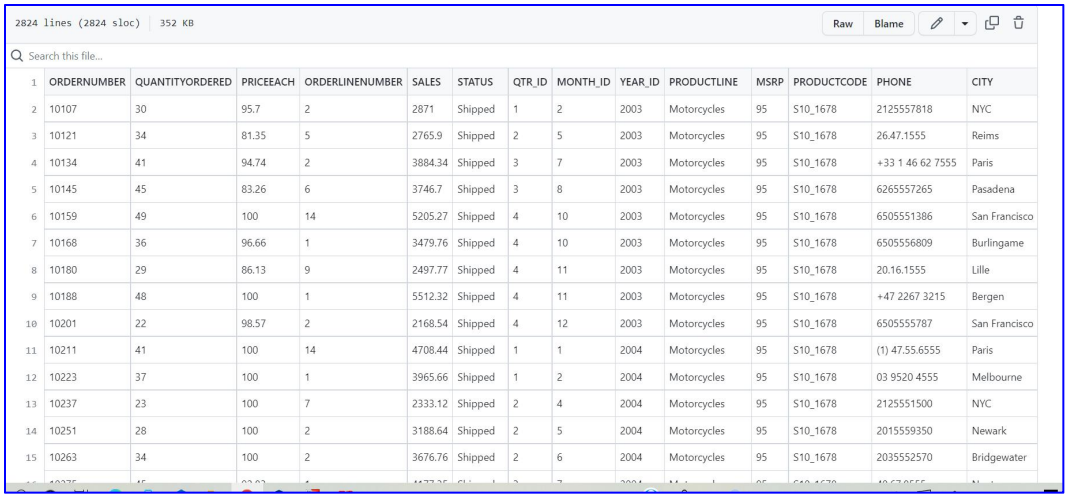
PROJECT DESCRIPTION:- The Project aims to display Vehicle Sales Data of Different Countries of the World. The dataset contains various attributes of the Sales data ranging from year 2003 to 2005. The Dataset is stored in the form of CSV file and the project aims to stage it to HDFS and further analysis is to be performed using HIVE.

DESCRIPTION OF DATASET:- The dataset contains almost 2900 rows and 21 columns(attributes) about the vehicle sales data. The Unique attributes of the dataset and their data-types are :

- ORDERNUMBER int,
- QUANTITYORDERED int,
- PRICEEACH float,
- ORDERLINENUMBER int,
- SALES float,
- STATUS string,
- QTR_ID int,
- MONTH_ID int,
- YEAR_ID int,
- PRODUCTLINE string,
- MSRP int,
- PRODUCTCODE string,
- PHONE string,
- CITY string,
- STATE string,
- POSTALCODE string,
- COUNTRY string,
- TERRITORY string,
- CONTACTLASTNAME string,
- CONTACTFIRSTNAME string,
- DEALSIZE string



2824 lines (2824 sloc) | 352 KB — Raw | Blame

Search this file...

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE	PHONE	CITY
2	10107	30	95.7	2	2871	Shipped	1	2	2003	Motorcycles	95	S10_1678	2125557818	NYC
3	10121	34	81.35	5	2765.9	Shipped	2	5	2003	Motorcycles	95	S10_1678	26.47.1555	Reims
4	10134	41	94.74	2	3884.34	Shipped	3	7	2003	Motorcycles	95	S10_1678	+33 1 46 62 7555	Paris
5	10145	45	83.26	6	3746.7	Shipped	3	8	2003	Motorcycles	95	S10_1678	6265557265	Pasadena
6	10159	49	100	14	5205.27	Shipped	4	10	2003	Motorcycles	95	S10_1678	6505551386	San Francisco
7	10168	36	96.66	1	3479.76	Shipped	4	10	2003	Motorcycles	95	S10_1678	6505556809	Burlingame
8	10180	29	86.13	9	2497.77	Shipped	4	11	2003	Motorcycles	95	S10_1678	20.16.1555	Lille
9	10188	48	100	1	5512.32	Shipped	4	11	2003	Motorcycles	95	S10_1678	+47 2267 3215	Bergen
10	10201	22	98.57	2	2168.54	Shipped	4	12	2003	Motorcycles	95	S10_1678	6505555787	San Francisco
11	10211	41	100	14	4708.44	Shipped	1	1	2004	Motorcycles	95	S10_1678	(1) 47.55.6555	Paris
12	10223	37	100	1	3965.66	Shipped	1	2	2004	Motorcycles	95	S10_1678	03 9520 4555	Melbourne
13	10237	23	100	7	2333.12	Shipped	2	4	2004	Motorcycles	95	S10_1678	2125551500	NYC
14	10251	28	100	2	3188.64	Shipped	2	5	2004	Motorcycles	95	S10_1678	2015559350	Newark
15	10263	34	100	2	3676.76	Shipped	2	6	2004	Motorcycles	95	S10_1678	2035552570	Bridgewater

The glimpse of the dataset is shown above.

PROBLEM STATEMENT:-

- Store raw data into hdfs location
- Create a internal hive table "sales_order_csv" which will store csv data sales_order_csv .. make sure to skip header row while creating table
- Load data from hdfs path into "sales_order_csv"
- Create an internal hive table which will store data in ORC format "sales_order_orc"
- Load data from "sales_order_csv" into "sales_order_orc".

Placing the given Dataset in the HDFS:- Step1 - Create a Directory in HDFS, ensuring all the daemons are started.

```
hadoop fs -mkdir /user/hive/project
```

Step2 - Copy the Dataset from local to HDFS directory.

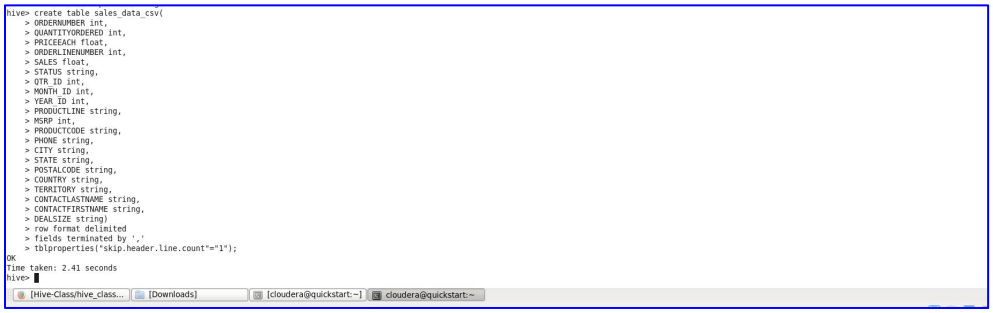
```
hadoop fs -copyFromLocal  /home/cloudera/Downloads/
Sales_order_data.csv /user/hive/project/sales_order_data.csv
```

Creating The Database: Create a database named ‘hive_assignment’, and use it for further purpose.

- Create database hive_assignment;
- Use hive_assignment;

Creating the internal hive table : After creating database , we have to create an internal table.

```
create table sales_order_data_csv
(ORDERNUMBER int, QUANTITYORDERED int,
PRICEEACH float, ORDERLINENUMBER int,
SALES float, STATUS string,
QTR_ID int, MONTH_ID int,
YEAR_ID int, PRODUCTLINE string,
MSRP int, PRODUCTCODE string,
PHONE string, CITY string,
STATE string, POSTALCODE string,
COUNTRY string, TERRITORY string,
CONTACTLASTNAME string,
CONTACTFIRSTNAME string, DEALSIZE string)
row format delimited
fields terminated by ','
tblproperties("skip.header.line.count"="1");
```



Creating Internal table

Loading data from HDFS path to the table created: After creating the table sales_order_csv, we have to load the data content inside it.

load data local inpath '/home/cloudera/Downloads/sales_order_data.csv' into table sales_order_data.csv;

Create an internal hive table which will store data in ORC format "sales_order_orc" :

```
create table sales_data_orc
(ORDERNUMBER int, QUANTITYORDERED int,
PRICEEACH float, ORDERLINENUMBER int,
SALES float, STATUS string,
QTR_ID int, MONTH_ID int,
YEAR_ID int, PRODUCTLINE string,
MSRP int, PRODUCTCODE string,
PHONE string, CITY string,
STATE string, POSTALCODE string,
COUNTRY string, TERRITORY string,
CONTACTLASTNAME string,
CONTACTFIRSTNAME string, DEALSIZE string)
Stored as orc;
```

Load data from "sales_order_csv" into "sales_order_orc": Now loading the content from csv table to orc table.

```
From sales_order_data.csv insert overwrite table sales_data_orc
select *;
```

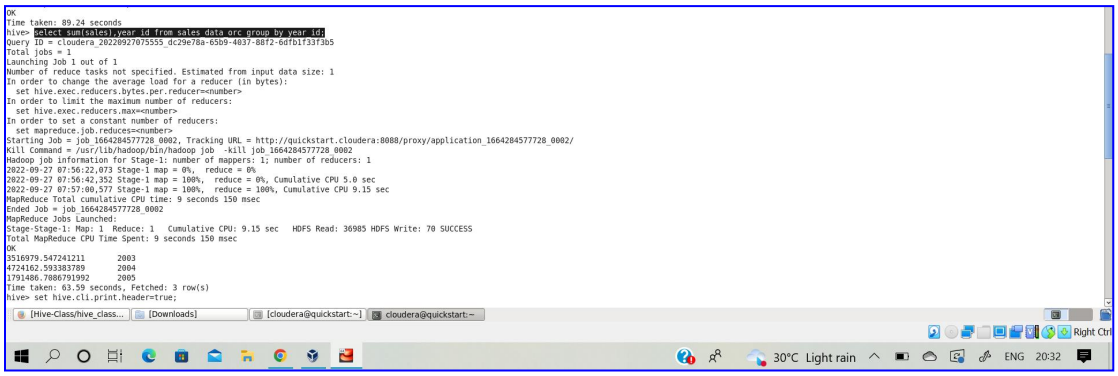
The above figure depicts the successful loading of data from csv table to sales_data_orc table. Now that the data is loaded in Hive table, further analysis is to be performed in Hive shell.

PROBLEM SCENARIO:

Below are some problem statements that are to be executed using Hive commands on the sales_orc table.

1) Calculate total sales per year.

Select sum(sales) , year_id from sales_data_orc
group by year_id;



After running the above query , we got the desired output of total sales year wise.

Total Sales	Year
3516979.5472	2003
47224162.5933	2004
1791486.7086	2005

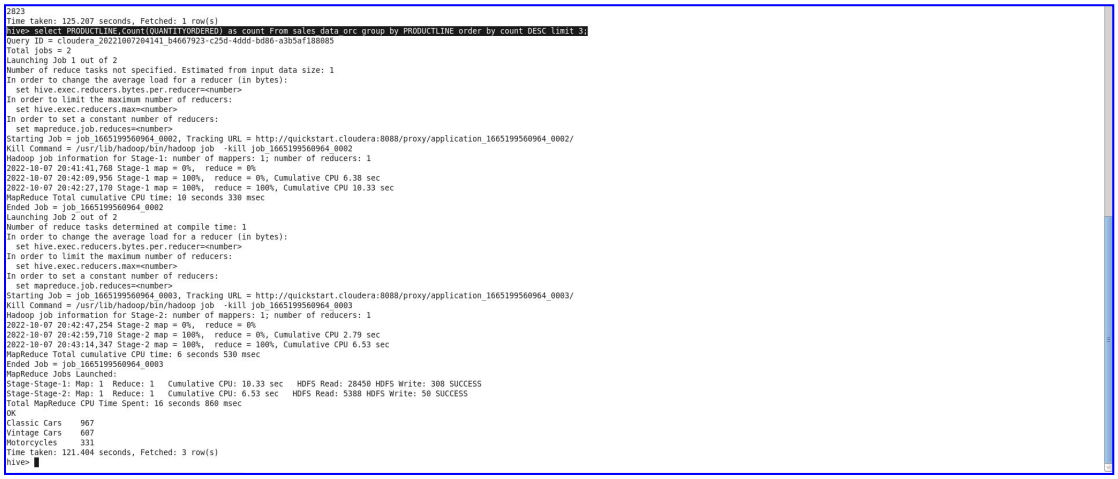
From the output, it can be concluded that the year 2004 had the highest amount of sales in comparison to other two years.

2) Find a product for which maximum orders were placed.

Select PRODUCTLINE, Count(QUANTITYORDERED) as count From sales_data_orc
Group by PRODUCTLINE
Order by count Desc limit 3;

After running the above query , the output is:

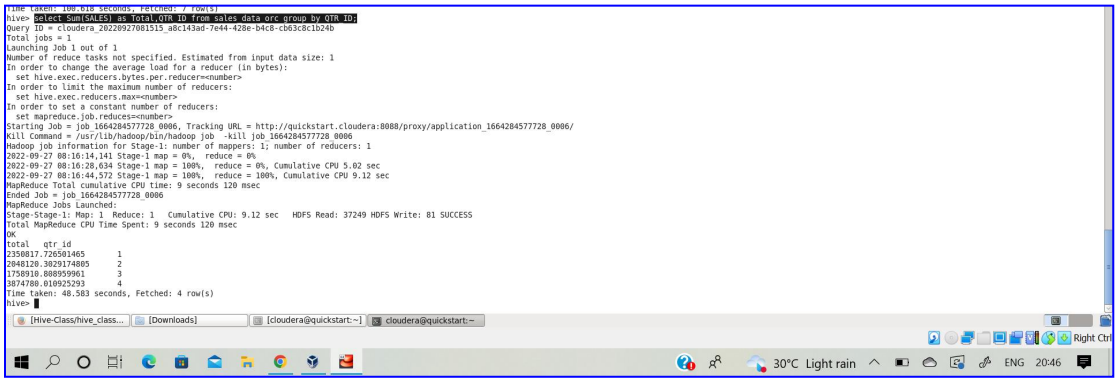
PRODUCTLINE	MAXIMUM
Classic Cars	967
Vintage Cars	607
Motorcycles	331



Here, it can be concluded that Classic cars were sold mostly in the year of 2003 - 2005 across different parts of the world.

3) Calculate Total sales for each quarter.

Select sum(SALES) as Total, QTR_ID from sales_data_orc Group by QTR_ID;

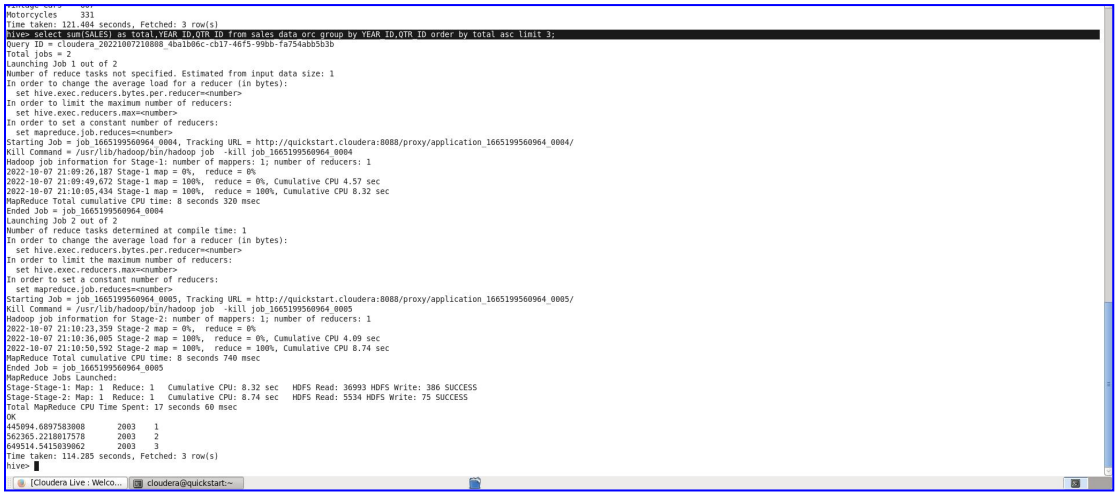


The result of the above query is shown in the above image. It can be concluded that the last quarter_id has the most amount of sales.

Total	Qtr_Id
2350817.726	1
2048120.302	2
1758910.808	3
3874780.010	4

4) In which quarter sales was minimum:

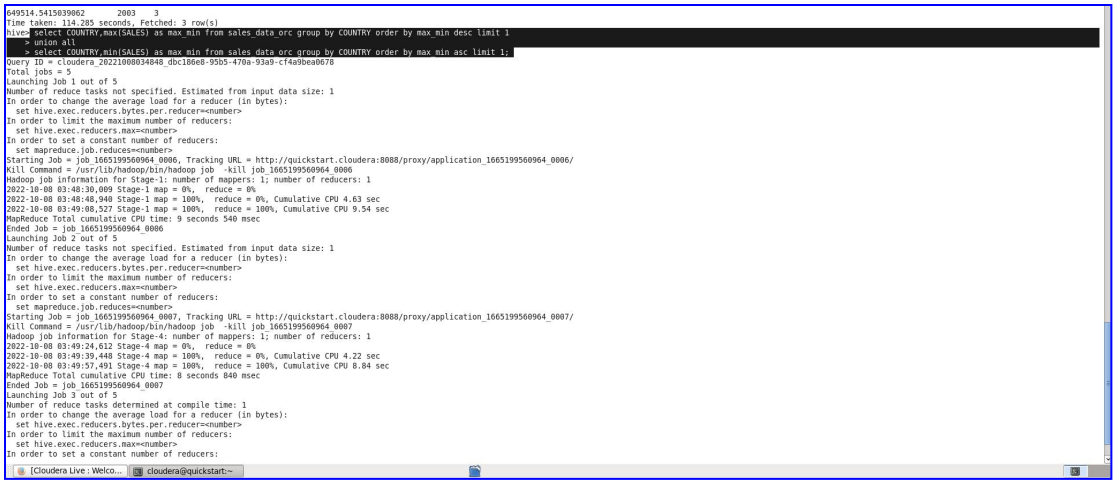
```
Select sum(SALES) as Sum, QTR_ID,YEAR_ID
from sales_data_orc
Group by QTR_ID,YEAR_ID
Order by sum Asc limit 3;
```

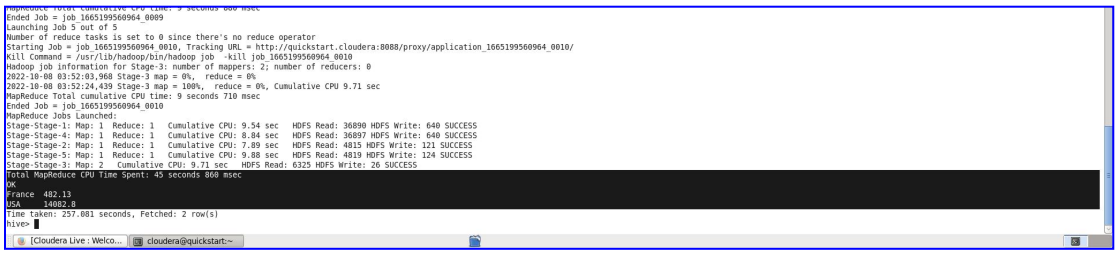


From the above output, Quarter Id 1 in year 2003 has the minimum amount of Sales recorded of value “445094.68”.

5) In which Country Sales was Maximum and in which country sales was minimum.

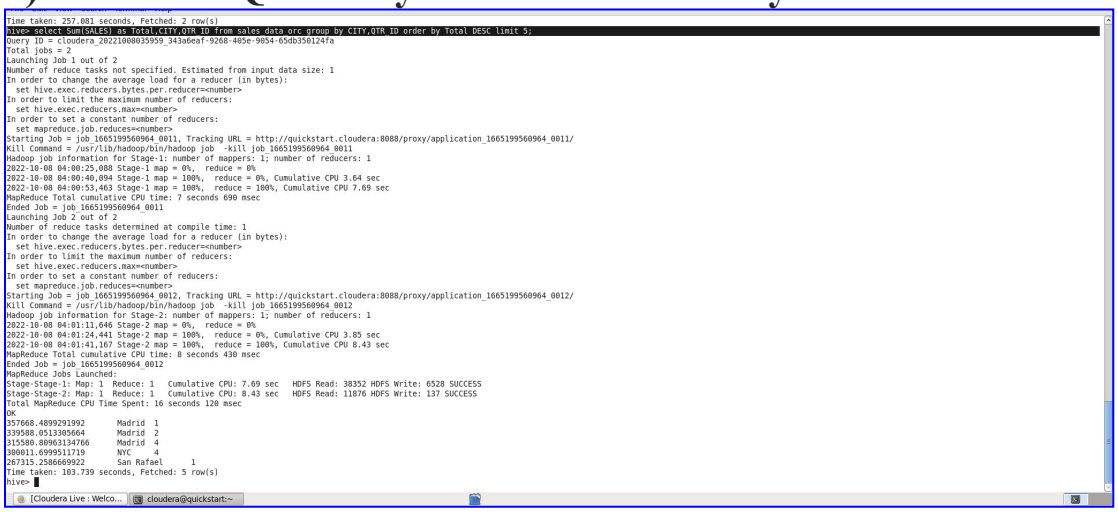
```
Select max(SALES) as Min_Max, COUNTRY
From Sales_data_orc group by COUNTRY
Order by Min_Max DESC limit 1
UNION ALL
Select Min(SALES) as Min_Max,COUNTRY
From Sales-data_orc group by COUNTRY
Order by Min_Max ASC limit 1;
```





The above two image depicts the final output. The Country of **USA** has the **maximum** amount of Sales of value “14082.8” while the Country of **France** has **lowest** amount of Sales recorded which was “482.3”.

6) Calculate Quarterly Sales For each City.



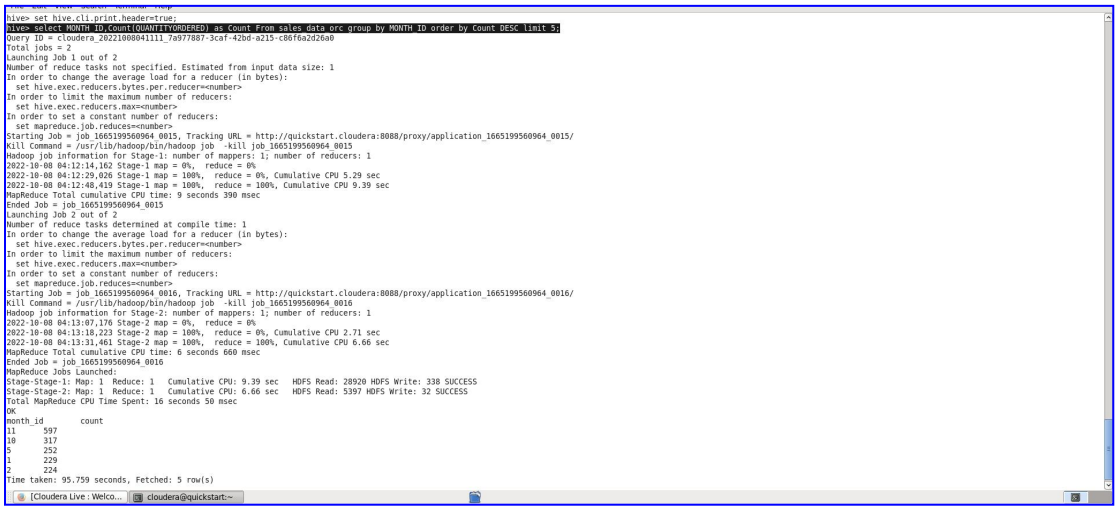

```
Select Sum(SALES) as Total , City,QTR_ID
From sales_data_orc
Group by City,QTR_ID
Order by Total Desc Limit 5;
```

The Top 5 sales city-wise recorded were:

Total	City	QTR_ID
357668.48	Madrid	1
339588.05	Madrid	2
315580.80	Madrid	4
300011.69	NYC	4
267315.25	San Rafael	1

7) Find the Month for each year in which maximum number of Quantities were sold.

```
Select MONTH_ID,Count(QUANTITYORDERED) as Count
From Sales_data_orc
Group by MONTH_ID
Order by Count Desc limit 5;
```



Month_id	Count
11	597
10	317
5	252
1	229
2	224

8) Show Distinct Countries from the dataset