DS8003 – Project Codes/Commands

AAPL (Apple) Stock Data

~ Group Members ~

Andrey Zhuravlev

Khushnud Ahmed

Momna Ali

Steven Luc

Table of Contents

HDFS and MapReducer

Upload the mapper and reducer python file into the Hadoop ecosystem

```
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put mapper_preprocess.py /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put reducer_preprocess.py /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put aapl_raw_data.csv /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put aapl_split_adjusted.csv /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -ls /user/root/aapl_stock_data
Found 4 items
-rw-r--r--   1 root root     813682 2024-11-23 08:14 /user/root/aapl_stock_data/aapl_raw_data.csv
-rw-r--r--   1 root root     968073 2024-11-23 08:15 /user/root/aapl_stock_data/aapl_split_adjusted.csv
-rw-r--r--   1 root root        408 2024-11-23 08:12 /user/root/aapl_stock_data/mapper_preprocess.py
-rw-r--r--   1 root root        110 2024-11-23 08:13 /user/root/aapl_stock_data/reducer_preprocess.py
```

Run the MapReducer command to handle any missing values by replacing them with appropriate defaults or using statistical measures.

```
[root@sandbox-hdp aapl_stock_data]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -files /root/aapl_stock_data/mapper_preproce
ss.py,/root/aapl_stock_data/reducer_preprocess.py -mapper mapper_preprocess.py -reducer reducer_preprocess.py -input /user/root/aapl_stock_data/aapl_raw_data.csv -outpu
t /user/root/aapl_stock_data/preprocessed_data
packageJobJar: [] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob8319286531481887335.jar tmpDir=null
24/11/23 08:24:54 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
24/11/23 08:24:54 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
24/11/23 08:24:54 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
24/11/23 08:24:54 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
24/11/23 08:24:55 INFO mapred.FileInputFormat: Total input paths to process : 1
24/11/23 08:24:55 INFO mapreduce.JobSubmitter: number of splits:2
24/11/23 08:24:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1732334798004_0011
24/11/23 08:24:56 INFO impl.YarnClientImpl: Submitted application application_1732334798004_0011
24/11/23 08:24:56 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1732334798004_0011/
24/11/23 08:24:56 INFO mapreduce.Job: Running job: job_1732334798004_0011
24/11/23 08:25:04 INFO mapreduce.Job: Job job_1732334798004_0011 running in uber mode : false
24/11/23 08:25:04 INFO mapreduce.Job:  map 0% reduce 0%
24/11/23 08:25:09 INFO mapreduce.Job:  map 100% reduce 0%
24/11/23 08:25:15 INFO mapreduce.Job:  map 100% reduce 100%
24/11/23 08:25:15 INFO mapreduce.Job: Job job_1732334798004_0011 completed successfully
24/11/23 08:25:15 INFO mapreduce.Job: Counters: 49

24/11/23 08:25:15 INFO streaming.StreamJob: Output directory: /user/root/aapl_stock_data/preprocessed_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -ls /user/root/aapl_stock_data/preprocessed_data
Found 2 items
-rw-r--r--   1 root root          0 2024-11-23 08:25 /user/root/aapl_stock_data/preprocessed_data/_SUCCESS
-rw-r--r--   1 root root     824668 2024-11-23 08:25 /user/root/aapl_stock_data/preprocessed_data/part-00000
```

Upload the second mapper and reducer python file that will analyze how stock splits have influenced stock pricing over the years.

```
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put mapper_compare_prices.py /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put reducer_compare_prices.py /user/root/aapl_stock_data
```

Run the MapReducer command again to obtain the output.

```
[root@sandbox-hdp aapl_stock_data]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -files /root/aapl_stock_data/mapper_compare_
prices.py,/root/aapl_stock_data/reducer_compare_prices.py -mapper mapper_compare_prices.py -reducer reducer_compare_prices.py -input /user/root/aapl_stock_data/preproce
ssed_data -output /user/root/aapl_stock_data/price_comparison
packageJobJar: [] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob7182087150932074464.jar tmpDir=null
24/11/23 09:30:55 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
24/11/23 09:30:55 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
24/11/23 09:30:55 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
24/11/23 09:30:55 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
24/11/23 09:30:56 INFO mapred.FileInputFormat: Total input paths to process : 1
24/11/23 09:30:57 INFO mapreduce.JobSubmitter: number of splits:2
24/11/23 09:30:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1732334798004_0017
24/11/23 09:30:58 INFO impl.YarnClientImpl: Submitted application application_1732334798004_0017
24/11/23 09:30:58 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1732334798004_0017/
24/11/23 09:30:58 INFO mapreduce.Job: Running job: job_1732334798004_0017
24/11/23 09:31:04 INFO mapreduce.Job: Job job_1732334798004_0017 running in uber mode : false
24/11/23 09:31:04 INFO mapreduce.Job:  map 0% reduce 0%
24/11/23 09:31:09 INFO mapreduce.Job:  map 100% reduce 0%
24/11/23 09:31:16 INFO mapreduce.Job:  map 100% reduce 100%
24/11/23 09:31:17 INFO mapreduce.Job: Job job_1732334798004_0017 completed successfully
24/11/23 09:31:17 INFO mapreduce.Job: Counters: 49

24/11/23 09:31:17 INFO streaming.StreamJob: Output directory: /user/root/aapl_stock_data/price_comparison
```

Finally upload the mapper and reducer python file required for calculating the yearly average and run the MapReduce task.

```
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put mapper_yearly_avg.py /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop fs -put reducer_yearly_avg.py /user/root/aapl_stock_data
[root@sandbox-hdp aapl_stock_data]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -files /root/aapl_stock_data/mapper_yearly_a
vg.py,/root/aapl_stock_data/reducer_yearly_avg.py -mapper mapper_yearly_avg.py -reducer reducer_yearly_avg.py -input /user/root/aapl_stock_data/preprocessed_data -outpu
t /user/root/aapl_stock_data/yearly_avg_price
packageJobJar: [] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob8344695766408210351.jar tmpDir=null
24/11/23 09:39:55 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
24/11/23 09:39:55 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
24/11/23 09:39:55 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
24/11/23 09:39:55 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
24/11/23 09:39:56 INFO mapred.FileInputFormat: Total input paths to process : 1
24/11/23 09:39:57 INFO mapreduce.JobSubmitter: number of splits:2
24/11/23 09:39:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1732334798004_0018
24/11/23 09:39:57 INFO impl.YarnClientImpl: Submitted application application_1732334798004_0018
24/11/23 09:39:57 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1732334798004_0018/
24/11/23 09:39:57 INFO mapreduce.Job: Running job: job_1732334798004_0018
24/11/23 09:40:04 INFO mapreduce.Job: Job job_1732334798004_0018 running in uber mode : false
24/11/23 09:40:04 INFO mapreduce.Job:  map 0% reduce 0%
24/11/23 09:40:10 INFO mapreduce.Job:  map 100% reduce 0%
24/11/23 09:40:18 INFO mapreduce.Job:  map 100% reduce 100%
24/11/23 09:40:20 INFO mapreduce.Job: Job job_1732334798004_0018 completed successfully
24/11/23 09:40:20 INFO mapreduce.Job: Counters: 49
```

Once all MapReduce tasks have been successfully executed. Export output files to local machine as csv file using the command:

```
[root@sandbox-hdp aapl_stock_data]# hdfs dfs -cat /user/root/aapl_stock_data/yearly_avg_price/part-00000 > yearly_avg_price_combined.csv
```

Repeat same command for the rest of the MapReducer file outputs.

Generate visuals using jupyter notebook.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the output data
price_comparison = pd.read_csv('price_comparison.csv', delimiter='\t', names=['date', 'ratio'])
price_comparison['date'] = pd.to_datetime(price_comparison['date'])

# Plot
plt.figure(figsize=(12, 6))
plt.plot(price_comparison['date'], price_comparison['ratio'], label='Raw/Adjusted Close Ratio', color='blue')
plt.axhline(y=1, color='red', linestyle='--', label='Ratio = 1')
plt.xlabel('Date')
plt.ylabel('Ratio')
plt.title('Comparison of Raw and Adjusted Closing Prices')
plt.legend()
plt.show()
```

```python
# Load yearly average data
yearly_avg = pd.read_csv('yearly_avg_price.csv', delimiter='\t', names=['year', 'avg_close'])
yearly_avg['year'] = yearly_avg['year'].astype(int)

# Plot
plt.figure(figsize=(12, 6))
plt.plot(yearly_avg['year'], yearly_avg['avg_close'], marker='o', color='green')
plt.xlabel('Year')
plt.ylabel('Average Closing Price (USD)')
plt.title('Yearly Average Closing Price of AAPL')
plt.grid(True)
plt.show()
```

ElasticSearch and Kibana

Go to elastic.co and open sign in to your account.

Setup your deployment and go to overview once a project deployment is made.

Ingest your csv files so that you can use elasticsearch queries on them.

Once all files have been successfully uploaded into the elasticsearch, click on the dev tools to begin.

1. Query to find the top 5 highest trading days.

```
GET aapl_raw/_search
{
    "size": 5,
    "sort": [
        { "volume": { "order": "desc" } }
    ],
    "_source": ["date", "volume", "close"]
}
```

2. Query to calculate the average closing price for 2023.

```
GET aapl_raw/_search
{
    "query": {
        "range": {
            "date": { "gte": "2023-01-01", "lte": "2023-12-31" }
        }
    },
    "aggs": {
        "average_close": { "avg": { "field": "close" } }
    },
    "size": 0
}
```

3. Query to calculate the average closing price for 1982.

```
GET aapl_raw/_search
{
    "query": {
        "range": {
            "date": { "gte": "1982-01-01", "lte": "1982-12-31" }
        }
    },
    "aggs": {
        "average_close": { "avg": { "field": "close" } }
    },
    "size": 0
}
```

4. Query to find the highest trading days.

```
GET aapl_raw/_search
{
    "size": 5,
    "sort": [
        { "volume": { "order": "desc" } }
    ],
    "_source": ["date", "volume", "close"]
}
```

5. Query to find the most volatile days.

```
GET aapl_raw/_search
{
    "query": {
        "script_score": {
            "query": { "match_all": {} },
            "script": {
                "source": "Math.abs((doc['close'].value - doc
                    ['open'].value) / doc['open'].value) * 100"
            }
        }
    },
    "_source": ["date", "open", "close"]
}
```

6. Query to get the distribution of price ranges.

```
GET aapl_raw/_search
{
    "aggs": {
        "price_range": {
            "range": {
                "field": "close",
                "ranges": [
                    { "to": 50 },
                    { "from": 50, "to": 150 },
                    { "from": 150, "to": 300 },
                    { "from": 300 }
                ]
            }
        }
    },
    "size": 0
}
```

After getting results from the queries, go to kibana dashboards to create visual insights.

Hive

Data Source 1: Kaggle

Datasets:

aapl_split_adjusted.csv - contains Open, High, Low, Close, and Volume (OHLCV) data for Apple stock for every trading day since Apple IPO in 1980 adjusted for stock splits.

Data Source 2: EDGAR (Electronic Data Gathering, Analysis, and Retrieval) - database system operated by SEC (the U.S. Securities and Exchange Commission).

Datasets:

aapl_EDGAR_10-K_and_10-Q - contains dates for all 10-K and 10-Q filings by Apple since the introduction of EDGAR in 1994.

10-Q is a quarterly report filed for each of the first three quarters of the fiscal year, 10-K is an annual report. The fourth quarter is usually covered in the annual report.

aapl_EDGAR_all_filings.csv - contains dates of all the filings by Apple with SEC since the introduction of EDGAR in 1994.

Data Source 3: Wikipedia (https://en.wikipedia.org/wiki/List_of_Apple_products)

Dataset:

Apple_products_release_dates.csv - contains release dates for all computers, phones, tablets, wearables, and other products made by Apple Inc along with the family types for those products.

(code that I used to parse and clean the dataset from wikipedia article)

```
url = "https://en.wikipedia.org/wiki/List_of_Apple_products"
response = requests.get(url)
soup = BeautifulSoup(response.content, features: 'html.parser')

# Find all tables with the 'wikitable' class
tables = soup.find_all( name: 'table', attrs: {'class': 'wikitable'})


combined_data = []

# function to convert date strings to 'YYYY-MM-DD' format
2 usages
def parse_date(date_str):
    try:
        return datetime.strptime(date_str, __format: '%B %d, %Y').strftime('%Y-%m-%d')
    except (ValueError, TypeError):
        return None  # Return None for invalid or missing dates


for table in tables[:-1]: # exclude the last table as it's unrelated
    df = pd.read_html(str(table))[0]
    df.columns = df.columns.str.strip()

    # preprocess the 'Released' and 'Discontinued' columns
    if 'Released' in df.columns:
        df['Released'] = df['Released'].apply(parse_date)
    if 'Discontinued' in df.columns:
        df['Discontinued'] = df['Discontinued'].apply(parse_date)

    # append the processed DataFrame to the combined list
    combined_data.append(df)

combined_table = pd.concat(combined_data, ignore_index=True)
combined_table.to_csv( path_or_buf: 'apple_products_release_dates.csv', index=False)
```

Data Loading and Preprocessing

First, let's load the main dataset into hive. I am going to go with aapl_split_adjusted.csv for analysis to account for stock splits and ensuring historical comparability):

```
CREATE TABLE apple_stock_data (
 `date` DATE,
 open FLOAT,
 high FLOAT,
 low FLOAT,
 close FLOAT,
 volume BIGINT,
 raw_close FLOAT,
 change_percent FLOAT,
 avg_vol_20d FLOAT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
TBLPROPERTIES ("skip.header.line.count"="1");

LOAD DATA INPATH '/user/root/project/aapl_split_adjusted.csv' INTO TABLE apple_stock_data;
```

```
    > LOAD DATA INPATH '/user/root/project/aapl_split_adjusted.csv' INTO TABLE apple_stock_data;
Loading data to table default.apple_stock_data
Table default.apple_stock_data stats: [numFiles=4, numRows=0, totalSize=3872292, rawDataSize=0]
OK
Time taken: 1.2 seconds
hive> CREATE TABLE sec_filings_10k_10q (
    >   form_type STRING,
    >   form_description STRING,
    >   filing_date DATE,
    >   reporting_date DATE
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '\t'
    > STORED AS TEXTFILE;
OK
Time taken: 0.582 seconds
```

Load datatable with quarterly and annual reports dates:

```
CREATE TABLE sec_filings_10k_10q (
 form_type STRING,
```

```
 form_description STRING,
 filing_date DATE,
 reporting_date DATE
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
STORED AS TEXTFILE
TBLPROPERTIES ("skip.header.line.count"="1");

LOAD DATA INPATH '/user/root/project/aapl_EDGAR_10-K_and_10-Q.csv' INTO TABLE sec_filings_10k_10q;
```

```
    > LOAD DATA INPATH '/user/root/project/aapl_EDGAR_10-K_and_10-Q.csv' INTO TABLE sec_filings_10k_10q;
Loading data to table default.sec_filings_10k_10q
Table default.sec_filings_10k_10q stats: [numFiles=1, numRows=0, totalSize=10195, rawDataSize=0]
OK
Time taken: 1.168 seconds
```

All SEC filings data:

```
CREATE TABLE sec_filings_all (
 form_type STRING,
 form_description STRING,
 filing_date DATE,
 reporting_date DATE
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
STORED AS TEXTFILE
TBLPROPERTIES ("skip.header.line.count"="1");

LOAD DATA INPATH '/user/root/project/aapl_EDGAR_all_filings.csv' INTO TABLE sec_filings_all;
```

Create table for product release data:

```
CREATE TABLE apple_product_releases (
 released DATE,
 model STRING,
 family STRING,
 discontinued DATE
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
 "separatorChar" = ",",  -- Fields are separated by commas
 "quoteChar" = "\""      -- Fields may be enclosed in double quotes
)
```

```
STORED AS TEXTFILE

TBLPROPERTIES ("skip.header.line.count"="1");


LOAD DATA INPATH '/user/root/project/apple_products_release_dates.csv'

INTO TABLE apple_product_releases;
```

Now all the tables are loaded:

```
SHOW TABLES;
```

```
+-------------------------+
| apple_product_releases  |
| apple_stock_data        |
| sec_filings_10k_10q     |
| sec_filings_all         |
+-------------------------+
4 rows selected (0.129 seconds)
```

Data Analysis

Impact of Product Releases on Stock Price

Assign a sequential number to each trading date to facilitate finding dates 5 trading days apart:

```
CREATE TABLE stock_data_with_index AS

SELECT

 apple_stock_data.`date`,

 apple_stock_data.open,

 apple_stock_data.close,

 ROW_NUMBER() OVER (ORDER BY apple_stock_data.`date`) AS trading_day_index

FROM

 apple_stock_data

ORDER BY

 Apple_stock_data.`date`;
```

```
10,926 rows affected (2.175 seconds)
```

Associate each release date with its trading day index and list of products released on that date:

```
CREATE TABLE release_dates_with_index AS

SELECT

 apple_product_releases.released AS release_date,

 COLLECT_SET(apple_product_releases.model) AS products,

 stock_data_with_index.trading_day_index

FROM

 apple_product_releases
```

```
JOIN
 stock_data_with_index
ON
 apple_product_releases.released = stock_data_with_index.`date`
GROUP BY
 apple_product_releases.released,
 stock_data_with_index.trading_day_index;
```

```
279 rows affected (2.217 seconds)
```

Calculate the percentage return from the open price on the release date to the close price 5 trading days later:

```
CREATE TABLE release_date_returns AS
SELECT
 release_dates_with_index.release_date,
 release_dates_with_index.products,
 stock_data_day1.open AS open_price_day1,
 stock_data_day5.close AS close_price_day5,
 ((stock_data_day5.close - stock_data_day1.open) / stock_data_day1.open) * 100 AS return_percent
FROM
 release_dates_with_index
JOIN
 stock_data_with_index AS stock_data_day1
ON
 release_dates_with_index.trading_day_index = stock_data_day1.trading_day_index
JOIN
 stock_data_with_index AS stock_data_day5
ON
 stock_data_day5.trading_day_index = release_dates_with_index.trading_day_index + 5
WHERE
 stock_data_day5.close IS NOT NULL;
```

```
279 rows affected (2.536 seconds)
```

Get top 5 best performances:

```
SELECT
 release_date,
 CONCAT_WS(', ', COLLECT_LIST(product)) as products,
 ROUND(open_price_day1, 2) as open_price,
 ROUND(close_price_day5, 2) as close_price,
```

```
 ROUND(return_percent, 2) as return_pct
FROM release_date_returns
LATERAL VIEW EXPLODE(products) products_table AS product
GROUP BY release_date, open_price_day1, close_price_day5, return_percent
ORDER BY return_pct DESC
LIMIT 5;
```

1982-10-01	Apple Dot Matrix Printer, Apple Daisy Wheel Printer	0.08	0.1	27.0
2003-05-02	iPod (3rd gen)	0.26	0.33	26.57
1999-08-31	Macintosh Server G4	0.56	0.67	19.02
1997-03-20	Twentieth Anniversary Macintosh	0.14	0.17	16.38
2006-07-13	Nike+iPod	1.86	2.16	16.28

Get top 5 worst performances:

```
SELECT
 release_date,
 products[0] as first_product,
 ROUND(open_price_day1, 2) as open_price,
 ROUND(close_price_day5, 2) as close_price,
 ROUND(return_percent, 2) as return_pct
FROM release_date_returns
ORDER BY return_percent ASC
LIMIT 5;
```

2008-01-15	MacBook Air (Early 2008)	6.35	4.97	-21.75
2008-10-14	MacBook Air (Late 2008)	4.15	3.27	-21.3
1993-06-07	PowerBook 145b	0.49	0.4	-18.13
2001-07-18	Power Mac G4 Quicksilver	0.39	0.33	-15.2
1984-01-24	Macintosh (128K)	0.13	0.11	-14.27

Impact of Product Release on Stock Price by Product Type

Now, let's look at the top 10 product categories (families) that, on average, had the best impact on the stock price and the top 10 with the worst impact for those families that pass the threshold of at least 3 different releases.

Map each release date's return to the product family:

```
CREATE TABLE family_returns_data AS
SELECT
 apple_product_releases.family,
 release_date_returns.return_percent
FROM
 release_date_returns
JOIN
```

```
 apple_product_releases
ON
 release_date_returns.release_date = apple_product_releases.released
WHERE
 apple_product_releases.released IS NOT NULL;
```

```
606 rows affected (4.019 seconds)
```

Compute the average return for each product family, considering only those with at least 3 release dates:

```
CREATE TABLE family_returns AS
SELECT
 family_returns_data.family,
 COUNT(DISTINCT apple_product_releases.released) AS release_count,
 AVG(family_returns_data.return_percent) AS avg_return
FROM
 family_returns_data
JOIN
 apple_product_releases
ON
 family_returns_data.family = apple_product_releases.family
GROUP BY
 family_returns_data.family
HAVING
 COUNT(DISTINCT apple_product_releases.released) >= 3;
```

```
46 rows affected (3.276 seconds)
```

Top 5 best performing families:

```
SELECT *
FROM family_returns
ORDER BY avg_return DESC
LIMIT 10;
```

```
+----------------------+------------------------------+--------------------------+
| 68000                | 3                            | 14.228188375874481       |
| Printers             | 9                            | 9.750150025174664        |
| AirPort Express      | 3                            | 6.79244363613175         |
| iPod Shuffle         | 5                            | 5.66147805653108         |
| Quadra               | 7                            | 4.831862819571727        |
| Display              | 7                            | 4.750477940590318        |
| PowerBook Duo        | 5                            | 4.498568057954815        |
| Macintosh Server     | 5                            | 4.4329376218376195       |
| Apple II             | 10                           | 4.285343175886791        |
| iPod Classic         | 9                            | 3.6918897736630236       |
+----------------------+------------------------------+--------------------------+
```

Top 5 worst performing families:

```
SELECT *
```

```
FROM family_returns

ORDER BY avg_return ASC

LIMIT 10;
```

```
+-----------------------+-------------------------------+----------------------------+
| Modems                | 4                             | -13.295102075972714        |
| MacBook               | 10                            | -4.5047819056035205        |
| PowerBook G3          | 4                             | -4.304842451578907         |
| PowerBook 100         | 8                             | -3.563522780730759         |
| Workgroup Server      | 5                             | -3.3391906390511212        |
| MacBook Air           | 17                            | -3.3354880299183822        |
| Keyboards             | 7                             | -2.7514171533025102        |
| Mac Mini              | 13                            | -2.237675184183968         |
| Speakers              | 9                             | -1.6654844906239905        |
| iPhone                | 30                            | -1.415226059154507         |
+-----------------------+-------------------------------+----------------------------+
```

Impact of Product Release on Stock Volatility

Now let's look at how product releases impact volatility in the first week after product releases. First, collect the change_percent values for the 5 trading days following each product release date:

```
CREATE TABLE release_date_daily_returns AS

SELECT

 release_dates_with_index.release_date,

 apple_stock_data.`date` AS stock_date,

 apple_stock_data.change_percent

FROM

 release_dates_with_index

JOIN

 apple_stock_data

ON

 apple_stock_data.`date` > release_dates_with_index.release_date

 AND apple_stock_data.`date` <= DATE_ADD(release_dates_with_index.release_date, 5);
```

```
884 rows affected (3.182 seconds)
```

Create a temporary table calculating volatility per release date:

```
CREATE TABLE release_date_volatility AS

SELECT

 release_date_daily_returns.release_date,

 STDDEV_SAMP(release_date_daily_returns.change_percent) AS volatility

FROM

 release_date_daily_returns

GROUP BY

 release_date_daily_returns.release_date;
```

```
279 rows affected (2.988 seconds)
```

Associate the calculated volatility with the products released on that date:

```
CREATE TABLE release_date_volatility_final AS

SELECT

 release_date_volatility.release_date,

 release_dates_with_index.products,

 release_date_volatility.volatility

FROM

 release_date_volatility

JOIN

 release_dates_with_index

ON

 release_date_volatility.release_date = release_dates_with_index.release_date;
```

```
279 rows affected (2.263 seconds)
```

10 highest volatility:

```
SELECT

 release_date,

 CONCAT_WS(', ', COLLECT_LIST(product)) as products,

 ROUND(volatility, 2) as volatility_pct

FROM release_date_volatility_final

LATERAL VIEW EXPLODE(products) products_table AS product

GROUP BY release_date, volatility

ORDER BY volatility_pct DESC

LIMIT 10;
```

```
+--------------+-----------------------------------------------------+----------------+
| 1997-08-05   | LaserWriter 8500, Apple ColorSync/AppleVision 750 Display | 20.69          |
| 1990-10-15   | Macintosh LC, Macintosh Classic, Macintosh IIsi     | 9.05           |
| 1983-01-19   | Apple Lisa[2]                                       | 8.59           |
| 1999-10-13   | Power Mac G4 Graphite                               | 8.44           |
| 2001-01-09   | PowerBook G4 Titanium, Power Mac G4 Digital Audio, Apple Pro Speakers (minijack) | 7.37           |
| 2000-09-13   | iBook (FireWire)                                    | 7.08           |
| 2000-07-19   | Power Mac G4 Cube, Cinema Display (22") (ADC), Apple Pro Speakers (USB) | 6.88           |
| 1994-07-18   | Quadra 630, PowerBook 150, Apple Multiple Scan 15 Display | 6.81           |
| 1999-01-05   | Power Macintosh G3 (Blue & White), Macintosh Server G3 (Blue & White) | 5.82           |
| 2022-10-26   | iPad (10th generation), iPad Pro (11-inch) (4th generation), iPad Pro (12.9-inch) (6th generation) | 5.74           |
+--------------+-----------------------------------------------------+----------------+
```

10 lowest volatility:

```
SELECT

 release_date,

 CONCAT_WS(', ', COLLECT_LIST(product)) as products,

 ROUND(volatility, 2) as volatility_pct

FROM release_date_volatility_final
```

```
LATERAL VIEW EXPLODE(products) products_table AS product

GROUP BY release_date, volatility

ORDER BY volatility_pct ASC

LIMIT 10;
```

```
+--------------+-----------------------------------------------------+----------------+
| 1999-10-05   | iMac (slot loading)                                 | 0.06           |
| 1996-04-01   | Power Macintosh 7600                                | 0.12           |
| 2018-07-12   | MacBook Pro (13-inch, 2018, Four Thunderbolt 3 ports), MacBook Pro (15-inch, 2018) | 0.26
| 2011-02-10   | iPhone 4 (CDMA) (16 & 32 GB)                        | 0.26           |
| 2003-11-18   | iMac G4 20"                                         | 0.26           |
| 2014-06-26   | iPod Touch 16GB (5th generation, Mid 2013)          | 0.28           |
| 1989-09-20   | Macintosh IIci, Macintosh Portable                  | 0.31           |
| 2014-04-29   | MacBook Air (Early 2014)                            | 0.34           |
| 2013-05-30   | iPod Touch (5th generation) (16 GB)                 | 0.34           |
| 2017-11-03   | iPhone X                                            | 0.36           |
+--------------+-----------------------------------------------------+----------------+
```

The volatility results further support the idea that Apple's early product launches were met with stronger market reactions, with only one release date after 2001 making to the top 10 of highest volatility weeks.

Impact of SEC Filing Type on Stock Price

For each SEC filing, let's calculate the average stock return over the next 5 days, then compute the average impact for each filing type, and then rank all the filing types by the average impact.

First, we associate each SEC filing date with its trading day index by creating a table linking filing dates to trading day indices (excluding amendments):

```
CREATE TABLE filings_with_index AS

SELECT

 sec_filings_all.filing_date AS filing_date,

 sec_filings_all.form_type,

 stock_data_with_index.trading_day_index

FROM

 sec_filings_all

JOIN

 stock_data_with_index

ON

 sec_filings_all.filing_date = stock_data_with_index.`date`

WHERE

 sec_filings_all.form_type NOT LIKE '%/A';
```

```
1,972 rows affected (3.4 seconds)
```

Then, we calculate the percentage return from the open price on the filing date to the close price 5 trading days later:

```
CREATE TABLE filing_returns AS

SELECT
```

```
 filings_with_index.filing_date,
 filings_with_index.form_type,
 stock_data_day1.open AS open_price_day1,
 stock_data_day5.close AS close_price_day5,
 ((stock_data_day5.close - stock_data_day1.open) / stock_data_day1.open) * 100 AS return_percent
FROM
 filings_with_index
JOIN
 stock_data_with_index AS stock_data_day1
ON
 filings_with_index.trading_day_index = stock_data_day1.trading_day_index
JOIN
 stock_data_with_index AS stock_data_day5
ON
 stock_data_day5.trading_day_index = filings_with_index.trading_day_index + 5
WHERE
 stock_data_day5.close IS NOT NULL;
```

```
1,968 rows affected (3.27 seconds)
```

Calculate the average return for each filing type:

```
CREATE TABLE filing_type_returns AS
SELECT
 form_type,
 COUNT(*) AS filing_count,
 AVG(return_percent) AS avg_return
FROM
 filing_returns
GROUP BY
 form_type;
```

```
38 rows affected (3.095 seconds)
```

First, let's look at all the filing types ranked by their average impact on stock returns:

```
SELECT
 form_type,
 filing_count,
 avg_return
FROM
```

```
 filing_type_returns

ORDER BY

 avg_return DESC;
```

NT 10-Q	1	7.404131328922311
25-NSE	2	7.364050252049568
PX14A6N	1	6.432746780273339
25	1	6.084830240780215
10-K405	2	5.105239520045617
CERT	2	2.6821310735795114
IRANNOTICE	2	2.1177869463452708
CERTNYS	4	1.4789439161716695
424B2	56	1.4716412388620346
SD	11	1.395124467774746
8-K	210	1.3165359268932026
FWP	28	1.298218048287302
8-A12B	6	1.2893221826264296
CORRESP	20	1.2093700248910146
PX14A6G	38	1.1171344711653537
144	13	1.0448388197899605
10-Q	90	0.8713677658883314
UPLOAD	28	0.7702679690139693
SC TO-I	2	0.7401480042102266
S-3ASR	4	0.7243570952551484
SC 13G	21	0.6601932797014202
4	1243	0.6199043431513676
DEFA14A	20	0.11719312412784669
S-8	29	0.05328170838707768
3	34	-0.03023085737074426
10-K	28	-0.09815999818078448
S-8 POS	3	-0.23492189660571552
DEF 14A	30	-0.4019918306917345
DFAN14A	1	-0.7318158079391958
NO ACT	17	-0.8937875998758936
PRE 14A	6	-1.077303838159431
424B3	6	-1.7923842987108618
424B5	2	-2.7793437673413144
S-3	2	-3.248999748103661
SC 13D	1	-3.606748396803263
S-4	1	-4.344457028417287
DEFR14A	2	-5.459823621292452
NT 10-K	1	-7.6620678996120795

Filing types with only one or two occurrences (like NT 10-Q or PX14A6N), even though quite interesting, might skew results due to small sample sizes, so let's look at filing types that have at least 5 filings:

```
SELECT *

FROM filing_type_returns

WHERE filing_count >= 5

ORDER BY avg_return DESC;
```

424B2	56	1.4716412388620346
SD	11	1.395124467774746
8-K	210	1.3165359268932026
FWP	28	1.298218048287302
8-A12B	6	1.2893221826264296
CORRESP	20	1.2093700248910146
PX14A6G	38	1.1171344711653537
144	13	1.0448388197899605
10-Q	90	0.8713677658883314
UPLOAD	28	0.7702679690139693
SC 13G	21	0.6601932797014202
4	1243	0.6199043431513676
DEFA14A	20	0.11719312412784669
S-8	29	0.05328170838707768
3	34	-0.03023085737074426
10-K	28	-0.09815999818078448
DEF 14A	30	-0.4019918306917345
NO ACT	17	-0.8937875998758936
PRE 14A	6	-1.077303838159431
424B3	6	-1.7923842987108618

Impact of SEC Filing Type on Stock Volatility

Collect daily percentage changes for the 5 trading days following each filing date:

```
CREATE TABLE filing_date_daily_returns AS
SELECT
 filings_with_index.filing_date,
 filings_with_index.form_type,
 apple_stock_data.`date` AS stock_date,
 apple_stock_data.change_percent
FROM
 filings_with_index
JOIN
 apple_stock_data
ON
 apple_stock_data.`date` > filings_with_index.filing_date
 AND apple_stock_data.`date` <= DATE_ADD(filings_with_index.filing_date, 5);
```

```
5,998 rows affected (5.281 seconds)
```

Compute the volatility of stock returns over the 5-day period after each filing date:

```
CREATE TABLE filing_volatility AS
SELECT
 filing_date_daily_returns.filing_date,
 filing_date_daily_returns.form_type,
 STDDEV_SAMP(filing_date_daily_returns.change_percent) AS volatility
FROM
 filing_date_daily_returns
GROUP BY
 filing_date_daily_returns.filing_date,
 filing_date_daily_returns.form_type;
```

```
1,263 rows affected (3.22 seconds)
```

Calculate the average volatility for each filing type:

```
CREATE TABLE filing_type_volatility AS
SELECT
 form_type,
 COUNT(*) AS filing_count,
 AVG(volatility) AS avg_volatility
FROM
```

```
 filing_volatility
GROUP BY
 form_type;
```

```
38 rows affected (3.13 seconds)
```

List all filing types ranked by their average volatility:

```
SELECT
 form_type,
 filing_count,
 avg_volatility
FROM
 filing_type_volatility
ORDER BY
 avg_volatility DESC;
```

PX14A6N	1	3.572286868245634
25-NSE	2	3.0797903684094665
SC 13D	1	3.065925579535839
S-4	1	2.967743246936032
10-K405	2	2.365272224372913
8-K	205	2.3206599819307634
DEFR14A	2	2.26903166643613
SC 13G	20	2.2687613757392024
NO ACT	16	2.121084426781071
S-8	26	2.1056544883994395
SC TO-I	2	2.0542070075328716
PRE 14A	6	1.967734982206389
10-Q	90	1.961155207690532
NT 10-K	1	1.8635807627768066
CORRESP	20	1.8275730498803628
10-K	28	1.8173877104619909
NT 10-Q	1	1.7360587822361764
S-8 POS	2	1.695779526946302
DEF 14A	30	1.687638373232248
3	28	1.5794836554354066
424B5	2	1.5469866464513737
DEFA14A	20	1.50258097742314
S-3ASR	4	1.4691933380942264
4	568	1.415391346051501
UPLOAD	28	1.3939612195444013
PX14A6G	29	1.3495263579680294
DFAN14A	1	1.317054779841815
25	1	1.1345042884437944
CERTNYS	4	1.105136590491358
8-A12B	6	1.0979227223166161
S-3	2	1.028530058689754
424B3	6	0.9928107177490073
CERT	2	0.9823424873170953
FWP	28	0.9454592271508357
424B2	56	0.8843112489268067
144	9	0.789013652201175
SD	11	0.7398316344979817
IRANNOTICE	2	0.583962683807224

With fewer than 5 filings, the average volatility could be heavily skewed by one-off events or unusual circumstances. For example, PX14A6N which is at the top of the list corresponds to a request to conduct a civil rights audit analyzing the company's adverse impact on stakeholders filed by an investment group. To find stronger patterns, it makes sense to look at the filtered table:

```
SELECT *
FROM filing_type_volatility
WHERE filing_count >= 5
ORDER BY avg_volatility DESC;
```

8-K	205	2.3206599819307634
SC 13G	20	2.2687613757392024
NO ACT	16	2.121084426781071
S-8	26	2.1056544883994395
PRE 14A	6	1.967734982206389
10-Q	90	1.961155207690532
CORRESP	20	1.8275730498803628
10-K	28	1.8173877104619909
DEF 14A	30	1.687638373232248
3	28	1.5794836554354066
DEFA14A	20	1.50258097742314
4	568	1.415391346051501
UPLOAD	28	1.3939612195444013
PX14A6G	29	1.3495263579680294
8-A12B	6	1.0979227223166161
424B3	6	0.9928107177490073
FWP	28	0.9454592271508357
424B2	56	0.8843112489268067
144	9	0.789013652201175
SD	11	0.7398316344979817

Spark

Starting inputs

```
export SPARK_HOME=/usr/hdp/2.6.5.0-292/spark2/


cd FinalProject


hadoop fs -mkdir /user/root/FinalProject
hadoop fs -put aapl_raw_data.csv /user/root/FinalProject/
hadoop fs -put aapl_split_adjusted.csv /user/root/FinalProject/


pyspark
```

Make the csv into a readable set of tables named appl and applsa

```
spark.read.option("header",
True).csv("/user/root/FinalProject/aapl_raw_data.csv").createOrReplaceTempView("aapl")
spark.read.option("header",
True).csv("/user/root/FinalProject/aapl_split_adjusted.csv").createOrReplaceTempView("aaplsa")
```

Some initial queries that will tell us date range and the max open/close

```
spark.sql("""
SELECT min(date), max(date)
FROM aapl
""").show(500)


--total price range
spark.sql("""
SELECT min(open),min(close),min(high),min(low)
FROM aapl
""").show(500)
```

This is a query to show volatility of the stock.

```
spark.sql("""
SELECT
substring(date,1,7) as month,
avg(open) as avg_open,
avg(close) as avg_close,
avg(high) as avg_high,
avg(low) as avg_low,
(sqrt(sum(pow(open-max_price.price,2))/count(*)) +
sqrt(sum(pow(close-max_price.price,2))/count(*)) +
sqrt(sum(pow(high-max_price.price2,2))/count(*)) +
sqrt(sum(pow(low-max_price.price2,2))/count(*)) )/4 as volitility
FROM aapl
left join
(select
substring(date,1,7) as month2,
avg((open+close)/2) as price,
avg((high+low)/2) as price2
from aapl
group by month2
) as max_price
on max_price.month2 = substring(date,1,7)
group by month
order by month
""").coalesce(1).write.mode('overwrite').option("header", "true").csv("/user/root/FinalProject/volitility.csv")
```

This code is run right after to get the data from hadoop

```
quit()
hadoop fs -get /user/root/FinalProject/volitility.csv
```

This query shows the seasonal changes of the stock, if month affects how this stock performs. Sometimes seasonal items like ice cream companies can fluctuate with season or month.

```
spark.sql("""
select
substring(yearmonth,1,4) as year,
max(case when substring(yearmonth,6,2) = '01' then avg_open end) as Jan,
max(case when substring(yearmonth,6,2) = '02' then avg_open end) as Feb,
max(case when substring(yearmonth,6,2) = '03' then avg_open end) as Mar,
max(case when substring(yearmonth,6,2) = '04' then avg_open end) as Apr,
max(case when substring(yearmonth,6,2) = '05' then avg_open end) as May,
max(case when substring(yearmonth,6,2) = '06' then avg_open end) as Jun,
max(case when substring(yearmonth,6,2) = '07' then avg_open end) as Jul,
max(case when substring(yearmonth,6,2) = '08' then avg_open end) as Aug,
max(case when substring(yearmonth,6,2) = '09' then avg_open end) as Sep,
max(case when substring(yearmonth,6,2) = '10' then avg_open end) as Oct,
max(case when substring(yearmonth,6,2) = '11' then avg_open end) as Nov,
max(case when substring(yearmonth,6,2) = '12' then avg_open end) as Dec
from
(
SELECT
substring(date,1,7) as yearmonth,
avg(open) as avg_open,
avg(close) as avg_close
FROM aapl
group by substring(date,1,7)
)
group by year
order by year
""").coalesce(1).write.mode('overwrite').option("header", "true").csv("/user/root/FinalProject/seasonal.csv")
```

This code is run right after to get the data from hadoop

```
quit()
hadoop fs -get /user/root/FinalProject/seasonal.csv
```

This query is the greatest differences in price per day, there are some very large changes

Interestingly enough most of them are in 2012 and 2020

```
spark.sql("""
SELECT
aapl.date,
a.high_open,
b.low_open,
c.high_close,
d.low_close,
e.open_close,
f.high_low
FROM aapl

left join
(select
date,
high-open as high_open,
Rank() over (ORDER BY (high-open) DESC ) as rank1
from aapl
) as a
on a.date = aapl.date

left join
(select
date,
low-open as low_open,
Rank() over (ORDER BY (low-open) ) as rank2
```

```
from aapl
) as b
on b.date = aapl.date

left join
(select
date,
high-close as high_close,
Rank() over (ORDER BY (high-close) DESC ) as rank3
from aapl
) as c
on c.date = aapl.date

left join
(select
date,
low-close as low_close,
Rank() over (ORDER BY (low-close) ) as rank4
from aapl
) as d
on d.date = aapl.date

left join
(select
date,
open-close as open_close,
Rank() over (ORDER BY (open-close) DESC ) as rank5
from aapl
) as e
on e.date = aapl.date
```

```
left join
(select
date,
high-low as high_low,
Rank() over (ORDER BY (high-low) DESC ) as rank6
from aapl
) as f
on f.date = aapl.date



where rank1<10
or rank2<10
or rank3<10
or rank4<10
or rank5<10
or rank6<10
order by aapl.date

""").coalesce(1).write.mode('overwrite').option("header", "true").csv("/user/root/FinalProject/largestdrop.csv")
```

This code is run right after to get the data from hadoop

```
quit()
hadoop fs -get /user/root/FinalProject/largestdrop.csv
```