Hive_Class_3_Assignment

Part-1

Below mentioned steps as been followed step by step.

1. Download vehicle sales data -> https://github.com/shashank-mishra219/Hive-Class/blob/main/sales_order_data.csv

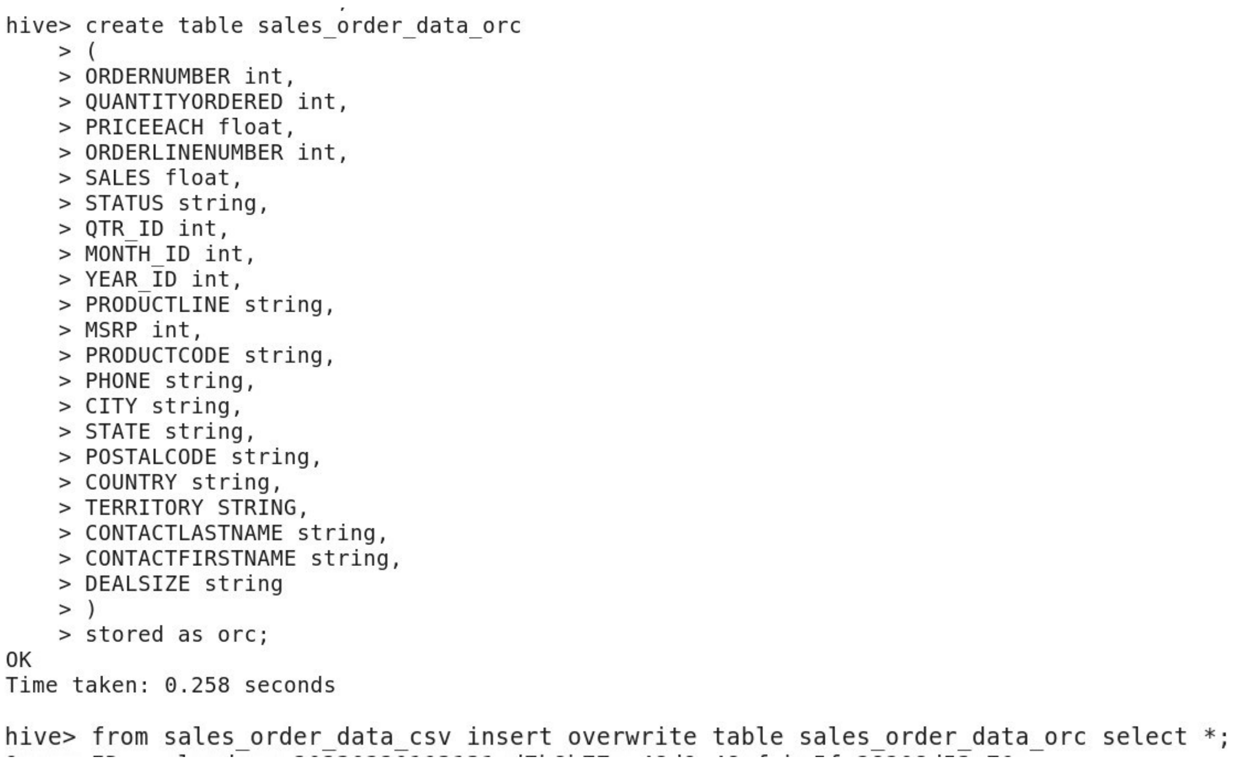
2. Store raw data into hdfs location

3. 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

4. Load data from hdfs path into "sales_order_csv"

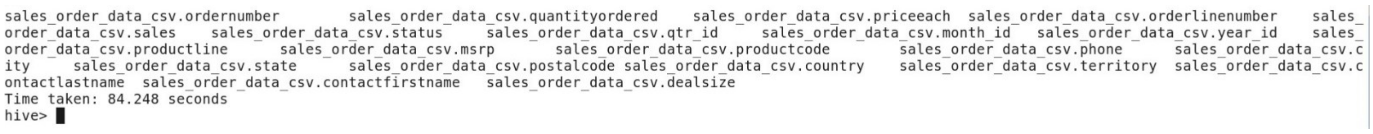
5. Create an internal hive table which will store data in ORC format "sales_order_orc"

6. Load data from "sales_order_csv" into "sales_order_orc"



```
hive> create table sales_order_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;
OK
Time taken: 0.258 seconds

hive> from sales_order_data_csv insert overwrite table sales_order_data_orc select *;
```



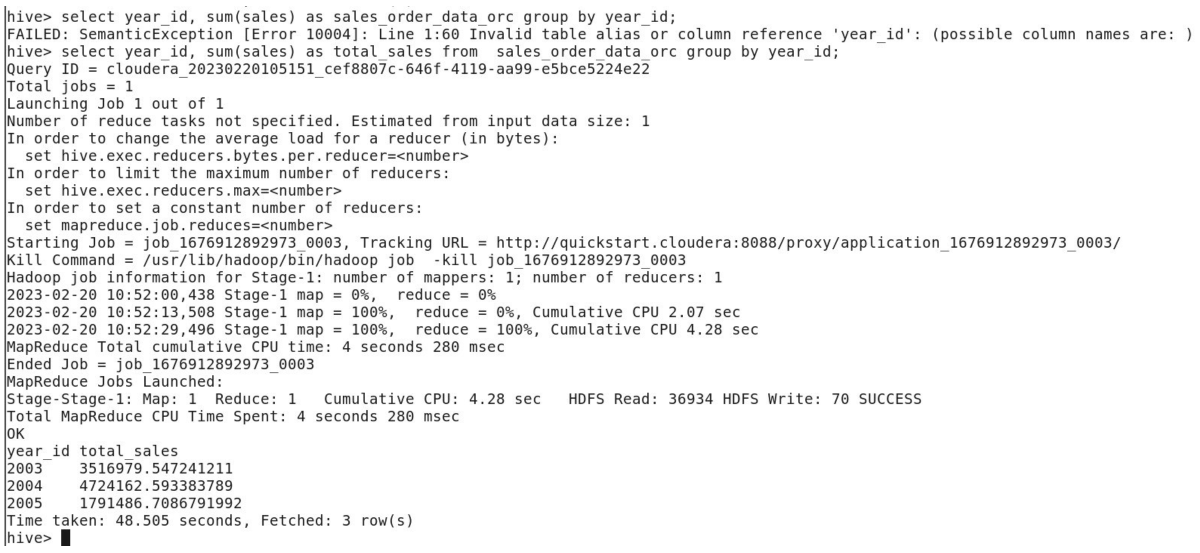
```
sales_order_data_csv.ordernumber        sales_order_data_csv.quantityordered    sales_order_data_csv.priceeach  sales_order_data_csv.orderlinenumber    sales_
order_data_csv.sales    sales_order_data_csv.status     sales_order_data_csv.qtr_id     sales_order_data_csv.month_id   sales_order_data_csv.year_id    sales_
order_data_csv.productline      sales_order_data_csv.msrp       sales_order_data_csv.productcode        sales_order_data_csv.phone      sales_order_data_csv.c
ity     sales_order_data_csv.state      sales_order_data_csv.postalcode sales_order_data_csv.country    sales_order_data_csv.territory  sales_order_data_csv.c
ontactlastname  sales_order_data_csv.contactfirstname   sales_order_data_csv.dealsize
Time taken: 84.248 seconds
hive> █
```

Perfect my ORC table is ready with data loaded in it.

Part-2

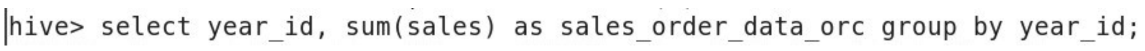
Perform below mentioned queries on "sales_order_orc" table:

a. Calculate total sales per year



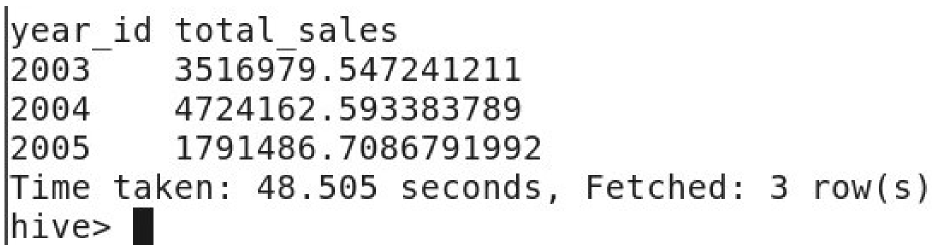
```
hive> select year_id, sum(sales) as sales_order_data_orc group by year_id;
FAILED: SemanticException [Error 10004]: Line 1:60 Invalid table alias or column reference 'year_id': (possible column names are: )
hive> select year_id, sum(sales) as total_sales from  sales_order_data_orc group by year_id;
Query ID = cloudera_20230220105151_cef8807c-646f-4119-aa99-e5bce5224e22
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1676912892973_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1676912892973_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1676912892973_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-02-20 10:52:00,438 Stage-1 map = 0%,  reduce = 0%
2023-02-20 10:52:13,508 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.07 sec
2023-02-20 10:52:29,496 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.28 sec
MapReduce Total cumulative CPU time: 4 seconds 280 msec
Ended Job = job_1676912892973_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.28 sec   HDFS Read: 36934 HDFS Write: 70 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 280 msec
OK
year_id total_sales
2003    3516979.547241211
2004    4724162.593383789
2005    1791486.7086791992
Time taken: 48.505 seconds, Fetched: 3 row(s)
hive>
```

The HQL query for this task is this as shown below figure.



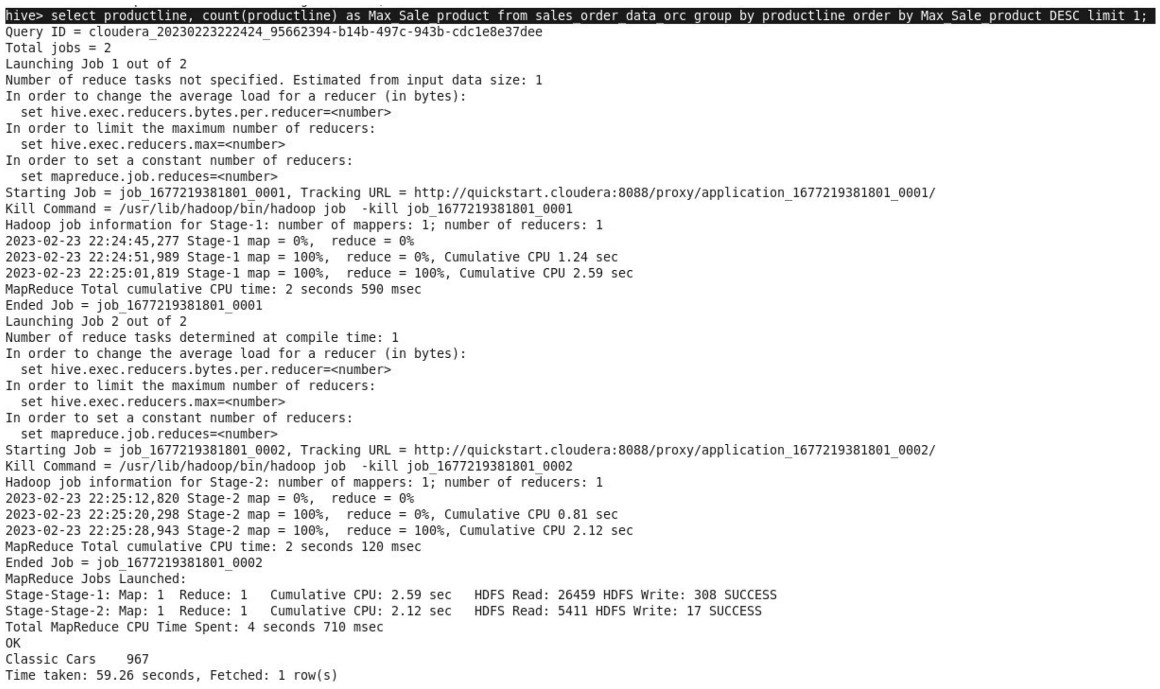
```
hive> select year_id, sum(sales) as sales_order_data_orc group by year_id;
```

Below figure shows the result.



```
year_id total_sales
2003    3516979.547241211
2004    4724162.593383789
2005    1791486.7086791992
Time taken: 48.505 seconds, Fetched: 3 row(s)
hive>
```

b. Find a product for which maximum orders were placed.



```
hive> select productline, count(productline) as Max_Sale_product from sales_order_data_orc group by productline order by Max_Sale_product DESC limit 1;
Query ID = cloudera_20230223222424_95662394-b14b-497c-943b-cdc1e8e37dee
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677219381801_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677219381801_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677219381801_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-02-23 22:24:45,277 Stage-1 map = 0%,  reduce = 0%
2023-02-23 22:24:51,989 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.24 sec
2023-02-23 22:25:01,819 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.59 sec
MapReduce Total cumulative CPU time: 2 seconds 590 msec
Ended Job = job_1677219381801_0001
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677219381801_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677219381801_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677219381801_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2023-02-23 22:25:12,820 Stage-2 map = 0%,  reduce = 0%
2023-02-23 22:25:20,298 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.81 sec
2023-02-23 22:25:28,943 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.12 sec
MapReduce Total cumulative CPU time: 2 seconds 120 msec
Ended Job = job_1677219381801_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.59 sec   HDFS Read: 26459 HDFS Write: 308 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.12 sec   HDFS Read: 5411 HDFS Write: 17 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 710 msec
OK
Classic Cars    967
Time taken: 59.26 seconds, Fetched: 1 row(s)
```

The HQL query for this task is this as shown below figure.

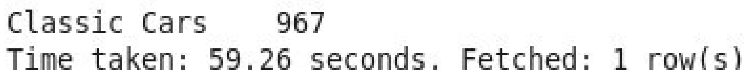
Select productline, count(prouductline) as Max_sale_product

from sales_order_data_orc

group by productline

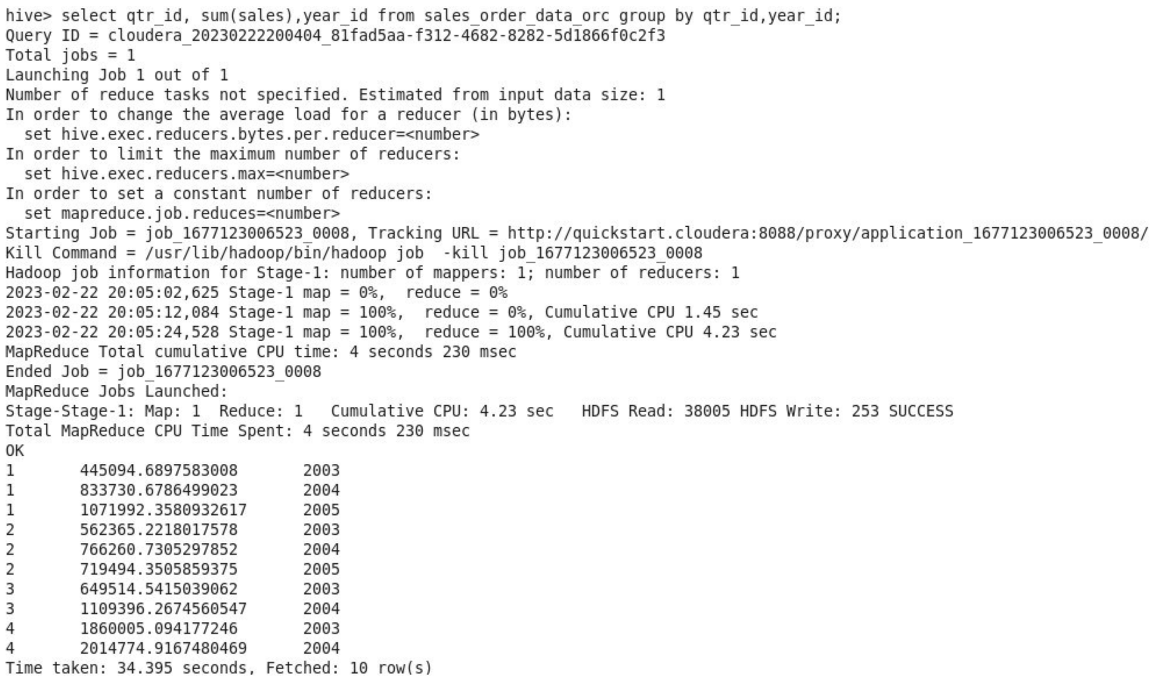
order by Max_Sale_product DESC limit 1;

Below figure shows the result.



```
Classic Cars    967
Time taken: 59.26 seconds. Fetched: 1 row(s)
```

c. Calculate the total sales for each quarter



```
hive> select qtr_id, sum(sales),year_id from sales_order_data_orc group by qtr_id,year_id;
Query ID = cloudera_20230222200404_81fad5aa-f312-4682-8282-5d1866f0c2f3
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677123006523_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677123006523_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677123006523_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-02-22 20:05:02,625 Stage-1 map = 0%,  reduce = 0%
2023-02-22 20:05:12,084 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.45 sec
2023-02-22 20:05:24,528 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.23 sec
MapReduce Total cumulative CPU time: 4 seconds 230 msec
Ended Job = job_1677123006523_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.23 sec   HDFS Read: 38005 HDFS Write: 253 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 230 msec
OK
1       445094.6897583008       2003
1       833730.6786499023       2004
1       1071992.3580932617      2005
2       562365.2218017578       2003
2       766260.7305297852       2004
2       719494.3505859375       2005
3       649514.5415039062       2003
3       1109396.2674560547      2004
4       1860005.094177246       2003
4       2014774.9167480469      2004
Time taken: 34.395 seconds, Fetched: 10 row(s)
```

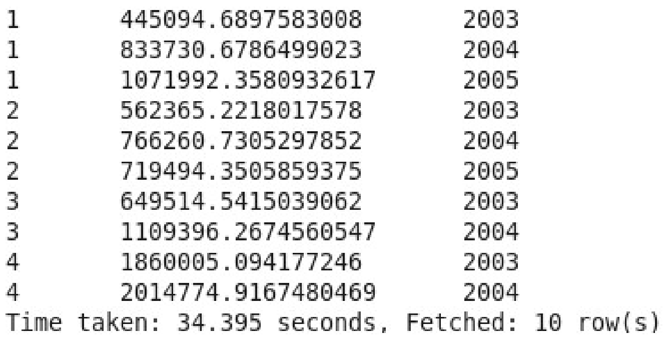
The HQL query for this task is this as shown below figure.

Select qtr_id, sum(sales), year_id

from sales_order_data_orc

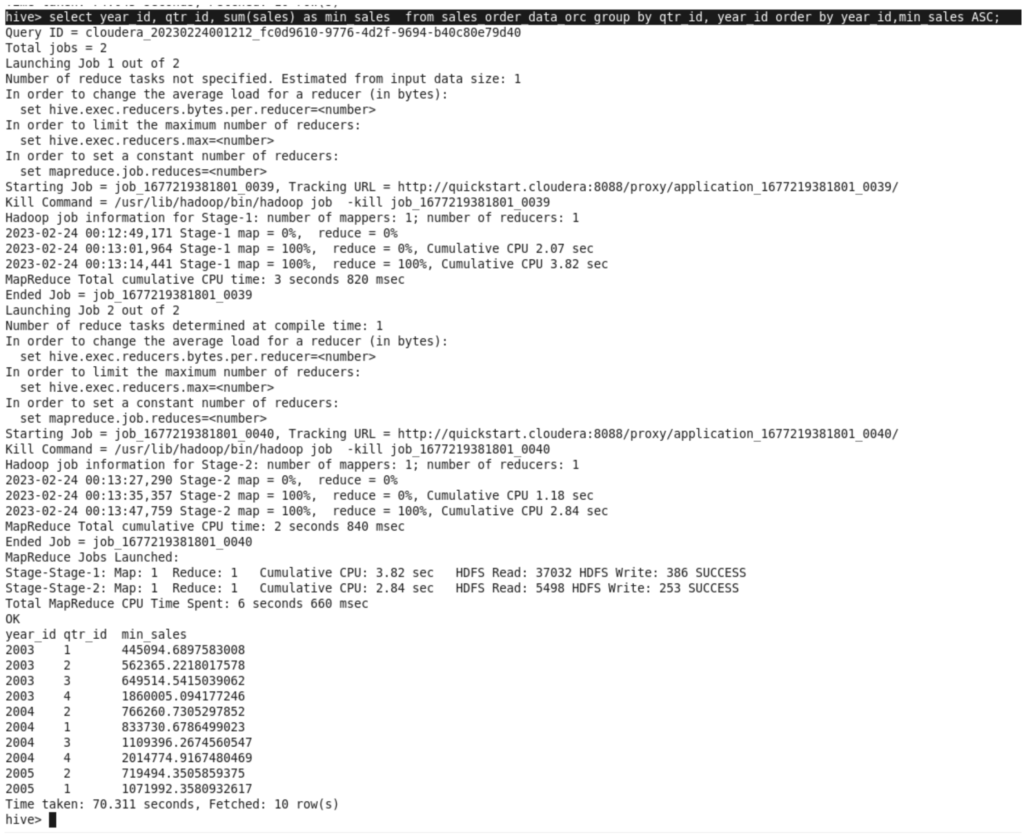
group by qtr_id, year_id

Below figure shows the result.



```
1       445094.6897583008       2003
1       833730.6786499023       2004
1       1071992.3580932617      2005
2       562365.2218017578       2003
2       766260.7305297852       2004
2       719494.3505859375       2005
3       649514.5415039062       2003
3       1109396.2674560547      2004
4       1860005.094177246       2003
4       2014774.9167480469      2004
Time taken: 34.395 seconds, Fetched: 10 row(s)
```

d. In which quarter sales was minimum



```
hive> select year_id, qtr_id, sum(sales) as min_sales  from sales_order_data_orc group by qtr_id, year_id order by year_id,min_sales ASC;
Query ID = cloudera_20230224001212_fc0d9610-9776-4d2f-9694-b40c80e79d40
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677219381801_0039, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677219381801_0039/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677219381801_0039
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-02-24 00:12:49,171 Stage-1 map = 0%,  reduce = 0%
2023-02-24 00:13:01,964 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.07 sec
2023-02-24 00:13:14,441 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.82 sec
MapReduce Total cumulative CPU time: 3 seconds 820 msec
Ended Job = job_1677219381801_0039
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677219381801_0040, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677219381801_0040/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677219381801_0040
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2023-02-24 00:13:27,290 Stage-2 map = 0%,  reduce = 0%
2023-02-24 00:13:35,357 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.18 sec
2023-02-24 00:13:47,759 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.84 sec
MapReduce Total cumulative CPU time: 2 seconds 840 msec
Ended Job = job_1677219381801_0040
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.82 sec   HDFS Read: 37032 HDFS Write: 386 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.84 sec   HDFS Read: 5498 HDFS Write: 253 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 660 msec
OK
year_id qtr_id  min_sales
2003    1       445094.6897583008
2003    2       562365.2218017578
2003    3       649514.5415039062
2003    4       1860005.094177246
2004    2       766260.7305297852
2004    1       833730.6786499023
2004    3       1109396.2674560547
2004    4       2014774.9167480469
2005    2       719494.3505859375
2005    1       1071992.3580932617
Time taken: 70.311 seconds, Fetched: 10 row(s)
hive> 
```

The HQL query for this task is this as shown below figure.

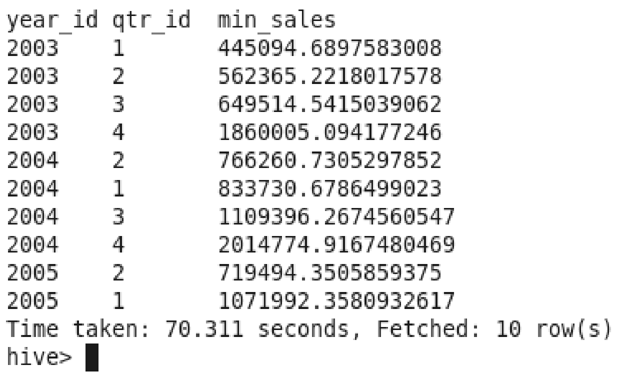
```
Select year_id, qtr_id, sum(sales) as min_sales

from sales_order_data_orc

 group by qtr_id, year_id

order by year_id, min_sales ASC;
```

Below figure shows the result.

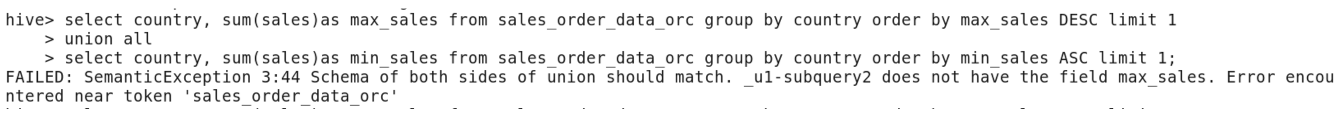


```
year_id qtr_id  min_sales
2003    1       445094.6897583008
2003    2       562365.2218017578
2003    3       649514.5415039062
2003    4       1860005.094177246
2004    2       766260.7305297852
2004    1       833730.6786499023
2004    3       1109396.2674560547
2004    4       2014774.9167480469
2005    2       719494.3505859375
2005    1       1071992.3580932617
Time taken: 70.311 seconds, Fetched: 10 row(s)
hive> 
```

In 2003 quarter 1 is less. In 2004 quarter 2 is less. In 2005 quarter 2 is less. I got the answer but I am unable to output only these tree minimum vales.

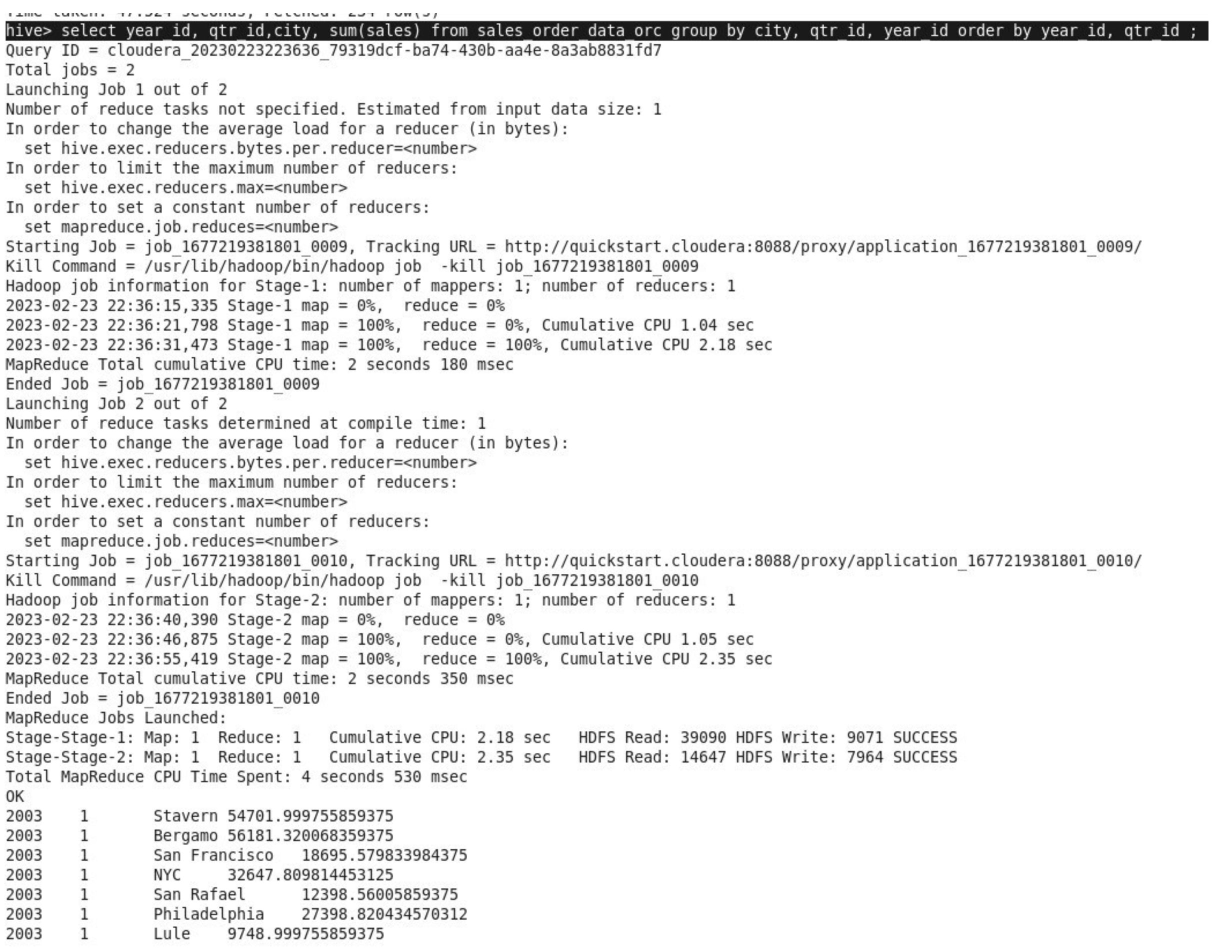
e. In which country sales was maximum and in which country sales was minimum

I have tried union all for this but it was giving me a error as shown below.



```
hive> select country, sum(sales)as max_sales from sales_order_data_orc group by country order by max_sales DESC limit 1
    > union all
    > select country, sum(sales)as min_sales from sales_order_data_orc group by country order by min_sales ASC limit 1;
FAILED: SemanticException 3:44 Schema of both sides of union should match. _u1-subquery2 does not have the field max_sales. Error encou
ntered near token 'sales_order_data_orc'
```

f. Calculate quarterly sales for each city



```
hive> select year_id, qtr_id,city, sum(sales) from sales_order_data_orc group by city, qtr_id, year_id order by year_id, qtr_id ;
Query ID = cloudera_20230223223636_79319dcf-ba74-430b-aa4e-8a3ab8831fd7
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677219381801_0009, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677219381801_0009/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677219381801_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2023-02-23 22:36:15,335 Stage-1 map = 0%,  reduce = 0%
2023-02-23 22:36:21,798 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.04 sec
2023-02-23 22:36:31,473 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.18 sec
MapReduce Total cumulative CPU time: 2 seconds 180 msec
Ended Job = job_1677219381801_0009
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1677219381801_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1677219381801_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1677219381801_0010
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2023-02-23 22:36:40,390 Stage-2 map = 0%,  reduce = 0%
2023-02-23 22:36:46,875 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.05 sec
2023-02-23 22:36:55,419 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.35 sec
MapReduce Total cumulative CPU time: 2 seconds 350 msec
Ended Job = job_1677219381801_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.18 sec   HDFS Read: 39090 HDFS Write: 9071 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.35 sec   HDFS Read: 14647 HDFS Write: 7964 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 530 msec
OK
2003    1       Stavern 54701.999755859375
2003    1       Bergamo 56181.320068359375
2003    1       San Francisco   18695.579833984375
2003    1       NYC     32647.809814453125
2003    1       San Rafael      12398.56005859375
2003    1       Philadelphia    27398.820434570312
2003    1       Lule    9748.999755859375
```

The HQL query for this task is this as shown below figure.

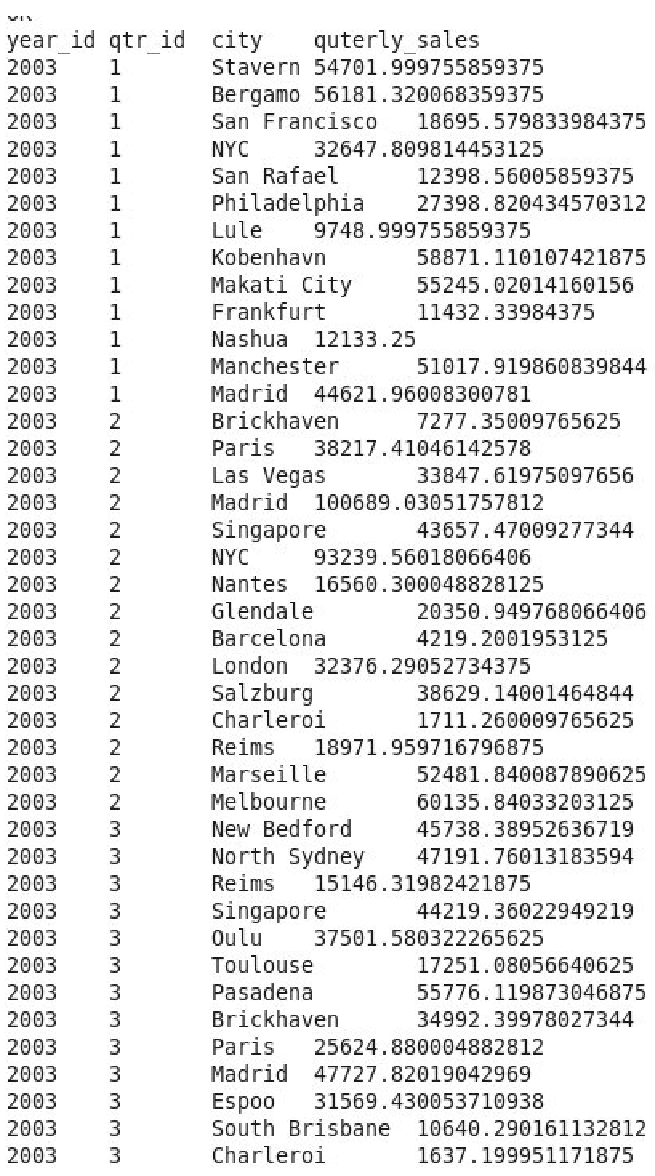
Select year_id, qtr_id, city, sum(sales)

From sales_order_data_orc

group by city, qtr_id, year_id

order by year_id, qtr_id;

Here year id also taken while querying this query because it going to be confusing which year quarter as been display. Year wise quarterly sale for the city is more sensible and it is easy to find some pattern in it. Dues to this reason the year id also added into the query. The result as been order by year id and quarter id. See below it is showing result.



```
year_id qtr_id  city    quterly_sales
2003    1       Stavern 54701.999755859375
2003    1       Bergamo 56181.320068359375
2003    1       San Francisco   18695.579833984375
2003    1       NYC     32647.809814453125
2003    1       San Rafael      12398.56005859375
2003    1       Philadelphia    27398.820434570312
2003    1       Lule    9748.999755859375
2003    1       Kobenhavn       58871.110107421875
2003    1       Makati City     55245.02014160156
2003    1       Frankfurt       11432.33984375
2003    1       Nashua  12133.25
2003    1       Manchester      51017.919860839844
2003    1       Madrid  44621.96008300781
2003    2       Brickhaven      7277.35009765625
2003    2       Paris   38217.41046142578
2003    2       Las Vegas       33847.61975097656
2003    2       Madrid  100689.03051757812
2003    2       Singapore       43657.47009277344
2003    2       NYC     93239.56018066406
2003    2       Nantes  16560.300048828125
2003    2       Glendale        20350.949768066406
2003    2       Barcelona       4219.2001953125
2003    2       London  32376.29052734375
2003    2       Salzburg        38629.14001464844
2003    2       Charleroi       1711.260009765625
2003    2       Reims   18971.959716796875
2003    2       Marseille       52481.840087890625
2003    2       Melbourne       60135.84033203125
2003    3       New Bedford     45738.38952636719
2003    3       North Sydney    47191.76013183594
2003    3       Reims   15146.31982421875
2003    3       Singapore       44219.36022949219
2003    3       Oulu    37501.580322265625
2003    3       Toulouse        17251.08056640625
2003    3       Pasadena        55776.119873046875
2003    3       Brickhaven      34992.39978027344
2003    3       Paris   25624.880004882812
2003    3       Madrid  47727.82019042969
2003    3       Espoo   31569.430053710938
2003    3       South Brisbane  10640.290161132812
2003    3       Charleroi       1637.199951171875
```

h. Find a month for each year in which maximum number of quantities were sold