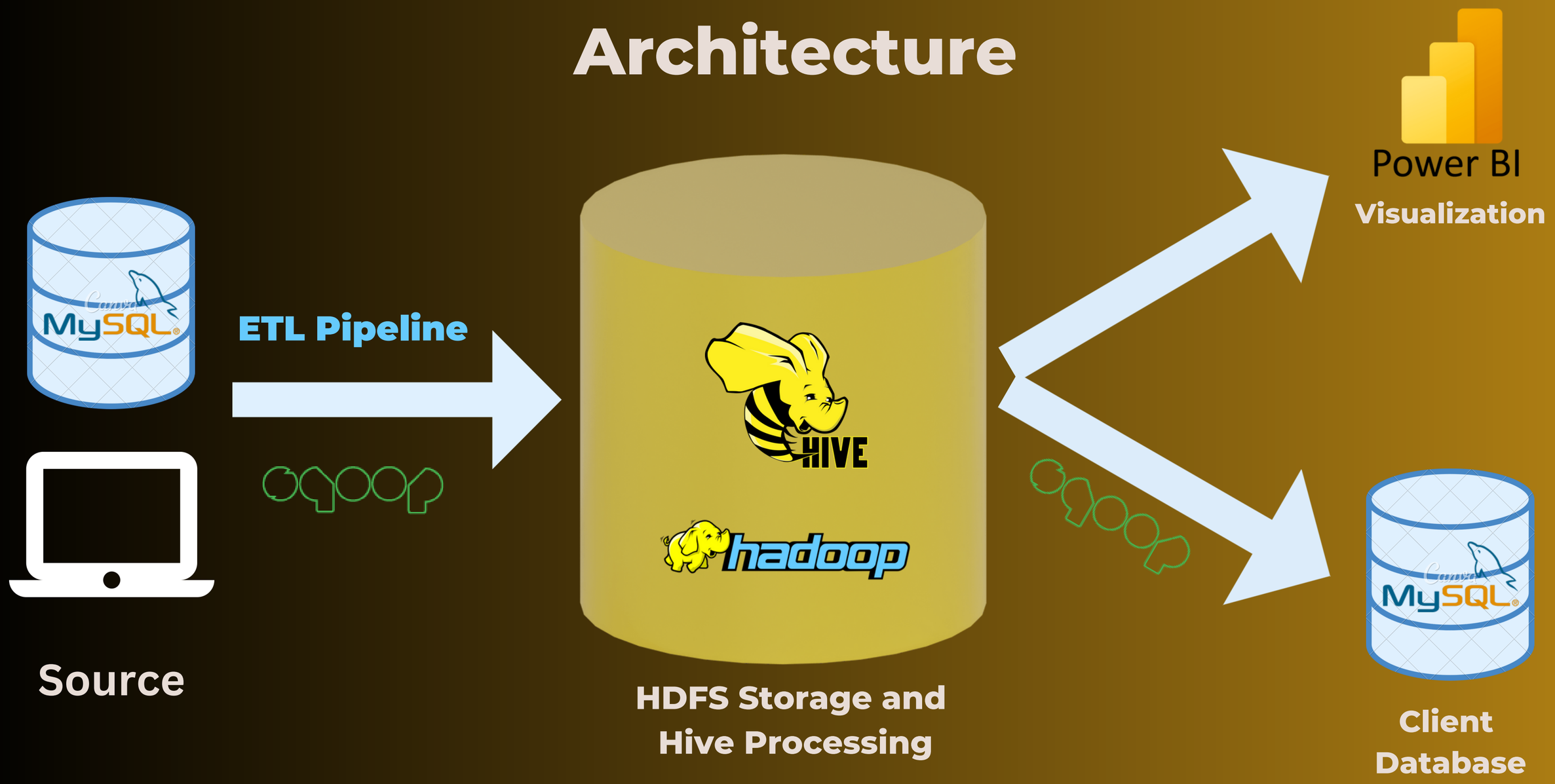
Big Data Analysis

on Ecommerce and Walmart Data

Presented By:

Abhishek Yadav
Anurag Yadav
Himanshu Dhiman
Kaviya R
Nikita Nannaware
Shreyash Gupta

Ecommerce Data Analysis Architecture



Tasks :

1. Create sqoop data pipeline to import data from mysql to hive
2. Data analysis with Hive Query Language
3. Save the Hive analysis as external tables
4. Export analyzed to client database using Sqoop
5. Load the output data in PowerBI for visualization

Import Data into HDFS using SQOOP ETL pipeline

```
sqoop import --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table ecom --target-dir /user/cloudera/ecom -m 1
```

CREATING A HIVE TABLE & LOADING DATA INTO IT

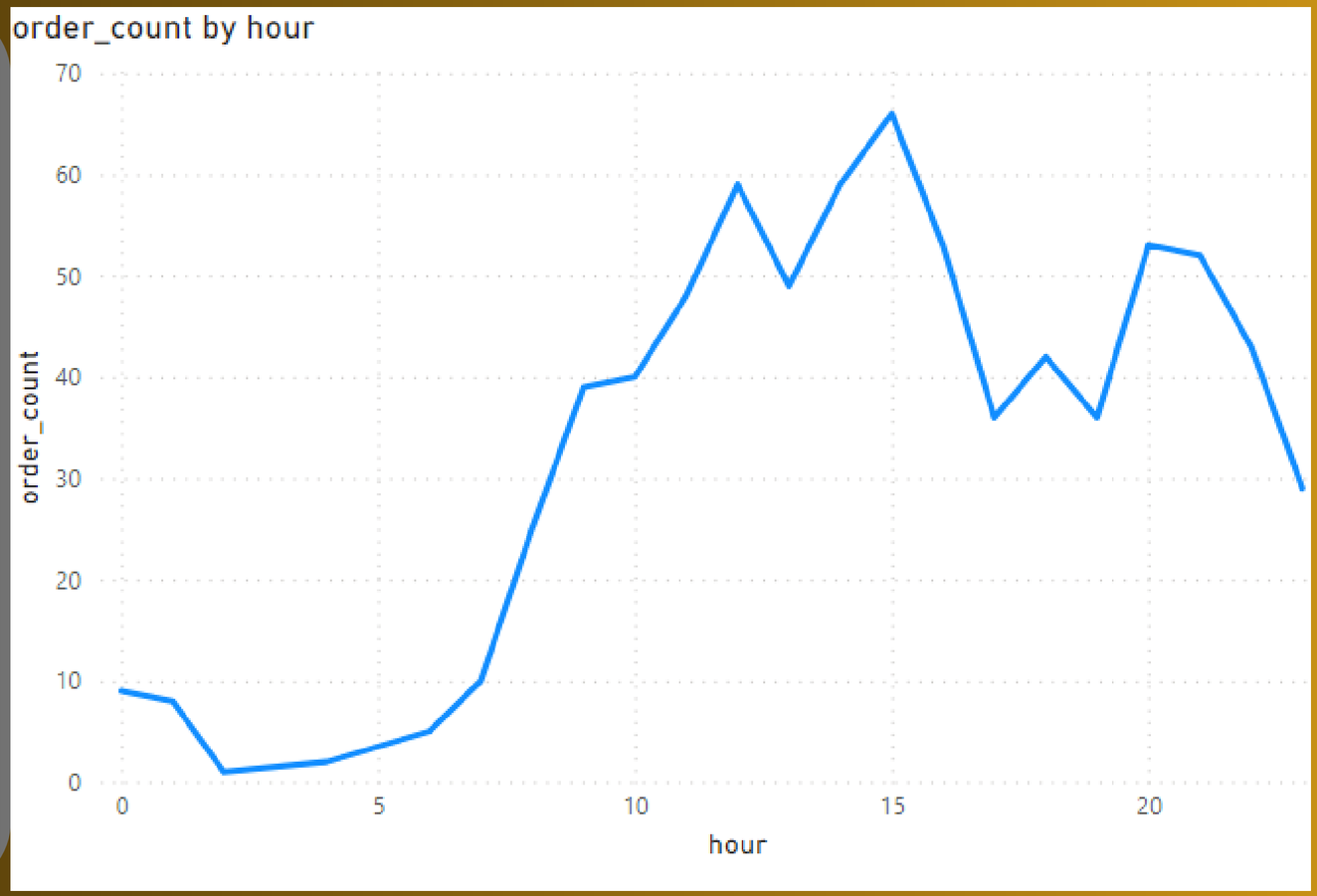
```
CREATE TABLE ecom_data (order_id STRING, customer_id STRING, quantity INT, price_MRP FLOAT, payment FLOAT, timestamp STRING, rating INT, product_category STRING, product_id STRING, payment_type STRING, order_status STRING, product_weight INT, product_length INT, product_height INT, product_width INT,customer_city STRING, customer_state STRING, seller_id STRING,seller_city STRING,payment_installments INT) row format delimited fields terminated by ',' ;

load data inpath '/user/cloudera/ecom/' into table ecom_data;
```

Hourly Sales Analysis

Query:

```
create external table op3 (hour int, product_category string, customer_state string, order_count int) row format delimited fields terminated by ',' location '/user/hive/warehouse/ecom_op/op3'

insert overwrite table op3 select substr(timestamp, 12,2) as hour, product_category, customer_state, count(distinct order_id) from ecom_data group by substr(timestamp,12,2), product_category, customer_state;
```

Visualization:

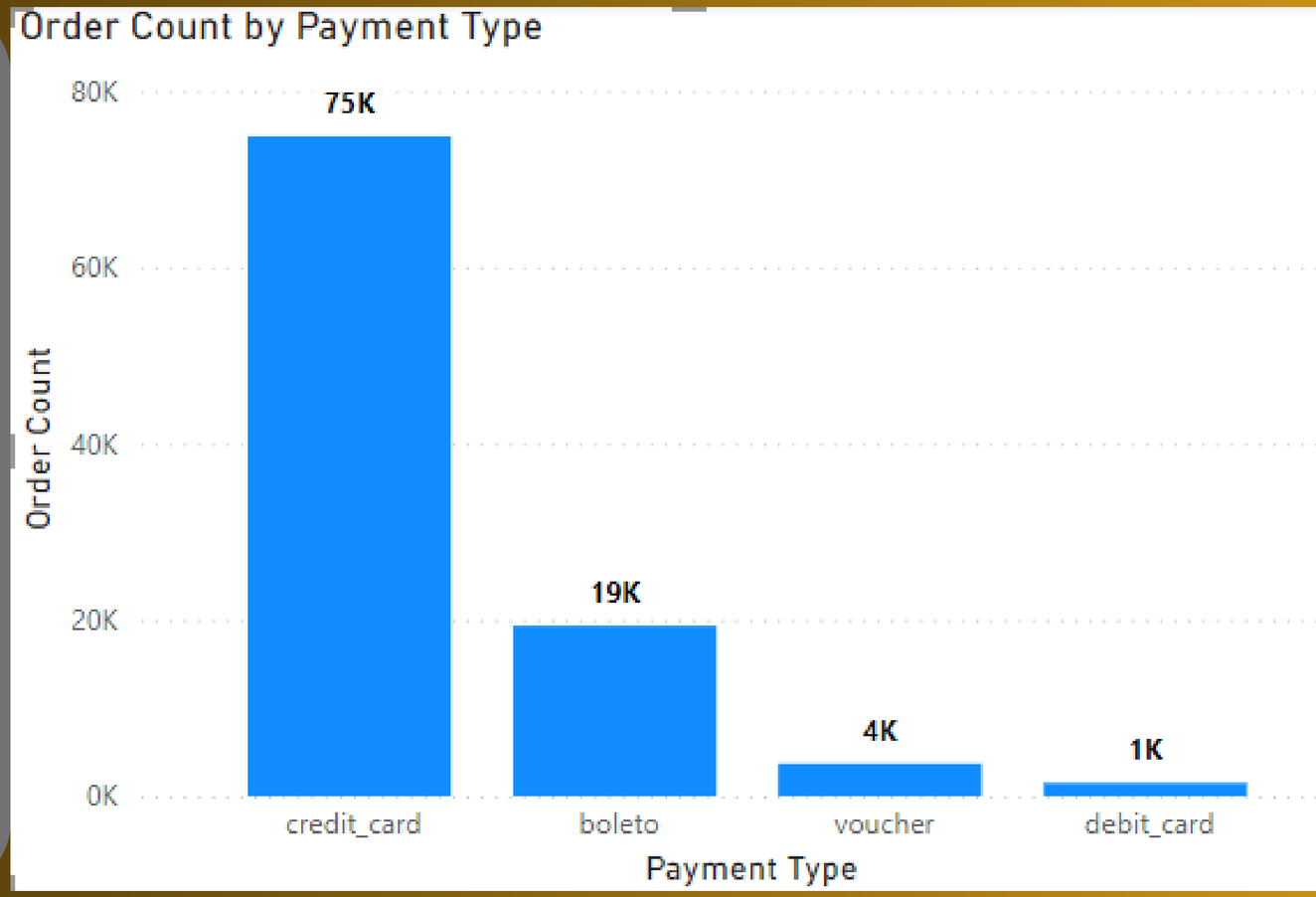


What are the most commonly used payment types?

Query:

```
create external table op5_1 (payment_type string,count_of_orders int) row format delimited fields terminated by ',' location '/user/hive/warehouse/ecom_op/op5_1';

insert overwrite table op5_1 select payment_type, count(distinct order_id) from ecom_data_orc group by payment_type;

create external table op5_2 (payment_installment int,count_of_orders int) row format delimited fields terminated by ',' location '/user/hive/warehouse/ecom_op/op5_2';

insert overwrite table op5_2 select coalesce(payment_installments, 'NO'), count(distinct order_id) from ecom_data_orc group by payment_installments;
```

Visualization:

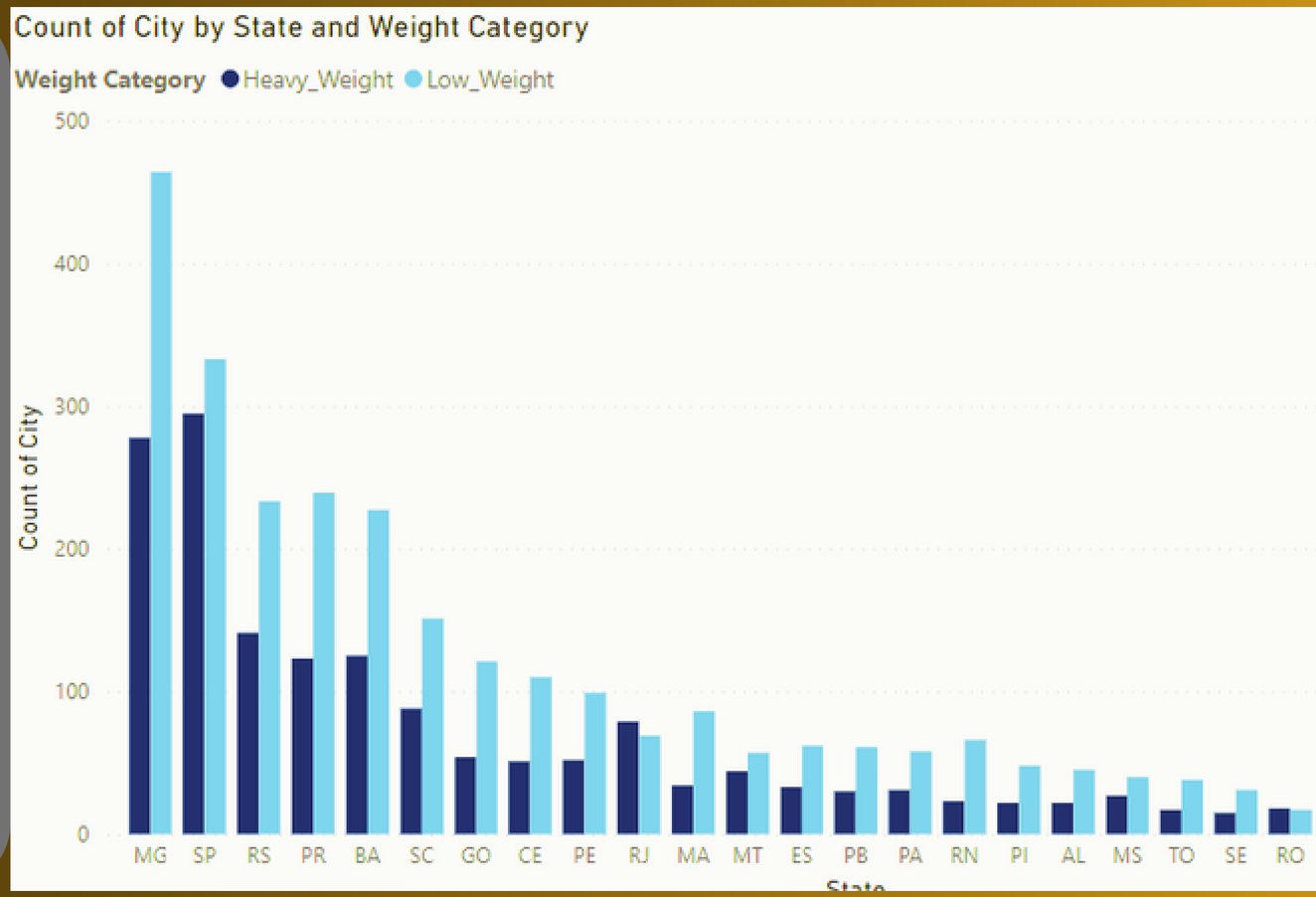


Which city buys heavy weight products and low weight products?

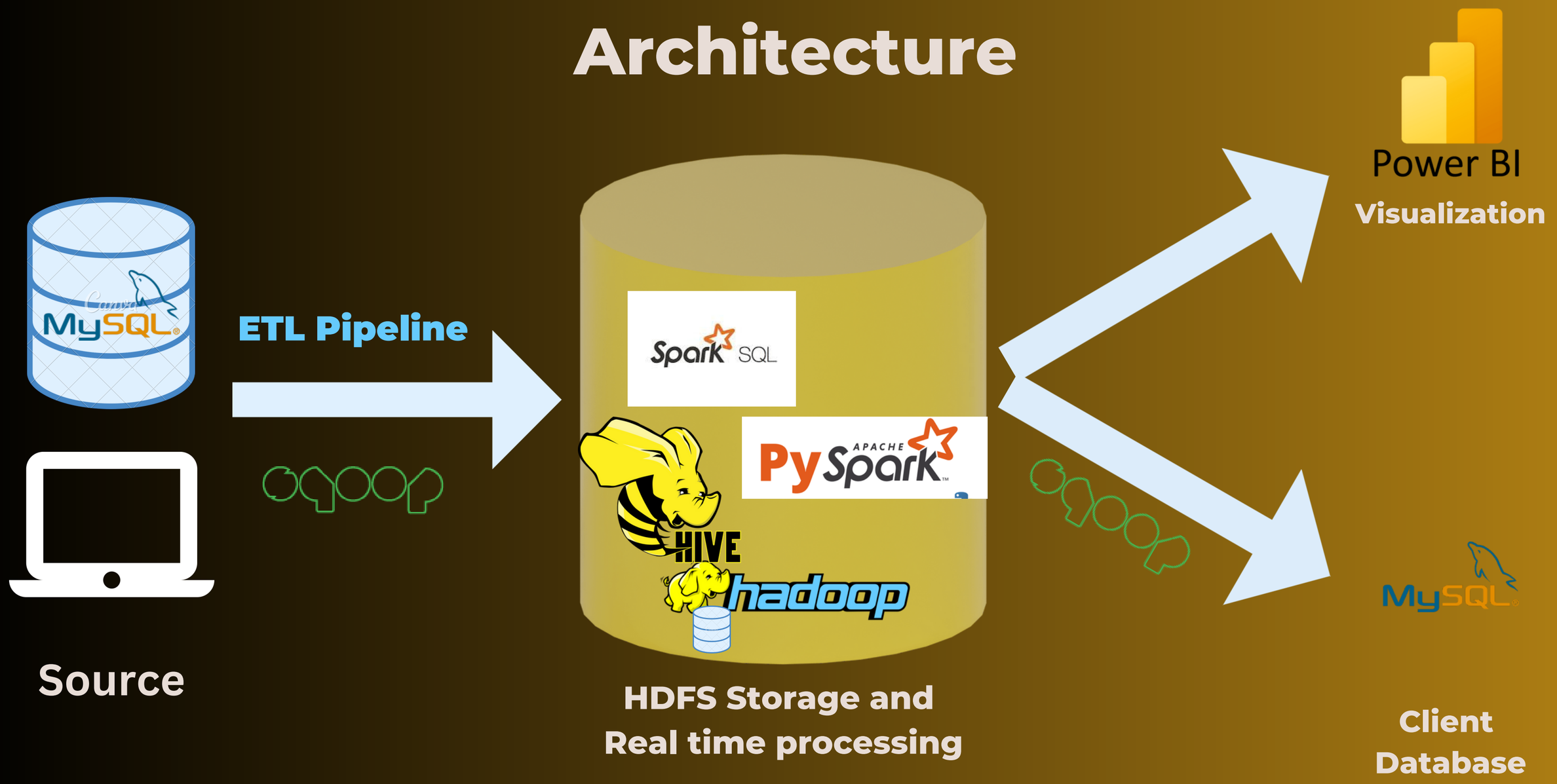
Query:

```
select avg(product_weight) from ecom_data_orc;    2018

create external table op8_1(city string, state string, weight_category string) row format delimited fields terminated by ',' location '/user/hive/warehouse/ecom_op/op8_1';

insert overwrite table op8_1 select customer_city,customer_state, if (avg(product_weight) >2018 , 'Heavy_Weight', 'Low_Weight') from ecom_data_orc group by customer_city, customer_state;
```

Visualization:



Walmart Stock Data Analysis Architecture



Tasks:

1. Loading data from MySql to HDFS using Sqoop Data pipeline
2. Data analysis with Hive, SparkSQL and PySpark
3. Exported the analysed data back to client database
4. Load the analyzed data in PowerBI for visualization

Import Data into HDFS using SQOOP ETL pipeline

```
sqoop import --connect jdbc:mysql://localhost:3306/walmart --username root --password cloudera --table stock --target-dir /user/cloudera/walmart -m 1
```

CREATING A HIVE TABLE & LOADING DATA INTO IT

```
create table walmart(date string, open double, high double, low double, close double, volume INT,
adj_close double) row format delimited fields terminated by ',';

load data inpath '/user/cloudera/walmart/' into table ecom_data;
```

Q1. Calculate VH Ratio that is the ratio of the High Price versus volume of stock traded for a day

Query:

```
create external table opt1(date string, HVratio float) row format delimited fields terminated by ',' location '/user/cloudera/walmart_analysis/opt1';

insert overwrite table opt1 select date , (volume / high) as HVratio from walmart;

sqoop export --connect jdbc:mysql://localhost:3306/stock --username root --password cloudera --table Vh_ratio --target-dir '/user/cloudera/walmart_analysis/opt1/'
```

Sample Output:

```
2012-01-03      207481.16139973773
2012-01-04      158961.06449967416
2012-01-05      214159.68179716906
2012-01-06      135734.2287038352
2012-01-09      112162.8897661376
2012-01-10      115680.79226107735
2012-01-11      106930.96137293732
2012-01-12      120606.66666666667
2012-01-13      129664.48449688105
```

Q2. Which 10 day had the Peak High in Price?

Query:

```
create external table opt2(date string, peak_high float) row format delimited fields terminated by ',' location '/user/cloudera/walmart_analysis/opt2';

insert overwrite table opt2 select date, high from walmart order by high DESC;

sqoop export --connect jdbc:mysql://localhost:3306/stock --username root --password cloudera --table peak_high --target-dir '/user/cloudera/walmart_analysis/opt2/'
```

Sample Output:

2015-01-13	90.97
2015-01-08	90.67
2015-01-09	90.39
2015-01-12	90.31
2015-01-23	89.26
2015-01-26	89.16
2015-01-07	88.68
2015-01-14	88.52
2015-01-27	88.46
2015-01-22	88.4

Data import from MySQL to pySpark Dataframe

Query

```
from pyspark.sql import SparkSession
```

```
spark=SparkSession.builder.appName("pyspark project").getOrCreate()
```

```
walmartdf = spark.read.format("jdbc") \
 .option("url", "jdbc:mysql://localhost:3306/sparkproject") \
 .option("driver","com.mysql.jdbc.Driver") \
 .option("dbtable", "walmartdata") \
 .option("user", "root").option("password", "cloudera") \
 .load()
```

```
walmartdf.show(3)
```

```
+-------------------+-----+-----+-----+-----+--------+------------------+
|               Date| Open| High|  Low|Close|  Volume|         Adj_Close|
+-------------------+-----+-----+-----+-----+--------+------------------+
|2012-01-03 00:00:00|59.97|61.06|59.87|60.33|12668800|52.619234999999996|
|2012-01-04 00:00:00|60.21|60.35|59.47|59.71| 9593300|         52.078475|
|2012-01-05 00:00:00|59.35|59.62|58.37|59.42|12768200|         51.825539|
+-------------------+-----+-----+-----+-----+--------+------------------+
only showing top 3 rows
```

```
walmartdf.createOrReplaceTempView("walmartdata")
```

Q3.What is the max and min of the Volume column?

Query

```
MaxMinVolume=spark.sql("select max(volume) as MaxVolume, min(volume) as MinVolume from walmartdata")
```

```
MaxMinVolume.show()
```

```
+---------+---------+
|MaxVolume|MinVolume|
+---------+---------+
| 80898100|  2094900|
+---------+---------+
```

```
MaxMinVolume.write.format("jdbc") \
                    .option("driver", "com.mysql.cj.jdbc.Driver") \
                    .option("url", "jdbc:mysql://localhost:3306/sparkproject") \
                    .option("dbtable","sol3") \
                    .option("user","root") \
                    .option("password","root") \
                    .save()
```

output

Result Grid | Filter Rows:

	MaxVolume	MinVolume
▶	80898100	2094900

Q4. How many days was the Close lower than 60 dollars?

Query

```
CloseLower=spark.sql("select date, close from walmartdata where close <=60")
```

```
CloseLower.show(5)
```

```
+-------------------+-----+
|               date|close|
+-------------------+-----+
|2012-01-04 00:00:00|59.71|
|2012-01-05 00:00:00|59.42|
|2012-01-06 00:00:00| 59.0|
|2012-01-09 00:00:00|59.18|
|2012-01-10 00:00:00|59.04|
+-------------------+-----+
only showing top 5 rows
```

```
CloseLower.write.format("jdbc") \
                    .option("driver", "com.mysql.cj.jdbc.Driver") \
                    .option("url", "jdbc:mysql://localhost:3306/sparkproject") \
                    .option("dbtable","sol4") \
                    .option("user","root") \
                    .option("password","root") \
                    .save()
```

output

Result Grid | Filter Rows:

	count_of_days
▶	81

Q5. What percentage of the time was the High greater than 80 dollars

Query

```
High_greater=spark.sql("SELECT round(sum(if(high>80,1,0))*100/count(*),2) AS percentage FROM walmartdata")
```

```
High_greater.show()
```

```
+----------+
|percentage|
+----------+
|      9.14|
+----------+
```

```
solution5=High_greater.write.format("jdbc") \
                     .option("driver", "com.mysql.cj.jdbc.Driver") \
                     .option("url", "jdbc:mysql://localhost:3306/sparkproject") \
                     .option("dbtable","sol5") \
                     .option("user","root") \
                     .option("password","root") \
                     .save()
```

output

Result Grid | Filter Rows:

	percentage
▶	9.14

sol5 31 ×

Q6. What is the Pearson correlation between High and Volume?

Query

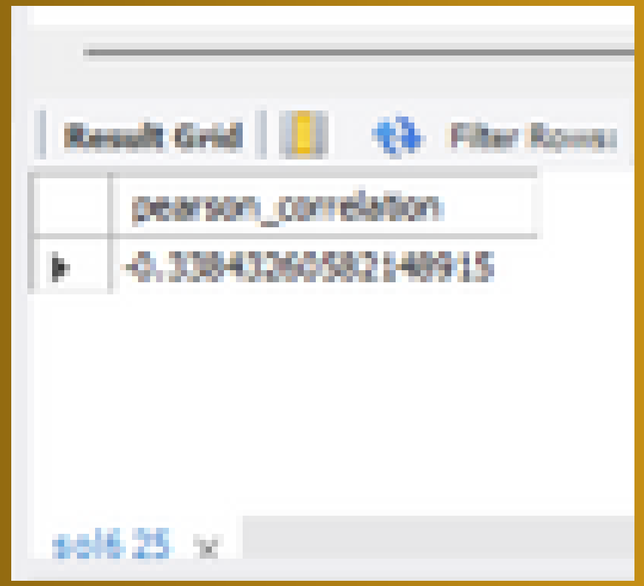
```
pearson_correlation = spark.sql("SELECT corr(high, volume) AS pearson_correlation FROM walmartdata")
```

```
pearson_correlation.show()
```

```
+--------------------+
| pearson_correlation|
+--------------------+
|-0.33843260582148915|
+--------------------+
```

```
pearson_correlation.write.format("jdbc") \
                     .option("driver", "com.mysql.cj.jdbc.Driver") \
                     .option("url", "jdbc:mysql://localhost:3306/sparkproject") \
                     .option("dbtable","sol6") \
                     .option("user","root") \
                     .option("password","root") \
                     .save()
```

output



Create Rdd for analysis with PySpark

```python
from pyspark.sql import SparkSession
from pyspark import SparkContext
```

```python
spark = SparkSession.builder.appName('Walmart Analysis')\
    .config('spark.jars', 'mysql-connector-j-8.0.32.jar').getOrCreate()
```

```python
sc = spark.sparkContext
```

```python
stock_data = sc.textFile('/user/cloudera/walmart/part-m-00000')
```

```python
stock_data.take(5)
```

```
['2012-01-03,59.970001,61.060001,59.869999,60.330002,12668800,52.619234999999996',
 '2012-01-04,60.209998999999996,60.349998,59.470001,59.709998999999996,9593300,52.078475',
 '2012-01-05,59.349998,59.619999,58.369999,59.419998,12768200,51.825539',
 '2012-01-06,59.419998,59.450001,58.869999,59.0,8069400,51.45922',
 '2012-01-09,59.029999,59.549999,58.919998,59.18,6679300,51.616215000000004']
```

```python
stock_data2 = stock_data.map(lambda x : x.split(','))
```

```python
stock_data2.take(1)
```

```
[['2012-01-03',
  '59.970001',
  '61.060001',
  '59.869999',
  '60.330002',
  '12668800',
  '52.619234999999996']]
```

Q7. What is the max High per year?

```
year_data = stock_data2.map(lambda x: [x[0][:4], float(x[2])])
```

```
year_data.take(5)
```

```
[['2012', 61.060001],
 ['2012', 60.349998],
 ['2012', 59.619999],
 ['2012', 59.450001],
 ['2012', 59.549999]]
```

```
year_data.reduceByKey(max).take(5)
```

```
[('2012', 77.599998),
 ('2013', 81.370003),
 ('2014', 88.089996),
 ('2015', 90.970001),
 ('2016', 75.190002)]
```

```
year_max = year_data.reduceByKey(max)
```

```
year_max.coalesce(1).saveAsTextFile('Output/year_max')
```

Output

2012	77.6
2013	81.37
2014	88.09
2015	90.97
2016	75.19

Q8. What is the average Close for each Calendar Month?

```
month_value = stock_data2.map(lambda x: [int(x[0].split('-')[1]), round(float(x[2]),2)])
```

```
month_value.take(2)
```

```
[[1, 61.06], [1, 60.35]]
```

```
stock_pair = month_value.groupByKey()
```

```
month_avg = stock_pair.mapValues(lambda x : round(sum(x)/len(x),2))
```

```
month_avg.take(2)
```

```
[(2, 71.73), (4, 73.45)]
```

```
month_avg_df = month_avg.toDF(['Month', 'Close_avg'])
```

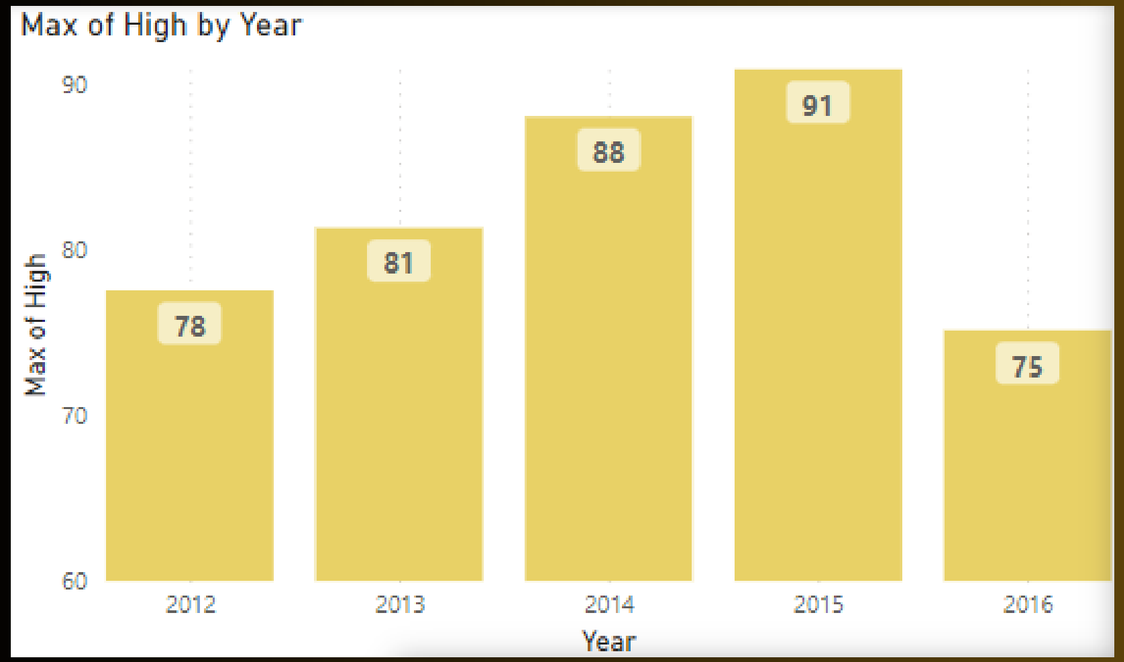
```
month_avg_df.write.format('jdbc')\
            .option('driver', "com.mysql.jdbc.Driver")\
            .option('url', 'jdbc:mysql://localhost:3306/walmart_data')\
            .option('dbtable','month_avg')\
            .option('user','root')\
            .option('password','dhiman')\
            .save()
```

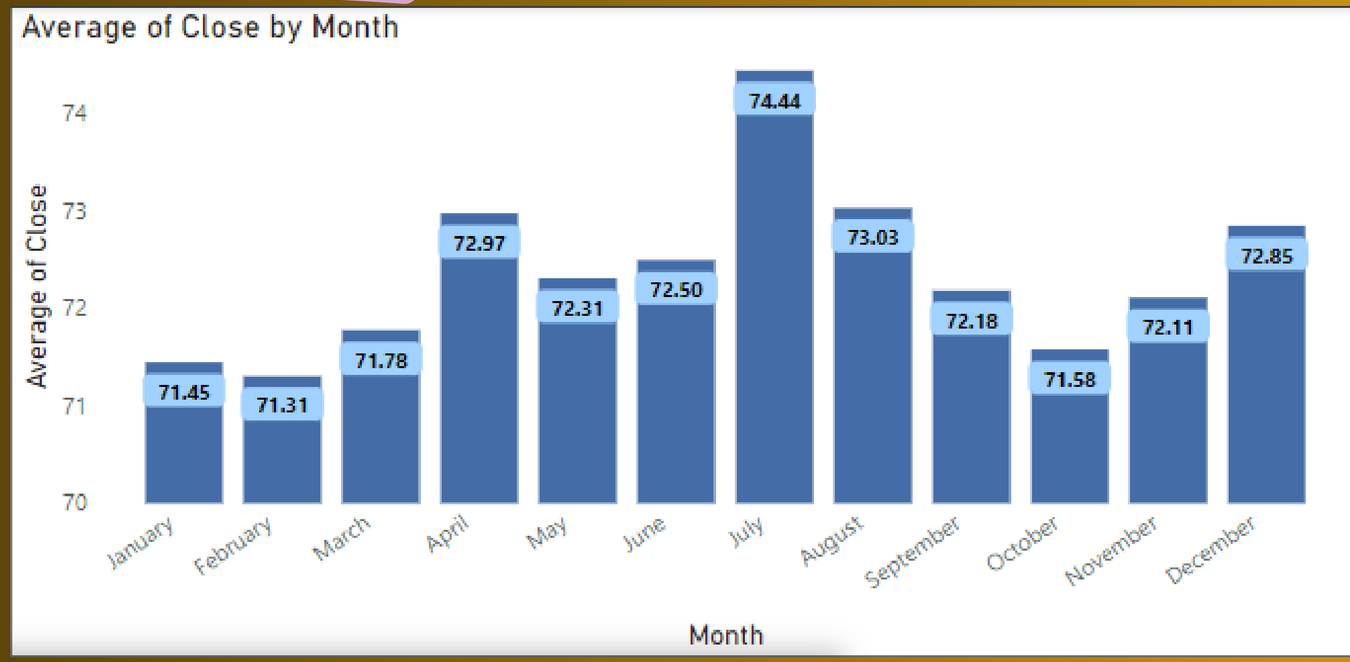
Output

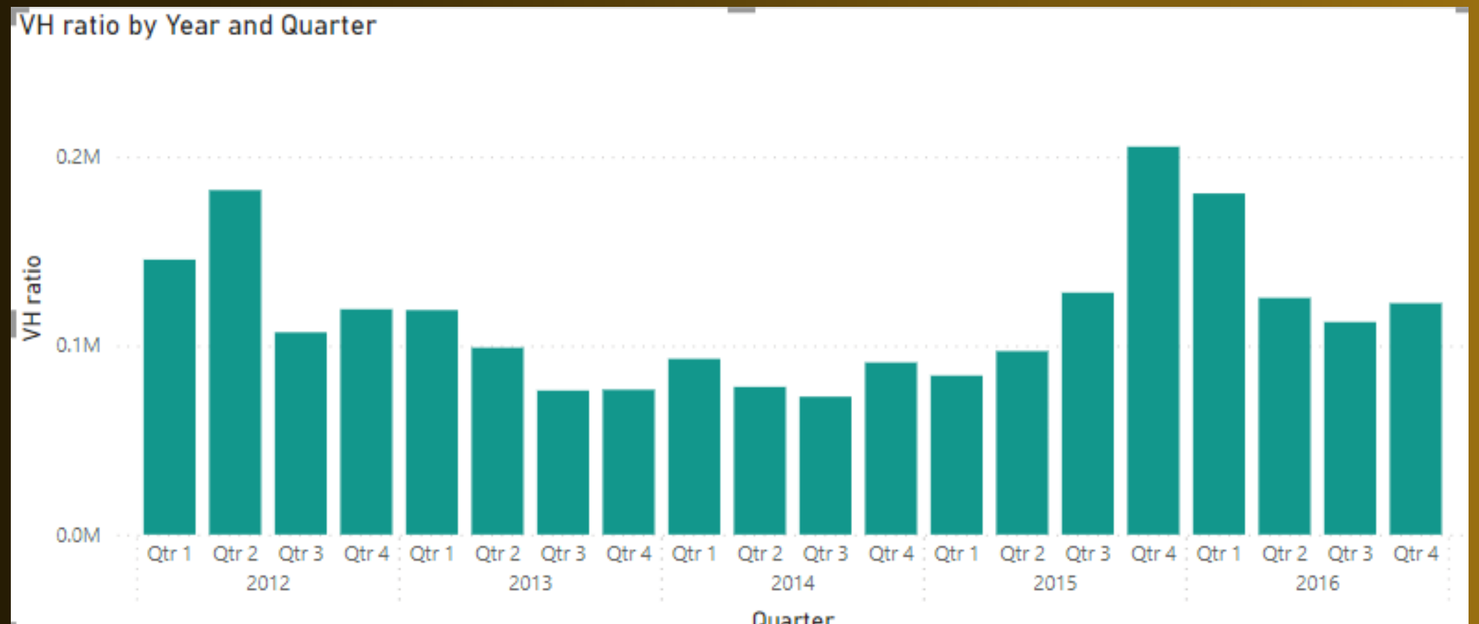
Result Grid | Filter Rows:

Month	Close_avg
1	71.97
3	72.2
5	72.72
7	74.78
9	72.6
11	72.53
2	71.73
4	73.45
6	72.93
8	73.52
10	72.13
12	73.36

Visualizations







THANK YOU