Project

Monday, July 10, 2023 8:23 PM

Stock Analysis Data Engineering Solution

Kousik Dutta (FFL19)

Introduction

The world of stock analysis is an intricate and fascinating realm that involves dissecting and evaluating various aspects of financial markets to make informed investment decisions. A stock analysis pipeline provides a structured framework for conducting thorough and systematic analyses of stocks, enabling investors to gain insights into the potential performance and value of individual companies.

The stock analysis pipeline encompasses a series of interconnected stages, each contributing to a comprehensive understanding of a stock's prospects. It involves gathering relevant data, applying analytical techniques, and interpreting the results to derive meaningful conclusions. By following a well-defined pipeline, investors can streamline their analysis process and improve the accuracy and effectiveness of their decision-making.

Let's first create database

```
mysql> create database stock_analysis;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bigbazar           |
| cm                 |
| firehose           |
| hue                |
| metastore          |
| mysql              |
| nav                |
| navms              |
| oozie              |
| retail_db          |
| rman               |
| sentry             |
| stock_analysis     |
| titanic            |
| travel_database    |
+--------------------+
```

Step - 1

First let's insert the table in the SQL.

- AAL (American Airlines Group)
- AAOI (Applied Optoelectronics Inc)
- ABIO (ABIO ENGINEERING LLP)
- ABMD (ABIOMED Inc.)

Table AAL (American Airlines Group):

Creating Table and Inserting the data in SQL

```
mysql> create table AAL_stock(date VARCHAR(255), low float, open float, volume b
igint, high float, close float, adjusted_close decimal(10,8));
Query OK, 0 rows affected (0.01 sec)

mysql> load data infile '/home/cloudera/Desktop/AAL.csv' into table AAL_stock fi
elds terminated by ',' lines terminated by '\n' ignore 1 lines;
Query OK, 4333 rows affected, 4283 warnings (0.07 sec)
Records: 4333  Deleted: 0  Skipped: 0  Warnings: 4283
```

Let's see the data:

```
mysql> select * from AAL_stock limit 5;
+------------+-------+-------+---------+-------+-------+----------------+
| date       | low   | open  | volume  | high  | close | adjusted_close |
+------------+-------+-------+---------+-------+-------+----------------+
| 27-09-2005 |  19.1 | 21.05 |  961200 |  21.4 |  19.3 |    18.19491005 |
| 28-09-2005 |  19.2 |  19.3 | 5747900 | 20.53 |  20.5 |    19.32620430 |
| 29-09-2005 |  20.1 |  20.4 | 1078200 | 20.58 | 20.21 |    19.05280495 |
| 30-09-2005 | 20.18 | 20.26 | 3123300 | 21.05 | 21.01 |    19.80700111 |
| 03-10-2005 |  20.9 |  20.9 | 1057900 | 21.75 |  21.5 |    20.26893997 |
+------------+-------+-------+---------+-------+-------+----------------+
5 rows in set (0.00 sec)
```

Table AAOI (Applied Optoelectronics Inc):

Creating Table and Inserting the data in SQL

```
mysql> create table AAOI_stock(date VARCHAR(255), low float, open float, volume
bigint, high float, close float, adjusted_close decimal(10,8));
mysql> load data infile '/home/cloudera/Desktop/AAOI.csv' into table AAOI_stock
fields terminated by ',' lines terminated by '\n' ignore 1 lines;
```

Let's see the data:

```
mysql> select * from AAOI_stock limit 5;
+------------+------+-------+--------+-------+-------+----------------+
| date       | low  | open  | volume | high  | close | adjusted_close |
+------------+------+-------+--------+-------+-------+----------------+
| 26-09-2013 | 9.37 |    10 | 946000 | 10.09 |  9.96 |     9.96000004 |
| 27-09-2013 |   10 | 10.44 | 253300 | 10.44 |  10.1 |    10.10000038 |
| 30-09-2013 | 9.71 |    10 |  84900 | 10.18 |    10 |    10.00000000 |
| 01-10-2013 | 9.92 |  9.95 |  74500 | 10.02 |    10 |    10.00000000 |
| 02-10-2013 | 9.89 |  9.99 |  94000 |    10 |  9.97 |     9.97000027 |
+------------+------+-------+--------+-------+-------+----------------+
5 rows in set (0.00 sec)
```

Table ABIO (ABIO ENGINEERING LLP):

Creating Table and Inserting the data in SQL

```
mysql> create table ABIO_stock(date VARCHAR(255), low float, open float, volume
bigint, high float, close float, adjusted_close DOUBLE);
Query OK, 0 rows affected (0.05 sec)

mysql> load data infile '/home/cloudera/Desktop/ABIO.csv' into table ABIO_stock
fields terminated by ',' lines terminated by '\n' ignore 1 lines;
Query OK, 6379 rows affected (0.08 sec)
Records: 6379  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from ABIO_stock limit 5;
+------------+--------+--------+--------+--------+--------+----------------+
| date       | low    | open   | volume | high   | close  | adjusted_close |
+------------+--------+--------+--------+--------+--------+----------------+
| 08-08-1997 | 657720 | 669060 |     57 | 708750 | 674730 |         674730 |
| 11-08-1997 | 680400 | 686070 |      4 | 708750 | 705915 |         705915 |
| 12-08-1997 | 652050 | 708750 |      2 | 708750 | 657720 |         657720 |
| 13-08-1997 | 635040 | 652050 |      4 | 669060 | 635040 |         635040 |
| 14-08-1997 | 635040 | 635040 |      3 | 663390 | 654885 |         654885 |
+------------+--------+--------+--------+--------+--------+----------------+
5 rows in set (0.00 sec)
```

Table ABMD (ABIOMED Inc.):

Creating Table and Inserting the data in SQL

```
mysql> create table ABMD_stock(date VARCHAR(255), low float, open float, volume
bigint, high float, close float, adjusted_close DOUBLE);
Query OK, 0 rows affected (0.02 sec)

mysql> load data infile '/home/cloudera/Desktop/ABMD.csv' into table ABMD_stock
fields terminated by ',' lines terminated by '\n' ignore 1 lines;
Query OK, 8916 rows affected (0.13 sec)
Records: 8916  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from ABMD_stock limit 5;
+------------+--------+--------+--------+--------+--------+----------------+
| date       | low    | open   | volume | high   | close  | adjusted_close |
+------------+--------+--------+--------+--------+--------+----------------+
| 29-07-1987 | 5.4375 |      0 | 201200 |    5.5 |    5.5 |            5.5 |
| 30-07-1987 | 5.4375 |    5.5 | 107000 | 5.5625 | 5.5625 |         5.5625 |
| 31-07-1987 |    5.5 | 5.5625 |  35400 | 5.5625 | 5.5625 |         5.5625 |
| 03-08-1987 |    5.5 | 5.5625 |  23800 | 5.5625 | 5.5625 |         5.5625 |
| 04-08-1987 |    5.5 | 5.5625 |  35800 | 5.5625 | 5.5625 |         5.5625 |
+------------+--------+--------+--------+--------+--------+----------------+
5 rows in set (0.00 sec)
```

List of all tables in Stock Analysis Database:

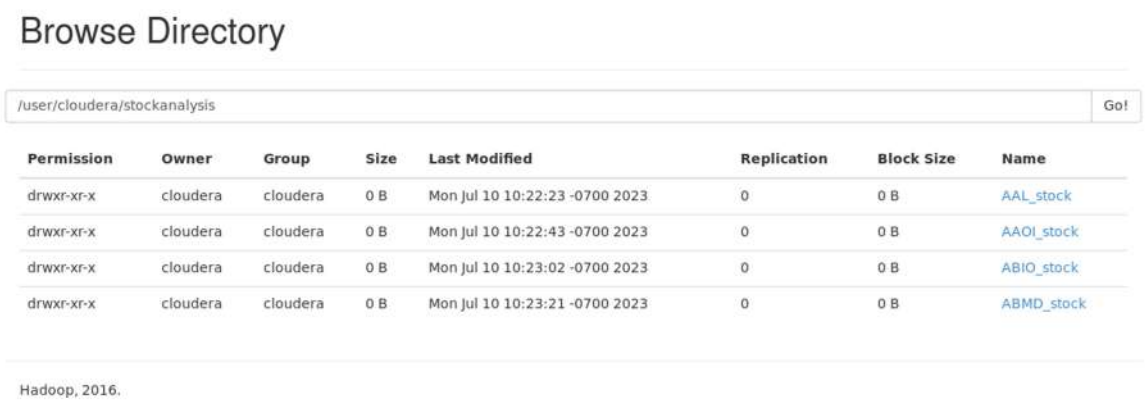
```
mysql> show tables;
+--------------------------+
| Tables_in_stock_analysis |
+--------------------------+
| AAL_stock                |
| AAOI_stock               |
| ABIO_stock               |
| ABMD_stock               |
+--------------------------+
4 rows in set (0.00 sec)
```

Step - 2

In the next step we will be exporting the data to HDFS.

```
[cloudera@quickstart Desktop]$ sqoop import-all-tables --connect jdbc:mysql://lo
calhost:3306/stock_analysis --username root --password cloudera --warehouse-dir
/user/cloudera/stockanalysis -m 1
```

HDFS Directory:



Step - 3

HIVE upload
In the next step we will be uploading the tables to HIVE (**Internal Table**)

1. **AAL table:**

```
hive> create table AAL_stock(date string, low float, open float, volume int, hig
h float, close float, adjusted_close float) row format delimited fields terminat
ed by ',';
hive> load data inpath 'stockanalysis/AAL_stock' into table AAL_stock;

hive> select * from AAL_stock limit 5;
OK
27-09-2005      19.1    21.05   961200  21.4    19.3    18.19491
28-09-2005      19.2    19.3    5747900 20.53   20.5    19.326204
29-09-2005      20.1    20.4    1078200 20.58   20.21   19.052805
30-09-2005      20.18   20.26   3123300 21.05   21.01   19.807001
03-10-2005      20.9    20.9    1057900 21.75   21.5    20.26894
Time taken: 0.109 seconds, Fetched: 5 row(s)
```

2. **AAOI Table:**

```
hive> create table AAOI_stock(date string, low float, open float, volume int, hi
gh float, close float, adjusted_close float) row format delimited fields termina
ted by ',';
OK
Time taken: 0.171 seconds
hive> load data inpath 'stockanalysis/AAOI_stock' into table AAOI_stock;
Loading data to table default.aaoi_stock
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehous
e/aaoi_stock/part-m-00000': User does not belong to supergroup
Table default.aaoi_stock stats: [numFiles=1, totalSize=121274]
OK
Time taken: 0.363 seconds
hive> select * from AAOI_stock limit 5;
OK
26-09-2013      9.37    10.0    946000  10.09   9.96    9.96
27-09-2013      10.0    10.44   253300  10.44   10.1    10.1
30-09-2013      9.71    10.0    84900   10.18   10.0    10.0
01-10-2013      9.92    9.95    74500   10.02   10.0    10.0
02-10-2013      9.89    9.99    94000   10.0    9.97    9.97
Time taken: 0.107 seconds, Fetched: 5 row(s)
hive>
```

3. **ABIO Table:**

```
hive> create table ABIO_stock(date string, low float, open float, volume int, hi
gh float, close float, adjusted_close float) row format delimited fields termina
ted by ',';
OK
Time taken: 0.11 seconds
hive> load data inpath 'stockanalysis/ABIO_stock' into table ABIO_stock;
Loading data to table default.abio_stock
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehous
e/abio_stock/part-m-00000': User does not belong to supergroup
Table default.abio_stock stats: [numFiles=1, totalSize=369893]
OK
Time taken: 0.291 seconds
hive> select * from ABIO_stock limit 5;
OK
08-08-1997      657720.0        669060.0        57      708750.0        674730.0
674730.0
11-08-1997      680400.0        686070.0        4       708750.0        705915.0
705915.0
12-08-1997      652050.0        708750.0        2       708750.0        657720.0
657720.0
13-08-1997      635040.0        652050.0        4       669060.0        635040.0
635040.0
14-08-1997      635040.0        635040.0        3       663390.0        654885.0
654885.0
Time taken: 0.097 seconds, Fetched: 5 row(s)
```

4. **ABMD Table:**

```
hive> create table ABMD_stock(date string, low float, open float, volume int, hi
gh float, close float, adjusted_close float) row format delimited fields termina
ted by ',';
OK
Time taken: 0.144 seconds
hive> load data inpath 'stockanalysis/ABMD_stock' into table ABMD_stock;
Loading data to table default.abmd_stock
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehous
e/abmd_stock/part-m-00000': User does not belong to supergroup
Table default.abmd_stock stats: [numFiles=1, totalSize=474705]
OK
Time taken: 0.583 seconds
hive> select * from ABMD_stock limit 5;
OK
29-07-1987      5.4375  0.0     201200  5.5     5.5     5.5
30-07-1987      5.4375  5.5     107000  5.5625  5.5625  5.5625
31-07-1987      5.5     5.5625  35400   5.5625  5.5625  5.5625
03-08-1987      5.5     5.5625  23800   5.5625  5.5625  5.5625
04-08-1987      5.5     5.5625  35800   5.5625  5.5625  5.5625
Time taken: 0.109 seconds, Fetched: 5 row(s)
```

Let's solve the scenarios :

Question 1:
Write a Hive query to identify the top three dates that experienced the largest percentage change in stock price (from open to close) for every stock.

Let's see the result for a single table (**AAL**), we can populate it to others as well

Code:

```
hive> select date, abs((open - close)/open*100) as percentage_change from AAL_st
ock order by percentage_change desc limit 3;
```

Output:

```
22-07-2008      45.23809176260581
10-10-2008      40.67278077870862
12-11-2008      30.266666412353516
Time taken: 62.219 seconds, Fetched: 3 row(s)
```

Question 2:
Write a Hive query to identify the dates where Low is less than average month low for every stock.

Let's see the result for a single table (**AAL**), we can populate it to others as well

Code:

```
hive> with cte as
    > (
    > select date, low , avg(low) over(partition by substr(date, 3, 9) order by
cast(substr(date,6,9) as int)) as average_value from AAL_stock
    > )
    > select date, low, average_value from cte where low<average_value
```

Output:

```
06-12-2021      16.91   17.216363473372027
03-12-2021      16.34   17.216363473372027
02-12-2021      16.15   17.216363473372027
01-12-2021      16.26   17.216363473372027
20-12-2021      16.45   17.216363473372027
17-12-2021      16.4    17.216363473372027
16-12-2021      16.43   17.216363473372027
15-12-2021      16.27   17.216363473372027
14-12-2021      16.81   17.216363473372027
13-12-2021      16.85   17.216363473372027
07-12-2022      13.53   13.649999976158142
08-12-2022      13.38   13.649999976158142
09-12-2022      13.42   13.649999976158142
12-12-2022      13.45   13.649999976158142
Time taken: 60.203 seconds, Fetched: 2152 row(s)
```

Question 3:
Write a Hive query to find the date with the longest consecutive streak of increasing closing prices for every stock.

Let's see the result for a single table (**AAL**), we can populate it to others as well

Code:

```sql
with cte as (
  SELECT
    Date,
    Close,
    lag(Close) over(
      order by
        unix_timestamp(Date, 'dd-MM-yyyy')
    ) as prev
  FROM
    aal_stock
),
cte1 as (
    select

    Date,
    Close,
    if(Close < prev, 0, 1) as is_inc
  from
    cte
),
cte2 as (
  select

    Date,
    Close,
    is_inc,
    lead(is_inc) over(
      order by
        unix_timestamp(Date, 'dd-MM-yyyy')
    ) as next_val
  from
    cte1
),
cte3 as (
  select

    Date,
    Close,
    is_inc,
    if(
      is_inc = 0
      and is_inc != next_val,
      1,
      0
    ) as is_chg
  from
    cte2
),
cte4 as (
  select

    Date,
    Close,
    is_inc,
    is_chg,
    sum(is_chg) over(
      order by  unix_timestamp(Date, 'dd-MM-yyyy')
    ) as bucket
  from
    cte3
  where
    not (
      is_inc = 0
      and is_chg = 0
    )
),
cte5 as (
  select

    Date,
    Close,
    is_inc,
    is_chg,
    bucket,
    count(*) over(partition by  bucket) as cnt
  from
    cte4
)
select
    *
from cte5
cross join (select  max(cnt) as max_cnt from cte5) cte6
where cte5.cnt = cte6.max cnt;
```

Output:

	cte5.date	cte5.close
1	04-11-2005	28.799999237060547
2	25-10-2005	22
3	26-10-2005	22.319999694824219
4	27-10-2005	22.559999465942383
5	28-10-2005	23.899999618530273
6	31-10-2005	24.680000305175781
7	01-11-2005	25.020000457763672
8	02-11-2005	27.549999237060547
9	03-11-2005	27.870000839233398
10	07-11-2005	28.930000305175781
11	08-11-2005	29.430000305175781
12	09-11-2005	31.299999237060547
13	10-11-2005	32.799999237060547
14	11-11-2005	33.349998474121094
15	14-11-2005	33.75
16	15-11-2005	33.880001068115234

Question 4:

Write a Hive query to find the dates where AAL open price is higher than AAOI open price OR AAL volume greater than ABMD (write your query in an optimised way).

Code:

```
hive> select AAL_stock.date, AAL_stock.open as aal_open_price, AAOI_stock.open a
aoi_open_price, AAL_stock.volume aal_volume_price, ABMD_stock.volume abmd_volume
_price
    > from
    > AAL_stock
    > join
    > AAOI_stock
    > on AAL_stock.date = AAOI_stock.date
    > join
    > ABMD_stock
    > on AAL_stock.date = ABMD_stock.date
    > where AAL_stock.open > AAOI_stock.open or AAL_stock.volume > ABMD_stock.vo
lume;
```

Output:

```
10-11-2022      14.46   2.23    35599100        2849100
11-11-2022      15.0    2.24    24134000        1665200
14-11-2022      14.79   2.28    26266600        2754200
15-11-2022      15.02   2.4     29423100        3660200
16-11-2022      14.45   2.3     28858000        1667800
17-11-2022      13.82   2.1     24041700        1621700
18-11-2022      14.26   2.18    25968300        1678600
21-11-2022      14.02   2.12    25708400        964500
22-11-2022      13.85   2.2     26387000        661500
23-11-2022      13.98   2.268   23686400        801800
25-11-2022      14.4    2.18    9903900 258900
28-11-2022      14.34   2.18    21313800        830300
29-11-2022      13.89   2.11    17335300        592000
30-11-2022      14.12   2.1     21195200        733900
01-12-2022      14.46   2.18    26519600        644000
02-12-2022      13.82   2.17    24094600        704800
05-12-2022      13.84   2.33    27029000        886200
06-12-2022      14.14   2.28    20781500        672300
07-12-2022      14.24   2.1     28161400        974100
08-12-2022      13.65   2.13    25300900        1506900
09-12-2022      13.52   2.15    18489800        990100
12-12-2022      13.49   2.11    8048550 412703
Time taken: 67.921 seconds, Fetched: 2320 row(s)
```

Question 5:

Write a Hive query to calculate VH ratio(volume to high ratio).

Let's see the result for a single table (**AAL**), we can populate it to others as well

Code:

```
hive> select date ,volume, high, (volume / high) as VH_ratio from aal_stock;
```

Output:

```
10-11-2022      35599100        14.99   2374856.6073089754
11-11-2022      24134000        15.08   1600397.8860809463
14-11-2022      26266600        15.03   1747611.4748278516
15-11-2022      29423100        15.18   1938280.5934442494
16-11-2022      28858000        14.47   1994333.0661683218
17-11-2022      24041700        14.02   1714814.4946519416
18-11-2022      25968300        14.58   1781090.5442994805
21-11-2022      25708400        14.27   1801569.6689072778
22-11-2022      26387000        14.03   1880755.5596732656
23-11-2022      23686400        14.43   1641469.1267542187
25-11-2022      9903900 14.7    673734.702619367
28-11-2022      21313800        14.44   1476024.9736402165
29-11-2022      17335300        14.16   1224244.36347492
30-11-2022      21195200        14.45   1466795.8671120491
01-12-2022      26519600        14.57   1820151.0333195617
02-12-2022      24094600        13.98   1723505.0635878402
05-12-2022      27029000        14.34   1884867.4834304077
06-12-2022      20781500        14.34   1449198.0319992977
07-12-2022      28161400        14.24   1977626.4362810934
08-12-2022      25300900        13.8    1833398.525384564
09-12-2022      18489800        13.66   1353572.4894977137
12-12-2022      8048550 13.93   577785.3426901584
Time taken: 0.217 seconds, Fetched: 4333 row(s)
```

Question 6:
Write a Hive query to find the dates where previous day close and current day open difference is greater than 0 for each stock.

Let's see the result for a single table (**AAL**), we can populate it to others as well

Code:

```
hive> with cte as
    > (
    > select date, open, lag(close) over(order by date) as previous_close, (lag(
Close) over(order by date) - open) as diff_prev_close from
    > (
    > select to_date(FROM_UNIXTIME(UNIX_TIMESTAMP(date, 'dd-MM-yyyy'), 'yyyy-MM-
dd')) as date, open, close from aal_stock) s
    > )
    >
    > select * from cte where diff_prev_close > 0;
```

Output:

```
2022-10-12      12.21   12.26   0.05000019
2022-10-13      12.5    12.7    0.19999981
2022-10-20      13.95   13.99   0.03999996
2022-10-21      13.36   13.46   0.10000038
2022-10-25      14.05   14.1    0.05000019
2022-10-26      14.14   14.29   0.14999962
2022-11-02      14.0    14.11   0.10999966
2022-11-03      13.36   13.58   0.22000027
2022-11-08      14.29   14.32   0.029999733
2022-11-09      14.12   14.25   0.13000011
2022-11-14      14.79   14.88   0.09000015
2022-11-16      14.45   14.61   0.15999985
2022-11-17      13.82   14.09   0.27000046
2022-11-21      14.02   14.05   0.029999733
2022-11-25      14.4    14.42   0.020000458
2022-11-28      14.34   14.5    0.15999985
2022-11-30      14.12   14.14   0.020000458
2022-12-02      13.82   13.98   0.15999985
2022-12-05      13.84   13.97   0.13000011
2022-12-07      14.24   14.33   0.09000015
2022-12-09      13.52   13.6    0.07999992
2022-12-12      13.49   13.53   0.03999996
Time taken: 28.945 seconds, Fetched: 1843 row(s)
```

Question 7:
Find median of volume for ABIO.

Code:

```
hive> WITH MedianQuery AS (
    >     SELECT volume,
    >            ROW_NUMBER() OVER (ORDER BY volume) AS RowNum,
    >            COUNT(*) OVER () AS TotalRows
    >     FROM ABIO_stock
    > )SELECT  avg(volume) as Median_value FROM MedianQuery where RowNum in (cei
ling(TotalRows/2.0), if(ceiling(TotalRows/2.0) = (TotalRows/2.0), (TotalRows/2.0
)+1, ceiling(TotalRows/2.0)));
```

Output:

```
MapReduce Total cumulative CPU time: 4 seconds 690 msec
Ended Job = job_1689150900907_0081
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.48 sec   HDFS Read: 377333
HDFS Write: 141078 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 9.87 sec   HDFS Read: 150139
HDFS Write: 127 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 4.69 sec   HDFS Read: 5105 HD
FS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 40 msec
OK
62.0
Time taken: 392.513 seconds, Fetched: 1 row(s)
```

Let's apply some of the optimization technique :

I would be applying below two technique

i. Indexing (Compact Indexing)
ii. Bucketing
iii. Compressing file format

Let's first apply Indexing:

- **Creating index on table : abio_stock**

```
hive> Create index index_abio on table abio_stock(date) as 'org.apache.hadoop.hi
ve.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;
OK
Time taken: 2.264 seconds
```

Let's check one particular query which was taking more time previously

User story : 6

Old Result: Time - 158 sec

```
2022-10-10      2.0     2.07    0.06999993
2022-10-11      2.01    2.04    0.029999971
2022-10-13      1.98    2.02    0.03999996
2022-10-18      2.06    2.1     0.03999996
2022-10-21      2.04    2.06    0.01999998
2022-10-24      2.05    2.08    0.029999971
2022-10-28      2.05    2.09    0.03999996
2022-10-31      2.12    2.18    0.06000018
2022-11-03      2.1     2.13    0.03000021
2022-11-04      2.05    2.08    0.029999971
2022-11-07      2.02    2.06    0.03999996
2022-11-14      2.0     2.01    0.00999999
2022-11-22      2.05    2.07    0.01999998
2022-11-23      2.07    2.08    0.00999999
2022-11-25      2.06    2.1     0.03999996
2022-11-28      2.01    2.08    0.06999993
2022-12-01      2.03    2.09    0.059999943
2022-12-02      2.03    2.07    0.03999996
2022-12-05      2.1     2.11    0.00999999
2022-12-06      2.11    2.13    0.02000022
2022-12-07      2.07    2.1     0.029999971
2022-12-09      2.14    2.2     0.059999943
Time taken: 158.059 seconds, Fetched: 2405 row(s)
```

New Result after applying Index: Time - 28 sec

```
2022-10-10      2.0     2.07    0.06999993
2022-10-11      2.01    2.04    0.029999971
2022-10-13      1.98    2.02    0.03999996
2022-10-18      2.06    2.1     0.03999996
2022-10-21      2.04    2.06    0.01999998
2022-10-24      2.05    2.08    0.029999971
2022-10-28      2.05    2.09    0.03999996
2022-10-31      2.12    2.18    0.06000018
2022-11-03      2.1     2.13    0.03000021
2022-11-04      2.05    2.08    0.029999971
2022-11-07      2.02    2.06    0.03999996
2022-11-14      2.0     2.01    0.00999999
2022-11-22      2.05    2.07    0.01999998
2022-11-23      2.07    2.08    0.00999999
2022-11-25      2.06    2.1     0.03999996
2022-11-28      2.01    2.08    0.06999993
2022-12-01      2.03    2.09    0.059999943
2022-12-02      2.03    2.07    0.03999996
2022-12-05      2.1     2.11    0.00999999
2022-12-06      2.11    2.13    0.02000022
2022-12-07      2.07    2.1     0.029999971
2022-12-09      2.14    2.2     0.059999943
Time taken: 28.721 seconds, Fetched: 2405 row(s)
```

User story : 7

Old Result: Time - 392 sec	New Result after applying Index: Time - 75 sec

```
MapReduce Total cumulative CPU time: 4 seconds 690 msec
Ended Job = job_1689150900907_0081
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.48 sec   HDFS Read: 377333
HDFS Write: 141078 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 9.87 sec   HDFS Read: 150139
HDFS Write: 127 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 4.69 sec   HDFS Read: 5105 HD
FS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 40 msec
OK
62.0
Time taken: 392.513 seconds, Fetched: 1 row(s)
```

```
MapReduce Total cumulative CPU time: 2 seconds 150 msec
Ended Job = job_1689395438209_0019
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.03 sec   HDFS Read: 377333
HDFS Write: 141078 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.18 sec   HDFS Read: 150139
HDFS Write: 127 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 2.15 sec   HDFS Read: 5105 HD
FS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 360 msec
OK
62.0
Time taken: 75.024 seconds, Fetched: 1 row(s)
```

Observation: As we can see the result from above, the time differences is quite significant in case of both the scenario, so indexing is working good.

Let's apply Bucketing:

Code:

```
hive> create table stock_bucket_aal(date string, low float, open float, volume i
nt, high float, close float, adjusted_close float) clustered by (date) into 3 bu
ckets row format delimited fields terminated by ',';
OK
```

As we can see below as I am creating 3 buckets for every table:

Browse Directory

/user/hive/warehouse/stock_bucket_aal Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxrwxrwx	cloudera	supergroup	71.39 KB	Sun Jul 16 03:26:47 -0700 2023	1	128 MB	000000_0
-rwxrwxrwx	cloudera	supergroup	72.06 KB	Sun Jul 16 03:26:48 -0700 2023	1	128 MB	000001_0
-rwxrwxrwx	cloudera	supergroup	71.55 KB	Sun Jul 16 03:26:50 -0700 2023	1	128 MB	000002_0

Hadoop, 2016.

Let's apply bucketing on a particular scenario and check if it works :

User story : 7

Old Result: Time - 392 sec

```
MapReduce Total cumulative CPU time: 4 seconds 690 msec
Ended Job = job_1689150900907_0081
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.48 sec   HDFS Read: 377333
HDFS Write: 141078 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 9.87 sec   HDFS Read: 150139
HDFS Write: 127 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 4.69 sec   HDFS Read: 5105 HD
FS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 24 seconds 40 msec
OK
62.0
Time taken: 392.513 seconds, Fetched: 1 row(s)
```

New Result after applying Bucketing : Time - 149 sec

```
2023-07-16 03:41:11,787 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 4.47
sec
MapReduce Total cumulative CPU time: 4 seconds 470 msec
Ended Job = job_1689395438209_0036
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.51 sec   HDFS Read: 333866
HDFS Write: 141078 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 9.11 sec   HDFS Read: 150153
HDFS Write: 127 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 4.47 sec   HDFS Read: 5105 HD
FS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 22 seconds 90 msec
OK
62.0
Time taken: 148.513 seconds, Fetched: 1 row(s)
```

Observation :

Here we are not getting that much of good benefit by using bucket. The reasons may be various :

- **Bucketing based on year:**
  - By bucketing the data based on the **year** from the **date** column, we can potentially improve performance by leveraging partition pruning on queries that specifically filter by year. Our user stories do not require filtering based on a specific year, bucketing by year here may not provide significant optimization benefits.

- **Querying a specific bucket:**
  - When writing a query, specifying a specific bucket to search on can improve performance by reducing the amount of data that needs to be processed.
  - The choice of which bucket to query should be provided by the administrator or the user running the query. They should have knowledge of the data distribution and the desired optimization criteria.

Now Let's apply scenario by compressing file format:

User story : 4

Code: I have compression all the four tables and saved into HIVE warehouse in **ORC** format. Below is the sample code for a particular table this can be populated to other table as well.

```
Time tcreate table myorc_abmd(date string, low float, open float, volume int, hi
gh float, close float, adjusted_close float) stored as orc;
```

Inserting the data:

```
Time tinsert overwrite table myorc_abmd select * from abmd_stock;
```

Old Result: Time - 68 sec

```
10-11-2022      14.46   2.23    35599100        2849100
11-11-2022      15.0    2.24    24134000        1665200
14-11-2022      14.79   2.28    26266600        2754200
15-11-2022      15.02   2.4     29423100        3660200
16-11-2022      14.45   2.3     28858000        1667800
17-11-2022      13.82   2.1     24041700        1621700
18-11-2022      14.26   2.18    25968300        1678600
21-11-2022      14.02   2.12    25708400        964500
22-11-2022      13.85   2.2     26387000        661500
23-11-2022      13.98   2.268   23686400        801800
25-11-2022      14.4    2.18    9903900 258900
28-11-2022      14.34   2.18    21313800        830300
29-11-2022      13.89   2.11    17335300        592000
30-11-2022      14.12   2.1     21195200        733900
01-12-2022      14.46   2.18    26519600        644000
02-12-2022      13.82   2.17    24094600        704800
05-12-2022      13.84   2.33    27029000        886200
06-12-2022      14.14   2.28    20781500        672300
07-12-2022      14.24   2.1     28161400        974100
08-12-2022      13.65   2.13    25300900        1506900
09-12-2022      13.52   2.15    18489800        990100
12-12-2022      13.49   2.11    8048550 412703
Time taken: 67.921 seconds, Fetched: 2320 row(s)
```

New Result after applying compression : Time - 26 sec

```
10-11-2022      14.46   2.23    35599100        2849100
11-11-2022      15.0    2.24    24134000        1665200
14-11-2022      14.79   2.28    26266600        2754200
15-11-2022      15.02   2.4     29423100        3660200
16-11-2022      14.45   2.3     28858000        1667800
17-11-2022      13.82   2.1     24041700        1621700
18-11-2022      14.26   2.18    25968300        1678600
21-11-2022      14.02   2.12    25708400        964500
22-11-2022      13.85   2.2     26387000        661500
23-11-2022      13.98   2.268   23686400        801800
25-11-2022      14.4    2.18    9903900 258900
28-11-2022      14.34   2.18    21313800        830300
29-11-2022      13.89   2.11    17335300        592000
30-11-2022      14.12   2.1     21195200        733900
01-12-2022      14.46   2.18    26519600        644000
02-12-2022      13.82   2.17    24094600        704800
05-12-2022      13.84   2.33    27029000        886200
06-12-2022      14.14   2.28    20781500        672300
07-12-2022      14.24   2.1     28161400        974100
08-12-2022      13.65   2.13    25300900        1506900
09-12-2022      13.52   2.15    18489800        990100
12-12-2022      13.49   2.11    8048550 412703
Time taken: 26.703 seconds, Fetched: 2320 row(s)
```

<u>Observation</u> :

Here we have a significant change in the time it took, so we can say if we compress a table into a particular file format, it will have a good performance as well.

Step - 4

Next, I will be creating external tables for the results to save in the MySQL database

Question 1:

Let's see the result for a single table (**AAL**), we can populate it to others as well

- **Lets Create External Table first:**

```
hive> create external table q1_aal(date string, percentage_change float) row for
mat delimited fields terminated by ',' location '/user/hive/warehouse/q1_aal/res
ults.txt';
OK
Time taken: 3.809 seconds
```

- **Lets Insert the data:**

```
hive> insert overwrite table q1_aal select date, abs((open - close)/open*100) as
 percentage_change from AAL_stock order by percentage_change desc limit 3;
Query ID = cloudera_20230712095858_0594c9f2-ac03-4d56-8b20-0bb5fb3899e7
Total jobs = 1
Launching Job 1 out of 1
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>
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q1_aal_percent -
-export-dir /user/hive/warehouse/q1_aal/results.txt/000000_0 --input-fields-term
inated-by ',';
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/07/12 10:16:46 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0
23/07/12 10:16:46 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
23/07/12 10:16:47 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
23/07/12 10:16:47 INFO tool.CodeGenTool: Beginning code generation
23/07/12 10:16:48 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `q1_aal_percent` AS t LIMIT 1
23/07/12 10:16:48 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `q1_aal_percent` AS t LIMIT 1
23/07/12 10:16:48 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/ha
doop-mapreduce
Note: /tmp/sqoop-cloudera/compile/73df2d3e398e53ef4aa4cd0bd34723f0/q1_aal_percen
t.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

- **Let's check the result in MYSQL database:**

```
mysql> select * from q1_aal_percent;
+------------+-------------------+
| date       | percentage_change |
+------------+-------------------+
| 10-10-2008 |           40.6728 |
| 22-07-2008 |           45.2381 |
| 12-11-2008 |           30.2667 |
+------------+-------------------+
3 rows in set (0.01 sec)
```

Question 2:

Let's see the result for a single table (**AAL**), we can populate it to others as well

- **Lets Create External Table first:**

```
hive> create external table q2_aal(date string, low float, average_value float)
row format delimited fields terminated by ',' location '/user/hive/warehouse/q2_
aal/results.txt';
OK
Time taken: 0.153 seconds
```

- **Lets Insert the data:**

```
hive> with cte as
    > (
    > select date, low , avg(low) over(partition by substr(date, 3, 9) order by
cast(substr(date,6,9) as int)) as average_value from AAL_stock
    > )
    > insert overwrite table q2_aal select date, low, average_value from cte whe
re low<average_value;
Query ID = cloudera_20230712104848_852cbbc2-b519-4056-81e2-430dae24df74
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_1689150900907_0090, Tracking URL = http://quickstart.cloudera
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q2_aal_diff --ex
port-dir /user/hive/warehouse/q2_aal/results.txt/000000_0 --input-fields-termina
ted-by ',';
```

- **Let's check the result in MYSQL database:**

```
mysql> select * from q2_aal_diff limit 10;
+------------+-------+-------------------+
| date       | low   | average_month_low |
+------------+-------+-------------------+
| 16-10-2007 | 27.53 |           27.8578 |
| 15-10-2007 |  27.7 |           27.8578 |
| 01-10-2007 | 26.35 |           27.8578 |
| 02-10-2007 | 27.79 |           27.8578 |
| 18-10-2007 | 27.71 |           27.8578 |
| 25-10-2007 | 27.04 |           27.8578 |
| 26-10-2007 |  26.4 |           27.8578 |
| 29-10-2007 |  26.3 |           27.8578 |
| 30-10-2007 | 27.12 |           27.8578 |
| 31-10-2007 |  27.3 |           27.8578 |
+------------+-------+-------------------+
10 rows in set (0.00 sec)
```

Question 3:

Let's see the result for a single table (**AAL**), we can populate it to others as well

- **Lets Create External Table first:**

```
hive> create external table q3_aal_str(date string, close_price float) row forma
t delimited fields terminated by ',' location '/user/hive/warehouse/q3_aal_str/r
esults.txt';
OK
Time taken: 4.857 seconds
```

- **Lets Insert the data:**

```
hive> with cte as (
    >   SELECT
    >     Date,
    >     Close,
    >     lag(Close) over(
    >       order by
    >         unix_timestamp(Date, 'dd-MM-yyyy')
    >     ) as prev
    >   FROM
    >     aal_stock
    > ),
    > cte1 as (
    >     select
    >
    >     Date,
    >     Close,
    >     if(Close < prev, 0, 1) as is_inc
    >   from
    >     cte
    > ),
    > cte2 as (
    >   select
    >
    >     Date,
    >     Close,
    >     is_inc,
    >     lead(is_inc) over(
    >       order by
    >         unix_timestamp(Date, 'dd-MM-yyyy')
    >     ) as next_val
    >   from
    >     cte1
    > ),
    > cte3 as (
    >   select
    >
    >     Date,
    >     Close,
    >     is_inc,
    >     if(
 >       is_inc = 0
 >       and is_inc != next_val,
 >       1,
 >       0
 >     ) as is_chg
 >   from
 >     cte2
 > ),
 > cte4 as (
 >   select
 >
 >     Date,
 >     Close,
 >     is_inc,
 >     is_chg,
 >     sum(is_chg) over(
 >       order by  unix_timestamp(Date, 'dd-MM-yyyy')
 >     ) as bucket
 >   from
 >     cte3
 >   where
 >     not (
 >       is_inc = 0
 >       and is_chg = 0
 >     )
 > ),
 > cte5 as (
 >   select
 >
 >     Date,
 >     Close,
 >     is_inc,
 >     is_chg,
 >     bucket,
 >     count(*) over(partition by  bucket) as cnt
 >   from
 >     cte4
 > ) insert overwrite table q3_aal_str
 > select
 >     Date, Close
```

```
    > from cte5
    > cross join (select  max(cnt) as max_cnt from cte5) cte6
    > where cte5.cnt = cte6.max_cnt
    > order by Close;
Query ID = cloudera_20230713065858_cb0d1e37-e41c-4ced-bd2a-f85a88c2d3a6
Total jobs = 13
Launching Job 1 out of 13
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_1689255701724_0001, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1689255701724_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1689255701724_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q3_aal_streak --
export-dir /user/hive/warehouse/q3_aal_str/results.txt/000000_0 --input-fields-t
erminated-by ',';
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/07/13 07:10:26 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0
23/07/13 07:10:26 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
23/07/13 07:10:26 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
23/07/13 07:10:26 INFO tool.CodeGenTool: Beginning code generation
23/07/13 07:10:27 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `q3_aal_streak` AS t LIMIT 1
23/07/13 07:10:27 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `q3_aal_streak` AS t LIMIT 1
23/07/13 07:10:27 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/ha
doop-mapreduce
```

- **Let's check the result in MYSQL database:**

```
mysql> select * from q3_aal_streak;
+------------+-------------+
| date       | close_price |
+------------+-------------+
| 10-11-2005 |        32.8 |
| 11-11-2005 |       33.35 |
| 14-11-2005 |       33.75 |
| 15-11-2005 |       33.88 |
| 25-10-2005 |          22 |
| 26-10-2005 |       22.32 |
| 27-10-2005 |       22.56 |
| 28-10-2005 |        23.9 |
| 31-10-2005 |       24.68 |
| 01-11-2005 |       25.02 |
| 02-11-2005 |       27.55 |
| 03-11-2005 |       27.87 |
| 04-11-2005 |        28.8 |
| 07-11-2005 |       28.93 |
| 08-11-2005 |       29.43 |
| 09-11-2005 |        31.3 |
+------------+-------------+
16 rows in set (0.01 sec)
```

Question 4:

- **Lets Create External Table first:**

```
hive> create external table q3_op(date string, aal_open_price float, aaoi_open_p
rice float, aal_volume bigint, abmd_volume bigint) row format delimited fields t
erminated by ',' location '/user/hive/warehouse/q3_op/results.txt';
OK
Time taken: 1.235 seconds
```

- **Lets Insert the data:**

```
hive> insert overwrite table q3_op select AAL_stock.date, AAL_stock.open as aal_
open_price, AAOI_stock.open aaoi_open_price, AAL_stock.volume aal_volume_price,
ABMD_stock.volume abmd_volume_price
    > from
    > AAL_stock
    > join
    > AAOI_stock
    > on AAL_stock.date = AAOI_stock.date
    > join
    > ABMD_stock
    > on AAL_stock.date = ABMD_stock.date
    > where AAL_stock.open > AAOI_stock.open or AAL_stock.volume > ABMD_stock.vo
lume;
Query ID = cloudera_20230712112525_0bcfdee1-f946-4c3d-ab70-93a7a1f039c1
Total jobs = 1
Execution log at: /tmp/cloudera/cloudera_20230712112525_0bcfdee1-f946-4c3d-ab70-
93a7a1f039c1.log
2023-07-12 11:25:34     Starting to launch local task to process map join;     m
aximum memory = 1013645312
2023-07-12 11:25:38     Dump the side-table for tag: 0 with group count: 4333 in
to file: file:/tmp/cloudera/135e5e77-9a3a-4b7f-bbf3-49b4321bf5a6/hive_2023-07-12
_11-25-18_115_6895647159276693814-1/-local-10002/HashTable-Stage-5/MapJoin-mapfi
le00--.hashtable
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q4_tables_comp -
-export-dir /user/hive/warehouse/q3_op/results.txt/000000_0 --input-fields-termi
nated-by ',';
```

- **Let's check the result in MYSQL database:**

```
mysql> select * from q4_tables_comp limit 10;
+------------+----------------+-----------------+------------+-------------+
| date       | aal_open_price | aaoi_open_price | aal_volume | abmd_volume |
+------------+----------------+-----------------+------------+-------------+
| 25-08-2020 |          13.69 |            11.7 |   79053400 |      253600 |
| 26-08-2020 |          13.11 |              12 |   44056800 |      240700 |
| 27-08-2020 |          13.43 |           11.64 |  108835700 |      207400 |
| 28-08-2020 |          13.59 |           11.46 |   54516400 |      278700 |
| 31-08-2020 |           13.6 |           11.68 |   45917200 |      320600 |
| 01-09-2020 |          12.86 |           11.63 |   72849700 |      320300 |
| 02-09-2020 |          12.94 |           11.74 |   58889600 |      395500 |
| 03-09-2020 |           13.4 |           12.41 |   86390800 |      381900 |
| 04-09-2020 |          13.65 |              11 |   64937000 |      511900 |
| 08-09-2020 |          13.36 |           10.17 |   72746000 |      392800 |
+------------+----------------+-----------------+------------+-------------+
10 rows in set (0.00 sec)
```

Question 5:

Let's see the result for a single table (**AAL**), we can populate it to others as well

- **Lets Create External Table first:**

```
hive> create external table q5_aal(date string, volume bigint, high float, VH_ra
tio float) row format delimited fields terminated by ',' location '/user/hive/wa
rehouse/q5_aal/results.txt';
OK
Time taken: 0.131 seconds
```

- **Lets Insert the data:**

```
hive> insert overwrite table q5_aal select date ,volume, high, (volume / high) a
s VH_ratio from aal_stock;
Query ID = cloudera_20230712114242_d84077cb-b309-40af-b0cf-ce9e9bdf7156
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1689150900907_0104, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1689150900907_0104/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1689150900907_0104
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2023-07-12 11:42:17,495 Stage-1 map = 0%,  reduce = 0%
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q5_aal_VH --expo
rt-dir /user/hive/warehouse/q5_aal/results.txt/000000_0 --input-fields-terminate
d-by ',';
```

- **Let's check the result in MYSQL database:**

```
mysql> select * from q5_aal_VH limit 10;
+------------+----------+-------+----------+
| date       | volume   | high  | VH_Ratio |
+------------+----------+-------+----------+
| 03-10-2018 |  6370300 | 39.26 |   162259 |
| 04-10-2018 |  5916500 | 39.01 |   151666 |
| 05-10-2018 |  9127000 | 38.13 |   239365 |
| 08-10-2018 |  7879300 | 36.85 |   213821 |
| 09-10-2018 | 19662600 | 36.39 |   540330 |
| 10-10-2018 | 20539000 |  33.4 |   614940 |
| 11-10-2018 | 17115800 | 32.75 |   522620 |
| 12-10-2018 | 12905200 | 32.13 |   401656 |
| 15-10-2018 | 11092500 | 32.24 |   344060 |
| 16-10-2018 | 11339200 | 33.44 |   339091 |
+------------+----------+-------+----------+
10 rows in set (0.00 sec)
```

Question 6:

Let's see the result for a single table (**AAL**), we can populate it to others as well

- **Lets Create External Table first:**

```
hive> create external table q6_aal(date string, open float, prev_close float, pr
ev_diff_close_open float) row format delimited fields terminated by ',' location
 '/user/hive/warehouse/q6_aal/results.txt';
OK
Time taken: 0.044 seconds
```

- **Lets Insert the data:**

```
hive> with cte as
    > (
    > select date, open, lag(close) over(order by date) as previous_close, (lag(
Close) over(order by date) - open) as diff_prev_close from
    > (
    > select to_date(FROM_UNIXTIME(UNIX_TIMESTAMP(date, 'dd-MM-yyyy'), 'yyyy-MM-
dd')) as date, open, close from aal_stock) s
    > )
    >
    > insert overwrite table q6_aal select * from cte where diff_prev_close > 0;

Query ID = cloudera_20230712120000_102a7e6f-d2b7-4c5e-b1a9-3e77a1185c69
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_1689150900907_0112, Tracking URL = http://quickstart.cloudera
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q6_aal_open_clos
e_diff --export-dir /user/hive/warehouse/q6_aal/results.txt/000000_0 --input-fie
lds-terminated-by ',';
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
23/07/12 12:02:51 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0
23/07/12 12:02:51 WARN tool.BaseSqoopTool: Setting your password on the command-
line is insecure. Consider using -P instead.
23/07/12 12:02:51 INFO manager.MySQLManager: Preparing to use a MySQL streaming
resultset.
23/07/12 12:02:51 INFO tool.CodeGenTool: Beginning code generation
23/07/12 12:02:51 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `q6_aal_open_close_diff` AS t LIMIT 1
23/07/12 12:02:52 INFO manager.SqlManager: Executing SQL statement: SELECT t.* F
ROM `q6_aal_open_close_diff` AS t LIMIT 1
23/07/12 12:02:52 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/lib/ha
doop-mapreduce
Note: /tmp/sqoop-cloudera/compile/3739123a08677cfc457194cfaddc1205/q6_aal_open_c
lose_diff.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

- **Let's check the result in MYSQL database:**

```
mysql> select * from q6_aal_open_close_diff limit 10;
+------------+-------+------------+----------------------+
| date       | open  | prev_close | prev_diff_close_open |
+------------+-------+------------+----------------------+
| 2018-12-27 | 31.71 |      32.29 |             0.580002 |
| 2019-01-02 | 31.46 |      32.11 |             0.650002 |
| 2019-01-03 | 31.69 |      32.48 |             0.789999 |
| 2019-01-07 | 31.99 |      32.04 |            0.0500011 |
| 2019-01-10 | 30.62 |      33.42 |                  2.8 |
| 2019-01-11 |  31.8 |      32.04 |             0.240002 |
| 2019-01-14 |  31.4 |       31.8 |                  0.4 |
| 2019-01-17 | 32.75 |      32.84 |            0.0900002 |
| 2019-01-22 | 33.76 |      33.97 |             0.210003 |
| 2019-01-28 | 34.53 |      34.98 |             0.450001 |
+------------+-------+------------+----------------------+
10 rows in set (0.00 sec)
```

Question 7:

- **Lets Create External Table first:**

```
hive> create external table q7_abio(median_value int) row format delimited field
s terminated by ',' location '/user/hive/warehouse/q7_abio/results.txt';
OK
Time taken: 0.068 seconds
```

- **Lets Insert the data:**

```
hive> WITH MedianQuery AS (
    >     SELECT volume,
    >            ROW_NUMBER() OVER (ORDER BY volume) AS RowNum,
    >            COUNT(*) OVER () AS TotalRows
    >     FROM ABIO_stock
    > ) insert overwrite table q7_abio SELECT avg(volume) as Median_value FROM M
edianQuery where RowNum in (ceiling(TotalRows/2.0), if(ceiling(TotalRows/2.0) =
(TotalRows/2.0), (TotalRows/2.0)+1, ceiling(TotalRows/2.0)));
WARNING: Comparing a bigint and a double may result in a loss of precision.
Query ID = cloudera_20230712122323_dde01007-4c04-4885-a5f3-74c81f81169f
Total jobs = 3
Launching Job 1 out of 3
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_1689150900907_0120, Tracking URL = http://quickstart.cloudera
```

- **Lets export the data to MYSQL database:**

```
[cloudera@quickstart Desktop]$ sqoop export --connect jdbc:mysql://localhost:330
6/stock_analysis_op --username root --password cloudera --table q7_abio_median -
-export-dir /user/hive/warehouse/q7_abio/results.txt/000000_0 --input-fields-ter
minated-by ',';
```

- **Let's check the result in MYSQL database**

```
mysql> select * from q7_abio_median;
+--------------+
| median_value |
+--------------+
|           62 |
+--------------+
1 row in set (0.00 sec)
```

Let's see all the tables in SQL Database:

```
mysql> show tables;
+-----------------------------+
| Tables_in_stock_analysis_op |
+-----------------------------+
| q1_aal_percent              |
| q1_aaoi_percent             |
| q1_abio_percent             |
| q1_abmd_percent             |
| q2_aal_diff                 |
| q2_aaoi_diff                |
| q2_abio_diff                |
| q2_abmd_diff                |
| q3_aal_streak               |
| q3_aaoi_streak              |
| q3_abio_streak              |
| q3_abmd_streak              |
| q4_tables_comp              |
| q5_aal_VH                   |
| q5_aaoi_VH                  |
| q5_abio_VH                  |
| q5_abmd_VH                  |
| q6_aal_open_close_diff      |
| q6_aaoi_open_close_diff     |
| q6_abio_open_close_diff     |
| q6_abmd_open_close_diff     |
| q7_abio_median              |
+-----------------------------+
22 rows in set (0.00 sec)
```

Above we have seperately for every question for every stock I have created separate tables, Now next I have assembled all the separate tables into 1 by reducing the tables.

```
mysql> show tables;
+---------------------------------+
| Tables_in_stock_analysis_output |
+---------------------------------+
| user_story_1                    |
| user_story_2                    |
| user_story_3                    |
| user_story_4                    |
| user_story_5                    |
| user_story_6                    |
| user_story_7                    |
+---------------------------------+
7 rows in set (0.00 sec)
```

Let's also check the output for every user stories also:

User Story 1:

Write a Hive query to identify the top three dates that experienced the largest percentage change in stock price (from open to close) for every stock.

```
mysql> select * from user_story_1;
+------------+--------------------+------------+
| date       | percentage_change  | stock_name |
+------------+--------------------+------------+
| 10-10-2008 |           40.6728  | AAL        |
| 22-07-2008 |           45.2381  | AAL        |
| 12-11-2008 |           30.2667  | AAL        |
| 16-09-2022 |           24.9169  | AAOI       |
| 08-11-2018 |           20.5556  | AAOI       |
| 19-03-2020 |           18.8748  | AAOI       |
| 26-03-2010 |           145.373  | ABIO       |
| 28-01-2009 |             122.5  | ABIO       |
| 28-05-2020 |             117.8  | ABIO       |
| 22-02-1995 |           29.6296  | ABMD       |
| 31-12-1987 |           28.5714  | ABMD       |
| 30-12-1994 |           27.7778  | ABMD       |
+------------+--------------------+------------+
12 rows in set (0.00 sec)
```

User Story 2:

Write a Hive query to identify the dates where Low is less than average month low for every stock.

```
mysql> select * from user_story_2 limit 10;
+------------+-------+-------------------+------------+
| date       | low   | average_month_low | stock_name |
+------------+-------+-------------------+------------+
| 16-10-2007 | 27.53 |           27.8578 | AAL        |
| 15-10-2007 |  27.7 |           27.8578 | AAL        |
| 01-10-2007 | 26.35 |           27.8578 | AAL        |
| 02-10-2007 | 27.79 |           27.8578 | AAL        |
| 18-10-2007 | 27.71 |           27.8578 | AAL        |
| 25-10-2007 | 27.04 |           27.8578 | AAL        |
| 26-10-2007 |  26.4 |           27.8578 | AAL        |
| 29-10-2007 |  26.3 |           27.8578 | AAL        |
| 30-10-2007 | 27.12 |           27.8578 | AAL        |
| 31-10-2007 |  27.3 |           27.8578 | AAL        |
+------------+-------+-------------------+------------+
10 rows in set (0.00 sec)
```

User Story 3: Write a Hive query to find the date with the longest consecutive streak of increasing closing prices for every stock.

```
mysql> select * from user_story_3;
+------------+-------------+------------+---------------+
| date       | close_price | stock_name | serial_number |
+------------+-------------+------------+---------------+
| 25-10-2005 |          22 | AAL        |             1 |
| 26-10-2005 |       22.32 | AAL        |             2 |
| 27-10-2005 |       22.56 | AAL        |             3 |
| 28-10-2005 |        23.9 | AAL        |             4 |
| 31-10-2005 |       24.68 | AAL        |             5 |
| 01-11-2005 |       25.02 | AAL        |             6 |
| 02-11-2005 |       27.55 | AAL        |             7 |
| 03-11-2005 |       27.87 | AAL        |             8 |
| 04-11-2005 |        28.8 | AAL        |             9 |
| 07-11-2005 |       28.93 | AAL        |            10 |
| 08-11-2005 |       29.43 | AAL        |            11 |
| 09-11-2005 |        31.3 | AAL        |            12 |
| 10-11-2005 |        32.8 | AAL        |            13 |
| 11-11-2005 |       33.35 | AAL        |            14 |
| 14-11-2005 |       33.75 | AAL        |            15 |
| 15-11-2005 |       33.88 | AAL        |            16 |
| 18-11-2013 |       12.21 | AAOI       |             1 |
| 19-11-2013 |       12.43 | AAOI       |             2 |
| 20-11-2013 |        12.5 | AAOI       |             3 |
| 21-11-2013 |       12.56 | AAOI       |             4 |
| 25-11-2013 |       12.57 | AAOI       |             5 |
| 22-11-2013 |       12.57 | AAOI       |             6 |
```

20-11-2013	12.5	AAOI	3
21-11-2013	12.56	AAOI	4
25-11-2013	12.57	AAOI	5
22-11-2013	12.57	AAOI	6
26-11-2013	12.63	AAOI	7
27-11-2013	12.96	AAOI	8
29-11-2013	13.19	AAOI	9
02-07-2013	162.54	ABIO	1
03-07-2013	163.8	ABIO	2
05-07-2013	165.06	ABIO	3
09-07-2013	168.84	ABIO	4
08-07-2013	168.84	ABIO	5
10-07-2013	170.1	ABIO	6
11-07-2013	171.36	ABIO	7
15-07-2013	173.88	ABIO	8
12-07-2013	173.88	ABIO	9
16-07-2013	175.14	ABIO	10
17-07-2013	176.4	ABIO	11
18-07-2013	178.92	ABIO	12
13-06-1988	4.0625	ABMD	1
14-06-1988	4.0625	ABMD	2
15-06-1988	4.0625	ABMD	3
16-06-1988	4.1875	ABMD	4
17-06-1988	4.1875	ABMD	5
23-06-1988	4.25	ABMD	6
22-06-1988	4.25	ABMD	7
21-06-1988	4.25	ABMD	8
20-06-1988	4.25	ABMD	9
27-06-1988	4.25	ABMD	10
24-06-1988	4.25	ABMD	11
29-06-1988	4.3125	ABMD	12
28-06-1988	4.3125	ABMD	13
01-07-1988	4.3125	ABMD	14
30-06-1988	4.3125	ABMD	15
05-07-1988	4.6875	ABMD	16
06-07-1988	4.6875	ABMD	17

User Story 4:
Write a Hive query to find the dates where AAL open price is higher than AAOI open price OR AAL volume greater than AMBD (write your query in an optimised way).

```
mysql> select * from user_story_4 limit 10;
```

date	aal_open_price	aaoi_open_price	aal_volume	abmd_volume
25-08-2020	13.69	11.7	79053400	253600
26-08-2020	13.11	12	44056800	240700
27-08-2020	13.43	11.64	108835700	207400
28-08-2020	13.59	11.46	54516400	278700
31-08-2020	13.6	11.68	45917200	320600
01-09-2020	12.86	11.63	72849700	320300
02-09-2020	12.94	11.74	58889600	395500
03-09-2020	13.4	12.41	86390800	381900
04-09-2020	13.65	11	64937000	511900
08-09-2020	13.36	10.17	72746000	392800

```
10 rows in set (0.00 sec)
```

User Story 5:
Write a Hive query to calculate VH ratio(volume to high ratio).

```
mysql> select * from user_story_5 limit 10;
+------------+----------+-------+----------+------------+
| date       | volume   | high  | VH_Ratio | stock_name |
+------------+----------+-------+----------+------------+
| 03-10-2018 |  6370300 | 39.26 |   162259 | AAL        |
| 04-10-2018 |  5916500 | 39.01 |   151666 | AAL        |
| 05-10-2018 |  9127000 | 38.13 |   239365 | AAL        |
| 08-10-2018 |  7879300 | 36.85 |   213821 | AAL        |
| 09-10-2018 | 19662600 | 36.39 |   540330 | AAL        |
| 10-10-2018 | 20539000 |  33.4 |   614940 | AAL        |
| 11-10-2018 | 17115800 | 32.75 |   522620 | AAL        |
| 12-10-2018 | 12905200 | 32.13 |   401656 | AAL        |
| 15-10-2018 | 11092500 | 32.24 |   344060 | AAL        |
| 16-10-2018 | 11339200 | 33.44 |   339091 | AAL        |
+------------+----------+-------+----------+------------+
10 rows in set (0.00 sec)
```

User Story 6:
Write a Hive query to find the dates where previous day close and current day open difference is greater than 0 for each stock.

```
mysql> select * from user_story_6 limit 10;
+------------+-------+------------+---------------------+------------+
| date       | open  | prev_close | prev_diff_close_open | stock_name |
+------------+-------+------------+---------------------+------------+
| 2018-12-27 | 31.71 |      32.29 |            0.580002 | AAL        |
| 2019-01-02 | 31.46 |      32.11 |            0.650002 | AAL        |
| 2019-01-03 | 31.69 |      32.48 |            0.789999 | AAL        |
| 2019-01-07 | 31.99 |      32.04 |           0.0500011 | AAL        |
| 2019-01-10 | 30.62 |      33.42 |                 2.8 | AAL        |
| 2019-01-11 |  31.8 |      32.04 |            0.240002 | AAL        |
| 2019-01-14 |  31.4 |       31.8 |                 0.4 | AAL        |
| 2019-01-17 | 32.75 |      32.84 |           0.0900002 | AAL        |
| 2019-01-22 | 33.76 |      33.97 |            0.210003 | AAL        |
| 2019-01-28 | 34.53 |      34.98 |            0.450001 | AAL        |
+------------+-------+------------+---------------------+------------+
10 rows in set (0.00 sec)
```

User Story 7:
Find median of volume for ABIO.

```
mysql> select * from user_story_7;
+------------+--------------+
| stock_name | median_value |
+------------+--------------+
| ABIO       |           62 |
+------------+--------------+
1 row in set (0.00 sec)
```

Walmart Stock Analysis

Scenario 1: Print out first 5 rows ?

First, we have received the data in MYSQL DB.

```
mysql> show tables;
+--------------------------+
| Tables_in_stock_analysis |
+--------------------------+
| AAL_stock                |
| AAOI_stock               |
| ABIO_stock               |
| ABMD_stock               |
| walmart_stock            |
+--------------------------+
5 rows in set (0.00 sec)
```

In the next step we have created an empty table in HBASE.
HBASE Table Creation:

```
hbase(main):003:0> create table 'walmart' values
```

Importing data from SQL DB to HBASE:

```
[cloudera@quickstart Desktop]$ sqoop import --connect jdbc:mysql://localhost:3306/stoc
k_analysis --username root --password cloudera --table walmart_stock --hbase-table wal
mart --columns "date,Open,High,Low,Close,Volume,Adj_Close"  --hbase-row-key date --col
umn-family values --hbase-create-table -m 1
```

Let's read from HBASE:

```
hbase(main):022:0> scan 'walmart', {LIMIT=>1}
ROW                   COLUMN+CELL
 2012-01-03           column=values:Adj_Close, timestamp=1689345828549, value=52
                      .6192
 2012-01-03           column=values:Close, timestamp=1689345828549, value=60.33
 2012-01-03           column=values:High, timestamp=1689345828549, value=61.06
 2012-01-03           column=values:Low, timestamp=1689345828549, value=59.87
 2012-01-03           column=values:Open, timestamp=1689345828549, value=59.97
 2012-01-03           column=values:Volume, timestamp=1689345828549, value=12668
                      800
1 row(s) in 0.0090 seconds
```

Last, Let's see the top 5 rows:

```
hbase(main):002:0> scan 'walmart', {LIMIT=>5}
ROW                   COLUMN+CELL
 2012-01-03           column=values:Adj_Close, timestamp=1689345828549, value=52
                      .6192
 2012-01-03           column=values:Close, timestamp=1689345828549, value=60.33
 2012-01-03           column=values:High, timestamp=1689345828549, value=61.06
 2012-01-03           column=values:Low, timestamp=1689345828549, value=59.87
 2012-01-03           column=values:Open, timestamp=1689345828549, value=59.97
 2012-01-03           column=values:Volume, timestamp=1689345828549, value=12668
                      800
 2012-01-04           column=values:Adj_Close, timestamp=1689345828549, value=52
                      .0785
 2012-01-04           column=values:Close, timestamp=1689345828549, value=59.71
 2012-01-04           column=values:High, timestamp=1689345828549, value=60.35
 2012-01-04           column=values:Low, timestamp=1689345828549, value=59.47
 2012-01-04           column=values:Open, timestamp=1689345828549, value=60.21
 2012-01-04           column=values:Volume, timestamp=1689345828549, value=95933
                      00
 2012-01-05           column=values:Adj_Close, timestamp=1689345828549, value=51
                      .8255
 2012-01-05           column=values:Close, timestamp=1689345828549, value=59.42
 2012-01-05           column=values:High, timestamp=1689345828549, value=59.62
 2012-01-05           column=values:Low, timestamp=1689345828549, value=58.37
 2012-01-05           column=values:Open, timestamp=1689345828549, value=59.35
 2012-01-05           column=values:Volume, timestamp=1689345828549, value=12768
                      200
 2012-01-06           column=values:Adj_Close, timestamp=1689345828549, value=51
                      .4592
 2012-01-06           column=values:Close, timestamp=1689345828549, value=59.0
 2012-01-06           column=values:High, timestamp=1689345828549, value=59.45
 2012-01-06           column=values:Low, timestamp=1689345828549, value=58.87
 2012-01-06           column=values:Open, timestamp=1689345828549, value=59.42
 2012-01-06           column=values:Volume, timestamp=1689345828549, value=80694
                      00
 2012-01-09           column=values:Adj_Close, timestamp=1689345828549, value=51
                      .6162
 2012-01-09           column=values:Close, timestamp=1689345828549, value=59.18
 2012-01-09           column=values:High, timestamp=1689345828549, value=59.55
 2012-01-09           column=values:Low, timestamp=1689345828549, value=58.92
 2012-01-09           column=values:Open, timestamp=1689345828549, value=59.03
 2012-01-09           column=values:Volume, timestamp=1689345828549, value=66793
                      00
5 row(s) in 0.0770 seconds

hbase(main):003:0>
```

Importing Required Packages:

```
# import required packages

import pyodbc
import pandas as pd
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField, IntegerType, StructType, StringType,FloatType,DateType
from pyspark.sql.functions import lit,max,mean,min, first,desc,col,format_number

# Declaring Path and Variables

# driver = "ODBC Driver 18 for SQL Server"
driver = "SQL Server"
server = "DESKTOP-TOEPTEF\SQL_SERVER"
port = 1433

# table Name
table_name = "walmart_stock"

# databse_name
db_name='walmart_stock_analysis'
```

Let's establish the connection between the SQL server and Python:

```
2]: # Let's build a connection string

    conn_url = f'DRIVER={driver};Server={server};Port={port}'
    conn = pyodbc.connect(conn_url)

3]: curs = conn.cursor()

4]: use_db = f"use {db_name}"
    curs.execute(use_db)
    curs.commit()

5]: curs.execute(
    """select
    * from
    walmart_stock
    """)

    query_results = curs.fetchall()
```

Next creating sparkcontext and also creating RDD for taking the data in memory :

-- PySpark --

```
[8]: # Now create spark context

     sc = pyspark.SparkContext()
     sc
```

t[8]: **SparkContext**

Spark UI

Version
`v3.3.2`
Master
`local[*]`
AppName
`pyspark-shell`

```
[9]: walmartrdd = sc.parallelize(query_results)
```

Let's solve the scenarios with the help of RDD:

-- RDD --

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

```
walmartrdd2 = walmartrdd.map(lambda line:(line[0],int(line[4])))\
.filter(lambda item:float(item[1]) < 60)\
.map(lambda x: (x[0],1))\
.count()
```

```
print(f"{walmartrdd2} days was the close lower than 60 dollars.")
```

```
81 days was the close lower than 60 dollars.
```

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

[13]:
```
walmartrdd3 = (walmartrdd.map(lambda line:(line[0],int(line[2])))\
.filter(lambda item:float(item[1]) > 80)\
.map(lambda x: (x[0],1))\
.count())/walmartrdd.count()*100
```

[14]:
```
print(f"{round(walmartrdd3, 2)} % of the time was the high greater than 80 dollars.")
```

```
8.43 % of the time was the high greater than 80 dollars.
```

Scenario 9: What is the max High per year?

[15]:
```
walmartrdd4 = walmartrdd.map(lambda x: (int(x[0].split('-')[0]), x[2]))\
.reduceByKey(lambda a, b: round(a,2) if a>b else round(b,2))
```

[16]:
```
print("Max High Per Year: %s"%walmartrdd4.collect())
```

```
Max High Per Year: [(2016, 75.19), (2012, 77.6), (2013, 81.37), (2014, 88.09), (2015, 90.97)]
```

Let's create a spark session and solve the problem with the Spark SQL and DSL language:

-- DSL & SparkSQL --

[17]:
```
ss = SparkSession.builder.appName("project").getOrCreate()
```

[18]:
```
ss
```

Out[18]:

SparkSession - in-memory

SparkContext

Spark UI

Version

`v3.3.2`

Master

`local[*]`

AppName

`pyspark-shell`

[19]:
```
# Let's define schema

data_schema = [StructField('Date',DateType(),True),
               StructField('Open',FloatType(),True),
               StructField('High',FloatType(),True),
               StructField('Low',FloatType(),True),
               StructField('Close',FloatType(),True),
              StructField('Volume',IntegerType(),True),
              StructField('Adj Close',FloatType(),True),]
```

[20]:
```
# Now create a structtype with the schema as field

df_schema = StructType(fields = data_schema)
```

[21]:
```
df = ss.read.csv(r'C:\Users\Futurense\Desktop\Walmart Project DE\Project documents\HIVE WALMART STOCK\walmart_stock.csv', schema
```

Walmart stock data:

```
df.show()
```

```
+----------+-----+-----+-----+-----+--------+---------+
|      Date| Open| High|  Low|Close|  Volume|Adj Close|
+----------+-----+-----+-----+-----+--------+---------+
|2012-01-03|59.97|61.06|59.87|60.33|12668800|52.619236|
|2012-01-04|60.21|60.35|59.47|59.71| 9593300|52.078476|
|2012-01-05|59.35|59.62|58.37|59.42|12768200| 51.82554|
|2012-01-06|59.42|59.45|58.87| 59.0| 8069400| 51.45922|
|2012-01-09|59.03|59.55|58.92|59.18| 6679300|51.616215|
|2012-01-10|59.43|59.71|58.98|59.04| 6907300| 51.49411|
|2012-01-11|59.06|59.53|59.04| 59.4| 6365600|51.808098|
|2012-01-12|59.79| 60.0| 59.4| 59.5| 7236400|51.895317|
|2012-01-13|59.18|59.61|59.01|59.54| 7729300|51.930202|
|2012-01-17|59.87|60.11|59.52|59.85| 8500000| 52.20058|
|2012-01-18|59.79|60.03|59.65|60.01| 5911400| 52.34013|
|2012-01-19|59.93|60.73|59.75|60.61| 9234600|52.863445|
|2012-01-20|60.75|61.25|60.67|61.01|10378800|53.212322|
|2012-01-23|60.81|60.98|60.51|60.91| 7134100|53.125103|
|2012-01-24|60.75| 62.0|60.75|61.39| 7362800|53.543755|
|2012-01-25|61.18|61.61|61.04|61.47| 5915800| 53.61353|
|2012-01-26| 61.8|61.84|60.77|60.97| 7436200|53.177437|
|2012-01-27|60.86|61.12|60.54|60.71| 6287300|52.950665|
|2012-01-30|60.47|61.32|60.35| 61.3| 7636900|53.465256|
|2012-01-31|61.53|61.57|60.58|61.36| 9761500| 53.51759|
+----------+-----+-----+-----+-----+--------+---------+
only showing top 20 rows
```

[23]:
```
# Let's create view for doing spark-sql programm

df.createOrReplaceTempView("walmartstock")
```

[24]:
```
# Let's do the printschema

df.printSchema()
```

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

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. Pay careful attention to the datatypes that .describe() returns, we didn't cover how to do this exact formatting, but we covered something very similar.

```
df.describe().show()
```

...

```
df.describe().withColumn("Open", format_number(col("Open").cast('float'),2))\
.withColumn("High", format_number(col("High").cast('float'),2))\
.withColumn("Low", format_number(col("Low").cast('float'),2))\
.withColumn("Close", format_number(col("Close").cast('float'),2))\
.withColumn("Volume", format_number(col("Volume").cast('float'),2))\
.withColumn("Adj Close", format_number(col("Adj Close").cast('float'),2))\
.show()
```

```
+-------+--------+--------+--------+--------+-------------+---------+
|summary|    Open|    High|     Low|   Close|       Volume|Adj Close|
+-------+--------+--------+--------+--------+-------------+---------+
|  count|1,258.00|1,258.00|1,258.00|1,258.00|     1,258.00| 1,258.00|
|   mean|   72.36|   72.84|   71.92|   72.39| 8,222,093.50|    67.24|
| stddev|    6.77|    6.77|    6.74|    6.76| 4,519,781.00|     6.72|
|    min|   56.39|   57.06|   56.30|   56.42| 2,094,900.00|    50.36|
|    max|   90.80|   90.97|   89.25|   90.47|80,898,096.00|    84.91|
+-------+--------+--------+--------+--------+-------------+---------+
```

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

```
# Spark - SQL Technique

ss.sql("select *, round(cast(Volume as float)/cast(High as float),2) as HV_Ratio from walmartstock").show()
```

```
+----------+-----+-----+-----+-----+--------+---------+---------+
|      Date| Open| High|  Low|Close|  Volume|Adj Close| HV_Ratio|
+----------+-----+-----+-----+-----+--------+---------+---------+
|2012-01-03|59.97|61.06|59.87|60.33|12668800|52.619236|207481.16|
|2012-01-04|60.21|60.35|59.47|59.71| 9593300|52.078476|158961.06|
|2012-01-05|59.35|59.62|58.37|59.42|12768200| 51.82554|214159.68|
|2012-01-06|59.42|59.45|58.87| 59.0| 8069400| 51.45922|135734.23|
|2012-01-09|59.03|59.55|58.92|59.18| 6679300|51.616215|112162.89|
|2012-01-10|59.43|59.71|58.98|59.04| 6907300| 51.49411|115680.79|
|2012-01-11|59.06|59.53|59.04| 59.4| 6365600|51.808098|106930.96|
|2012-01-12|59.79| 60.0| 59.4| 59.5| 7236400|51.895317|120606.67|
```

```
# DSL Technique

df.withColumn('HV Ratio',round(lit(col('Volume').cast('float')/col('High').cast('float')),2)).show()
```

```
+----------+-----+-----+-----+-----+--------+---------+---------+
|      Date| Open| High|  Low|Close|  Volume|Adj Close| HV Ratio|
+----------+-----+-----+-----+-----+--------+---------+---------+
|2012-01-03|59.97|61.06|59.87|60.33|12668800|52.619236|207481.16|
|2012-01-04|60.21|60.35|59.47|59.71| 9593300|52.078476|158961.06|
|2012-01-05|59.35|59.62|58.37|59.42|12768200| 51.82554|214159.68|
|2012-01-06|59.42|59.45|58.87| 59.0| 8069400| 51.45922|135734.23|
|2012-01-09|59.03|59.55|58.92|59.18| 6679300|51.616215|112162.89|
|2012-01-10|59.43|59.71|58.98|59.04| 6907300| 51.49411|115680.79|
|2012-01-11|59.06|59.53|59.04| 59.4| 6365600|51.808098|106930.96|
|2012-01-12|59.79| 60.0| 59.4| 59.5| 7236400|51.895317|120606.67|
```

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

[28]:
```
# Spark-SQL techinique

ss.sql("select Date, High from walmartstock order by High desc limit 1").show()
```

```
+----------+-----+
|      Date| High|
+----------+-----+
|2015-01-13|90.97|
+----------+-----+
```

[29]:
```
# DSL

df.orderBy(desc('High')).select("Date", "High").limit(1).show()
```

```
+----------+-----+
|      Date| High|
+----------+-----+
|2015-01-13|90.97|
+----------+-----+
```

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

:
```
from pyspark.sql.functions import round
```

:
```
# Spark - SQL Technique

ss.sql("select round(avg(Close),2) as mean_close from walmartstock").show()
```

```
+----------+
|mean_close|
+----------+
|     72.39|
+----------+
```

:
```
# DSL Technique

df.select(round(mean(df.Close),2).alias("Mean_Close")).show()
```

```
+----------+
|Mean_Close|
+----------+
|     72.39|
+----------+
```

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

```
# Spark - Sql Technique

ss.sql("select round(max(Volume)) as max_volume, round(min(Volume)) as min_volume from walmartstock").show()
```

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

```
# DSL Technique

df.select(round(max(df.Volume),2).alias("Max_Volume"), round(min(df.Volume),2).alias("Min_Volume")).show()
```

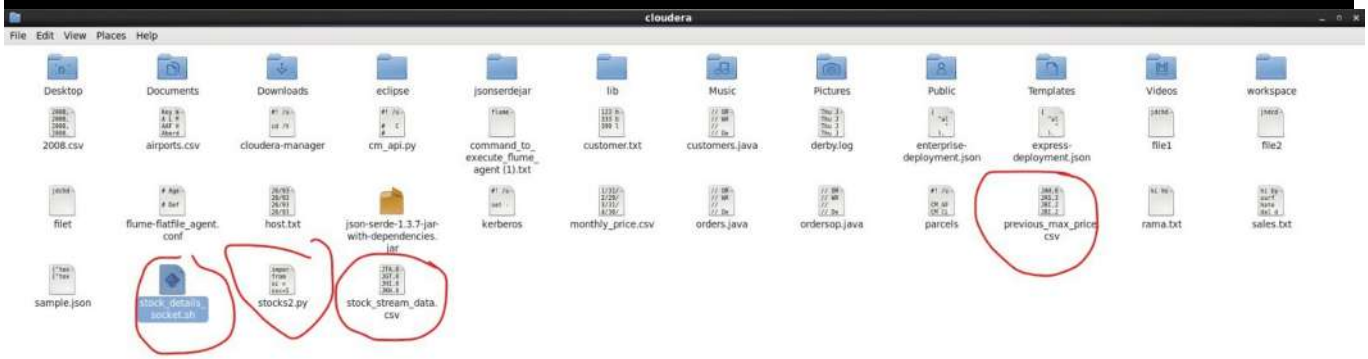
```
+----------+----------+
|Max_Volume|Min_Volume|
+----------+----------+
|  80898100|   2094900|
+----------+----------+
```

Spark Streaming analysis

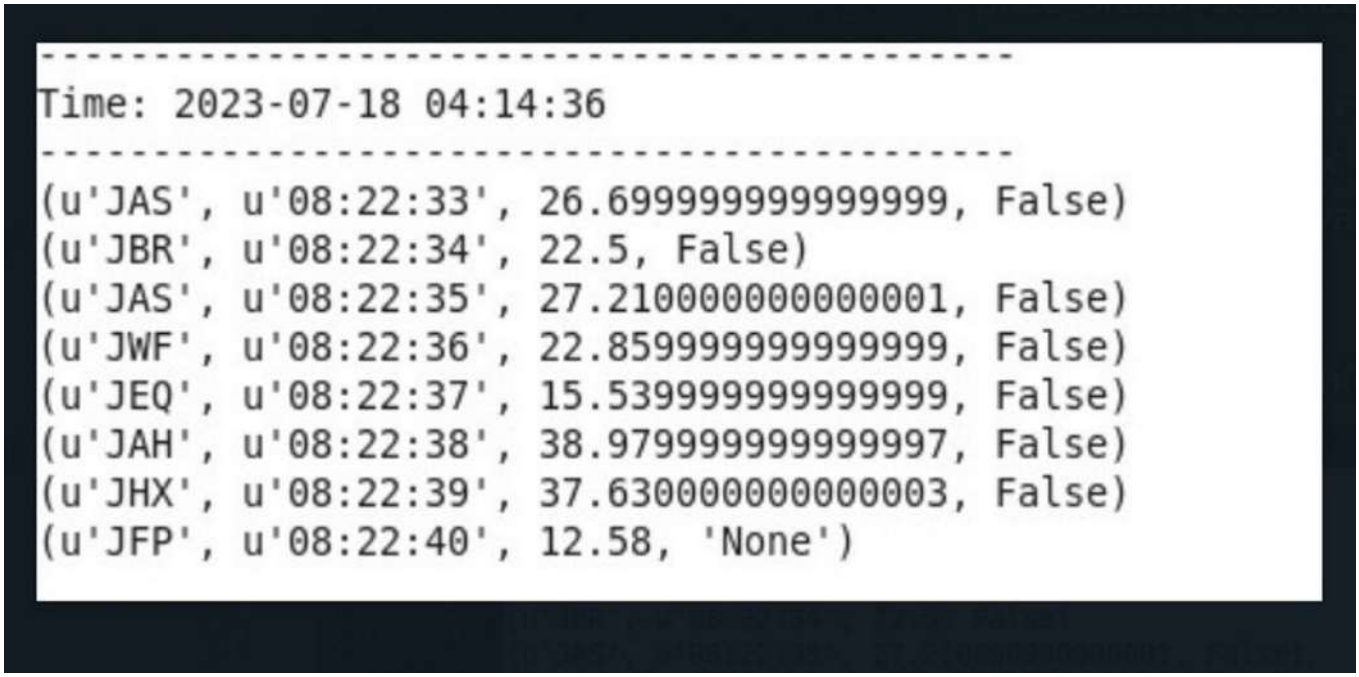
Code:

```
import pyspark
from pyspark.streaming import StreamingContext
sc = pyspark.SparkContext('local[2]','StreamingWordCount')
ssc=StreamingContext(sc,4)
sts=ssc.socketTextStream('localhost',9999)
price=sc.textFile("/user/cloudera/previous_max_price.csv")
p1=price.map(lambda x:x.split(',')).map(lambda x:(x[0],float(x[1])))
p1.take(2)
p2=p1.collectAsMap()
price_broad=sc.broadcast(p2)

r1=sts.map(lambda x:x.split(',')).map(lambda x:(x[0],x[1],float(x[2])))
r2=r1.map(lambda x: (x[0],x[1],x[2],x[2]>=price_broad.value.get(x[0]) if x[0] in price_broad.value else 'None')
)
r2.pprint()
ssc.start()
ssc.awaitTermination()
```



Output:



Conclusion

In conclusion, the stock analysis project serves as a valuable tool for individuals who aim to earn from the stock market but may lack the capacity for in-depth analysis. With its customizable dashboard and market-ready insights, it empowers common investors to confidently navigate the stock market and make informed investment decisions, even without extensive knowledge of the market. By providing accessible and user-friendly analytics, this project opens doors for individuals to participate in the stock market with ease and potential success.