

Project 1 Stock Exchange Data Analysis

DESCRIPTION

Objective: To use hive features for data engineering or analysis and sharing the actionable insights

Problem Statement:

New York stock exchange data of seven years, between 2010 to 2016, is captured for 500+ listed companies. The data set comprises of intra-day prices and volume traded for each listed company. The data serves both for machine learning and exploratory analysis projects, to automate the trading process and to predict the next trading-day winners or losers.. The scope of this project is limited to exploratory data analysis.

Domain: BFSI

Analysis to be done: Exploratory analysis to understand how MoM or YoY companies from different sectors or industries and states have progressed in a period of 7 years

Content: This data set contains prices.csv and securities.csv files having the following features:

Prices.csv:

1. Date: Trading date
2. Symbol: Ticker code or listed company code on NY stock exchange
3. Open: Intra-day opening price for each listed company
4. Close: Intra-day closing price for each listed company
5. Low: Intra-day lowest price for each listed company
6. High: Intra-day highest price for each listed company
7. Volume: Number of shares traded per day per company

Securities.csv:

1. Ticker_Symbol: Country to which the customer belongs
2. Security: Legal name of the listed company
3. Sector: Business vertical of the listed company
4. Sub_Industry: Business domain of the listed company within a Sector.
5. Headquarter: Headquarters address

Steps to perform:

- 1) Create a data pipeline using sqoop to pull the data from the table below from MYSQL server into Hive.
 - a. MYSQL DATABASE NAME: BDHS_PROJECT
 - i. Stock_prices
 - ii. Stock_companies

Check the TABLE description: STOCK_PRICES

Column Name	Datatype
Trading_date	Date
Symbol	String
Open	double
Close	double
Low	double
High	double
Volume	int

TABLE: STOCK_COMPANIES

Column Name	Datatype
Symbol	String

Company_name	String
Sector	String
Sub_industry	String
Headquarter	String

2) Create a new hive table with the following fields by joining the above two hive tables.
Please use appropriate Hive built-in functions for columns (a,b,e and h to l).

- Trading_year: Should contain YYYY for each record
- Trading_month: Should contain MM or MMM for each record
- Symbol: Ticker code
- CompanyName: Legal name of the listed company
- State: State to be extracted from headquarters value.
- Sector: Business vertical of the listed company
- Sub_Industry: Business domain of the listed company within a sector
- Open: Average of intra-day opening price by month and year for each listed company
- Close: Average of intra-day closing price by month and year for each listed company
- Low: Average of intra-day lowest price by month and year for each listed company
- High: Average of intra-day highest price by month and year for each listed company
- Volume: Average of number of shares traded by month and year for each listed company

Project Resolution

Load mysql data tables into Hive using Sqoop.

Load the STOCK_COMPANIES table

```
sqoop import --connect jdbc:mysql://ip-10-0-1-10.ec2.internal/BDHS_PROJECT --username labuser --password simplilearn --table STOCK_COMPANIES --hive-import --hive-table sg_TestHive.stock_companies -m 1;
```

Load the STOCK_PRICES table

```
sqoop import --connect jdbc:mysql://ip-10-0-1-10.ec2.internal/BDHS_PROJECT --username labuser --password simplilearn --table STOCK_PRICES --hive-import --hive-table sg_TestHive.stock_prices -m 1;
```

Direct option could be used however this ran quickly.

Create new Hive table with the joined data of the above 2 tables

--Open: Average of intra-day opening price by month and year for each listed company

--Close: Average of intra-day closing price by month and year for each listed company

--Low: Average of intra-day lowest price by month and year for each listed company

--High: Average of intra-day highest price by month and year for each listed company

--Volume: Average of number of shares traded by month and year for each listed company

create table stock_analysis_data

as

SELECT

```
    year(sp.trading_date)          trading_year
, month(sp.trading_date)          trading_month
, sc.ticker_symbol ticker_code
, sc.securityCompany_name
, substr(sc.headquarter, instr(sc.headquarter, ';')+1) state
, sc.sector
, sc.sub_industry
, avg(sp.open)   avg_opening_price
, avg(sp.close)  avg_closing_price
, avg(sp.low)    avg_low_price
, avg(sp.high)   avg_highest_price
, avg(sp.volumn) avg_volume
```

from

stock_companiessc

, stock_pricessp

where

sc.ticker_symbol = sp.symbol

group by

```
    year(sp.trading_date)
, month(sp.trading_date)
, sc.ticker_symbol
, sc.security
, substr(sc.headquarter, instr(sc.headquarter, ';')+1)
, sc.sector
, sc.sub_industry
```

;

You are accessing a non-optimized Hue, please switch to one of the available addresses: <http://ip-10-0-1-10.ec2.internal:8889>, <http://ip-10-0-1-11>.

Hue Query Search data and saved documents...

Hive Add a name... Add a description...

29.54s sg_testhive text ?

```

10 | year(sp.trading_date)          trading_year
11 | month(sp.trading_date)        trading_month
12 | sc.ticker_symbol              ticker_code
13 | sc.security                   Company_name
14 | substr(sc.headquarter, instr(sc.headquarter, ';')+1) state
15 | sc.sector
16 | sc.sub_industry
17 | avg(sp.open)                  avg_opening_price
18 | avg(sp.close)                 avg_closing_price
19 | avg(sp.low)                   avg_low_price
20 | avg(sp.high)                  avg_highest_price
21 | avg(sp.volume)                avg_volume
22 from
23 | stock_companies sc
24 | , stock_prices sp
25 where
26 | sc.ticker_symbol = sp.symbol
27 group by
28 | year(sp.trading_date)
29 | month(sp.trading_date)
30 | sc.ticker_symbol
31 | sc.security
32 | substr(sc.headquarter, instr(sc.headquarter, ';')+1)
33 | sc.sector
34 | sc.sub_industry

```

INFO : MapReduce Jobs Launched:
 INFO : Stage-Stage-2: Map: 2 Reduce: 2 Cumulative CPU: 29.42 sec HDFS Read: 84718801 HDFS Write: 6633315 SUCCESS [job_1588139849525_6900](#)
 INFO : Total MapReduce CPU Time Spent: 29 seconds 420 msec
 INFO : Completed executing command(queryId=hive_20200521174141_b605bb89-9363-437d-9499-b289bebea82a); Time taken: 26.607 seconds
 INFO : OK

✓ Success.

Query History Saved Queries

2 minutes ago create table stock_analysis_data as select year(sp.trading_date) trading_year , month(sp.trading_date) trading_month , sc.ticker_symbol ticker_code , sc.security Company_name , substr(sc.headquarter, instr(sc.headquarter, ';')+1) state , sc.sector

Analysis on the loaded data

The top five companies that are good for investment

The rational is that the companies with highest average growth i.e. avg. closing price minus avg. opening price, are the ones which are good for investment

```

SELECT op.company_name from (
SELECT gr.company_name,
rank() OVER(
ORDER BY gr.growth DESC) overall_rank
FROM
(SELECT company_name,
max(avg_closing_price - avg_opening_price) growth
FROM stock_analysis_data
GROUP BY company_name) gr
) op
where
overall_rank< 6

```

You are accessing a non-optimized Hue, please switch to one of the available addresses: <http://ip-10-0-1-10.ec2.internal:8889>, <http://ip-10-0-1-11.4>

Hive Query Search data and saved documents...

Add a name... Add a description...

32.20s sg_testhive text ?

```

1 select op.company_name from (
2 SELECT gr.company_name,
3       rank() OVER(
4         ORDER BY gr.growth DESC) overall_rank
5 FROM
6   (SELECT company_name,
7         max(avg_closing_price - avg_opening_price) growth
8    FROM stock_analysis_data
9   GROUP BY company_name) gr
10 ) op
11 where
12 overall_rank < 6

```

INFO : Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 9.35 sec HDFS Read: 6648860 HDFS Write: 21570 SUCCESS
 INFO : Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.34 sec HDFS Read: 29183 HDFS Write: 90 SUCCESS
 INFO : Total MapReduce CPU Time Spent: 13 seconds 690 msec
 INFO : Completed executing command(queryId=hive_20200521191212_80ce6bda-9780-43a8-8ece-147baae7581d); Time taken: 30.54 seconds
 INFO : OK

Query History Saved Queries Results (5)

op.company_name
1 Priceline.com Inc
2 Chipotle Mexican Grill
3 Regeneron
4 Intuitive Surgical Inc.
5 Amazon.com Inc

The best-growing industry by each state, having at least two or more industries mapped.

```

select sa2.state, sa2.sector
from
(
  select sa1.state, sa1.sector, rank() over (partition by sa1.state order by (sa1.avg_closing_price -
sa1.avg_opening_price)) rank_in_state
  from
stock_analysis_data sa1
  where sa1.state in
  (
    select substr(sc.headquarter, instr(sc.headquarter,',')+1) state
    from
stock_companiessc
    group by substr(sc.headquarter, instr(sc.headquarter,',')+1)
    having count(sc.sector) >= 2
  )
) sa2
where
sa2.rank_in_state = 1

```


Hive Query interface showing a query to find the worst year for each sector.

```

1 -- For each sector find the following.
2 -- Worst year e.g.Consumer Disc 2011
3 -- Best year
4 -- Stable year
5
6 select wy.sector, wy.trading_year worst_year
7 from
8 (
9 select sa.sector, sa.trading_year, rank() over (partition by sa.sector order by sa.avg_low_price) rank_in_sector
10 from
11 stock_analysis_data sa
12 ) wy
13 where
14 wy.rank_in_sector = 1

```

INFO : MapReduce Jobs Launched:
 INFO : Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 11.26 sec HDFS Read: 6649123 HDFS Write: 220 SUCCESS
 INFO : Total MapReduce CPU Time Spent: 11 seconds 260 msec
 INFO : Completed executing command(queryId=hive_20200521200404_126b9564-604c-4f00-8db0-046c0c4656dc); Time taken: 15.684 seconds
 INFO : OK

Query History Saved Queries Results (11)

wy.sector	worst_year
1 Consumer Discretionary	2011
2 Consumer Staples	2010
3 Energy	2016
4 Financials	2011
5 Health Care	2011
6 Industrials	2011
7 Information Technology	2011
8 Materials	2011

Best year

Got the year with the maximum average high price for a given sector

select wy.sector, wy.trading_year best_year

from

(

select sa.sector, sa.trading_year, rank() over (partition by sa.sector order by sa.avg_highest_price desc)

rank_in_sector

from

stock_analysis_data sa

) wy

where

wy.rank_in_sector = 1

```

2 from
3 (
4 select sa.sector, sa.trading_year, rank() over (partition by sa.sector order by sa.avg_highest_price desc) rank_in_sector
5 from
6 stock_analysis_data sa
7 ) wy
8 where
9 wy.rank_in_sector = 1

```

INFO : MapReduce Jobs Launched:
 INFO : Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 10.8 sec HDFS Read: 6649131 HDFS Write: 220 SUCCESS
 INFO : Total MapReduce CPU Time Spent: 10 seconds 800 msec
 INFO : Completed executing command(queryId=hive_20200521200606_b0fc4524-c021-406b-a691-2bf2ded33fad); Time taken: 14.692 seconds
 INFO : OK

Query History Saved Queries Results (11)

wy.sector	worst_year
1 Consumer Discretionary	2016
2 Consumer Staples	2016
3 Energy	2014
4 Financials	2016
5 Health Care	2016
6 Industrials	2016
7 Information Technology	2016
8 Materials	2016

Stable year

An year with minimum difference between the average high price and average low price would be called as a stable year

```
select wy.sector, wy.trading_year most_stable_year
from
(
select sa.sector, sa.trading_year, rank() over (partition by sa.sector order by (sa.avg_highest_price -
sa.avg_low_price)) rank_in_sector
from
stock_analysis_data sa
) wy
where
wy.rank_in_sector = 1
```

The screenshot shows the Hive Query Editor interface. At the top, there is a search bar and a 'Query' button. Below the search bar, there are tabs for 'Query History', 'Saved Queries', and 'Results (11)'. The 'Results (11)' tab is active, displaying a table with two columns: 'wy.sector' and 'mst_stable_year'. The table contains 11 rows of data, representing different sectors and their most stable years. Below the table, there is a section for 'Query History' and 'Saved Queries'. The 'Query History' section shows a list of queries with their execution times and status. The 'Saved Queries' section shows a list of saved queries with their names and descriptions.

```
1 select wy.sector, wy.trading_year most_stable_year
2 from
3 (
4 select sa.sector, sa.trading_year, rank() over (partition by sa.sector order by (sa.avg_highest_price - sa.avg_low_price)) rank_in_sector
5 from
6 stock_analysis_data sa
7 ) wy
8 where
9 wy.rank_in_sector = 1
10
```

INFO : MapReduce Jobs Launched:
INFO : Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 13.96 sec HDFS Read: 6651381 HDFS Write: 220 SUCCESS
INFO : Total MapReduce CPU Time Spent: 13 seconds 960 msec
INFO : Completed executing command(queryId=hive_20200521201717_Sedca214-6a1f-4cad-881f-32f6813f0a46); Time taken: 16.715 seconds
INFO : OK

wy.sector	mst_stable_year
1 Consumer Discretionary	2012
2 Consumer Staples	2010
3 Energy	2010
4 Financials	2012
5 Health Care	2012
6 Industrials	2013
7 Information Technology	2013
8 Materials	2013

<End of Project>