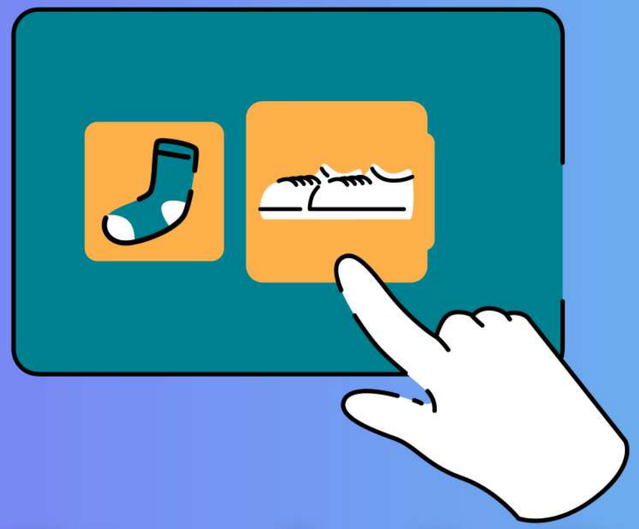


Walmart Stock & Ecommerce Data Analysis

-----Dataengineering Gang-2-----

Nahush Dhavalikar

Venkat Sir DE Expert

Shriya Gandreti

Shivam Kumar

Jyoti Biswas

Sejal

Deepak Mehta

Ecommerce Dataset

Column Description:

1 order_id unique id for each order (32 fixed-size number)
2 customer_id unique id for each customer (32 fixed-size number)
3 quantity 1-21
4 price_MRP cost price, 0.85-6735
5 payment selling price, 0-13664.8
6 timestamp order purchase time (local, day-month-year hour:min:sec AM/PM)
7 rating 1-5
8 product_category category under which product belongs
9 product_id unique id for each product (32 fixed-size number)
10 payment_typeType of payment - credit card/debit card/boleto/voucher
11 order_status delivered/shipped/invoiced
12 product_weight_g weight of product (in grams), 0-40425
13 product_length_cm length of product (in centimeter), 7-105
14 product_height_cm height of product (in centimeter), 2-105
15 product_width_cm width of product (in centimeter), 6-118
16 customer_city city where order is placed
17 customer_state state where order is placed
18 seller_id unique id for each seller (32 fixed-size number)
19 seller_city city where order is picked up
20 seller_state state where order is picked up
21 payment_installments no. of installments taken by customer to pay bill, 0-24

Ecommerce Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	order_id	customer_	quantity	price_MRP	payment	timestamp	rating	product_category	product_id	payment_type	order_status	product_weight_g	product_length_cm	product_h	product_w	cust
2	9045fa841	2f8d6af8c	1	1074.38	1095.65	21-11-2017 22:34	5	health_beauty	0a37e055289	credit_card	delivered	800	40	20	30	osas
3	404c087c1	1ce709108	1	145	161.71	24-02-2017 22:55	5	health_beauty	67473aa97e9	credit_card	delivered	400	38	12	25	cam
4	d6d7c4312	5b477d525	1	145	161.71	19-01-2017 14:28	4	health_beauty	67473aa97e9	credit_card	delivered	400	38	12	25	nova
5	c0e02613	c2f181778	1	555	585.95	01-06-2018 22:22	4	health_beauty	b60a0c8bd03	boleto	delivered	650	16	10	11	port
6	1bf38e345	21a9772e7	1	226.8	250.32	15-06-2018 20:36	5	health_beauty	08462528607	credit_card	delivered	650	16	10	11	mon
7	d1ff908b4	190508c5	1	207.9	238.61	04-08-2018 21:57	5	health_beauty	08462528607	credit_card	delivered	650	16	10	11	sao
8	591eeea2	ed0e1816	1	860	882.46	20-06-2017 14:53	4	health_beauty	2483416c14a	credit_card	delivered	500	40	20	30	uber
9	c1808aac	2d88e5d5	1	589	606.87	24-11-2017 10:23	5	health_beauty	2483416c14a	credit_card	delivered	500	40	20	30	grav
10	4da0b6d6	6d51e47d	1	589	607.87	28-11-2017 09:37	3	health_beauty	2483416c14a	credit_card	delivered	500	40	20	30	ribei
11	5d4f3f9b6	44ce310a	1	650	696.58	06-12-2017 13:05	4	health_beauty	2483416c14a	credit_card	delivered	500	40	20	30	lago
12	9f1dcbf0e	9b9681cfb	1	1597.35	1650.56	20-08-2017 11:40	5	health_beauty	76951acb342	credit_card	delivered	650	40	20	30	mar
13	e21a007c	e3e4e76e	1	1597.35	1629.33	18-07-2017 14:26	5	health_beauty	76951acb342	credit_card	delivered	650	40	20	30	salv
14	0974465	753217178	1	364	428.8	16-07-2018 12:41	5	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	belo
15	0713f1dd	e1e49b43	1	349.9	365.37	01-08-2017 07:31	5	health_beauty	6cdd5384349	boleto	delivered	900	25	12	38	casc
16	f810273bc	6b8d7030	1	349.9	381.88	22-02-2018 17:06	4	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	reci
17	b2bc998b	c6747b39	1	364	398.69	19-06-2018 20:01	2	health_beauty	6cdd5384349	credit_card	shipped	900	25	12	38	arac
18	897f6132	790edc01	1	349.9	369.6	05-02-2018 17:04	5	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	mari
19	91a31ae0	a7e7ac71	1	349.9	378.89	08-06-2017 17:50	5	health_beauty	6cdd5384349	boleto	delivered	900	25	12	38	guac
20	6ae8d599	43e4c6f57	1	349.9	379.2	22-07-2017 12:59	4	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	arac
21	4389819b	0d3740ed	1	349.9	373.17	04-04-2018 12:47	4	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	itu
22	093c11a6	77a6136f4	1	364	797.38	13-06-2018 08:52	5	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	jabo
23	093c11a6	77a6136f4	2	364	797.38	13-06-2018 08:52	5	health_beauty	6cdd5384349	credit_card	delivered	900	25	12	38	jabo
24	d710dadc	6d1ad1d9	1	349.9	368.29	16-05-2018 17:46	4	health_beauty	6cdd5384349	boleto	delivered	900	25	12	38	vito

Total columns: 21 columns

Total records: 116580

Walmart Dataset

Column Description:

1. Date: Date on which the stock is traded.

2. Open: The opening price is the price from the first transaction of a business day

3. High: High is the highest price at which a stock traded during the course of the trading day and is typically higher than the closing or equal to the opening price

4. Low: Low is the lowest price at which a stock trades over the course of a trading day

5. Close: The close is a reference to the end of a trading session in the financial markets when the markets close for the day

6. Volume: Volume is the number of shares of a security traded during a given period of time

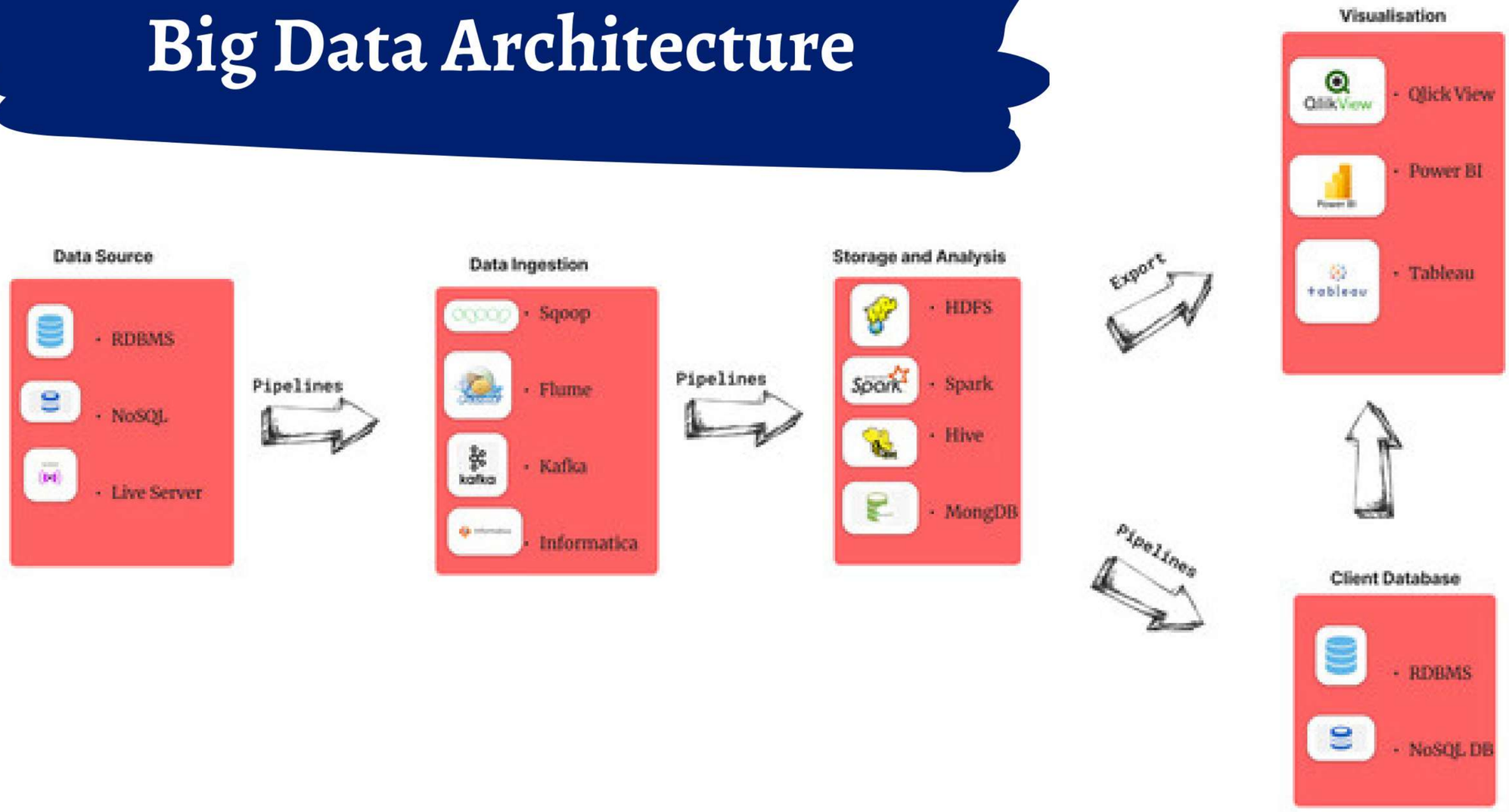
7. Adjusted Close: Adjusted closing price refers to the price of the stock after paying off the dividends

Walmart Dataset

Total Number of records:- 1258

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	Adj Close
2	03-01-2012	59.97	61.06	59.87	60.33	12668800	52.61923
3	04-01-2012	60.21	60.35	59.47	59.71	9593300	52.07848
4	05-01-2012	59.35	59.62	58.37	59.42	12768200	51.82554
5	06-01-2012	59.42	59.45	58.87	59	8069400	51.45922
6	09-01-2012	59.03	59.55	58.92	59.18	6679300	51.61622
7	10-01-2012	59.43	59.71	58.98	59.04	6907300	51.49411
8	11-01-2012	59.06	59.53	59.04	59.4	6365600	51.8081
9	12-01-2012	59.79	60	59.4	59.5	7236400	51.89532
10	13-01-2012	59.18	59.61	59.01	59.54	7729300	51.9302
11	17-01-2012	59.87	60.11	59.52	59.85	8500000	52.20058
12	18-01-2012	59.79	60.03	59.65	60.01	5911400	52.34013
13	19-01-2012	59.93	60.73	59.75	60.61	9234600	52.86345
14	20-01-2012	60.75	61.25	60.67	61.01	10378800	53.21232
15	23-01-2012	60.81	60.98	60.51	60.91	7134100	53.1251
16	24-01-2012	60.75	62	60.75	61.39	7362800	53.54375
17	25-01-2012	61.18	61.61	61.04	61.47	5915800	53.61353
18	26-01-2012	61.8	61.84	60.77	60.97	7436200	53.17744
19	27-01-2012	60.86	61.12	60.54	60.71	6287300	52.95067
20	30-01-2012	60.47	61.32	60.35	61.3	7636900	53.46526
21	31-01-2012	61.53	61.57	60.58	61.36	9761500	53.51759
22	01-02-2012	61.79	62.63	61.79	62.18	12130600	54.23279
23	02-02-2012	62.4	62.47	61.82	61.94	6211300	54.02346

Big Data Architecture



Ecommerce Project

Source Data

Table Creation in MySQL

```
CREATE TABLE ecom(
order_id VARCHAR(100),
cust_id VARCHAR(100), qty int, MRP float,
payment float, order_pchTime VARCHAR(100),
rating int, prodCat VARCHAR(100), prodID VARCHAR(100),
paymt_type VARCHAR(100), order_status VARCHAR(100),
prod_weightGm int, prod_len_cm int, prod_height_cm int,
prod_width_cm int, cust_city VARCHAR(100),
cust_state VARCHAR(100), seller_id VARCHAR(100),
seller_city VARCHAR(100), seller_state VARCHAR(100),
paymt_insmt int)
```

LOADING Data FROM LFS to MySQL table

```
LOAD DATA LOCAL INFILE '/home/cloudera/ecom_data.csv' INTO TABLE
ecom
FIELDS TERMINATED BY ' , ' ;
```

SQOOP IMPORT

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

Walmart Stock Project

Source Data

CREATE TABLE in MySQL

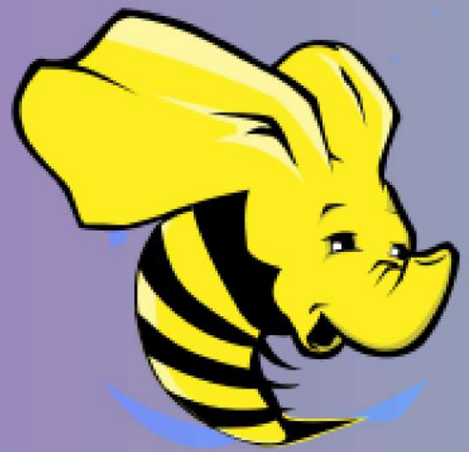
```
CREATE TABLE walmart(
date DATE, open DOUBLE,
 high DOUBLE, low DOUBLE,
close DOUBLE, volume INT,
adj_close DOUBLE
);
```

Loading file from LFS MySQL

```
LOAD DATA LOCAL INFILE \
'/home/cloudera/walmart_stock.csv' INTO \
TABLE walmart FIELDS TERMINATED BY ',' ;
```

SQOOP IMPORT

```
sqoop import \
--connect jdbc:mysql://localhost:3306/BDSpark_pro \
--username root --password cloudera
--table walmart \
--hive-import -m 1
```



HIVE

What is HIVE?

- A system for managing and querying structured data built on top of Hadoop
  - Uses Map-Reduce for execution
  - HDFS for storage
  - SQL on structured data as a familiar data warehousing tool
  - Extensibility
    (Pluggable map/reduce scripts in the language of your choice, Rich and User Defined data types, User Defined Functions)
  - Interoperability (Extensible framework to support different file and data formats)

Ecommerce Project

JOB 1: Customer Segmentation: Categorizing customers based on their spendings

Query

```
SELECT cust_id, ROUND(SUM(payment),3) AS spendings,
CASE
WHEN ROUND(SUM(payment),3) > 20000 THEN 'GROUP 1'
WHEN ROUND(SUM(payment),3) > 5000 THEN 'GROUP 2'
WHEN ROUND(SUM(payment),3) > 1500 THEN 'GROUP 3'
WHEN ROUND(SUM(payment),3) > 100 THEN 'GROUP 4'
ELSE 'GROUP 5'
END AS cust_cat
FROM ecom GROUP BY cust_id ORDER BY spendings DESC;
```

Output

```
mysql> select * from job1 limit 10;;
+----------------------------------+-----------+-------------------+
| customer_id                      | Spendings | Customer_category |
+----------------------------------+-----------+-------------------+
| d2a88d4cd954a41795819308c136a315 |     63.33 | GROUP 5           |
| 028ef41231e80887d8be309e5a28aaf2 |     63.33 | GROUP 5           |
| 744a80b67af618adfecb19c2b77e3618 |     63.33 | GROUP 5           |
| b52dc9b0b38a7c414ffb6da8aec8d27d |     63.33 | GROUP 5           |
| d30a43ef4613415982f5d40a4d0e47bf |     63.31 | GROUP 5           |
| 5667ce9b9fb068ffcc82902f8ad64c63 |     63.31 | GROUP 5           |
| 1addc7d0ea1d5587f4cec79230ef78c5 |     63.31 | GROUP 5           |
| 68f1d888d94100d13b58f401de3bb5b4 |     63.31 | GROUP 5           |
| c42eeec226b4a998f0655d5bd633cbb3 |     63.31 | GROUP 5           |
| 1471f2464edb13ceccc7b52cb1a9def4 |     63.31 | GROUP 5           |
+----------------------------------+-----------+-------------------+
10 rows in set (0.00 sec)
```

JOB 2: Monthly Trend Forecasting: the monthly trend of sales

Query

```
SELECT yr, CASE WHEN mon='01' THEN 'January'
WHEN mon='02' THEN 'February'
WHEN mon='03' THEN 'March'
WHEN mon='04' THEN 'April'
WHEN mon='05' THEN 'May'
WHEN mon='06' THEN 'June'
WHEN mon='07' THEN 'July'
WHEN mon='08' THEN 'August'
WHEN mon='09' THEN 'September'
WHEN mon='10' THEN 'October'
WHEN mon='11' THEN 'November'
WHEN mon='12' THEN 'December'
ELSE mon END AS month, sales ,
ROUND((sales - (LAG(sales) OVER()))) AS sales_trend
from(SELECT SUBSTRING(tstamp,7,4) as yr, SUBSTRING(tstamp,4,2) as mon, ROUND(SUM(qty*mrp),3) as sales from ec
GROUP BY SUBSTRING(tstamp,7,4), SUBSTRING(tstamp,4,2) ORDER BY yr DESC, mon DESC) t;
```

Output

```
mysql> select * from job2;
+------+-----------+-------------+-------------+
| Year | Month     | sales       | sales_trend |
+------+-----------+-------------+-------------+
| 2016 | September |      165.29 |        NULL |
| 2016 | October   |     58085.9 |       57921 |
| 2016 | December  |        10.9 |      -58075 |
| 2017 | January   |      148450 |      148439 |
| 2017 | February  |      276258 |      127808 |
| 2017 | March     |      431747 |      155489 |
| 2017 | April     |      421563 |      -10184 |
| 2017 | May       |      595437 |      173874 |
| 2017 | June      |      489714 |     -105723 |
| 2017 | July      |      596520 |      106806 |
| 2017 | August    |      689649 |       93129 |
| 2017 | September |      792980 |      103331 |
| 2017 | October   |      782518 |      -10462 |
| 2017 | November  | 1.20964e+06 |      427119 |
| 2017 | December  |      829291 |     -380345 |
| 2018 | January   | 1.09532e+06 |      266032 |
| 2018 | February  | 1.01098e+06 |      -84344 |
| 2018 | March     |  1.1382e+06 |      127219 |
| 2018 | April     | 1.15867e+06 |       20474 |
| 2018 | May       |  1.1711e+06 |       12432 |
| 2018 | June      | 1.02328e+06 |     -147827 |
| 2018 | July      | 1.04366e+06 |       20384 |
| 2018 | August    |      976500 |      -67161 |
| 2018 | September |         145 |     -976355 |
+------+-----------+-------------+-------------+
24 rows in set (0.00 sec)
```

JOB 8 : Logistics based Optimization Insights:

Query

8.1-> Which city buys heavy weight products and low weight products?

Low Weight Products:

```
WITH cte AS (
SELECT order_id, cust_id, cust_city, prod_weightgm, CASE WHEN prod_weightgm > 22000 THEN 'High Weight Product'
WHEN prod_weightgm < 2000 THEN 'Low Weight Product'
ELSE 'Medium Weight Product'
END AS pwt_cat
FROM ecom
) SELECT cust_city, pwt_cat, COUNT(*) AS count from cte  where pwt_cat='Low Weight Product' GROUP BY cust_city, pwt_cat ORDER BY count desc;
```

High Weight Products:

```
WITH cte AS (
SELECT order_id, cust_id, cust_city, prod_weightgm, CASE WHEN prod_weightgm > 22000 THEN 'High Weight Product'
WHEN prod_weightgm < 2000 THEN 'Low Weight Product'
ELSE 'Medium Weight Product'
END AS pwt_cat
FROM ecom
) SELECT cust_city, pwt_cat, COUNT(*) AS count from cte  where pwt_cat='High Weight Product' GROUP BY cust_city, pwt_cat ORDER BY count desc;
```

JOB 8 : Logistics based Optimization Insights:

Output

```
mysql> select * from job8a_HighWt;
+----------------------+-----------------------+--------------+
| Customer_City        | Product_weightCategory | No_of_Orders |
+----------------------+-----------------------+--------------+
| guarulhos            | Low Weight Product     |         1041 |
| sao bernardo do campo | Low Weight Product     |          878 |
| sao paulo            | Low Weight Product     |        14745 |
| rio de janeiro       | Low Weight Product     |         6057 |
| belo horizonte       | Low Weight Product     |         2511 |
| brasilia             | Low Weight Product     |         1979 |
| curitiba             | Low Weight Product     |         1424 |
| campinas             | Low Weight Product     |         1338 |
| porto alegre         | Low Weight Product     |         1251 |
| salvador             | Low Weight Product     |         1140 |
+----------------------+-----------------------+--------------+
10 rows in set (0.00 sec)

mysql> select * from job8a_lowWt;
+---------------+-----------------------+--------------+
| Customer_City | Product_weightCategory | No_of_Orders |
+---------------+-----------------------+--------------+
| sao paulo     | High Weight Product    |          102 |
| rio de janeiro | High Weight Product    |           53 |
| palmas        | High Weight Product    |           16 |
| belo horizonte | High Weight Product    |           14 |
| niteroi       | High Weight Product    |           13 |
| brasilia      | High Weight Product    |            9 |
| campinas      | High Weight Product    |            9 |
| santos        | High Weight Product    |            9 |
| salvador      | High Weight Product    |            9 |
| registro      | High Weight Product    |            8 |
+---------------+-----------------------+--------------+
10 rows in set (0.00 sec)
```

JOB 8 : Logistics based Optimization Insights:

Query

8.2-> How much products sold within seller state?

```
SELECT COALESCE(prodCat, 'TOTAL') AS prodCat, COUNT(order_id) as no_orders FROM ecom WHERE cust_state = seller_state GROUP BY prodCat WITH ROLLUP ;
```

8.2.1-> Ext Table Creation:

```
CREATE EXTERNAL TABLE job8c_op(prod_cat string, no_orders INT);
```

8.2.2-> Transferring o/p to ext table:

```
insert overwrite table job8c_op SELECT COALESCE(prodCat, 'TOTAL') AS prodCat, COUNT(order_id) as no_orders FROM ecom WHERE cust_state = seller_state
GROUP BY prodCat WITH ROLLUP;
```

JOB 8 : Logistics based Optimization Insights:

Output

```
mysql> select * from job8b;
+---------------------------------------+--------------+
| Product_Category                      | No_of_Orders |
+---------------------------------------+--------------+
| TOTAL                                 |        42052 |
| agro_industry_and_commerce            |           67 |
| air_conditioning                      |          107 |
| art                                   |           96 |
| arts_and_craftmanship                 |           17 |
| audio                                 |          146 |
| auto                                  |         1655 |
| baby                                  |         1124 |
| bed_bath_table                        |         5202 |
| books_general_interest                |          203 |
| books_imported                        |           37 |
| books_technical                       |          111 |
| cds_dvds_musicals                     |            6 |
| christmas_supplies                    |           49 |
| cine_photo                            |           29 |
| computers                             |           41 |
| computers_accessories                 |         2177 |
| consoles_games                        |          413 |
| construction_tools_construction       |          392 |
| construction_tools_lights             |          147 |
| flowers                               |           11 |
| food                                  |          280 |
| food_drink                            |          108 |
| furniture_bedroom                     |           51 |
| furniture_decor                       |         3202 |
| furniture_living_room                 |          168 |
| furniture_mattress_and_upholstery     |           24 |
| garden_tools                          |         1424 |
| health_beauty                         |         3709 |
| home_appliances                       |          355 |
| home_appliances_2                     |           74 |
```

1.1.1-> Ext Table Creation:

```
CREATE EXTERNAL TABLE job1_op(cust_id string, spendings float, cust_cat string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hive/warehouse/ecommerce/job1_op.txt';
```

2.1-> Ext Table Creation:

```
CREATE EXTERNAL TABLE job2_op(year INT, month string, sales float, sales_trend INT);
```

8.1.1-> Ext Table Creation:

Low Weight Products:

```
CREATE EXTERNAL TABLE job8a_op(cust_city string, prod_weightGm INT, count INT);
```

High Weight Products:

```
CREATE EXTERNAL TABLE job8b_op(cust_city string, prod_weightGm INT, count INT);
```

Overwritten the External Table :

1.1.2-> Transferring o/p to ext table:

```
insert overwrite table job1_op SELECT cust_id, ROUND(SUM(payment),3) AS spendings,
CASE
WHEN ROUND(SUM(payment),3) > 20000 THEN 'GROUP 1'
WHEN ROUND(SUM(payment),3) > 5000 THEN 'GROUP 2'
WHEN ROUND(SUM(payment),3) > 1500 THEN 'GROUP 3'
WHEN ROUND(SUM(payment),3) > 100 THEN 'GROUP 4'
ELSE 'GROUP 5'
END AS cust_cat
FROM ecom GROUP BY cust_id ORDER BY spendings DESC;
```

2.2-> Transferring o/p to ext table:

```
insert overwrite table job2_op
SELECT yr, CASE WHEN mon='01' THEN 'January'
WHEN mon='02' THEN 'February'
WHEN mon='03' THEN 'March'
WHEN mon='04' THEN 'April'
WHEN mon='05' THEN 'May'
WHEN mon='06' THEN 'June'
WHEN mon='07' THEN 'July'
WHEN mon='08' THEN 'August'
WHEN mon='09' THEN 'September'
WHEN mon='10' THEN 'October'
WHEN mon='11' THEN 'November'
WHEN mon='12' THEN 'December'
ELSE mon END AS month, sales ,
ROUND((sales - (LAG(sales) OVER()))) AS sales_trend
from(SELECT SUBSTRING(tstamp,7,4) as yr, SUBSTRING(tstamp,4,2) as mon, ROUND(SUM(qty*mrp),3) as sales from ecom
GROUP BY SUBSTRING(tstamp,7,4), SUBSTRING(tstamp,4,2) ORDER BY yr DESC, mon DESC) t;
```

Overwritten the External Table :

8.1.2-> Transferring o/p to ext table:

Low Weight Products:

```
WITH cte AS (
SELECT order_id, cust_id, cust_city, prod_weightgm, CASE WHEN prod_weightgm > 22000 THEN 'High Weight Product'
WHEN prod_weightgm < 2000 THEN 'Low Weight Product'
ELSE 'Medium Weight Product'
END AS pwt_cat
FROM ecom
) insert overwrite table job8a_op
SELECT cust_city, pwt_cat, COUNT(*) AS count from cte  where pwt_cat='Low Weight Product' GROUP BY cust_city, pwt_cat ORDER BY count desc LIMIT
10;
```

High Weight Products:

```
WITH cte AS (
SELECT order_id, cust_id, cust_city, prod_weightgm, CASE WHEN prod_weightgm > 22000 THEN 'High Weight Product'
WHEN prod_weightgm < 2000 THEN 'Low Weight Product'
ELSE 'Medium Weight Product'
END AS pwt_cat
FROM ecom
) insert overwrite table job8b_op
SELECT cust_city, pwt_cat, COUNT(*) AS count from cte  where pwt_cat='High Weight Product' GROUP BY cust_city, pwt_cat ORDER BY count desc LIMIT
10;
```

Walmart Data Analysis

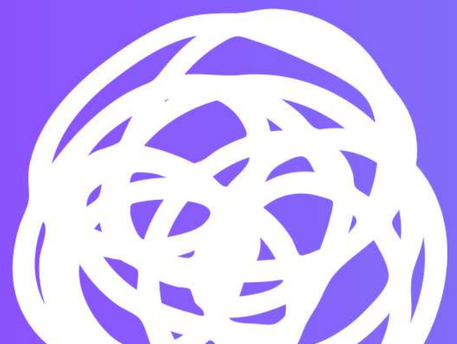
What is the Pearson correlation between High and Volume?

1. Ext Table Creation:

```
CREATE EXTERNAL TABLE pearson_corr(pearson_corr_high_volume
DOUBLE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
```

2.Transferring o/p to ext table:

```
insert overwrite table pearson_corr SELECT (Avg(high * volume) -
(Avg(high) * Avg(volume))) / (stddev_pop(high) * stddev_pop(volume))
AS ‘Pearsonsr’ from walmart;
```



What is the Pearson correlation between High and Volume?

```
mysql> show tables;
+-------------------------+
| Tables_in_BDSpark_pro |
+-------------------------+
| max_min_volume          |
| pearson_corr            |
| walmart                 |
+-------------------------+
3 rows in set (0.01 sec)

mysql> select * from pearson_corr;
+---------------------------+
| pearson_corr_high_volume |
+---------------------------+
|       -0.338432606173703 |
+---------------------------+
1 row in set (0.00 sec)
```

What is the max and min of the Volume column?

1-> Ext Table Creation:

```
CREATE EXTERNAL TABLE max_min_volume(maxV INT, minV INT) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

2-> Transferring o/p to ext table:

```
insert overwrite table max_min_volume select MAX(volume) as max_volume,
MIN(volume) as min_volume from walmart ;
```

```
ive> CREATE EXTERNAL TABLE max_min_volume(maxV INT, minV INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
   > ;
K
ime taken: 0.266 seconds
ive> insert overwrite table max_min_volume select MAX(volume) as max_volume, MIN(volume) as min_volume from walmart
   > ;
uery ID = cloudera_20230525065858_dcbfd0f0-fa99-422d-96be-ac2a6436b00f
otal jobs = 1
aunching Job 1 out of 1
umber of reduce tasks determined at compile time: 1
n order to change the average load for a reducer (in bytes):
 set hive.exec.reducers.bytes.per.reducer=<number>
n order to limit the maximum number of reducers:
 set hive.exec.reducers.max=<number>
n order to set a constant number of reducers:
 set mapreduce.job.reduces=<number>
tarting Job = job_1685004878679_0004, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1685004878679_0004/
ill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1685004878679_0004
adoop job information for Stage-1: number of mappers: 1; number of reducers: 1
023-05-25 06:58:27,561 Stage-1 map = 0%,  reduce = 0%
```

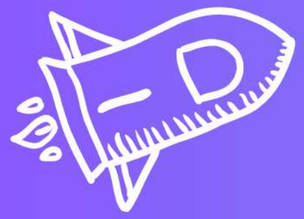
What is the max and min of the Volume column?

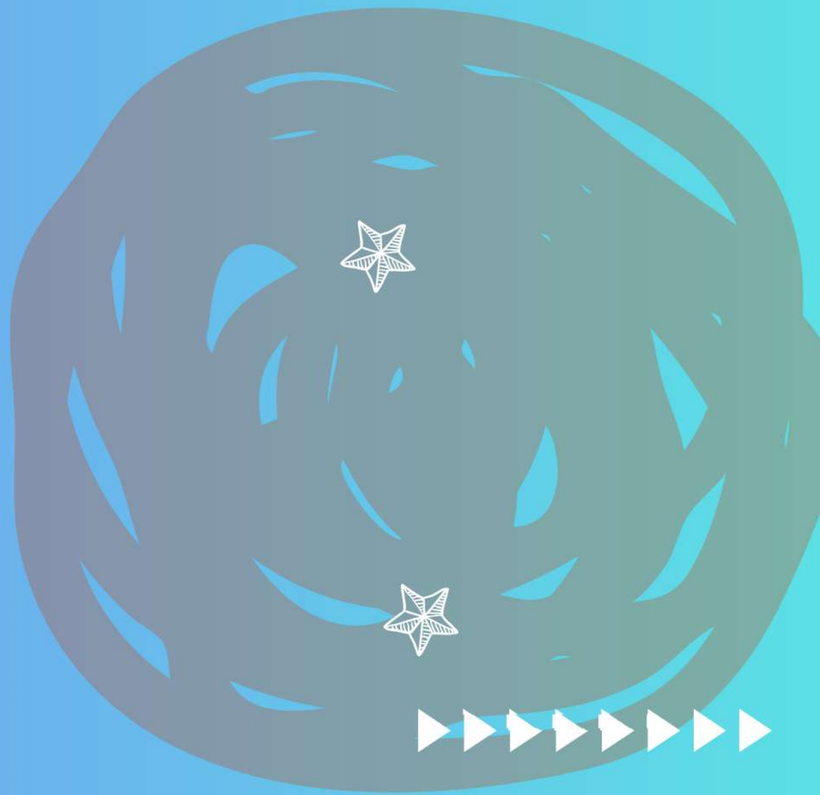
```
Loading data to table bdspark_pro.max_min_volume
Table bdspark_pro.max_min_volume stats: [numFiles=1, numRows=1, totalSize=17, rawDataSize=16]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.9 sec   HDFS Read: 91516 HDFS Write: 99 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 900 msec
OK
Time taken: 63.453 seconds
```

```
hive> show tables;
OK
max_min_volume
walmart
Time taken: 0.035 seconds, Fetched: 2 row(s)
hive> select * from max_min_volume;
OK
80898100        2094900
Time taken: 0.265 seconds, Fetched: 1 row(s)
```

Create a new table in Client Database and then use Sqoop pipeline to transfer the output.

PySpark





Walmart Data Analysis with Pyspark

1. Start a simple Spark Session

In [1]:
```
import findspark
findspark.init()
import pyspark

from pyspark.sql import SparkSession
```

In [3]:
```
spark=SparkSession.builder.appName('Big data Hadoop Spark Mini-Project').getOrCreate()
```

In [4]:
```
sc = spark.sparkContext
```

In [5]:
```
sc
```

Out[5]:

SparkContext

Spark UI

Version

`v3.3.2`

Master

`local[*]`

AppName

`Python Spark SQL Lesson`

Walmart Data Analysis with Pyspark

2. Load the Walmart Stock CSV File, have Spark infer the data types. What are the column names?

```
walmartDF = spark.read.load('walmart_stock.csv', format='csv', sep=',',
                            inferSchema='true', header='true')
```

```
walmartDF.show()
```

```
+-------------------+------------------+------------------+------------------+------------------+--------+------------------+
|               Date|              Open|              High|               Low|             Close|  Volume|         Adj Close|
+-------------------+------------------+------------------+------------------+------------------+--------+------------------+
|2012-01-03 00:00:00|         59.970001|         61.060001|         59.869999|         60.330002|12668800|52.619234999999996|
|2012-01-04 00:00:00|60.209998999999996|         60.349998|         59.470001|59.709998999999996| 9593300|         52.078475|
|2012-01-05 00:00:00|         59.349998|         59.619999|         58.369999|         59.419998|12768200|         51.825539|
|2012-01-06 00:00:00|         59.419998|         59.450001|         58.869999|              59.0| 8069400|          51.45922|
|2012-01-09 00:00:00|         59.029999|         59.549999|         58.919998|             59.18| 6679300|51.616215000000004|
|2012-01-10 00:00:00|             59.43|59.709998999999996|             58.98|59.040001000000004| 6907300|         51.494109|
```

3. What does the Schema look like?

```
In [7]: walmartDF.printSchema()
```

```
root
 |-- Date: timestamp (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)
```

Walmart Data Analysis with Pyspark

4. Print out the first 5 columns.

```
In [11]: walmartDF.show(5)
```

```
+-------------------+-----------------+---------+---------+-----------------+--------+-----------------+
|               Date|             Open|     High|      Low|            Close|  Volume|        Adj Close|
+-------------------+-----------------+---------+---------+-----------------+--------+-----------------+
|2012-01-03 00:00:00|        59.970001|61.060001|59.869999|        60.330002|12668800|52.619234999999996|
|2012-01-04 00:00:00|60.209998999999996|60.349998|59.470001|59.709998999999996| 9593300|        52.078475|
|2012-01-05 00:00:00|        59.349998|59.619999|58.369999|        59.419998|12768200|        51.825539|
|2012-01-06 00:00:00|        59.419998|59.450001|58.869999|             59.0| 8069400|         51.45922|
|2012-01-09 00:00:00|        59.029999|59.549999|58.919998|            59.18| 6679300|51.616215000000004|
+-------------------+-----------------+---------+---------+-----------------+--------+-----------------+
only showing top 5 rows
```

OR

```
In [99]: walmartDF.take(5)
```

```
Out[99]: [Row(Date=datetime.datetime(2012, 1, 3, 0, 0), Open=59.970001, High=61.060001, Low=59.869999, Close=60.330002, Volume=12668800,
Adj Close=52.619234999999996),
 Row(Date=datetime.datetime(2012, 1, 4, 0, 0), Open=60.209998999999996, High=60.349998, Low=59.470001, Close=59.709998999999996
, Volume=9593300, Adj Close=52.078475),
 Row(Date=datetime.datetime(2012, 1, 5, 0, 0), Open=59.349998, High=59.619999, Low=58.369999, Close=59.419998, Volume=12768200,
Adj Close=51.825539),
 Row(Date=datetime.datetime(2012, 1, 6, 0, 0), Open=59.419998, High=59.450001, Low=58.869999, Close=59.0, Volume=8069400, Adj C
lose=51.45922),
 Row(Date=datetime.datetime(2012, 1, 9, 0, 0), Open=59.029999, High=59.549999, Low=58.919998, Close=59.18, Volume=6679300, Adj
Close=51.616215000000004)]
```

Walmart Data Analysis with Pyspark

7. **Create a new dataframe with a column called HV Ratio that is the ratio of the High Price versusvolume of stock traded for a day.**

```
In [131]: wmt_HVRatioDF = walmartDF.withColumn("HV Ratio", walmartDF.High/walmartDF.Volume)
```

```
In [133]: wmt_HVRatioDF.select("HV Ratio").show()
```

```
+--------------------+
|            HV Ratio|
+--------------------+
|4.819714653321546E-6|
|6.290848613094555E-6|
|4.669412994783916E-6|
|7.367338463826307E-6|
|8.915604778943901E-6|
|8.644477436914568E-6|
|9.351828421515645E-6|
| 8.29141562102703E-6|
|7.712212102001476E-6|
|7.071764823529412E-6|
|1.015495466386981E-5|
|6.576354146362592...|
| 5.90145296180676E-6|
|8.547679455011844E-6|
|8.420709512685392E-6|
|1.041448341728929...|
|8.316075414862431E-6|
|9.721183814992126E-6|
```

Walmart Data Analysis with Pyspark(RDD)

```
# 1. Creating RDD for walmart dataset

walmart = sc.textFile('walmart_stock.csv')
```

```
# 2. Removing 1st row of headers from RDD

first_row_index = 0
walmartRDD = walmart.zipWithIndex().filter(lambda x: x[1] > first_row_index).map(lambda x: x[0])
```

```
# 2.1 What zipWWithIndex() did to RDD

first_row_index = 0
w1 = walmart.zipWithIndex()
w1.take(5)
```

```
[('Date,Open,High,Low,Close,Volume,Adj Close', 0),
 ('03-01-2012,59.970001,61.060001,59.869999,60.330002,12668800,52.619235', 1),
 ('04-01-2012,60.209999,60.349998,59.470001,59.709999,9593300,52.078475', 2),
 ('05-01-2012,59.349998,59.619999,58.369999,59.419998,12768200,51.825539', 3),
 ('06-01-2012,59.419998,59.450001,58.869999,59,8069400,51.45922', 4)]
```

Headers removed with Method 1:

```
# 2.2 What filter() did with indexed RDD

w2 = walmart.zipWithIndex().filter(lambda x: x[1] > first_row_index)
w2.take(5)
```

```
[('03-01-2012,59.970001,61.060001,59.869999,60.330002,12668800,52.619235', 1),
 ('04-01-2012,60.209999,60.349998,59.470001,59.709999,9593300,52.078475', 2),
 ('05-01-2012,59.349998,59.619999,58.369999,59.419998,12768200,51.825539', 3),
 ('06-01-2012,59.419998,59.450001,58.869999,59,8069400,51.45922', 4),
 ('09-01-2012,59.029999,59.549999,58.919998,59.18,6679300,51.616215', 5)]
```

```
# 2.3 Finally getting real workable RDD

w3 = walmart.zipWithIndex().filter(lambda x: x[1] > first_row_index).map(lambda x: x[0])
w3.take(5)[0]
```

```
'03-01-2012,59.970001,61.060001,59.869999,60.330002,12668800,52.619235'
```

Headers removed with Method 2:

```
# 3. Removing headers using alternative method

header = walmart.first()
w4 = walmart.filter(lambda row: row!=header)
w4.take(2)
```

```
['03-01-2012,59.970001,61.060001,59.869999,60.330002,12668800,52.619235',
 '04-01-2012,60.209999,60.349998,59.470001,59.709999,9593300,52.078475']
```

To get each record

```
# 4. Splitting records using comma -> " , " as delimiter

walmart_split = walmartRDD.map(lambda x : x.split(','))
walmart_split.take(1)[0][1]
```

```
# 5. Converting Data types for required columns using function

def asInt(x):
    return [x[0], float(x[1]), float(x[2]), float(x[3]), float(x[4]),float(x[5]), float(x[6])]
#       return x[0], x[1]

walmart_fnl = walmart_split.map(asInt)
walmart_fnl.take(1)
```

```
[['03-01-2012',
  59.970001,
  61.060001,
  59.869999,
  60.330002,
  12668800.0,
  52.619235]]
```

Finally we got RDD for walmart dataset,

Problem Statements

1. **What is the mean of the Close column? (RDD)**

```
# Getting only values of column "Close"

close_prices = walmart_fnl.map(lambda x: x[4])      #  close_prices is of type --> pyspark.rdd.PipelinedRDD
close_prices.mean()

# Finding mean of Close prices

meanClose = (close_prices.mean())
print(f"\nMean of Close column in Walmart : \n{meanClose}")
```

```
Mean of Close column in Walmart :
72.38844998012719
```

Problem Statements

2. What is the max High per year? ----> (RDD)

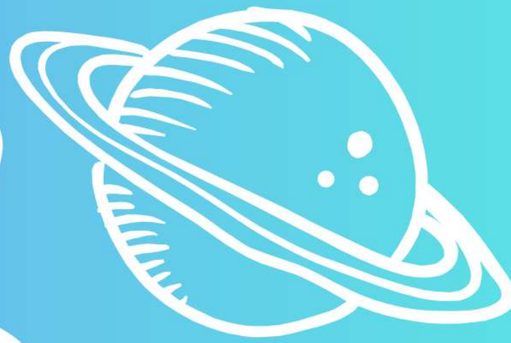
```
# Using map() to make K,V pairs of (year, High)

yearHigh_paired = walmart_fnl.map(lambda x: (int(x[0][6:]), x[2]))
yearHigh_paired.take(5)
```

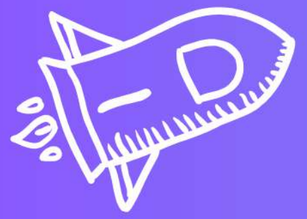
```
[(2012, 61.060001),
 (2012, 60.349998),
 (2012, 59.619999),
 (2012, 59.450001),
 (2012, 59.549999)]
```

```
# Using reduceByKey() to group year by max of High

maxHigh_perYr = yearHigh_paired.reduceByKey(max)
print('Max High of Walmart stocks every Year(2012-2016): \n\n' ,maxHigh_perYr.sortBy(lambda x: x[1], ascending=False).take(5))
```

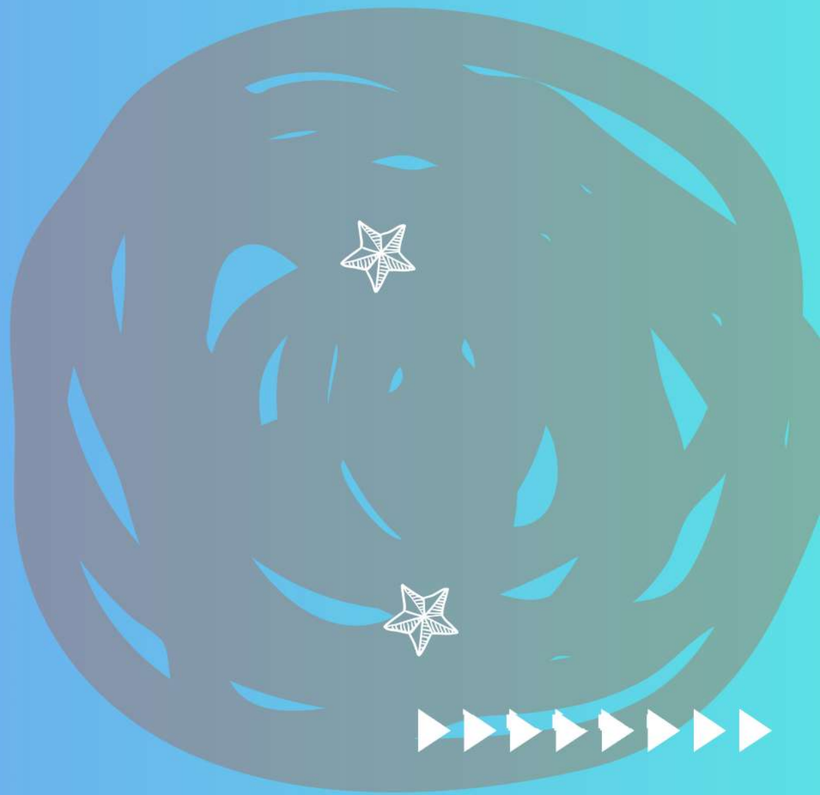
```
Max High of Walmart stocks every Year(2012-2016):

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

Spark SQL







Creating a Temp View for Walmart Dataframe

Creating SQL table from DF

19]:
```
walmartDF.createOrReplaceTempView("walmart")
spark.sql("select * from walmart limit 5").show()
```

```
+-------------------+------------------+---------+---------+------------------+--------+------------------+
|               Date|              Open|     High|      Low|             Close|  Volume|         Adj Close|
+-------------------+------------------+---------+---------+------------------+--------+------------------+
|2012-01-03 00:00:00|         59.970001|61.060001|59.869999|         60.330002|12668800|52.619234999999996|
|2012-01-04 00:00:00|60.209998999999996|60.349998|59.470001|59.709998999999996| 9593300|         52.078475|
|2012-01-05 00:00:00|         59.349998|59.619999|58.369999|         59.419998|12768200|         51.825539|
|2012-01-06 00:00:00|         59.419998|59.450001|58.869999|              59.0| 8069400|          51.45922|
|2012-01-09 00:00:00|         59.029999|59.549999|58.919998|             59.18| 6679300|51.616215000000004|
+-------------------+------------------+---------+---------+------------------+--------+------------------+
```

Walmart Data Analysis with SparkSQL

1. What day had the Peak High in Price?

In [45]:
```
# walmartDF.orderBy(walmartDF['High'].desc()) \
#                                  .select(['Date']) \
#                                  .head(1)[0]['Date']

peakHighDF = spark.sql("SELECT Date FROM walmart WHERE High = (SELECT MAX(High) FROM walmart)")
peakHighDF.head()['Date']
```

Out[45]:
```
datetime.datetime(2015, 1, 13, 0, 0)
```

Walmart Data Analysis with SparkSQL

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

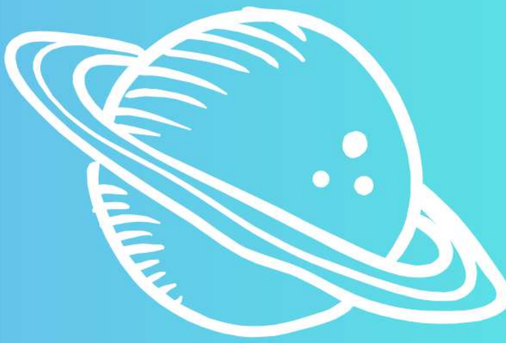
In [81]:
```
avgCloseDF = spark.sql("SELECT YEAR(Date) AS year, MONTH(Date) AS month, AVG(Close) AS avg_close\
                        FROM walmart GROUP BY YEAR(Date), month(Date)")
avgCloseDF.show(60)
```

```
+----+-----+------------------+
|year|month|         avg_close|
+----+-----+------------------+
|2012|   10| 75.30619061904761|
|2015|    2| 85.52315805263159|
|2014|    4| 77.80857085714285|
|2015|   12| 59.98681827272728|
|2016|    7| 73.54149939999999|
|2016|   11| 70.30476261904762|
|2012|    8| 73.04478265217392|
|2013|    2| 70.62315857894738|
|2012|    4|60.149000150000006|
|2012|   12| 69.71100009999999|
|2014|   10| 76.48869486956522|
|2016|    5| 68.05285676190476|
|2014|   12|  85.1259102727273|
|2013|    9|        74.4395005|
|2013|   10| 74.97913104347826|
|2014|    5| 77.38095276190477|
|2016|    2| 66.24800044999999|
|2013|   12|  78.7752382857143|
|2014|    1| 76.53142833333334|
|2013|    3| 73.43649940000002|
|2014|    8| 74.67666623809525|
|2013|    6| 74.97800020000001|
|2012|    7| 73.40666661904762|
```

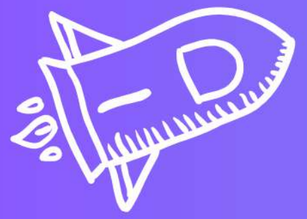
Walmart Data Analysis with SparkSQL

5. What percentage of the time was the High greater than 80 dollars ? In other words, (Number of Days High>80)/(Total Days in the dataset)

```
In [69]: high_greater80DF = spark.sql("SELECT Date FROM walmart WHERE High > 80")
         (high_greater80DF.count() / walmartDF.count()) * 100

Out[69]: 9.141494435612083
```



SQOOP EXPORT



EXPORT data to Target DB: (MySQL)

Ecommerce Dataset Sqoop Export Commands

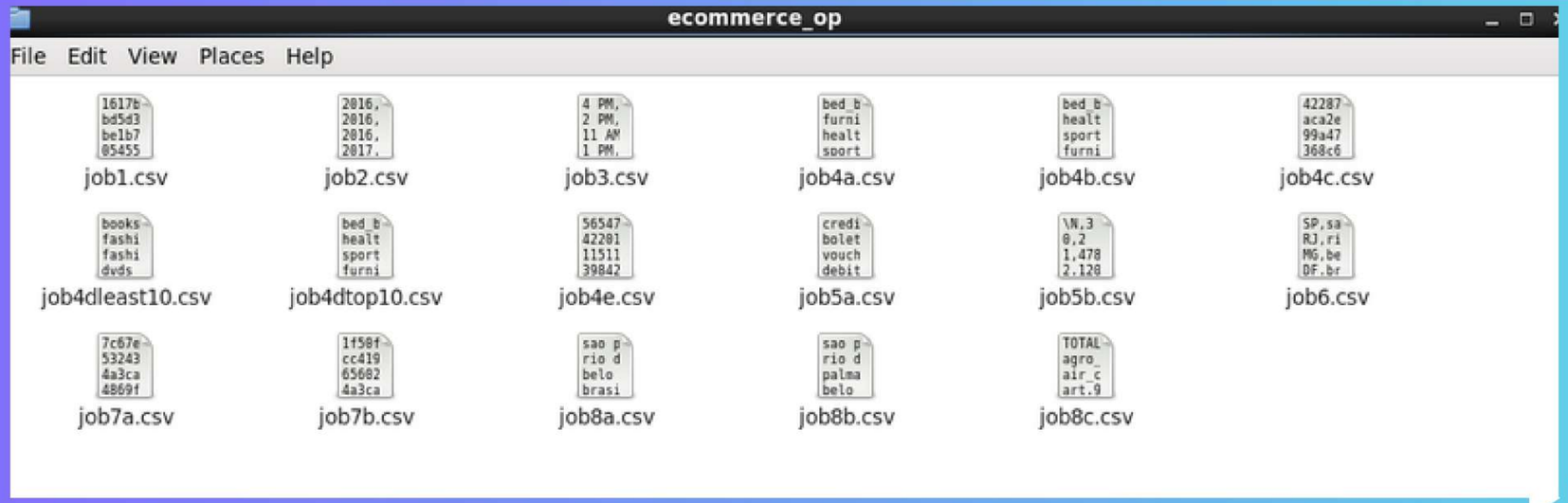
```
sqoop export \
--connect jdbc:mysql://localhost:3306/ecommerce \
--username root --password cloudera \
--table job1 \
--export-dir /user/hive/warehouse/ecommerce/job1_op1.txt
```

Sqoop Export for Ecommerce

```
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job1 --export-dir
/user/hive/warehouse/ecommerce/job1_op.txt;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job2 --export-dir
/user/hive/warehouse/ecommerce.db/job2_op --input-null-string '\\N'  --input-null-non-string '\\N';
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job3 --export-dir
/user/hive/warehouse/ecommerce.db/job3_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job4a --export-dir
/user/hive/warehouse/ecommerce.db/job4a_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job4b --export-dir
/user/hive/warehouse/ecommerce.db/job4b_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job4c --export-dir
/user/hive/warehouse/ecommerce.db/job4c_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job4d_Least10 --export-dir
/user/hive/warehouse/ecommerce.db/job4dleast10_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job4d_Top10 --export-dir
/user/hive/warehouse/ecommerce.db/job4dtop10_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job4e --export-dir
/user/hive/warehouse/ecommerce.db/job4e;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job5a --export-dir
/user/hive/warehouse/ecommerce.db/job5a_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job5b --export-dir
/user/hive/warehouse/ecommerce.db/job5b_op --input-null-string '\\N'  --input-null-non-string '\\N'; ;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job6 --export-dir
/user/hive/warehouse/ecommerce.db/job6_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job7a --export-dir
/user/hive/warehouse/ecommerce.db/job7a_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job7b --export-dir
/user/hive/warehouse/ecommerce.db/job7b_op;
sqoop export --connect jdbc:mysql://localhost:3306/ecommerce --username root --password cloudera --table job8a_HighWt --export-dir
/user/hive/warehouse/ecommerce.db/job8a_op;
```

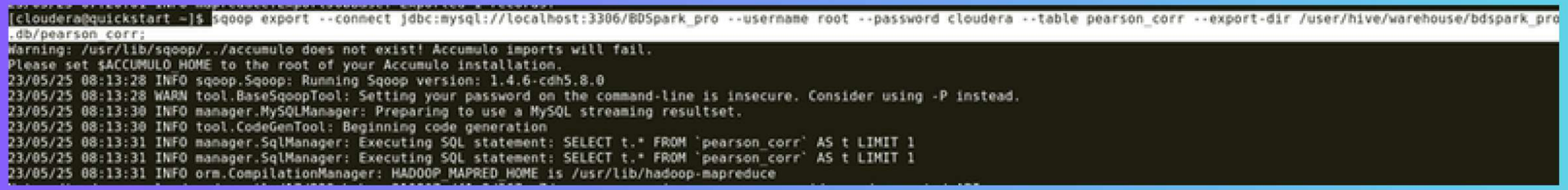
Getting files from Client DB(MySQL) to Local machine

```
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job5a_op/000000_0 /home/cloudera/ecommerce/job5a.csv
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job5b_op/000000_0 /home/cloudera/ecommerce/job5b.csv
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job6_op/000000_0 /home/cloudera/ecommerce/job6.csv
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job7a_op/000000_0 /home/cloudera/ecommerce/job7a.csv
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job7b_op/000000_0 /home/cloudera/ecommerce/job7b.csv
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job8a_op/000000_0 /home/cloudera/ecommerce/job8a.csv
cloudera@quickstart ~]$
cloudera@quickstart ~]$ hdfs dfs -get /user/hive/warehouse/ecommerce.db/job8b_op/000000_0 /home/cloudera/ecommerce/job8b.csv
```

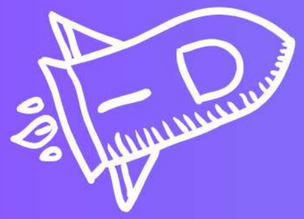


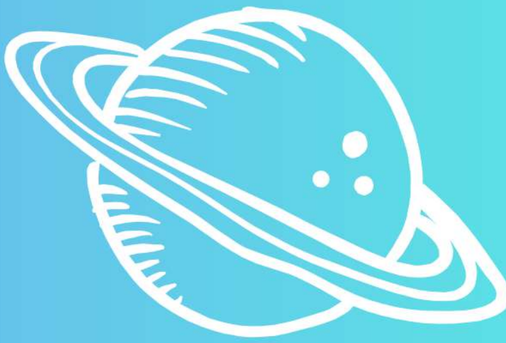
Walmart Output Data Transfer

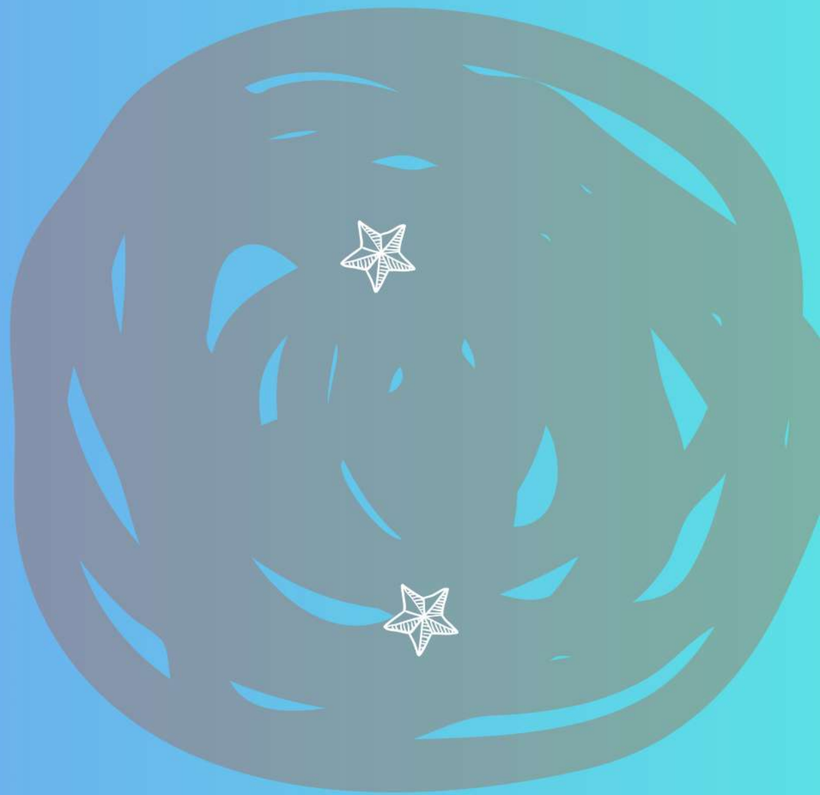
```
sqoop export --connect jdbc:mysql://localhost:3306/BDSpark_pro --username root  \
--password cloudera --table max_min_volume --export-dir \
/user/hive/warehouse/bdspark_pro.db/max_min_volume;
```



Visualization

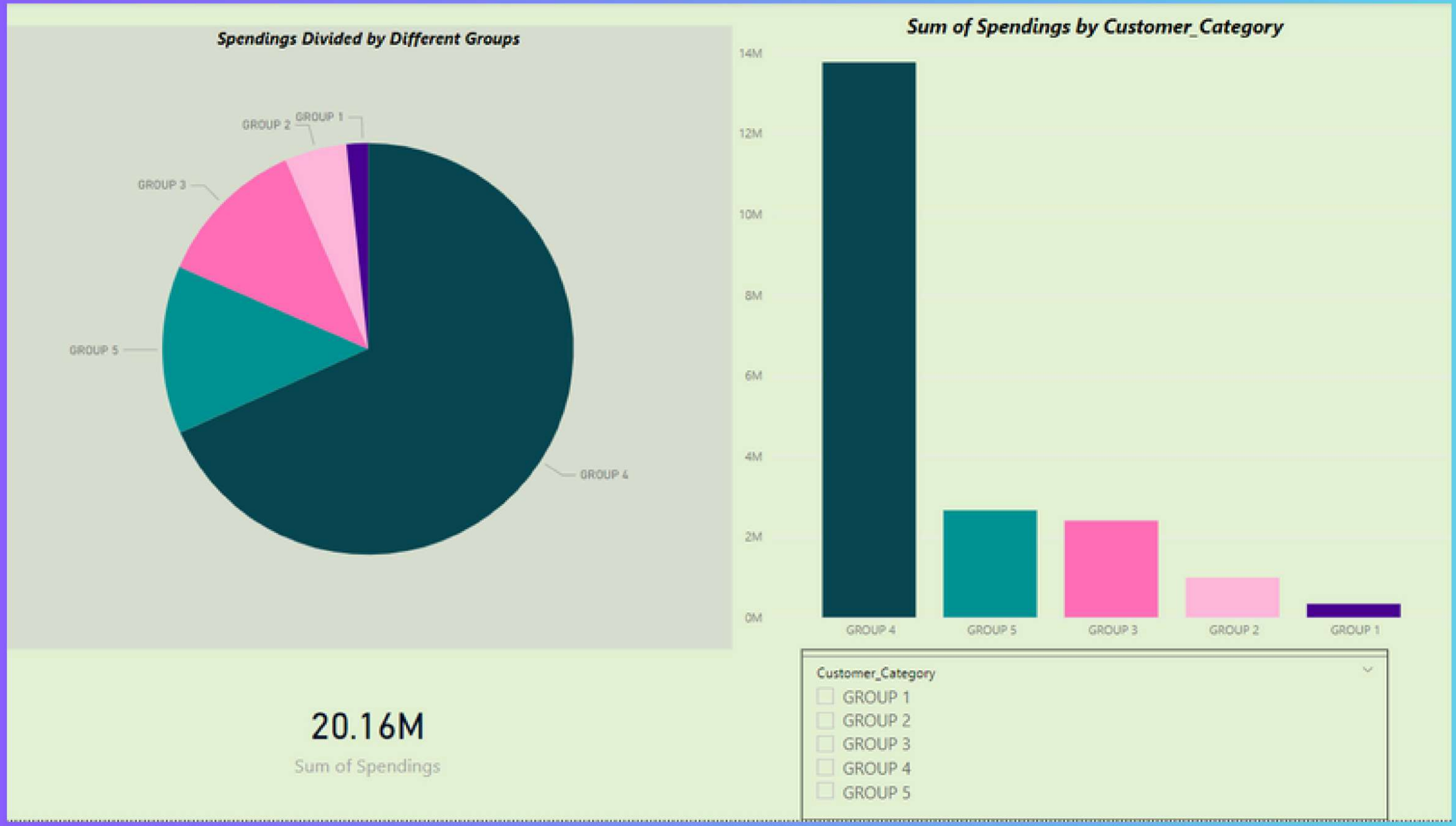




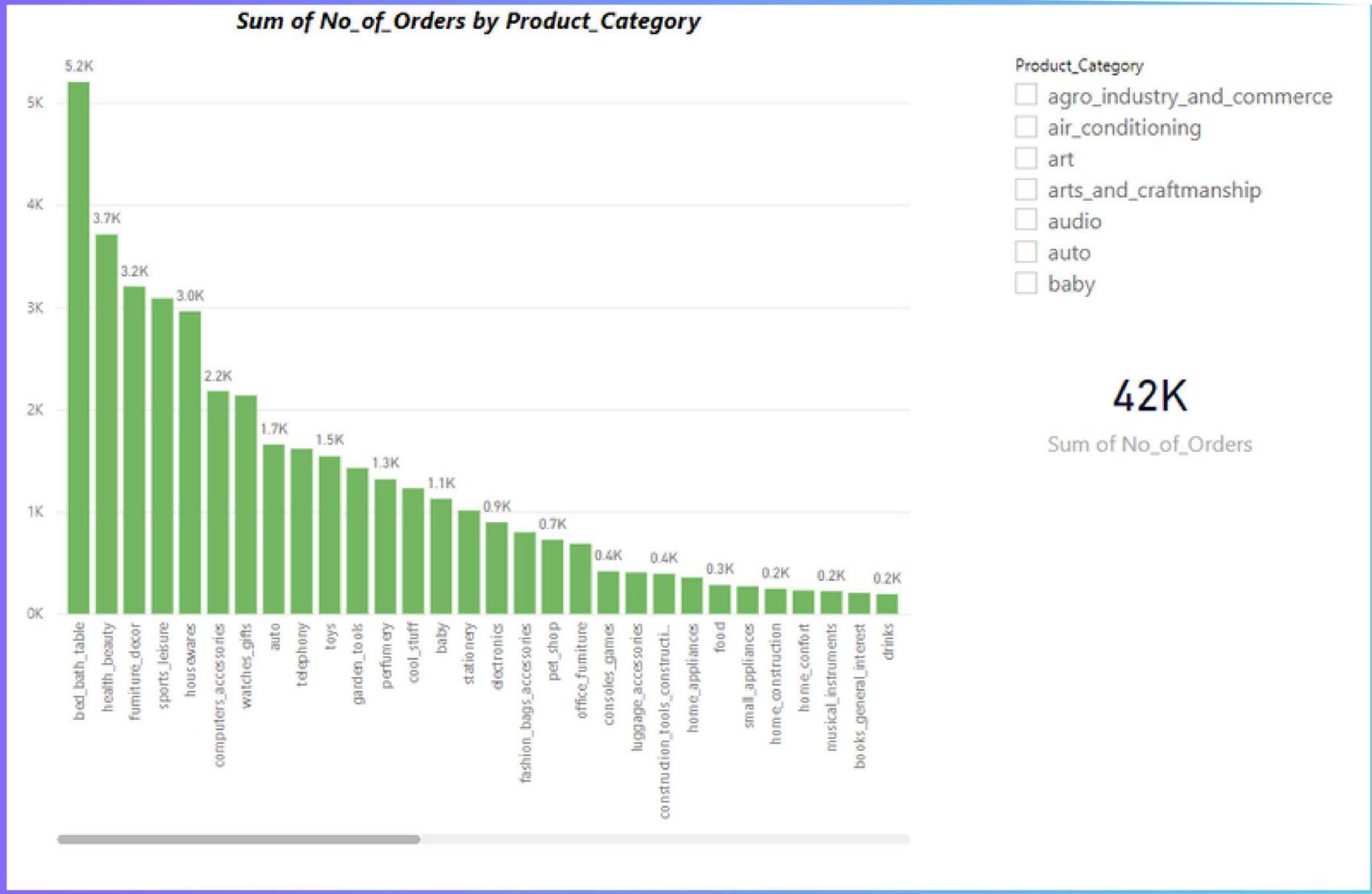


Power BI

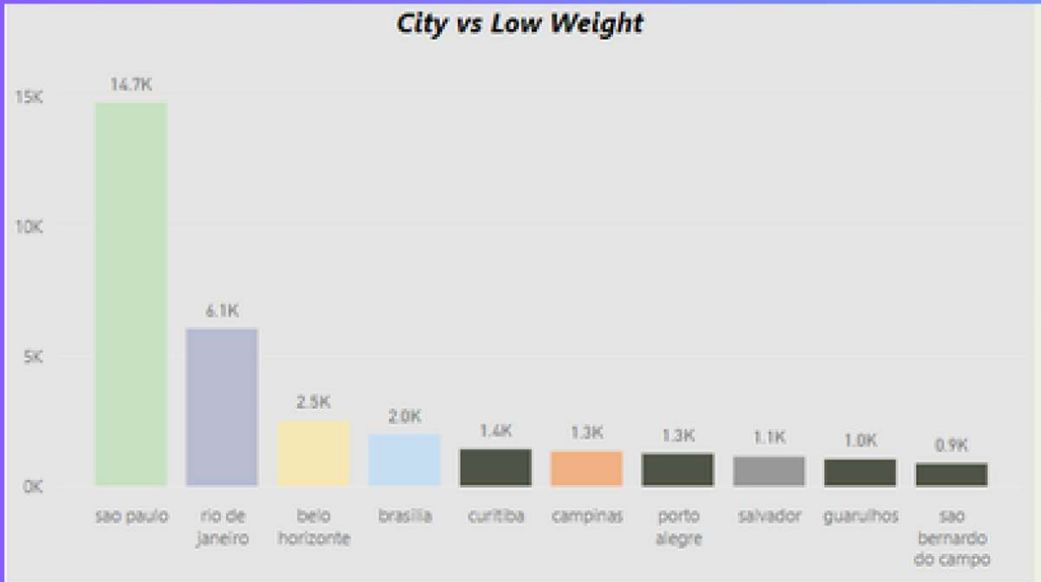
Customer Segmentation:
Categorizing customers based on their spending

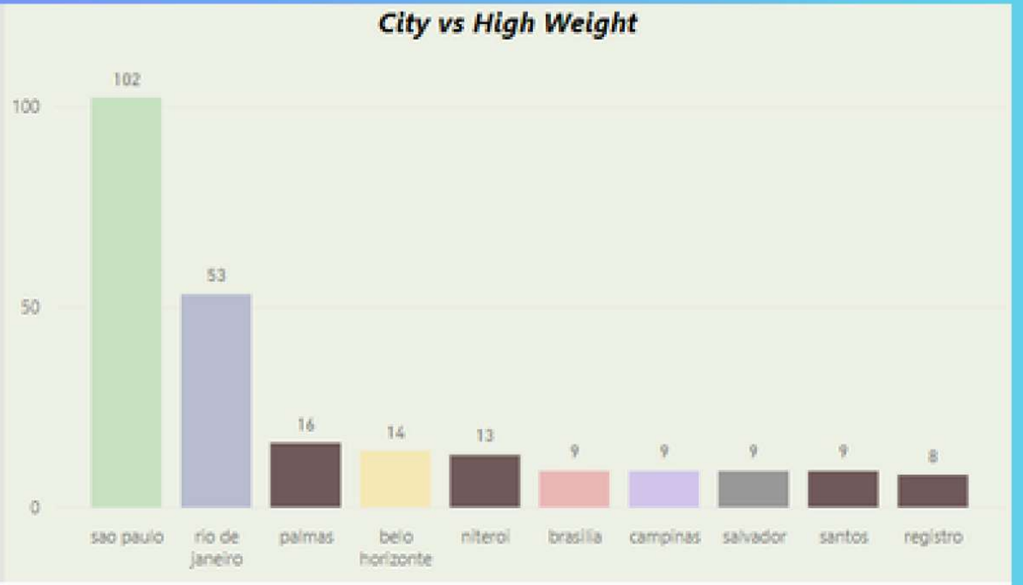


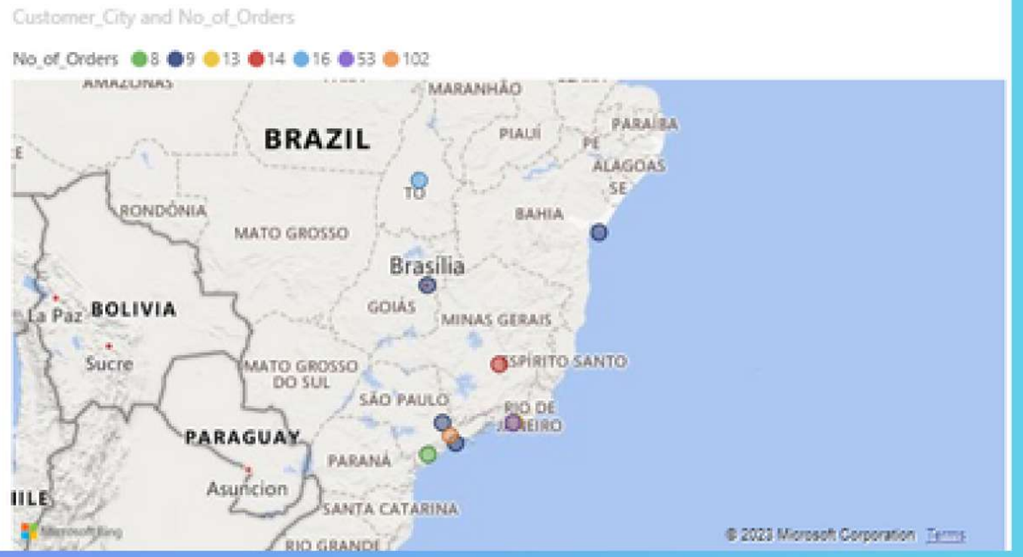
Plotting total orders purchased in particular category



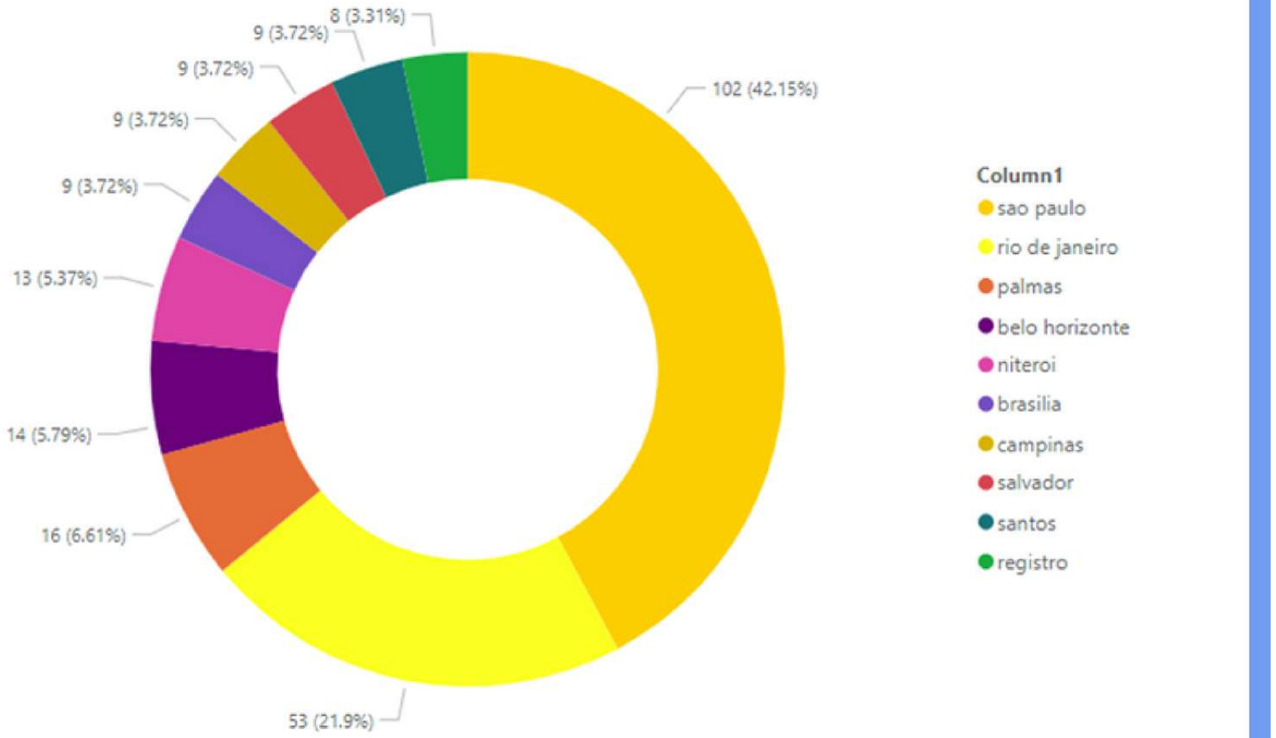
Potential Customer's Location:
Where do most customers come from?

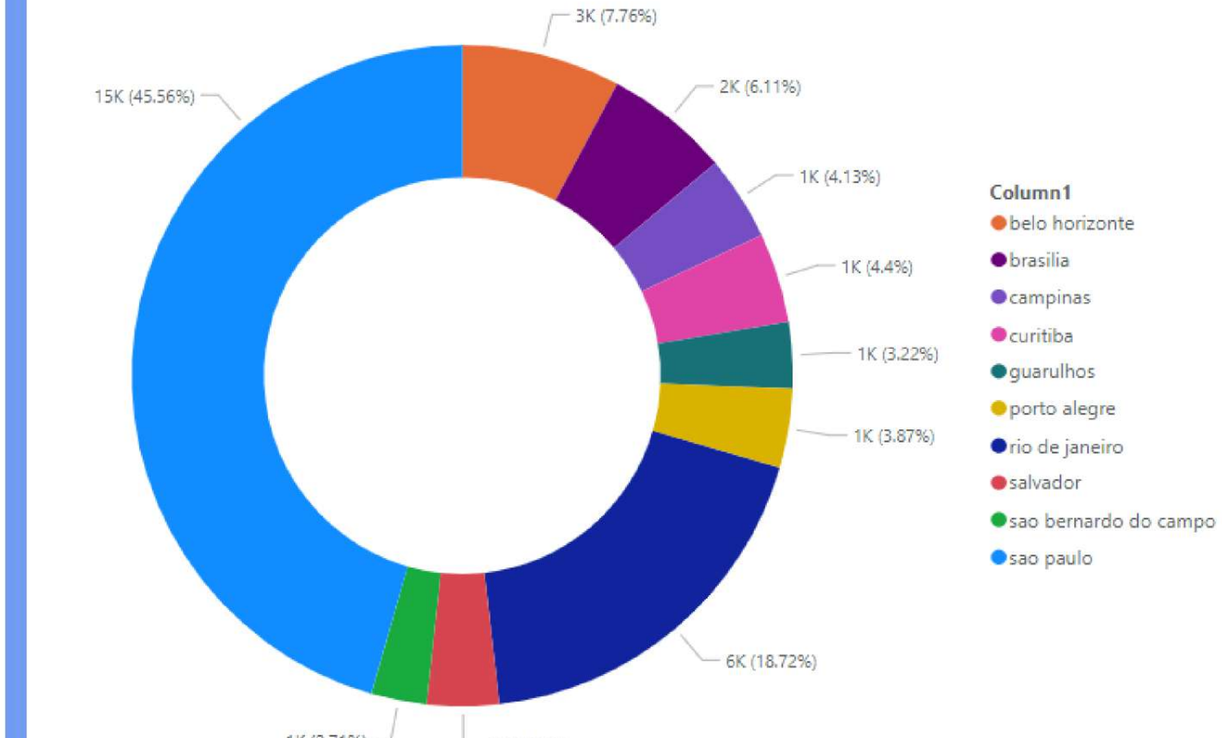




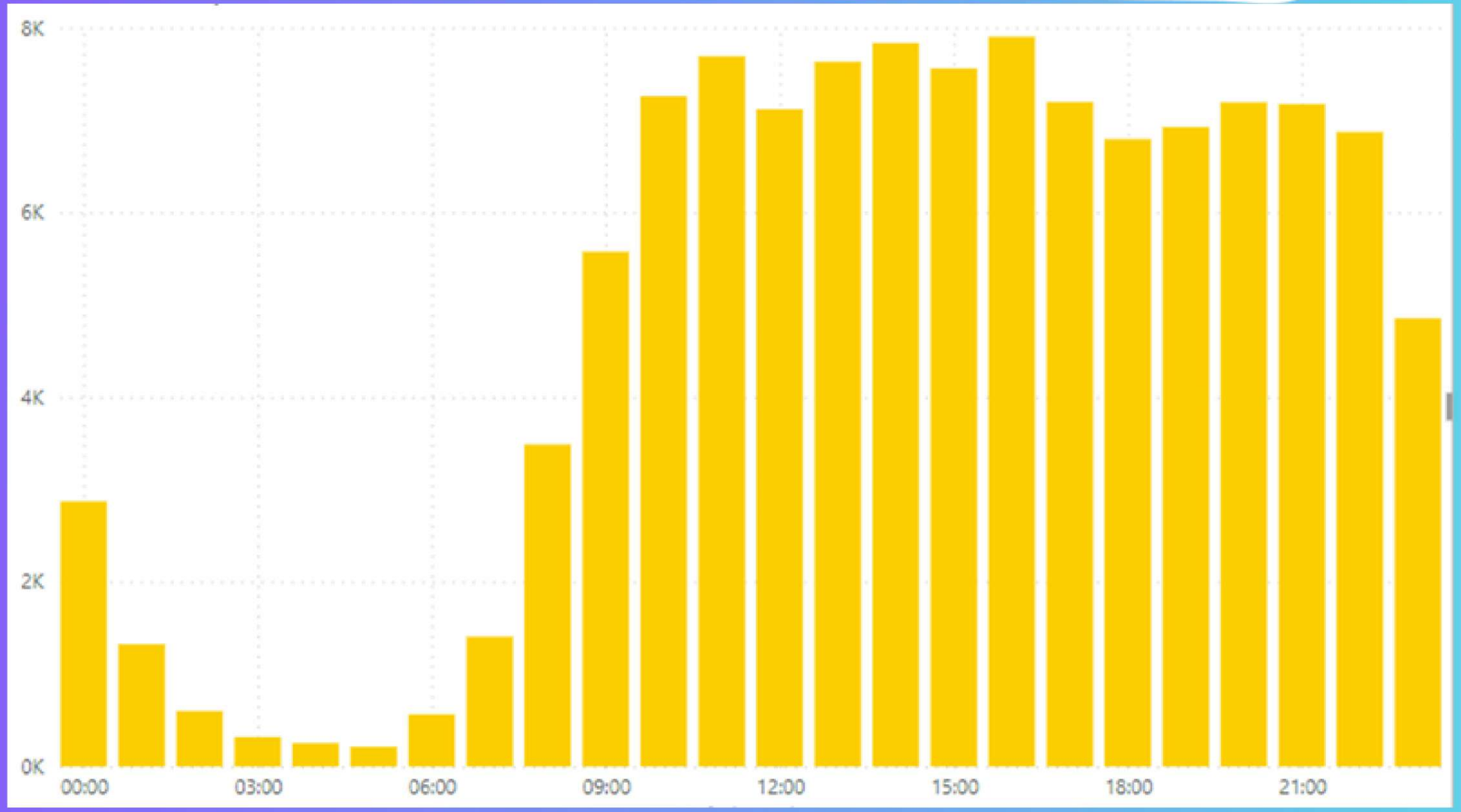


Logistics based Optimization Insights:
Which city buys heavy weight products and light weight products?

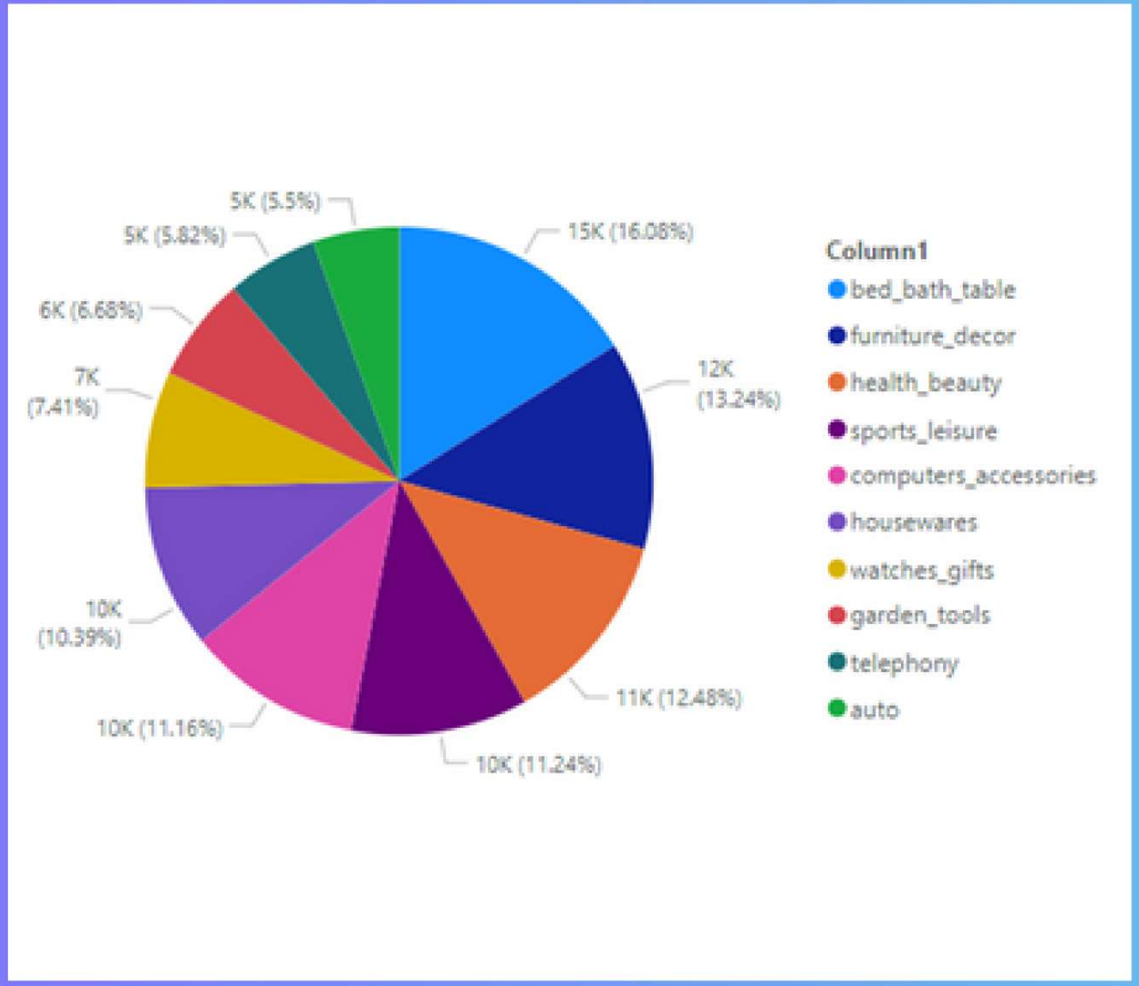




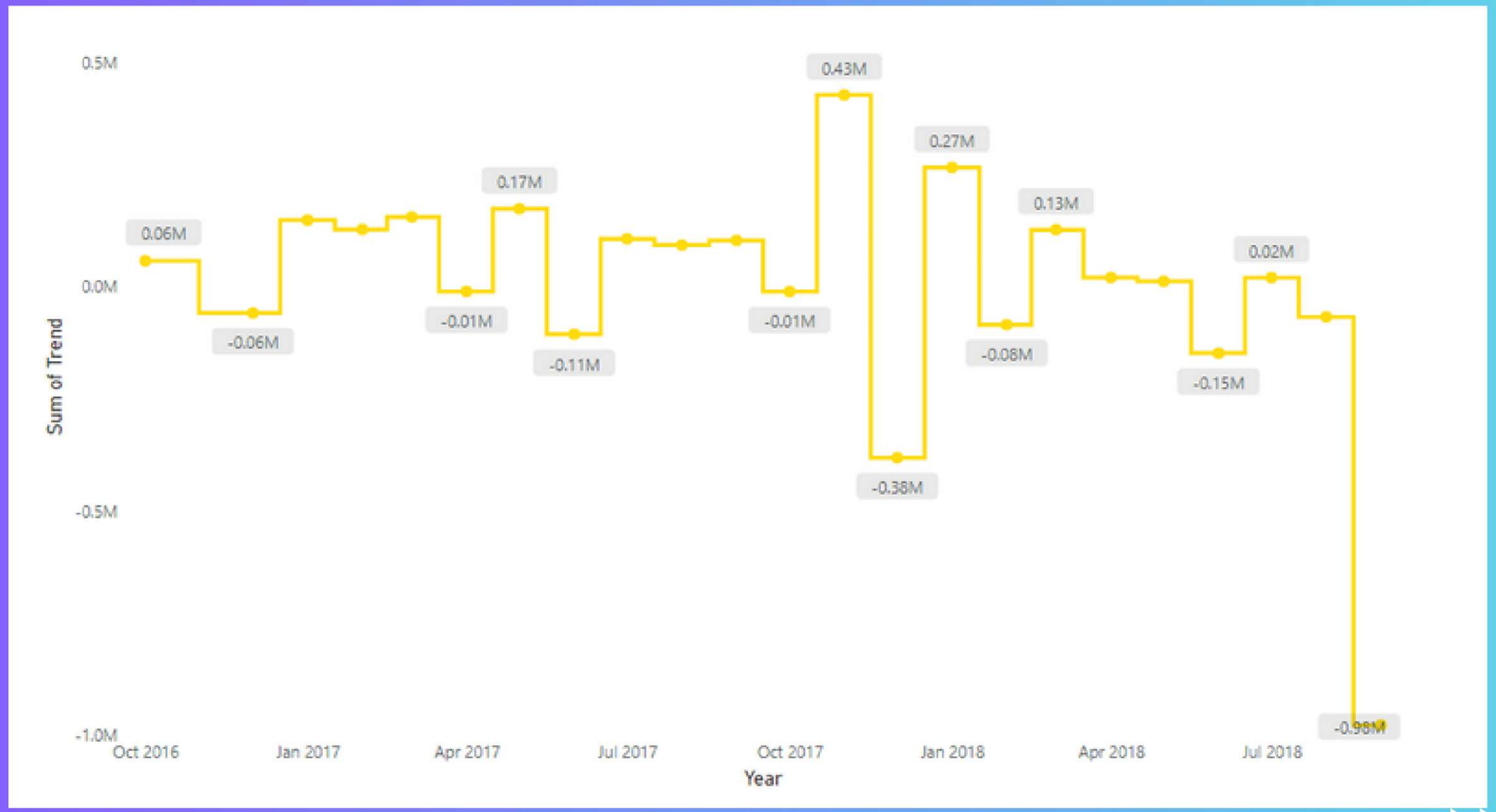
Hourly Sales Analysis:
Which hour has more number of sales?



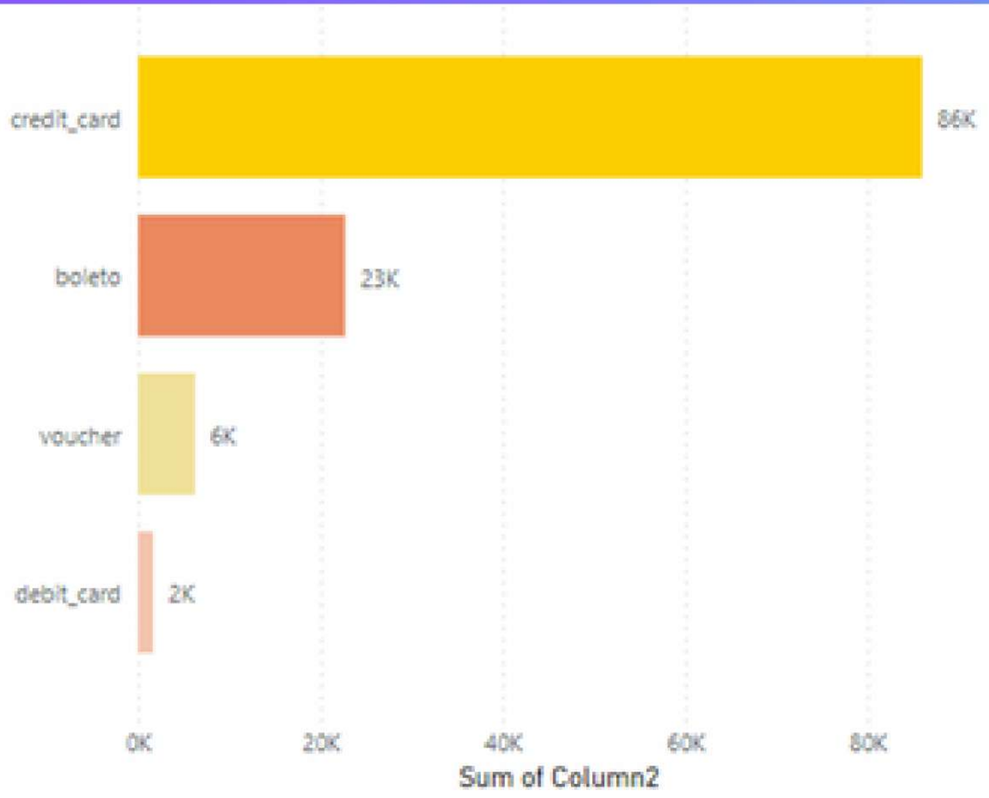
Product Based Analysis:
Which category product has sold more?

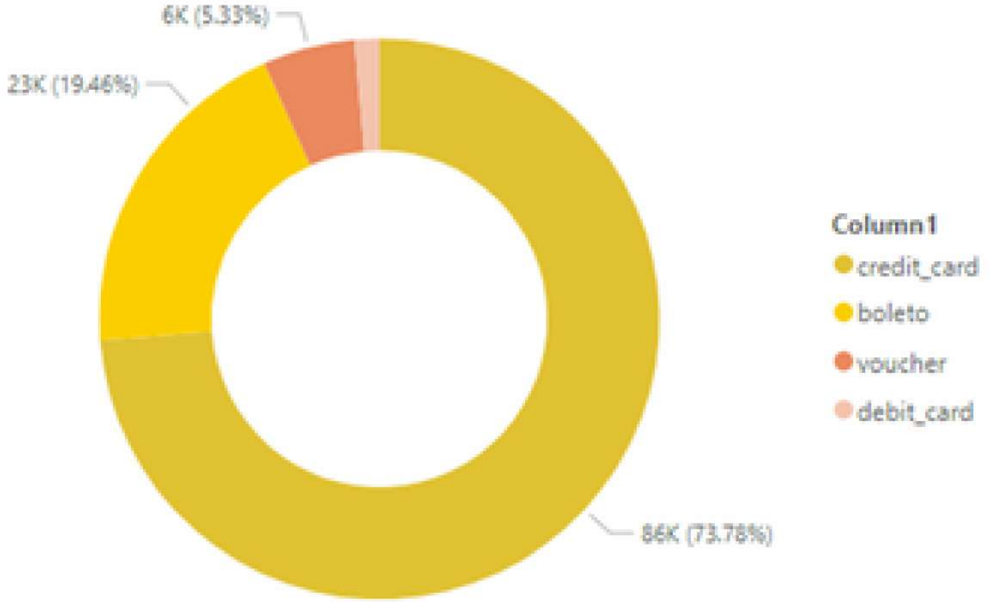


Ecommerce Data Analysis

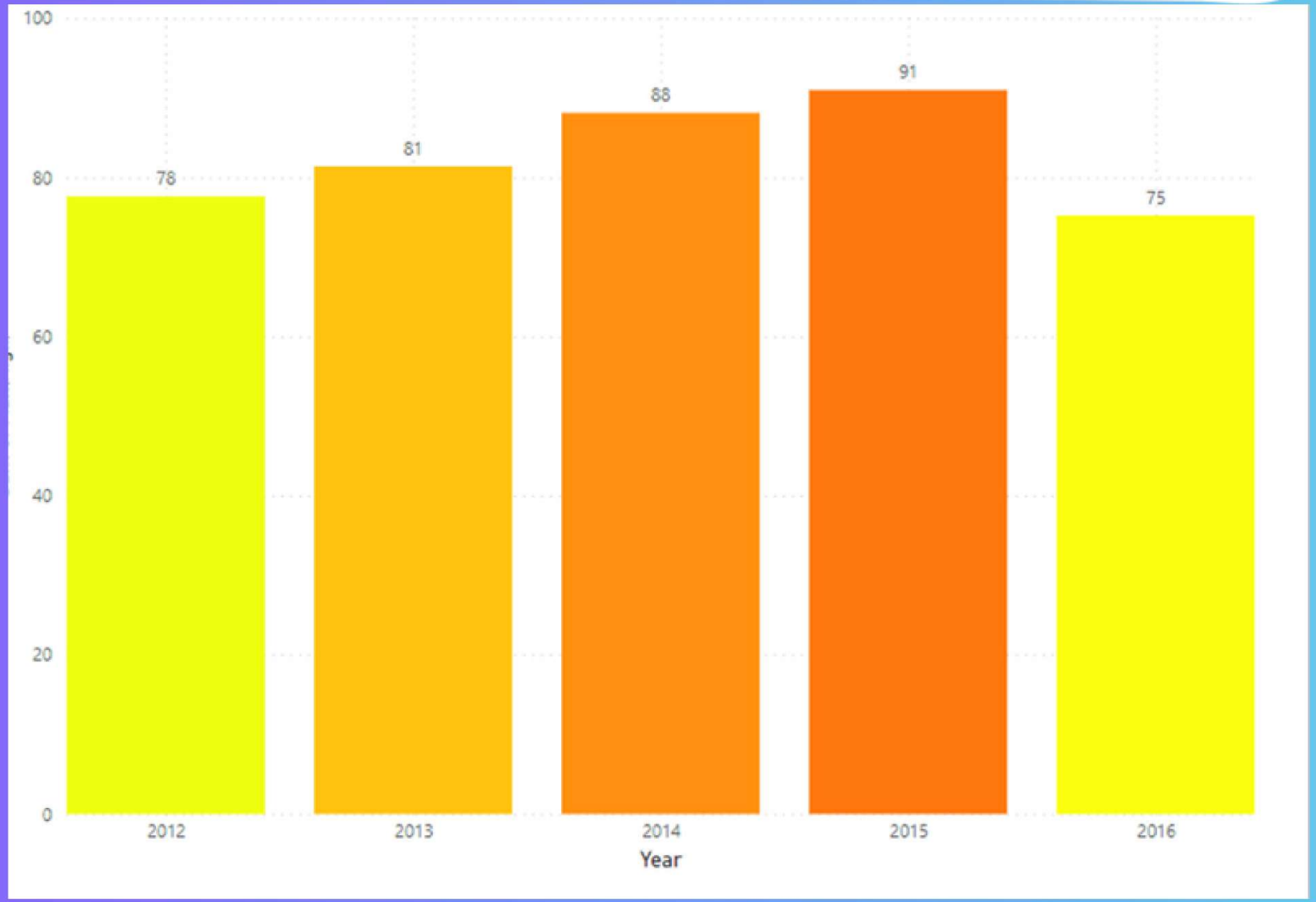


Payment Preference:
What are the most commonly used payment types?





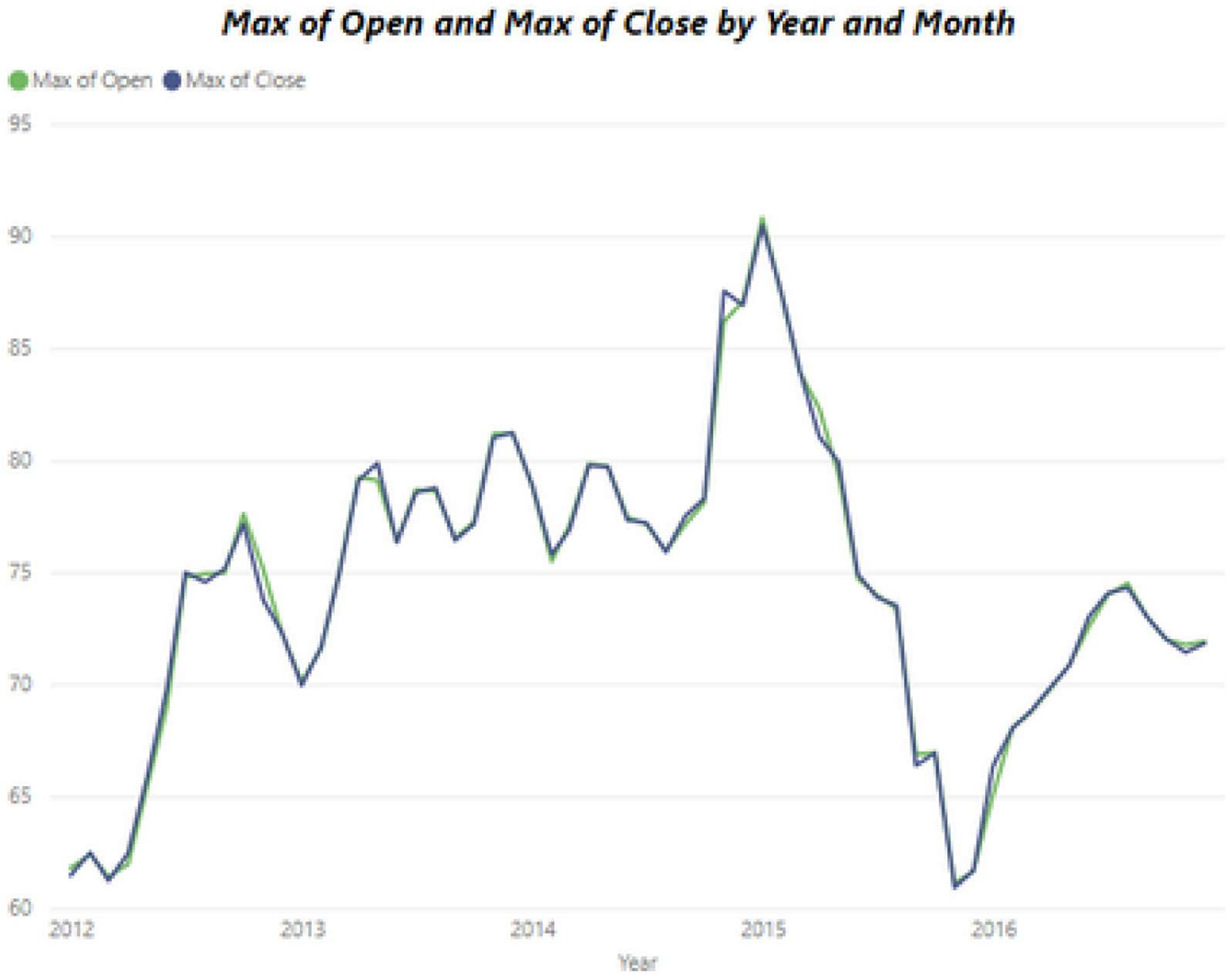
Walmart: Max High per Year



Month wise max and min price of stocks over 5 years

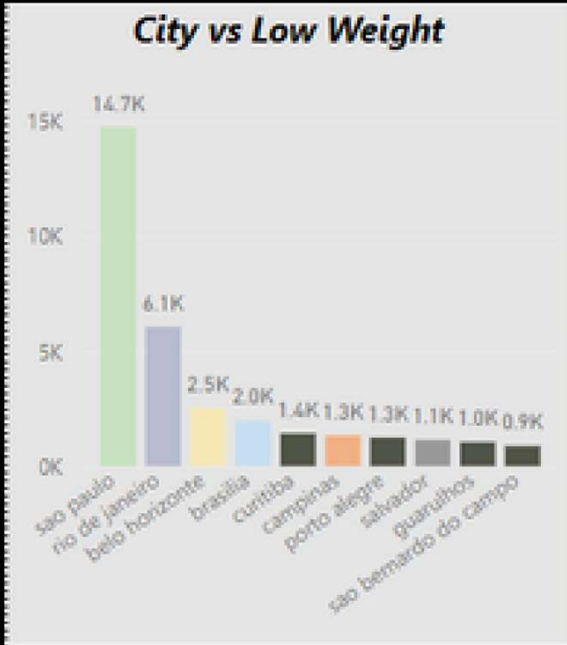
January			
2012	Year	1,211.44	Sum of High
		1,196.06	Sum of Low
February			
2012	Year	1,224.48	Sum of High
		1,211.65	Sum of Low
March			
2012	Year	1,334.13	Sum of High
		1,322.24	Sum of Low
April			
2012	Year	1,211.06	Sum of High
		1,194.54	Sum of Low
May			
2012	Year	1,358.98	Sum of High
		1,341.68	Sum of Low
June			
2012	Year	1,424.17	Sum of High
		1,407.09	Sum of Low
July			

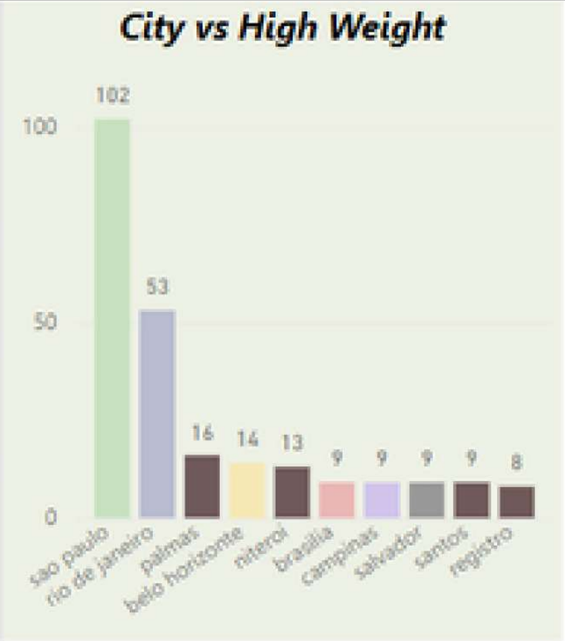
Month wise opening and closing price of stocks over 5 years

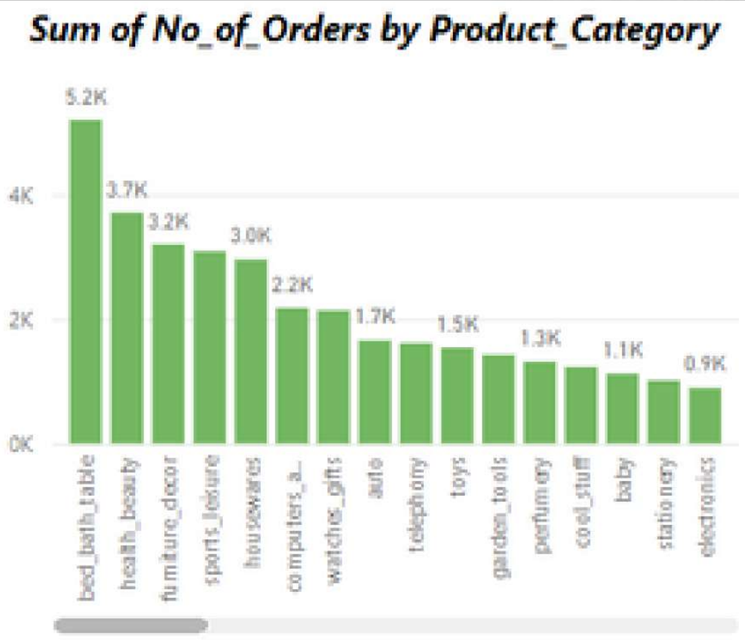


January			
2012	Year	61.80	Max of Open
		61.47	Max of Close
February			
2012	Year	62.40	Max of Open
		62.48	Max of Close
March			
2012	Year	61.41	Max of Open
		61.23	Max of Close
April			
2012	Year	61.96	Max of Open
		62.45	Max of Close
May			
2012	Year	65.41	Max of Open
		65.82	Max of Close
June			
2012	Year	68.94	Max of Open
		69.72	Max of Close

Ecommerce Data Analysis Report

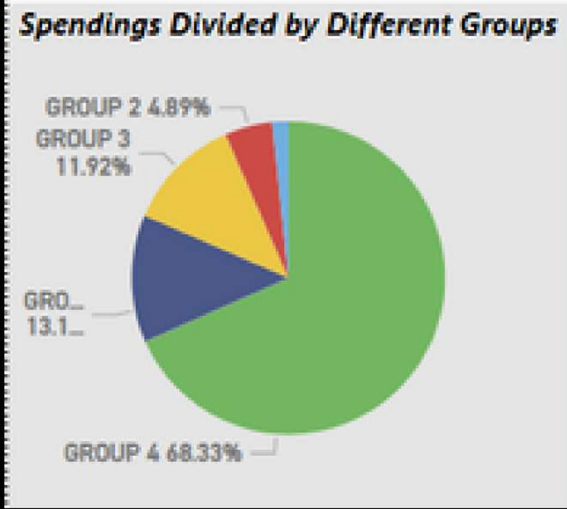


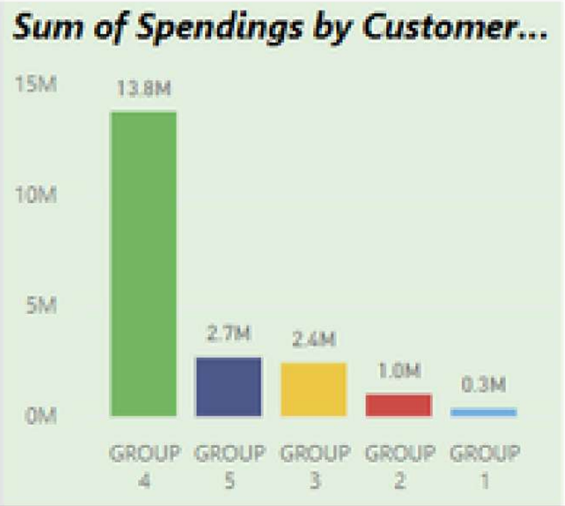


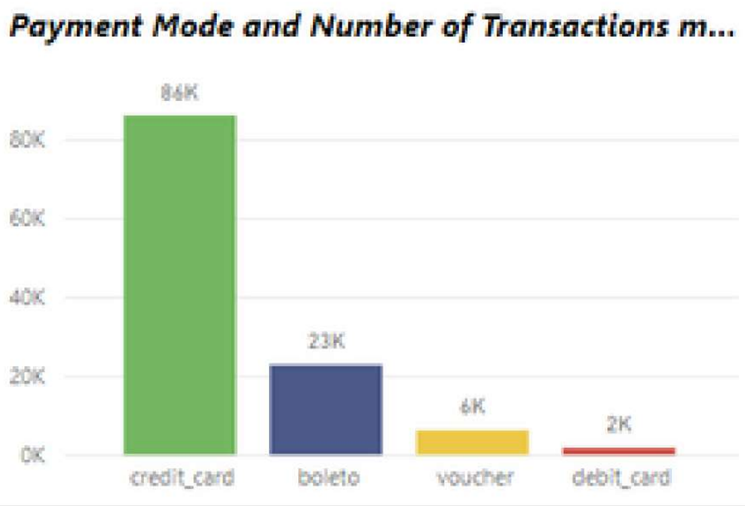


42K

Sum of No_of_Orders



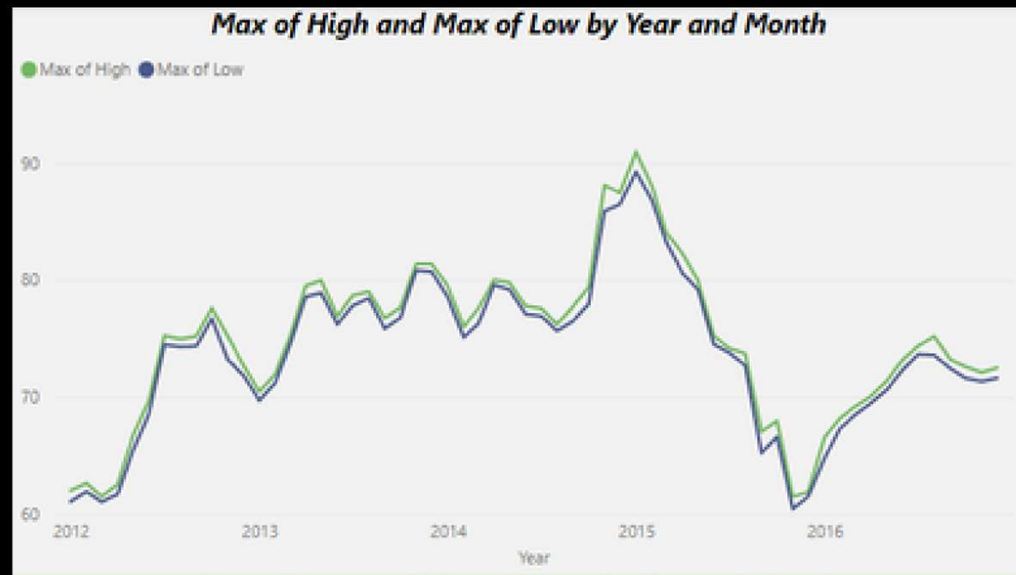




117K

Sum of No_of_Count

Walmart Stock Data Analysis Report



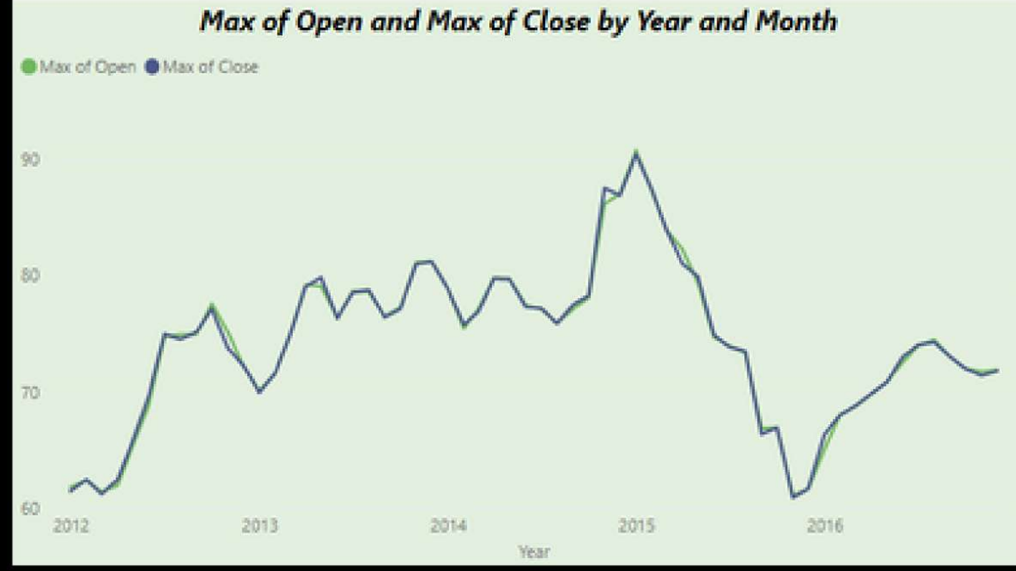
January
2012 Year | 1,211.44 Sum of High | 1,196.06 Sum of Low

February
2012 Year | 1,224.48 Sum of High | 1,211.65 Sum of Low

March
2012 Year | 1,334.13 Sum of High | 1,322.24 Sum of Low

April
2012 Year | 1,211.06 Sum of High | 1,194.54 Sum of Low

May



January
2012 Year | 61.80 Max of Open | 61.47 Max of Close

February
2012 Year | 62.40 Max of Open | 62.48 Max of Close

March
2012 Year | 61.41 Max of Open | 61.23 Max of Close

April
2012 Year | 61.96 Max of Open | 62.45 Max of Close

May

Thank You