effects.
We have tried to add a column by

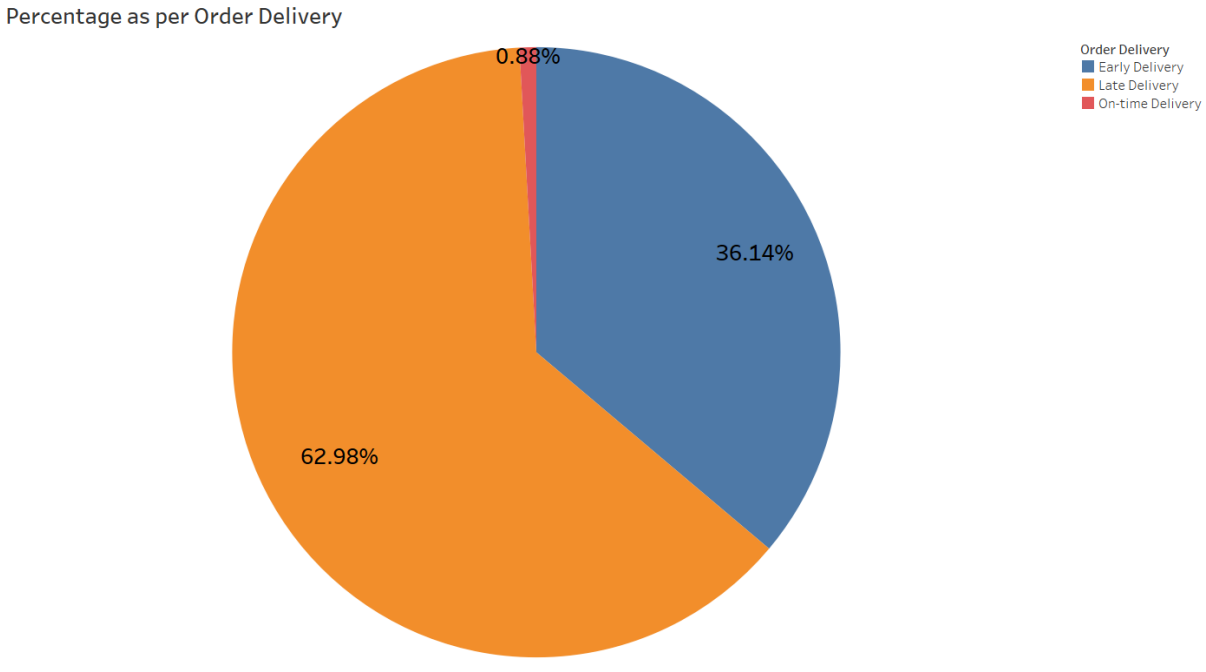
Query 5: Company wants to know the percentage of items delivered on-time, early or late.

```
# Define the function to be registered
ord_del_udf = udf(lambda commit_date,receipt_date:
ord_del(commit_date,receipt_date), StringType())

# Register the udf function
spark.udf.register("ord_del_udf",ord_del_udf)

# Define the function mentioned in udf
from datetime import date
```

```
def ord_del(commit_date,receipt_date):
  commit_year,commit_month,commit_date = (int(x) for x in
str(commit_date).split("-"))
  receipt_year,receipt_month,receipt_date = (int(y) for y
in str(receipt_date).split("-"))
  commit_date =
date(commit_year,commit_month,commit_date)
  receipt_date =
date(receipt_year,receipt_month,receipt_date)
  if commit_date > receipt_date:
          return 'Early Delivery'
  elif commit_date == receipt_date:
          return 'On-time Delivery'
  else:
          return 'Late Delivery'
```

```
# Add Column to Dataframe by using udf
lineitemDfSat =
lineitemDf.withColumn("Order_Delivery",ord_del_udf(col("l
_commitdate"),col("l_receiptdate")))

#Compute Total
total = lineitemDf.count()

# Compute Percentage
lineitemDfSat.groupBy("Order_Delivery").agg(((count("l_or
derkey")/total)*100).alias("Percentage")).show()
```

```
+---------------+------------------+
|  Order_Delivery|        Percentage|
+---------------+------------------+
|On-time Delivery|0.8807644370585791|
|   Late Delivery|62.977980889073535|
|  Early Delivery|36.141254673867884|
+---------------+------------------+
```



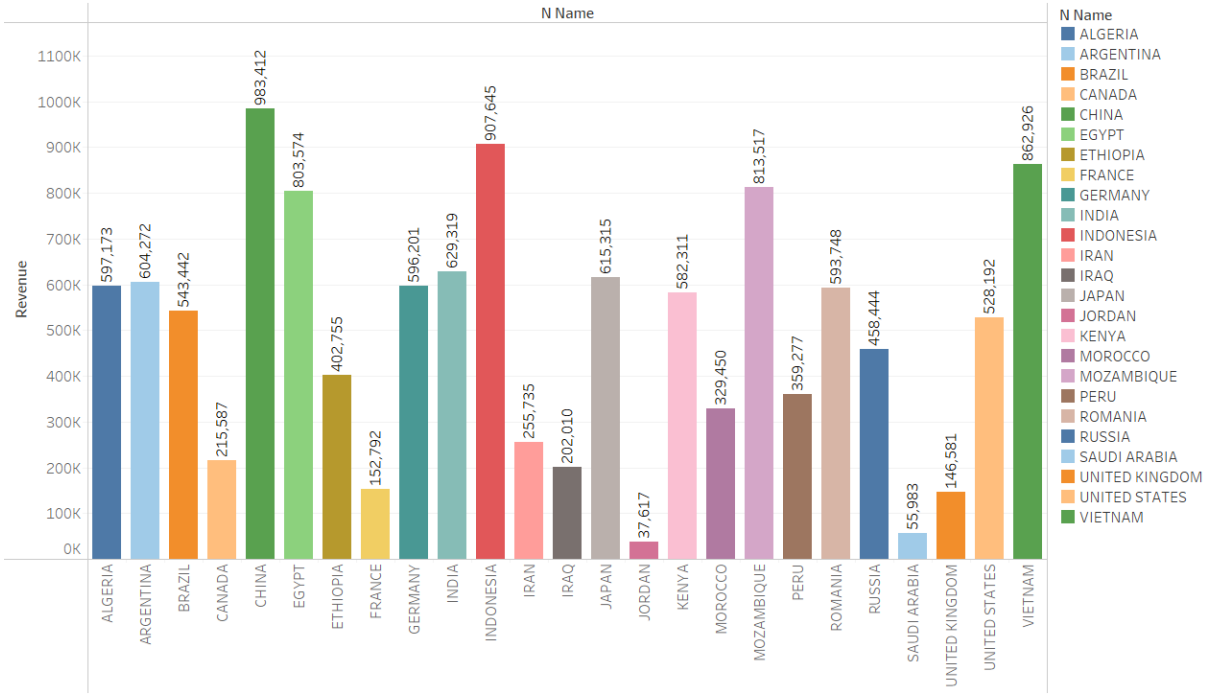
9.1.4 Using spark-sql

Query 6: Company wants to know the total revenue earned through local suppliers in the year 1993

```
spark.sql("select n_name, round(sum(l_extendedprice * (1 - l_discount)),3) as revenue from customer, orders, lineitem, supplier, nation where c_custkey = o_custkey and l_orderkey = o_orderkey and l_suppkey = s_suppkey and c_nationkey = s_nationkey and s_nationkey = n_nationkey and year(o_orderdate)=1993 group by n_name order by n_name").show()
```

```
+----------+----------+
|    n_name|   revenue|
+----------+----------+
|   ALGERIA|597173.266|
| ARGENTINA|604272.055|
|    BRAZIL|543441.829|
|    CANADA|215587.186|
|     CHINA|983411.702|
|     EGYPT|803573.957|
|  ETHIOPIA|402755.016|
|    FRANCE|152791.536|
|   GERMANY| 596200.72|
|     INDIA|629319.033|
| INDONESIA|907645.494|
|      IRAN| 255735.05|
|      IRAQ| 202009.81|
|     JAPAN|615315.029|
|    JORDAN| 37617.233|
|     KENYA|582310.683|
|   MOROCCO|329450.095|
|MOZAMBIQUE|813517.219|
|      PERU|359277.291|
|   ROMANIA|593747.754|
+----------+----------+
only showing top 20 rows
```

Total Revenue earned through local suppliers

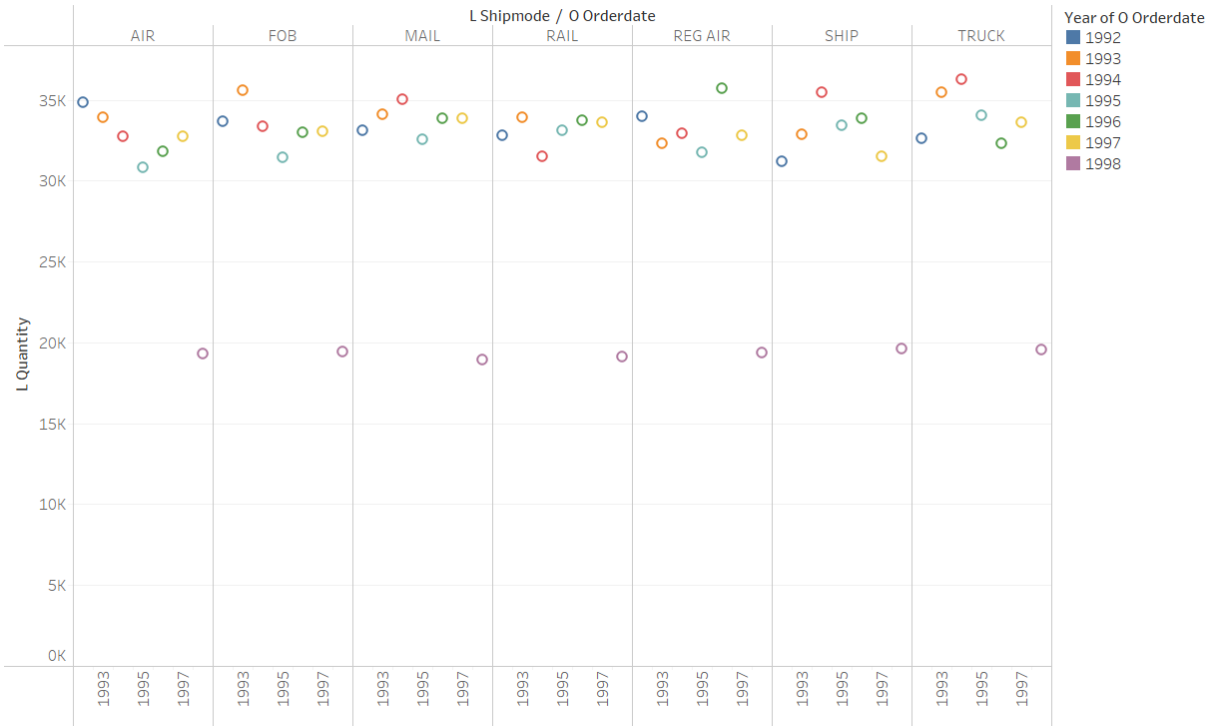


Query 7: Company wants to know quantity of items shipped each year by different ship modes

spark.sql("select year(o_orderdate) as Year,l_shipmode as ShipMode,sum(l_quantity) as Quantity from lineitem,orders where l_orderkey=o_orderkey group by year(o_orderdate),l_shipmode order by Year,ShipMode").show()

```
+----+--------+--------+
|Year|ShipMode|Quantity|
+----+--------+--------+
|1992|     AIR| 34864.0|
|1992|     FOB| 33655.0|
|1992|    MAIL| 33100.0|
|1992|    RAIL| 32814.0|
|1992| REG AIR| 34018.0|
|1992|    SHIP| 31199.0|
|1992|   TRUCK| 32644.0|
|1993|     AIR| 33944.0|
|1993|     FOB| 35614.0|
|1993|    MAIL| 34124.0|
|1993|    RAIL| 33940.0|
|1993| REG AIR| 32316.0|
|1993|    SHIP| 32869.0|
|1993|   TRUCK| 35452.0|
|1994|     AIR| 32771.0|
|1994|     FOB| 33369.0|
|1994|    MAIL| 35062.0|
|1994|    RAIL| 31498.0|
|1994| REG AIR| 32965.0|
|1994|    SHIP| 35461.0|
+----+--------+--------+
only showing top 20 rows
```

Quantity of items shipped each year through different ship modes



9.1.5 Using Rank and Dense Rank

- **RANK**: This gives you the ranking within your ordered partition. Ties are assigned the same rank, with the next ranking(s) skipped. Therefore, if you have 3 items at rank 2, the next rank listed will be ranked 5.
- **DENSE_RANK**: This gives you the ranking within your ordered partition, but the ranks are consecutive in it. Also, no ranks are skipped if there are ranks with multiple items.

Query 8: Company has decided to reward the customers having highest account balance in each nation. Also some coupons for it's top 10 customers worldwide. So we need to provide the details of these customers to the company.

```
from pyspark.sql import Window
```

- **Details of Top Customer in Each Nation**

```
customerDf.join(broadcast(nationDf),customerDf.c_nationkey==nationDf.n_nationkey,"left").withColumn("Country_Ranking",rank().over(Window.partitionBy("n_name").orderBy(col("c_acctbal").desc()))).filter("Country_Ranking<=1").select(col("c_custkey").alias("Key"),col("c_name").alias("Name"),col("c_address").alias("Address"),col("c_phone").alias("Contact No"),col("c_acctbal").alias("Account Balance"),col("n_name").alias("Nation")).show()
```

```
+----+------------------+--------------------+--------------+---------------+--------------+
| Key|              Name|             Address|    Contact No|Account Balance|        Nation|
+----+------------------+--------------------+--------------+---------------+--------------+
|1403|Customer#000001403|,ql804gtMc3uxTfP,...|22-458-624-2509|        9782.34|         JAPAN|
| 145|Customer#000000145|kQjHmt2kcec cy3hf...|23-562-444-8454|        9748.93|        JORDAN|
|1441|Customer#000001441|u0YYZb46w,pwKo5H9...|33-681-334-4499|        9465.15|UNITED KINGDOM|
|1312|Customer#000001312|f5zgMB4MHLMSHaX0t...|13-153-492-9898|         9459.5|        CANADA|
| 295|Customer#000000295|mk649IH6njR14woTV...|10-340-773-4322|        9497.89|       ALGERIA|
| 157|Customer#000000157|HGEouzCcFrNd nBAd...|25-207-442-1556|        9768.73|       MOROCCO|
| 683|Customer#000000683|G0, q8c6vBykpiLvc...|16-566-251-5446|        9120.93|        FRANCE|
| 200|Customer#000000200|x1 H5c66DUgH2pgNT...|26-472-302-4189|         9967.6|    MOZAMBIQUE|
|1370|Customer#000001370|WN7onCgcC,,Lt4dC4...|28-575-379-5893|        9802.04|         CHINA|
| 100|Customer#000000100|      fptUABXcmkC5Wx|30-749-445-4907|        9889.89|  SAUDI ARABIA|
| 197|Customer#000000197|   UeVqssepNuXmtZ38D|11-107-312-6585|        9860.22|     ARGENTINA|
| 468|Customer#000000468|   IcbihAtOVWcnswfyE|20-489-960-5023|        9834.19|          IRAN|
|1130|Customer#000001130|60zzrBpFXjvHzyv0W...|32-503-721-8203|        9519.36|        RUSSIA|
| 780|Customer#000000780|      CMxcdzgEUkCWP1|18-844-576-7345|        9874.12|         INDIA|
| 800|Customer#000000800|mpI6pkdnWLZsBbQi4...|24-555-630-2261|        9443.39|         KENYA|
| 219|Customer#000000219|eTjiL01eyoKiAe2WQ...|21-159-138-6090|        9858.57|          IRAQ|
|1106|Customer#000001106| WZEExIU9g2smcowcinj|31-214-739-2409|        9977.62|       VIETNAM|
| 381|Customer#000000381|    w3zVseYDbjBbzLld|15-860-208-7093|        9931.71|      ETHIOPIA|
|  45|Customer#000000045|4v3OcpFgoOmMG,Cbn...|19-715-298-9917|        9983.38|     INDONESIA|
| 213|Customer#000000213|   NpqMYBhBcWk8mnEta|34-768-700-9764|        9987.71| UNITED STATES|
+----+------------------+--------------------+--------------+---------------+--------------+
```

- **Top 10 Customer Details**

```
customerDf.join(broadcast(nationDf),customerDf.c_nationkey==nationDf.n_nationkey,"left").withColumn("Ranking",dense_rank().over(Window.orderBy(col("c_acctbal").desc()))).filter("Ranking<=10").select(col("c_custkey").alias("Key"),col("c_name").alias("Name"),col("c_address").alias("Address"),col("c_phone").alias("Contact No"),col("c_acctbal").alias("Account Balance"),col("n_name").alias("Nation"),"Ranking").show()
```

```
+----+------------------+--------------------+--------------+--------------+-------------+-------+
| Key|              Name|             Address|    Contact No|Account Balance|       Nation|Ranking|
+----+------------------+--------------------+--------------+--------------+-------------+-------+
| 213|Customer#000000213|   NpqMYBhBcWk8mnEta|34-768-700-9764|        9987.71|UNITED STATES|      1|
|  45|Customer#000000045|4v3OcpFgoOmMG,Cbn...|19-715-298-9917|        9983.38|    INDONESIA|      2|
|1106|Customer#000001106| WZEExIU9g2smcowcinj|31-214-739-2409|        9977.62|      VIETNAM|      3|
| 200|Customer#000000200|x1 H5c66DUgH2pgNT...|26-472-302-4189|         9967.6|   MOZAMBIQUE|      4|
| 140|Customer#000000140|XRqEPiKgcETII,iOL...|14-273-885-6505|        9963.15|        EGYPT|      5|
| 381|Customer#000000381|    w3zVseYDbjBbzLld|15-860-208-7093|        9931.71|     ETHIOPIA|      6|
|  43|Customer#000000043|ouSbjHk8lh5fKX3zG...|29-316-665-2897|        9904.28|      ROMANIA|      7|
| 100|Customer#000000100|      fptUABXcmkC5Wx|30-749-445-4907|        9889.89| SAUDI ARABIA|      8|
| 780|Customer#000000780|      CMxcdzgEUkCWP1|18-844-576-7345|        9874.12|        INDIA|      9|
| 518|Customer#000000518|EsCrt4chk,3IRIzwM...|27-651-256-7682|        9871.66|         PERU|     10|
+----+------------------+--------------------+--------------+--------------+-------------+-------+
```

Details of Top 10 Customers WorldWide

C Acctbal	C Address	C Custkey	C Name	C Phone	N Name	
9987.71	NpqMYBhBcWk8mn..	213	Customer#0000002..	34-768-700-9764	UNITED STATES	1
9983.38	4v3OcpFgoOmMG,C..	45	Customer#0000000..	19-715-298-9917	INDONESIA	2
9977.62	WZEExIU9g2smcow..	1106	Customer#0000011..	31-214-739-2409	VIETNAM	3
9967.6	x1 H5c66DUgH2pgN..	200	Customer#0000002..	26-472-302-4189	MOZAMBIQUE	4
9963.15	XRqEPiKgcETII,iOLD..	140	Customer#0000001..	14-273-885-6505	EGYPT	5
9931.71	w3zVseYDbjBbzLld	381	Customer#0000003..	15-860-208-7093	ETHIOPIA	6
9904.28	ouSbjHk8lh5fKX3zG..	43	Customer#0000000..	29-316-665-2897	ROMANIA	7
9889.89	fptUABXcmkC5Wx	100	Customer#0000001..	30-749-445-4907	SAUDI ARABIA	8
9874.12	CMxcdzgEUkCWP1	780	Customer#0000007..	18-844-576-7345	INDIA	9
9871.66	EsCrt4chk,3IRIzwM..	518	Customer#0000005..	27-651-256-7682	PERU	10

Rank_AcctBal

1 10

9.1.6 Using spark-sql and DSL in a single query

Query 9: Company wants to know the details of suppliers who were not able to ship required parts on time

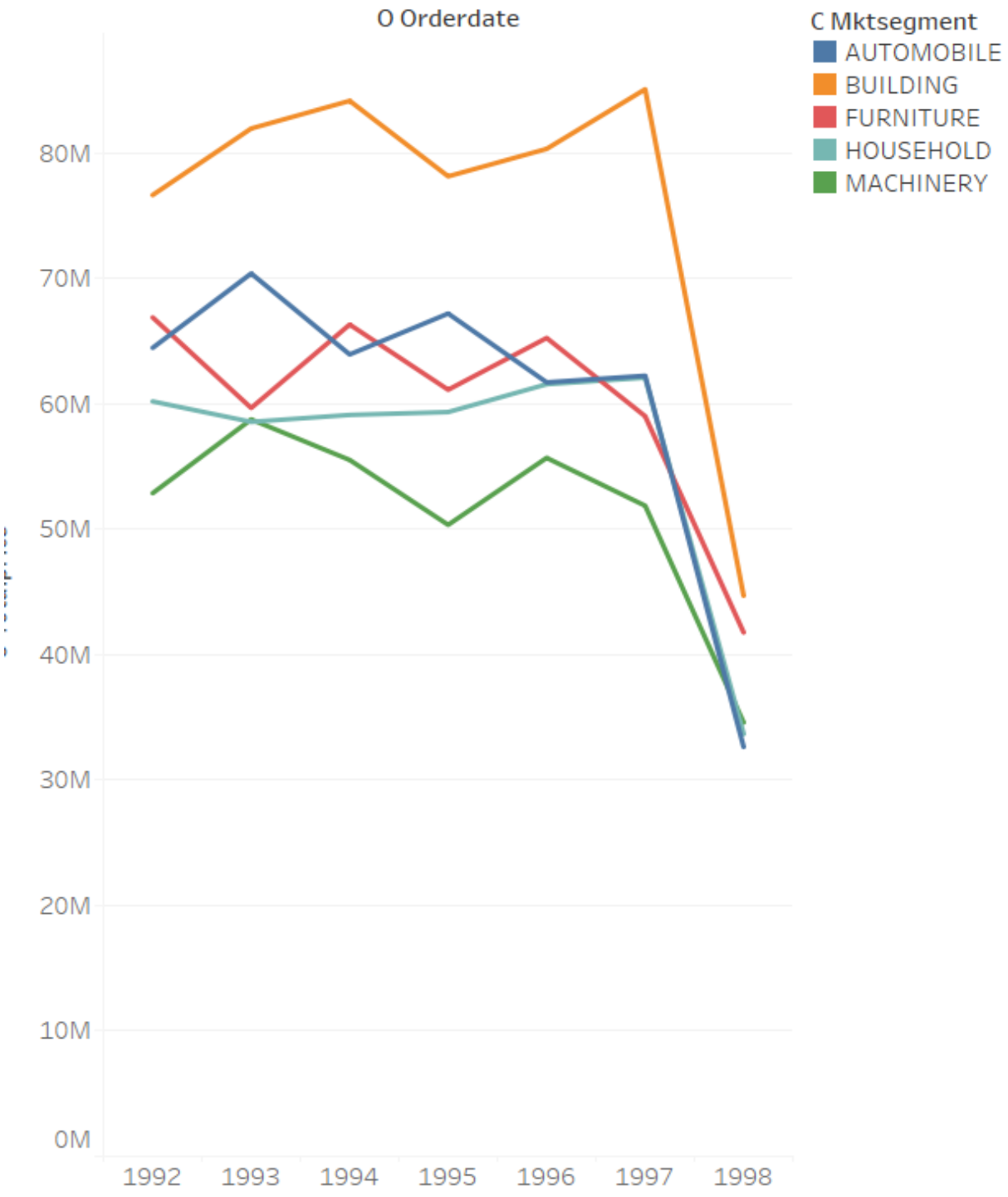
- ❏ Orderstatus = ‘F’ (Failed)
- ❏ Commitdate < Receiptdate
- ❏ Part of a multi supplier order where they were the only supplier who failed to meet the deadline.

```
spark.sql("select s_name, count(*) as numwait from supplier,
lineitem l1, orders, nation where s_suppkey = l1.l_suppkey and
o_orderkey = l1.l_orderkey and o_orderstatus = 'F' and
l1.l_receiptdate > l1.l_commitdate and exists ( select * from
lineitem l2 where l2.l_orderkey = l1.l_orderkey and
l2.l_suppkey <> l1.l_suppkey ) and not exists ( select * from
lineitem l3 where l3.l_orderkey = l1.l_orderkey and
l3.l_suppkey <> l1.l_suppkey and l3.l_receiptdate >
l3.l_commitdate ) and s_nationkey = n_nationkey group by
s_name").withColumnRenamed("s_name","Supplier
Name").orderBy("numwait",ascending=False).withColumnRenamed("n
umwait","Number of Items Waiting").show()
```

```
+------------------+-----------------------+
|     Supplier Name|Number of Items Waiting|
+------------------+-----------------------+
|Supplier#000000028|                     21|
|Supplier#000000024|                     20|
|Supplier#000000081|                     19|
|Supplier#000000038|                     18|
|Supplier#000000056|                     18|
|Supplier#000000090|                     17|
|Supplier#000000066|                     16|
|Supplier#000000043|                     16|
|Supplier#000000070|                     16|
|Supplier#000000094|                     14|
|Supplier#000000026|                     14|
|Supplier#000000012|                     14|
|Supplier#000000099|                     14|
|Supplier#000000039|                     14|
|Supplier#000000050|                     13|
|Supplier#000000017|                     13|
|Supplier#000000033|                     13|
|Supplier#000000073|                     13|
|Supplier#000000036|                     13|
|Supplier#000000086|                     13|
+------------------+-----------------------+
only showing top 20 rows
```

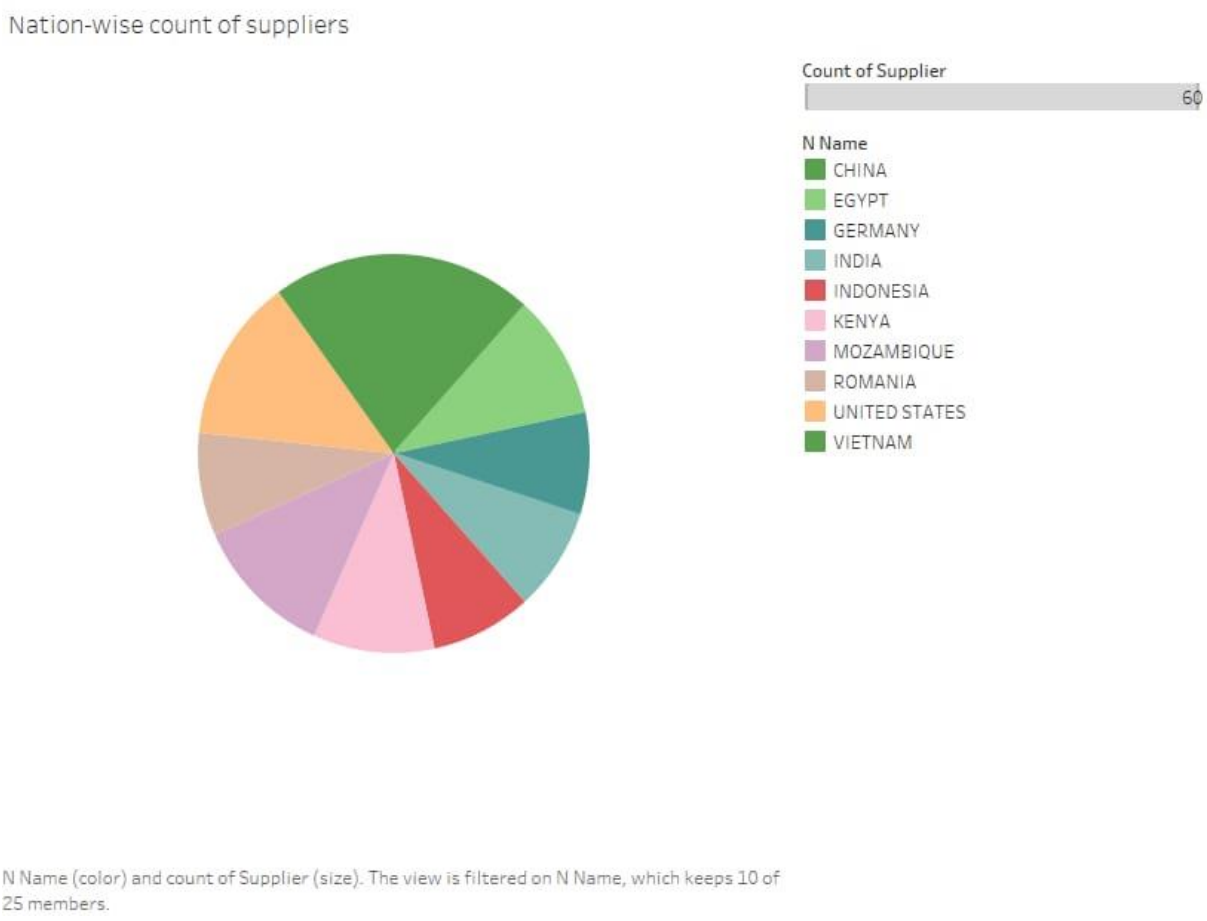
Query 10: Company wants to know the Market Segments that are profitable

```
customerDf.join(ordersDf,customerDf.c_custkey == ordersDf.o_custkey,"inner").groupBy(year("o_orderdate"),"c_mktsegment").sum("o_totalprice").orderBy(year("o_orderdate"),"c_mktsegment").withColumnRenamed("year(o_orderdate)","Year").withColumnRenamed("sum(o_totalprice)","Sum").show()
```



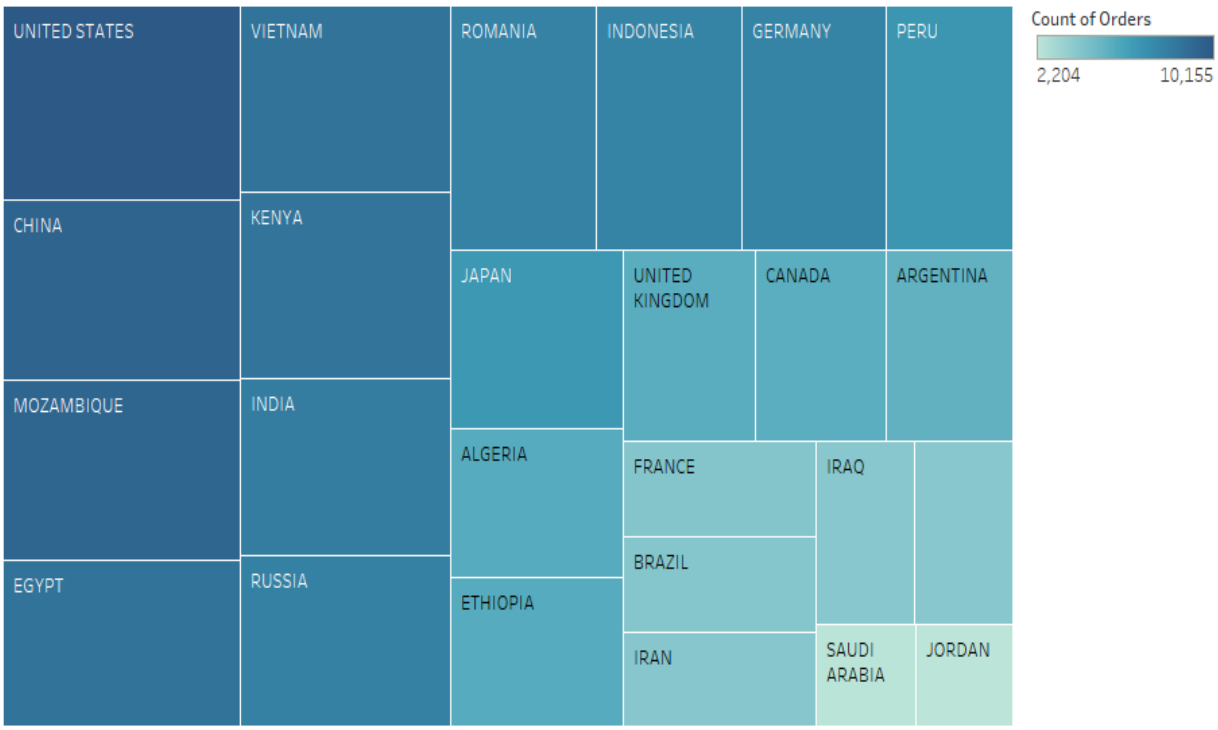
Query 11: Company wants to know Nation-wise count of suppliers

```
supplierDf.groupBy(supplierDf.S_NATIONKEY).count().show()
```



Query 12: Company wants to know nation-wise count of orders

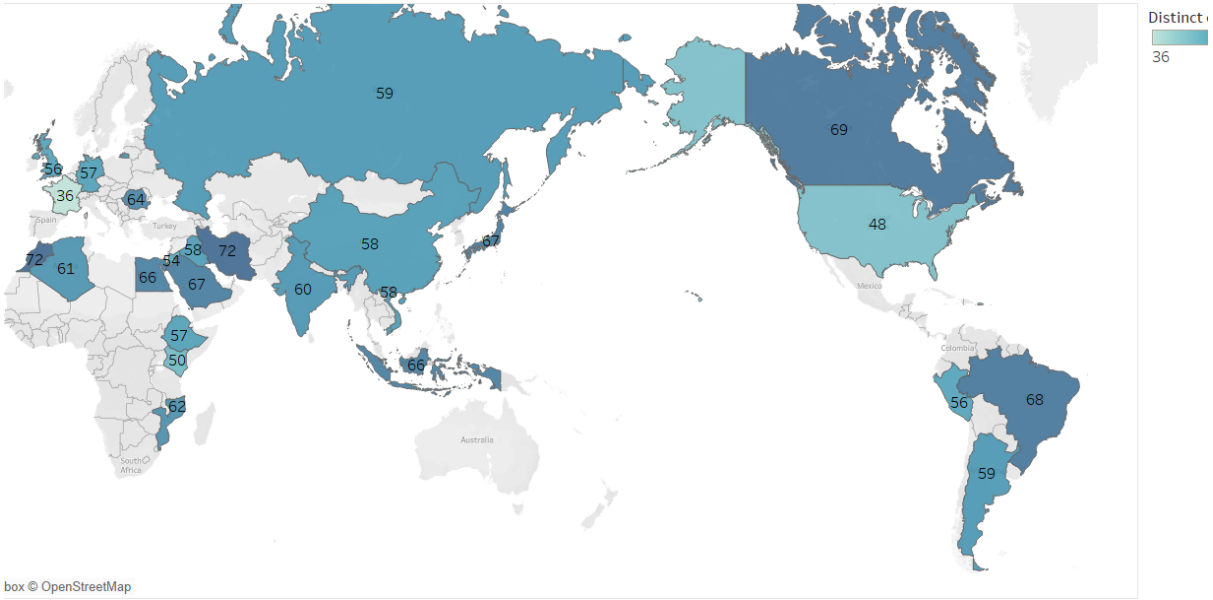
```
ordersDf.join(broadcast(customerDf),customerDf.c_custkey==ordersDf.o_custkey,"inner").join(broadcast(nationDf),nationDf.n_nationkey==customerDf.c_nationkey,"inner").groupBy("n_name").agg(count("o_orderkey").alias("Count")).show()
```



N Name. Color shows count of Orders. Size shows count of Orders. The marks are labeled by N Name.

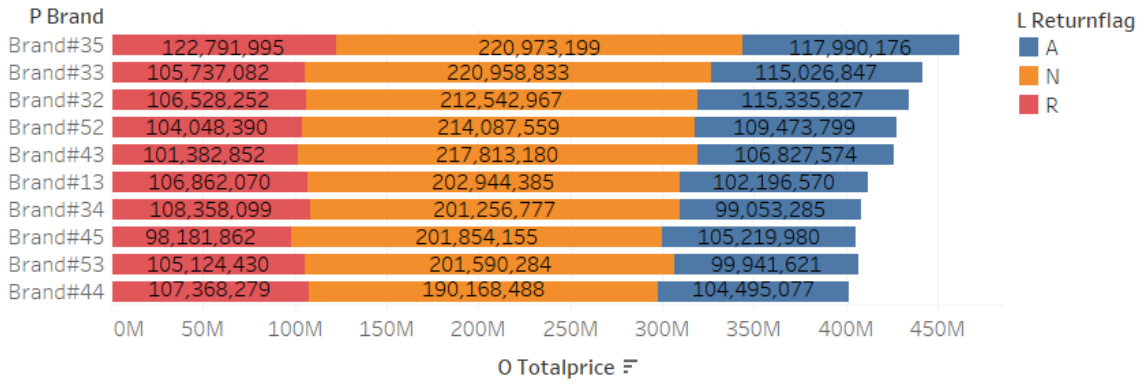
Query 13: Company wants to know Concentration of customers in each nation

```
customerDf.join(nationDf,
customerDf.C_NATIONKEY==nationDf.N_NATIONKEY,
"inner").groupBy(customerDf.C_NATIONKEY,
nationDf.N_NAME).count().show()
```



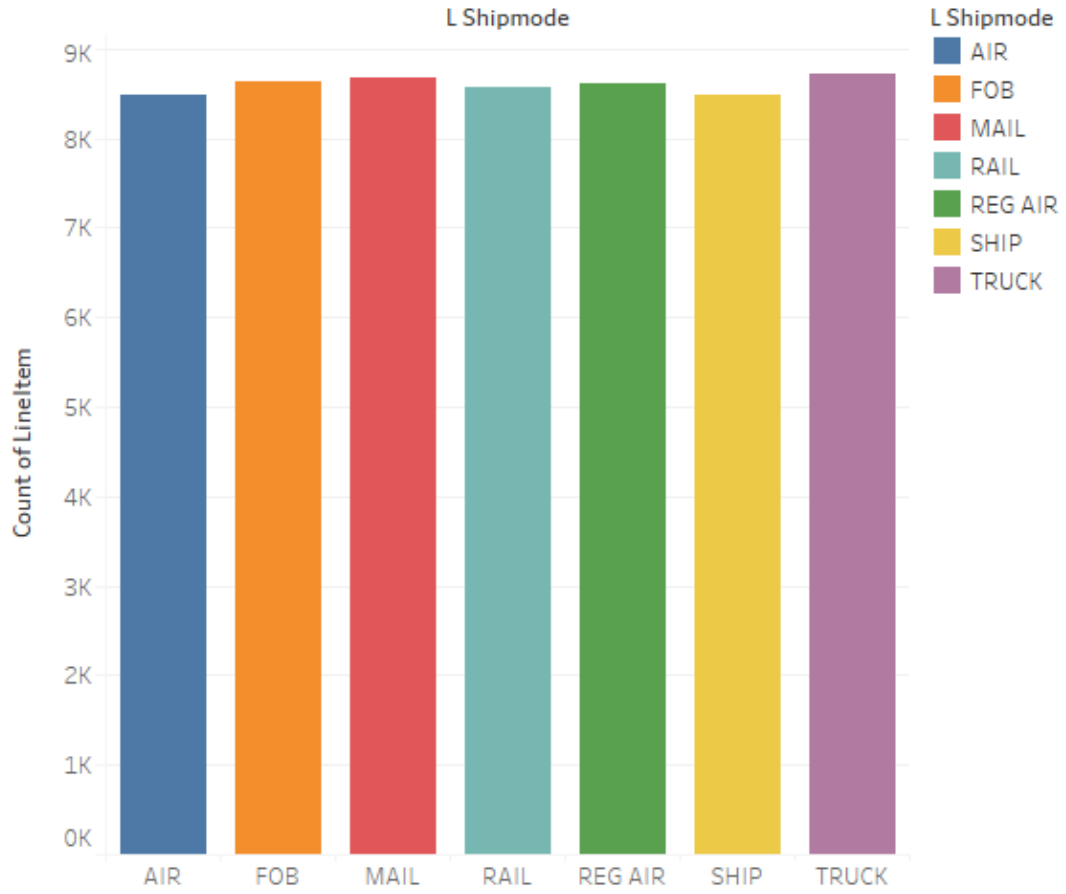
Query 14: Company wants to know their top 10 brands ordered as per their contribution to sales and segregated by Returnflag

```
lineitemDf.join(ordersDf,lineitemDf.l_orderkey==ordersDf.o_orderkey,"inner").join(broadcast(partDf),partDf.p_partkey==lineitemDf.l_partkey,"inner").groupBy("p_brand","l_returnflag").agg(sum("o_totalprice").alias("Sales")).orderBy("Sales").show(10)
```



Query15: Count of orders with respect to Ship-Modes available

```
lineitemDf.groupBy(lineitemDf.L_SHIPMODE).count().show()
```

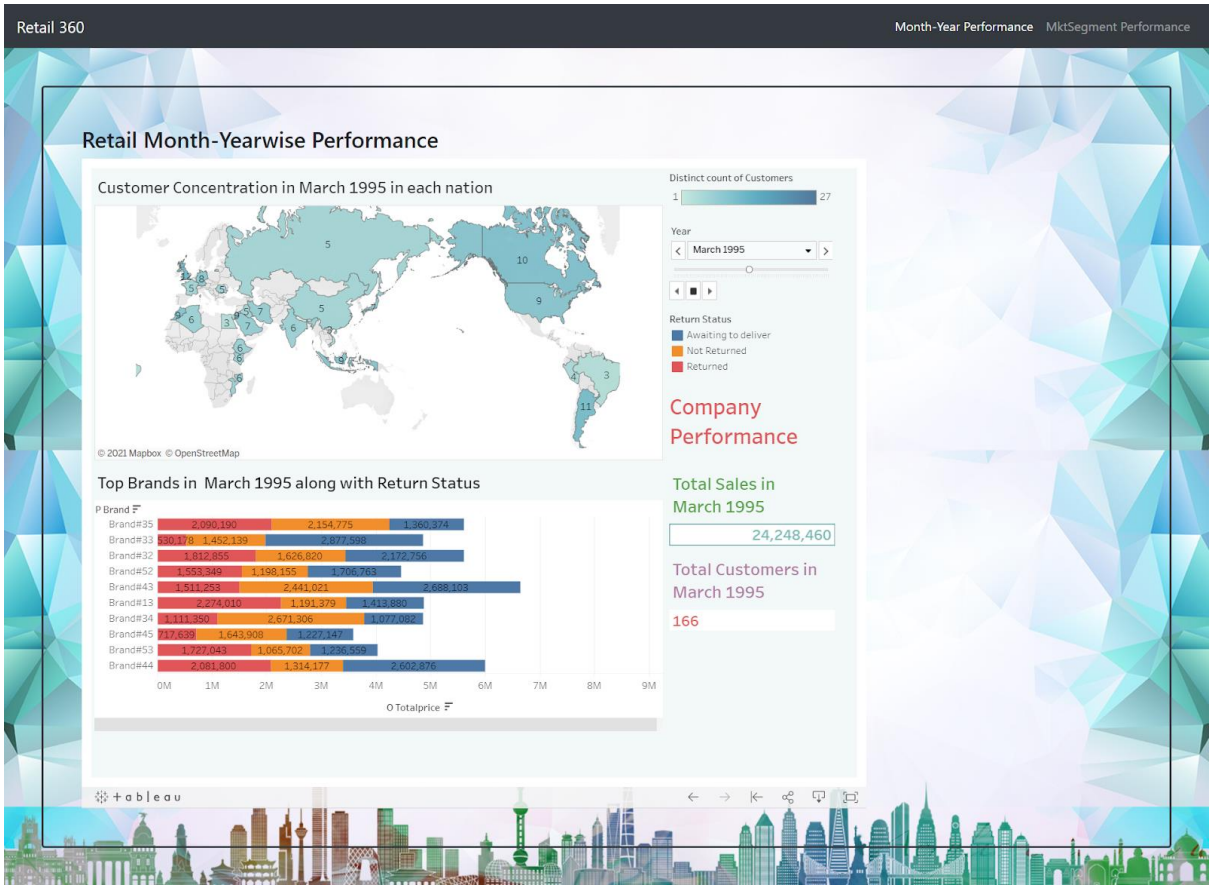


Count of LineItem for each L Shipmode. Color shows details about L Shipmode.

Task 10: Embedding interactive tableau dashboards and worksheets into website

==========

Embedded views update as the underlying data changes, or as their workbooks are updated on Tableau Server or Tableau Online, so it would become easier to keep the users updated with the latest information.



CONCLUSION:

==========

We have successfully completed the analysis of the provided retail data using the big data tools we learned during the course of our training.