Contents

Data exploration using sql server

This process involves querying the data, examining its structure, and summarizing its characteristics to gain a deeper understanding of its patterns and trends using SQL Server.

> See data visualization here

min max caluclation

This SQL query retrieves summary statistics for the 'close_price' column of the 'Apple' table. It calculates the minimum, maximum, and average values of the 'close_price'.

```
select
min(close_price) as min_price,
max(close_price) as max_price,
avg(close_price) as avg_price

from Apple
```

The result provides insights into the range of prices, the highest and lowest prices recorded, and the average price for the specified dataset. By running this query, you can quickly understand key characteristics of the price distribution, helping you gain a basic understanding of the data's variability and central tendency.

Trend identification

This SQL code examines changes in the 'close_price' of the 'Apple' dataset using a Common Table Expression (CTE). It computes 'previous_close' using the lag function to get the previous day's price. The 'price_change' is found by subtracting yesterday's price from today's.

```
with CTE as(
select
date_only,close_price,
lag(close_price) over (order by date_only) as previous_close,
close_price - lag(close_price) over (order by date_only) as price_change
from [Apple])
select *
from cte
```

```
where price_change >0
order by price_change desc
```

This helps understand daily price shifts. The final query highlights positive price changes, suggesting days when Apple's stock price rose. Results are sorted by change size, unveiling the most notable upward shifts. This approach uncovers significant positive trends in stock value, aiding insights into Apple's market performance.

stock price vs the divided

This SQL query retrieves specific data from the 'Apple' dataset. It selects the 'date_only', 'close_price', 'volume', and 'dividends' columns from the table. The query focuses on rows where the 'Dividends' value is greater than zero, which means there were dividend payments on those days. The results are organized in descending order based on the 'Dividends' value.

```
select
date_only,close_price, volume,dividends
from Apple
where Dividends >0
order by Dividends desc
```

By running this query, we can see the dates, closing prices, trading volumes, and dividend amounts on days when Apple paid dividends. This helps us understand when Apple distributed dividends and the associated stock market activities on those occasions.

impact of stock split

It focuses on days when there were stock splits, which are changes in the number of shares outstanding. For each of these days, it calculates adjusted values to account for the split effect. It divides the original open, high, low, and close prices by the stock split , resulting in adjusted prices that reflect the impact of the split.

```
select date_only,[Stock Splits],
open_price / [stock splits] as adjusted_open,
high_price / [stock splits] as adjusted_high,
low_price / [stock splits] as adjusted_low,
close_price / [stock splits] as adjusted_close,
volume * [stock splits] as adjusted_volume
from [apple]
where [Stock Splits] >0
```

Similarly, it multiplies the original volume by the stock split , providing an adjusted volume that considers the split effect. By running this query, we can see how stock splits influenced the trading data and gain insights into how the stock's value was adjusted to accommodate these events.

Measuring stock performance

- This SQL query helps us find the best months for Apple's stock over the years 2015 to 2023. It does this by first organizing the data into a neat summary called a Common Table Expression (CTE). This summary calculates the average stock price and average trading volume for each month of each year.
- The main part of the query works with this summary. It picks out the year, month, average trading volume, and average stock price from the CTE. Using a trick called "window ranking," it gives each month a rank based on its average stock price. This means it puts the months with the highest stock prices at the top of the list.

```
with cte as (
select
datepart(year,date_only)  as year,
datepart(month, date_only) as month,
avg(close_price) as avg_stock_price,
avg(volume) as avg_volume
from Apple
group by
datepart(year,date_only),
datepart(month, date_only)
)
select*
from(
select year,month,avg_volume,avg_stock_price,
row_number() over (partition by year order by avg_stock_price desc) as rank
from cte
where year >= '2015' and year <= '2023' ) as ranked_cte
where rank =1
```

- The query pays attention only to years between 2015 and 2023, so it filters out the other years
- Lastly, it checks which months are ranked as number one (the top rank) for each year. This gives us the months when Apple's stock had the highest average price for each year in the specified time range.

> In short, the query helps us find the months when Apple's stock did really well each year from 2015 to 2023. It's like picking out the stars of each year based on their stock prices

outlier detection using IQR technique

In summary, this query helps us spot unusual days in Apple's stock price history days where the price is either remarkably low or incredibly high compared to the typical pattern. It's like finding the days when something really unique happened with Apple's stock price.

```
with outlier_detection as (
select close_price,
avg(close_price) over () as mean,
stdev(close_price) over() as stdev,
PERCENTILE_CONT(0.25) within group (order by close_price) over () as q1,
PERCENTILE_CONT(0.75) within group (order by close_price) over () as q3
from Apple)
select
close_price,
```

```
case
when close_price < q1-1.5 * (q3 - q1) or close_price > q3 + 1.5 * (q3 -q1) then
'outlier'
else 'not an outlier'
end as outlier_status
from outlier_detection
```

- For each day's stock price, it calculates the average of all the stock prices (mean) and how much the prices typically vary from this average (standard deviation).
- It also figures out where the 25th percentile (Q1) and 75th percentile (Q3) of the prices are. This means it's looking at prices that are lower than 25% of all prices and prices that are higher than 75% of all prices.

> Formula:
> IQR = Q3 -Q1