WALMART STOCK ANALYSIS

Loading data from Local system to Mysql

1. Creating table named Walmart

```
mysql> create table walmart(Date varchar(20),open decimal(20,5),high decimal(20,5),low decimal(20,5),close decimal(20,5),volum
imal(20,10));
Query OK, 0 rows affected (0.02 sec)
```

2. Loading data from local to the mysql table

```
mysql> load data infile '/home/cloudera/Downloads/walmart_case_study_data/walmart_stock.csv' into table walmart fields termina
 lines;
Query OK, 1258 rows affected, 4534 warnings (0.04 sec)
Records: 1258  Deleted: 0  Skipped: 0  Warnings: 4534
```

3. Checking the table data

```
mysql> select * from walmart limit 4;
+------------+----------+----------+----------+----------+----------+---------------+
| Date       | open     | high     | low      | close    | volumne  | adj_close     |
+------------+----------+----------+----------+----------+----------+---------------+
| 2012-01-03 | 59.97000 | 61.06000 | 59.87000 | 60.33000 | 12668800 | 52.6192350000 |
| 2012-01-04 | 60.21000 | 60.35000 | 59.47000 | 59.71000 |  9593300 | 52.0784750000 |
| 2012-01-05 | 59.35000 | 59.62000 | 58.37000 | 59.42000 | 12768200 | 51.8255390000 |
| 2012-01-06 | 59.42000 | 59.45000 | 58.87000 | 59.00000 |  8069400 | 51.4592200000 |
+------------+----------+----------+----------+----------+----------+---------------+
4 rows in set (0.00 sec)
```


```
mysql> select * from walmart limit 4;
+------------+----------+----------+----------+----------+----------+---------------+
| Date       | open     | high     | low      | close    | volumne  | adj_close     |
+------------+----------+----------+----------+----------+----------+---------------+
| 2012-01-03 | 59.97000 | 61.06000 | 59.87000 | 60.33000 | 12668800 | 52.6192350000 |
| 2012-01-04 | 60.21000 | 60.35000 | 59.47000 | 59.71000 |  9593300 | 52.0784750000 |
| 2012-01-05 | 59.35000 | 59.62000 | 58.37000 | 59.42000 | 12768200 | 51.8255390000 |
| 2012-01-06 | 59.42000 | 59.45000 | 58.87000 | 59.00000 |  8069400 | 51.4592200000 |
+------------+----------+----------+----------+----------+----------+---------------+
4 rows in set (0.00 sec)
```


Loading data into the HABSE

1. Create table in hbase

```
hbase(main):002:0> create 'walmart_stock','info'
0 row(s) in 1.5600 seconds

=> Hbase::Table - walmart_stock
```

Checking if the table got created.

```
hbase(main):004:0> list
TABLE
bykes
cars
laptops
walmart_stock
4 row(s) in 0.0220 seconds

=> ["bykes", "cars", "laptops", "walmart_stock"]
```

2. Load data from mysql to hbase using sqoop pipeline -

```
[cloudera@quickstart Desktop]$ sqoop import --connect jdbc:mysql://localhost:3306/w
almartdb --username root --password cloudera --table walmart --hbase-table walmart_
stock --hbase-row-key Date --column-family info --hbase-create-table -m 1;
```

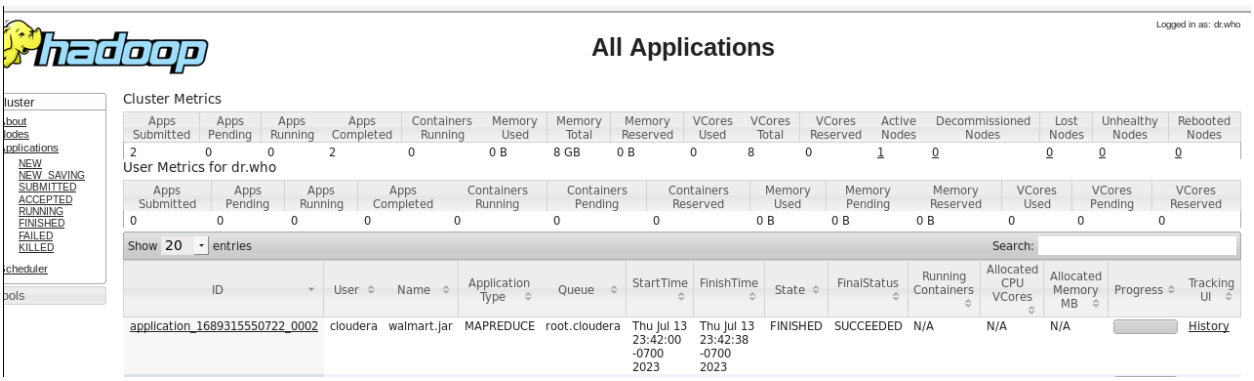
Checking if the data has transferred successfully:

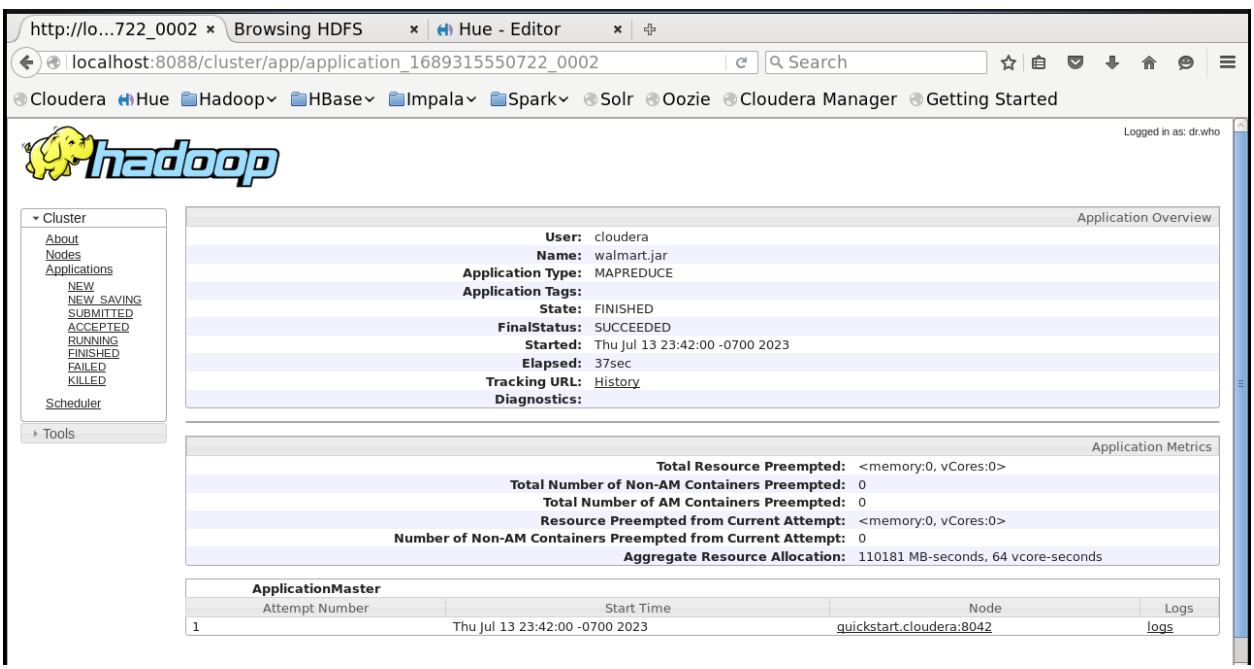
```
hbase(main):018:0> scan 'walmart_stock', {LIMIT=>5}
ROW                                    COLUMN+CELL
 2012-01-03                            column=info:adj_close, timestamp=1689316956016, value=52.6192350000
 2012-01-03                            column=info:close, timestamp=1689316956016, value=60.33000
 2012-01-03                            column=info:high, timestamp=1689316956016, value=61.06000
 2012-01-03                            column=info:low, timestamp=1689316956016, value=59.87000
 2012-01-03                            column=info:open, timestamp=1689316956016, value=59.97000
 2012-01-03                            column=info:volumne, timestamp=1689316956016, value=12668800
 2012-01-04                            column=info:adj_close, timestamp=1689316956016, value=52.0784750000
 2012-01-04                            column=info:close, timestamp=1689316956016, value=59.71000
 2012-01-04                            column=info:high, timestamp=1689316956016, value=60.35000
 2012-01-04                            column=info:low, timestamp=1689316956016, value=59.47000
 2012-01-04                            column=info:open, timestamp=1689316956016, value=60.21000
 2012-01-04                            column=info:volumne, timestamp=1689316956016, value=9593300
 2012-01-05                            column=info:adj_close, timestamp=1689316956016, value=51.8255390000
 2012-01-05                            column=info:close, timestamp=1689316956016, value=59.42000
 2012-01-05                            column=info:high, timestamp=1689316956016, value=59.62000
 2012-01-05                            column=info:low, timestamp=1689316956016, value=58.37000
 2012-01-05                            column=info:open, timestamp=1689316956016, value=59.35000
 2012-01-05                            column=info:volumne, timestamp=1689316956016, value=12768200
 2012-01-06                            column=info:adj_close, timestamp=1689316956016, value=51.4592200000
 2012-01-06                            column=info:close, timestamp=1689316956016, value=59.00000
 2012-01-06                            column=info:high, timestamp=1689316956016, value=59.45000
 2012-01-06                            column=info:low, timestamp=1689316956016, value=58.87000
```

Getting specific data:

```
hbase(main):022:0> get 'walmart_stock','2012-03-01','info:close','info:open'
COLUMN                                 CELL
 info:close                            timestamp=1689316956016, value=58.82000
 info:open                             timestamp=1689316956016, value=59.36000
2 row(s) in 0.0070 seconds
```

JOB status:





Scenarios:

scenario 1: print out first 5 columns?

WITH HBASE:

```
hbase(main):013:0> scan 'walmart_stock',{COLUMN=>['info:open','info:high','info:low','info:close','info:volumne']}
 2016-12-22                                     column=info:volumne, timestamp=1689316956016, value=12106800
 2016-12-23                                     column=info:close, timestamp=1689316956016, value=69.54000
 2016-12-23                                     column=info:high, timestamp=1689316956016, value=69.75000
 2016-12-23                                     column=info:low, timestamp=1689316956016, value=69.36000
 2016-12-23                                     column=info:open, timestamp=1689316956016, value=69.43000
 2016-12-23                                     column=info:volumne, timestamp=1689316956016, value=4803900
 2016-12-27                                     column=info:close, timestamp=1689316956016, value=69.70000
 2016-12-27                                     column=info:high, timestamp=1689316956016, value=69.82000
 2016-12-27                                     column=info:low, timestamp=1689316956016, value=69.25000
 2016-12-27                                     column=info:open, timestamp=1689316956016, value=69.30000
 2016-12-27                                     column=info:volumne, timestamp=1689316956016, value=4435700
 2016-12-28                                     column=info:close, timestamp=1689316956016, value=69.31000
 2016-12-28                                     column=info:high, timestamp=1689316956016, value=70.00000
 2016-12-28                                     column=info:low, timestamp=1689316956016, value=69.26000
 2016-12-28                                     column=info:open, timestamp=1689316956016, value=69.94000
 2016-12-28                                     column=info:volumne, timestamp=1689316956016, value=4875700
 2016-12-29                                     column=info:close, timestamp=1689316956016, value=69.26000
 2016-12-29                                     column=info:high, timestamp=1689316956016, value=69.52000
 2016-12-29                                     column=info:low, timestamp=1689316956016, value=69.12000
 2016-12-29                                     column=info:open, timestamp=1689316956016, value=69.21000
 2016-12-29                                     column=info:volumne, timestamp=1689316956016, value=4298400
 2016-12-30                                     column=info:close, timestamp=1689316956016, value=69.12000
 2016-12-30                                     column=info:high, timestamp=1689316956016, value=69.43000
 2016-12-30                                     column=info:low, timestamp=1689316956016, value=68.83000
 2016-12-30                                     column=info:open, timestamp=1689316956016, value=69.12000
 2016-12-30                                     column=info:volumne, timestamp=1689316956016, value=6889500
1258 row(s) in 2.1910 seconds
```

Creating a Spark Session

```python
# doing it with dataframe
print('Before:')
walmart.describe().show()
print('After:')

walmart.describe().select(
    "summary",
    round('Open',2).alias("Open"),
    round('High',2).alias("High"),
    round('Low',2).alias("Low"),
    round('Close',2).alias("Close"),
    round('Volume',2).alias("Volume"),
    round('Adj Close',2).alias("Adj Close")).show()
```

WALMART STOCK ANALYSIS

Creating a SparkSession

In [11]:

```python
import findspark
findspark.init('C:\spark-3.3.2-bin-hadoop3\spark-3.3.2-bin-hadoop3')
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark.sql.types import *
sparkS=SparkSession.builder.master('local[*]').appName('spark_mysql_connection').getOrCreate()
sparkC = sparkSess.sparkContext
sparkS
```

SparkSession - in-memory

SparkContext

Spark UI

Version
`v3.3.2`
Master
`local[*]`
AppName
`spark_mysql_connection`

Loading Walmart_stock dataset in a dataframe of Spark using SparkSession.

Loading dataset into a dataframe using SparkSession

In [19]:

```
walmart = sparkS.read.csv(r"C:\Users\Admin\Downloads\walmart_case_study_data\walmart_stock.csv",\
                          inferSchema=True,header=True)
walmart.show(5)
print(f"columns in walmart are: {walmart.columns}")
```

```
+-------------------+------------------+---------+---------+------------------+--------+------------------+
|               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

columns in walmart are: ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
```

CREATING TEMP VIEW OF the DF to Run SPARKSQL

CREATING A TEMP VIEW OF THE DATAFRAME TO RUN SPARKSQL

```
walmart.createOrReplaceTempView("walmart_table")
s = sparkS.sql("SELECT * FROM walmart_table limit 5")
# cross checking the data
s.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|
+-------------------+------------------+---------+---------+------------------+--------+------------------+
```

Scenario 1: Print out first 5 columns.

scenario 1: Print out first 5 columns ?

```python
# doing it with DATAFRAME
for row in walmart.collect():
    print(row[:5])
```

```
(datetime.datetime(2012, 1, 3, 0, 0), 59.970001, 61.060001, 59.869999, 60.330002)
(datetime.datetime(2012, 1, 4, 0, 0), 60.209998999999996, 60.349998, 59.470001, 59.709998999999996)
(datetime.datetime(2012, 1, 5, 0, 0), 59.349998, 59.619999, 58.369999, 59.419998)
(datetime.datetime(2012, 1, 6, 0, 0), 59.419998, 59.450001, 58.869999, 59.0)
(datetime.datetime(2012, 1, 9, 0, 0), 59.029999, 59.549999, 58.919998, 59.18)
(datetime.datetime(2012, 1, 10, 0, 0), 59.43, 59.709998999999996, 58.98, 59.040001000000004)
(datetime.datetime(2012, 1, 11, 0, 0), 59.060001, 59.529999, 59.040001000000004, 59.400002)
(datetime.datetime(2012, 1, 12, 0, 0), 59.790001000000004, 60.0, 59.400002, 59.5)
(datetime.datetime(2012, 1, 13, 0, 0), 59.18, 59.610001000000004, 59.009997999999996, 59.540001000000004)
(datetime.datetime(2012, 1, 17, 0, 0), 59.869999, 60.110001000000004, 59.52, 59.849998)
(datetime.datetime(2012, 1, 18, 0, 0), 59.790001000000004, 60.029999, 59.650002, 60.009997999999996)
(datetime.datetime(2012, 1, 19, 0, 0), 59.93, 60.73, 59.75, 60.610001000000004)
(datetime.datetime(2012, 1, 20, 0, 0), 60.75, 61.25, 60.669998, 61.009997999999996)
(datetime.datetime(2012, 1, 23, 0, 0), 60.810001, 60.98, 60.509997999999996, 60.91)
(datetime.datetime(2012, 1, 24, 0, 0), 60.75, 62.0, 60.75, 61.389998999999996)
(datetime.datetime(2012, 1, 25, 0, 0), 61.18, 61.610001000000004, 61.040001000000004, 61.470001)
(datetime.datetime(2012, 1, 26, 0, 0), 61.799999, 61.84, 60.77, 60.970001)
(datetime.datetime(2012, 1, 27, 0, 0), 60.860001000000004, 61.119999, 60.540001000000004, 60.709998999999996)
(datetime.datetime(2012, 1, 30, 0, 0), 60.470001, 61.32, 60.349998, 61.299999)
(datetime.datetime(2012, 1, 31, 0, 0), 61.529999, 61.57, 60.580002, 61.360001000000004)
```

Scenario 2: There are too many decimal places for mean and stddev in the describe() dataframe. Format the numbers to just show up to two decimal places.

```
Before:
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|summary|              Open|             High|              Low|            Close|           Volume|        Adj Close|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|  count|              1258|             1258|             1258|             1258|             1258|             1258|
|   mean| 72.35785375357709|72.83938807631165| 71.9186009594594|72.38844998012726|8222093.481717011|67.23883848728146|
| stddev|  6.76809024470826|6.768186808159218|6.744075756255496|6.756859163732991|  4519780.8431556|6.722609449996857|
|    min|56.389998999999996|        57.060001|        56.299999|        56.419998|          2094900|        50.363689|
|    max|         90.800003|        90.970001|            89.25|        90.470001|         80898100|84.91421600000001|
+-------+------------------+-----------------+-----------------+-----------------+-----------------+-----------------+

After:
+-------+------+------+------+------+----------+---------+
|summary|  Open|  High|   Low| Close|    Volume|Adj Close|
+-------+------+------+------+------+----------+---------+
|  count|1258.0|1258.0|1258.0|1258.0|    1258.0|   1258.0|
|   mean| 72.36| 72.84| 71.92| 72.39|8222093.48|    67.24|
| stddev|  6.77|  6.77|  6.74|  6.76|4519780.84|     6.72|
|    min| 56.39| 57.06|  56.3| 56.42| 2094900.0|    50.36|
|    max|  90.8| 90.97| 89.25| 90.47| 8.08981E7|    84.91|
+-------+------+------+------+------+----------+---------+
```

Scenario 3: Create a new data-frame with a column called HV Ratio that is the ratio of the High Price versus volume of stock traded for a day?

```python
# doing it with DATAFRAME
walmart = walmart.withColumn('HV Ratio',walmart['high']/walmart['volume'])
walmart.show()
```

```
+-------------------+-----+-----+-----+-----+--------+---------+-------------------+
|               Date| Open| High|  Low|Close|  Volume|Adj Close|           HV Ratio|
+-------------------+-----+-----+-----+-----+--------+---------+-------------------+
|2012-01-03 00:00:00|59.97|61.06|59.87|60.33|12668800|    52.62|4.819714574387472E-6|
|2012-01-04 00:00:00|60.21|60.35|59.47|59.71| 9593300|    52.08|6.290848821573389...|
|2012-01-05 00:00:00|59.35|59.62|58.37|59.42|12768200|    51.83|4.669413073103491E-6|
|2012-01-06 00:00:00|59.42|59.45|58.87| 59.0| 8069400|    51.46|7.367338339901356E-6|
|2012-01-09 00:00:00|59.03|59.55|58.92|59.18| 6679300|    51.62|8.915604928660188E-6|
|2012-01-10 00:00:00|59.43|59.71|58.98|59.04| 6907300|    51.49|8.644477581688938E-6|
|2012-01-11 00:00:00|59.06|59.53|59.04| 59.4| 6365600|    51.81| 9.35182857861003E-6|
|2012-01-12 00:00:00|59.79| 60.0| 59.4| 59.5| 7236400|     51.9| 8.29141562102703E-6|
|2012-01-13 00:00:00|59.18|59.61|59.01|59.54| 7729300|    51.93|7.712211972623653E-6|
|2012-01-17 00:00:00|59.87|60.11|59.52|59.85| 8500000|     52.2|7.071764705882352...|
|2012-01-18 00:00:00|59.79|60.03|59.65|60.01| 5911400|    52.34|1.015495483303447...|
|2012-01-19 00:00:00|59.93|60.73|59.75|60.61| 9234600|    52.86|6.576354146362592...|
|2012-01-20 00:00:00|60.75|61.25|60.67|61.01|10378800|    53.21| 5.90145296180676E-6|
|2012-01-23 00:00:00|60.81|60.98|60.51|60.91| 7134100|    53.13|8.547679455011844E-6|
|2012-01-24 00:00:00|60.75| 62.0|60.75|61.39| 7362800|    53.54|8.420709512685392E-6|
|2012-01-25 00:00:00|61.18|61.61|61.04|61.47| 5915800|    53.61|1.041448324825044...|
|2012-01-26 00:00:00| 61.8|61.84|60.77|60.97| 7436200|    53.18|8.316075414862431E-6|
|2012-01-27 00:00:00|60.86|61.12|60.54|60.71| 6287300|    52.95|9.721183974042911E-6|
|2012-01-30 00:00:00|60.47|61.32|60.35| 61.3| 7636900|    53.47|8.029436027707578E-6|
|2012-01-31 00:00:00|61.53|61.57|60.58|61.36| 9761500|    53.52|6.307432259386365E-6|
+-------------------+-----+-----+-----+-----+--------+---------+-------------------+
only showing top 20 rows
```

```python
# Doing it with SparkSQL
sparkS.sql("Select Date,(high/volume) as HV_Ratio from walmart_table order by date").show()
```

```
+-------------------+--------------------+
|               Date|            HV_Ratio|
+-------------------+--------------------+
|2012-01-03 00:00:00|4.819714653321546E-6|
|2012-01-04 00:00:00|6.290848613094555E-6|
|2012-01-05 00:00:00|4.669412994783916E-6|
|2012-01-06 00:00:00|7.367338463826307E-6|
|2012-01-09 00:00:00|8.915604778943901E-6|
|2012-01-10 00:00:00|8.644477436914568E-6|
|2012-01-11 00:00:00|9.351828421515645E-6|
|2012-01-12 00:00:00| 8.29141562102703E-6|
|2012-01-13 00:00:00|7.712212102001476E-6|
|2012-01-17 00:00:00|7.071764823529412E-6|
|2012-01-18 00:00:00|1.015495466386981E-5|
|2012-01-19 00:00:00|6.576354146362592...|
|2012-01-20 00:00:00| 5.90145296180676E-6|
|2012-01-23 00:00:00|8.547679455011844E-6|
|2012-01-24 00:00:00|8.420709512685392E-6|
|2012-01-25 00:00:00|1.041448341728929...|
|2012-01-26 00:00:00|8.316075414862431E-6|
|2012-01-27 00:00:00|9.721183814992126E-6|
|2012-01-30 00:00:00|8.029436027707578E-6|
|2012-01-31 00:00:00|6.307432259386365E-6|
+-------------------+--------------------+
only showing top 20 rows
```

Scenario 4: What day had the Peak High in Price?

```python
# Doing it with DATAFRAME
condition = walmart.select(max(walmart['High'])).first()[0]
walmart.filter(walmart['High']==condition).show()


```

```
+-------------------+---------+---------+-----+---------+-------+---------+
|               Date|     Open|     High|  Low|    Close| Volume|Adj Close|
+-------------------+---------+---------+-----+---------+-------+---------+
|2015-01-13 00:00:00|90.800003|90.970001|88.93|89.309998|8215400|83.825448|
+-------------------+---------+---------+-----+---------+-------+---------+
```

```python
# Doing it with SparkSQL
result = sparkS.sql("select date,high from walmart_table where high = \
(select max(high) from walmart_table)")
result.show()

print(f"Day that had the peak high in price:{result.collect()[0][0]}")
```

```
+-------------------+---------+
|               date|     high|
+-------------------+---------+
|2015-01-13 00:00:00|90.970001|
+-------------------+---------+

Day that had the peak high in price:2015-01-13 00:00:00
```

Scenario 5: What is the mean of the Close column?

```
# doing it with DATAFRAME
result = walmart.agg({'Close':'mean'}).select(col('avg(Close)').alias('mean_close'))
result.show()
```

```
+-----------------+
|       mean_close|
+-----------------+
|72.38844992050863|
+-----------------+
```

```
# Doing it with SparkSQL
result = sparkS.sql("select avg(close) from walmart_table")
result.show()
```

```
+-----------------+
|       avg(close)|
+-----------------+
|72.38844998012726|
+-----------------+
```

Scenario 6: What is the max and min of the Volume column?

```python
# Doing it with DATAFRAME
result = walmart.agg(
    max('Volume').alias('max_volume'),
    min('Volume').alias('min_volume'))
result.show()
```

```
+----------+----------+
|max_volume|min_volume|
+----------+----------+
|  80898100|   2094900|
+----------+----------+
```

```python
# Doing it with SparkSQL
result =  sparkS.sql("select max(Volume) as MAX_VOLUME, min(Volume) \
                      as MIN_VOLUME from walmart_table")
result.show()
```

```
+----------+----------+
|MAX_VOLUME|MIN_VOLUME|
+----------+----------+
|  80898100|   2094900|
+----------+----------+
```

Scenario 7: How many days was the Close lower than 60 dollars?

```
# doing with DATAFRAME
walmart.filter(walmart['Close']<60).count()
```

```
81
```

```
# Doing it with SparkSQL
result= sparkS.sql("Select count(Date) from walmart_table where close<60")
result.show()
```

```
+-----------+
|count(Date)|
+-----------+
|         81|
+-----------+
```

Scenario 8: What percentage of the time was the High greater than 80 dollars?

```python
# Doing it with DATAFRAME
favoured = walmart.filter(walmart['High']>80).count()
total = walmart.agg(count('High')).collect()[0][0]
print(f"percentage of the time was the High greater than 80 dollars is {favoured*100/total}")
```

```
percentage of the time was the High greater than 80 dollars is 9.141494435612083
```

```python
# Doing it with SparkSQL
result = sparkS.sql("""
    SELECT
        (COUNT(high) *100/ (SELECT COUNT(high) FROM walmart_table)) AS percentage
    FROM
        walmart_table
    WHERE
        high > 80
""")
result.show()
```

```
+-----------------+
|       percentage|
+-----------------+
|9.141494435612083|
+-----------------+
```

Scenario 9: What is the max High per year

```
# Doing it with DATAFRAME
walmart.groupby(year('Date').alias('YEAR'))\
.agg(max('High').alias("Max HIGH"))\
.orderBy(year('Date')).show()
```

```
+----+---------+
|YEAR| Max HIGH|
+----+---------+
|2012|77.599998|
|2013|81.370003|
|2014|88.089996|
|2015|90.970001|
|2016|75.190002|
+----+---------+
```

?

```
# doing it with SparkSQL
result = sparkS.sql(""" select year(date) as YEAR, max(high) as MAX_HIGH
    from walmart_table
    group by 1
    order by 1 """)
result.show()
```

```
+----+---------+
|YEAR| MAX_HIGH|
+----+---------+
|2012|77.599998|
|2013|81.370003|
|2014|88.089996|
|2015|90.970001|
|2016|75.190002|
+----+---------+
```