

TASK-B

SQL -QUERY

1. Print the highest opening and the lowest closing values of each month for Google.

Query

```
SELECT max([Open]) as Highest_opening ,min([Close]) as  
Lowest_closing,MONTH(Date) as Month  
FROM Stocks_Google  
GROUP BY MONTH(Date);
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
SELECT * FROM Stocks_Netf;  
  
SELECT max([Open]) as Highest_opening ,min([Close]) as Lowest_closing,MONTH(Date) as Month  
FROM Stocks_Google  
GROUP BY MONTH(Date);  
  
select STDEV(Volume) as SD_Vol,Year(Date) as Year  
FROM Stocks_Netflix  
Group by year(Date);
```

The Results pane displays the output of the third query, showing the standard deviation of volume for each year:

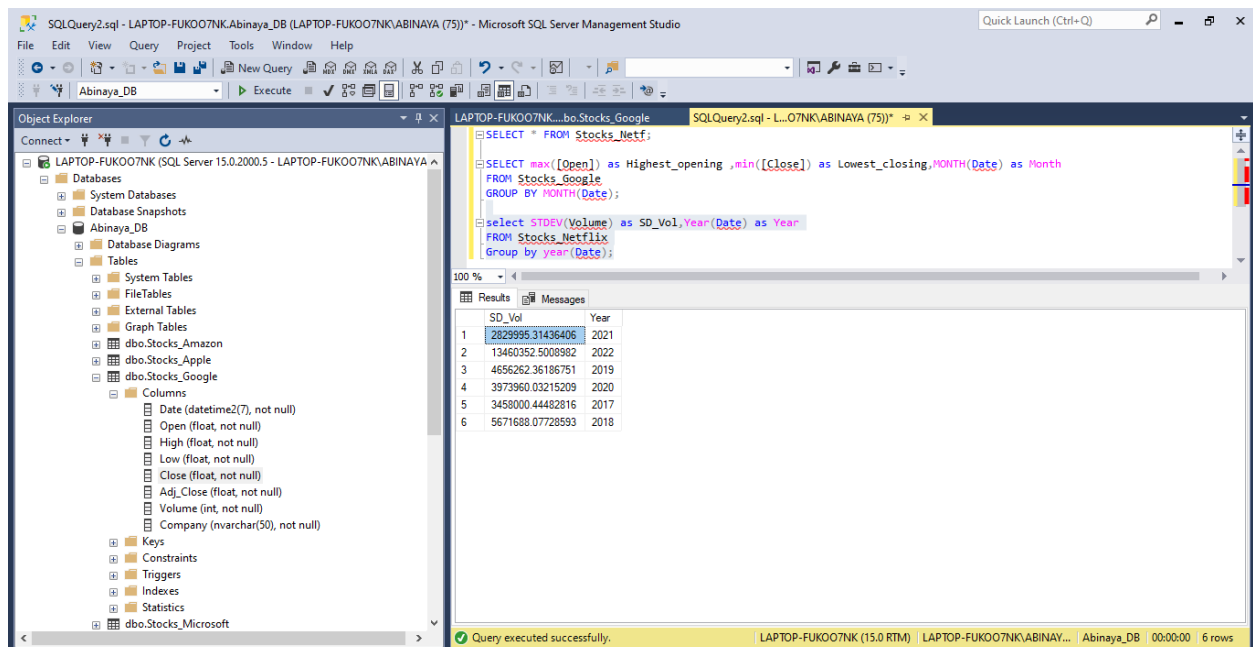
SD_Vol	Year
2829995.31436406	2021
13460352.5008982	2022
4656262.36186751	2019
3973960.03215209	2020
3458000.44482816	2017
5671688.07728593	2018

The status bar at the bottom indicates: Query executed successfully. LAPTOP-FUKOO7NK (15.0 RTM) | LAPTOP-FUKOO7NK\ABINAY... | Abinaya_DB | 00:00:00 | 6 rows

2. Find the standard deviation of Volume per year for Netflix.

Query

```
SELECT STDEV(Volume) as SD_Vol, Year(Date) as Year
FROM Stocks_Netflix
GROUP BY year(Date);
```



The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'LAPTOP-FUKOO7NK' (SQL Server 15.0.2000.5). The query editor in the center contains the following SQL query:

```
SELECT * FROM Stocks_Netflix;
SELECT max([Open]) as Highest_opening ,min([Close]) as Lowest_closing, MONTH(Date) as Month
FROM Stocks_Google
GROUP BY MONTH(Date);
select STDEV(Volume) as SD_Vol, Year(Date) as Year
FROM Stocks_Netflix
Group by year(Date);
```

The Results pane at the bottom shows the output of the query, displaying a table with two columns: 'SD_Vol' and 'Year'. The table contains six rows of data:

SD_Vol	Year
2829995.31436406	2021
13460352.5008982	2022
4656262.36186751	2019
3973960.03215209	2020
3458000.44482816	2017
5671688.07728593	2018

The status bar at the bottom indicates that the query was executed successfully, showing the server name 'LAPTOP-FUKOO7NK (15.0 RTM)', the database 'Abinaya_DB', and the execution time '00:00:00' with '6 rows' returned.

3. Find the difference between the opening values of Amazon and Apple.

Query

```
SELECT (am.[Open])-(a.[Open]) as Difference  
FROM Stocks_Amazon am, Stocks_Apple a
```

The screenshot displays the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows the database structure for 'LAPTOP-FUKOO7NK' (SQL Server 15.0.2000.5). The 'Query Editor' on the right contains the following SQL query:

```
FROM Stocks_Bogle  
GROUP BY MONTH(Date);  
  
select STDEV(Volume) as SD_Vol, Year(Date) as Year  
FROM Stocks_Netflix  
Group by year(Date);  
  
select (am.[Open])-(a.[Open]) as Difference  
from Stocks_Amazon am, Stocks_Apple a
```

The 'Results' pane shows the output of the query, which is a single column named 'Difference' with 17 rows of data:

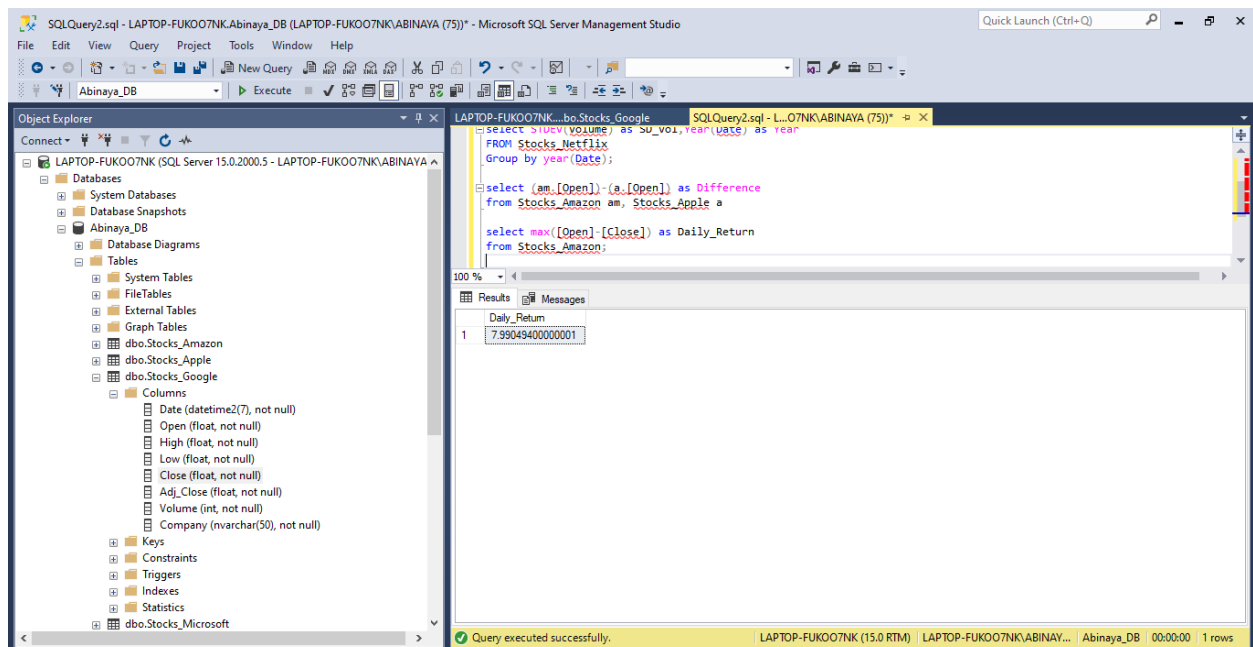
Difference
10.267502
9.882499
9.717502
9.557503
10.382499
9.702499
9.367500000000001
9.047500000000001
9.4025
10.067501
10.157501
9.975002
9.764999
9.425003
9.620003
9.497501000000001
9.507499

The status bar at the bottom indicates 'Query executed successfully' and shows the execution time as 00:00:24 with 15,82,564 rows.

4. Find the largest daily return for Amazon. (Daily return is calculated by subtracting the opening price from the closing price)

Query

```
SELECT max([Open]-[Close]) as Daily_Return  
FROM Stocks_Amazon;
```



The screenshot displays the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows the database structure for 'Abinaya_DB', including tables like 'dbo.Stocks_Amazon', 'dbo.Stocks_Apple', and 'dbo.Stocks_Google'. The 'Query Editor' window shows a SQL query that calculates the daily return for Amazon by subtracting the closing price from the opening price and finding the maximum value. The 'Results' pane shows a single row with the value 7.990494000000001 under the column 'Daily_Return'. The status bar at the bottom indicates that the query was executed successfully.

```
--select stock_volume as sv_vol, year(Date) as year  
FROM Stocks_Netflix  
Group by year(Date);  
  
--select (am.[Open])-(a.[Open]) as Difference  
from Stocks_Amazon am, Stocks_Apple a  
  
select max([Open]-[Close]) as Daily_Return  
from Stocks_Amazon;
```

Daily_Return
7.990494000000001

Query executed successfully. LAPTOP-FUKOO7NK (15.0 RTM) LAPTOP-FUKOO7NK\ABINAY... Abinaya_DB 00:00:00 1 rows

5. Print the company name with the highest opening value for each day.

Query

```
SELECT Date,MAX([UpdateOpen]) AS Highest_Open
FROM
(
    SELECT Date,Company, [Open] AS UpdateOpen
    FROM Stocks_Amazon
    UNION
    SELECT Date,Company, [Open] AS UpdateOpen
    FROM Stocks_Apple
    UNION
    SELECT Date,Company, [Open] AS UpdateOpen
    FROM Stocks_netflix
    UNION
    SELECT Date,Company, [Open] AS UpdateOpen
    FROM Stocks_Microsoft
    UNION
    SELECT Date,Company, [Open] AS UpdateOpen
    FROM Stocks_Tesla
    UNION
    SELECT Date,Company, [Open] AS UpdateOpen
    FROM Stocks_Google
) ud

GROUP BY Date
```

SQLQuery2.sql - LAPTOP-FUKOO7NK.Abinaya_DB (LAPTOP-FUKOO7NK\ABINAYA (75)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Connect Abinaya_DB Execute

Object Explorer

- Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.Stocks_Amazon
 - dbo.Stocks_Apple
 - dbo.Stocks_Google
 - Columns
 - Date (datetime2(7))
 - Open (float, not null)
 - High (float, not null)
 - Low (float, not null)
 - Close (float, not null)
 - Adj_Close (float, not null)
 - Volume (int, not null)
 - Company (nvarchar(100))
 - Keys
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.Stocks_Microsoft
 - dbo.Stocks_Netflix
 - dbo.Stocks_Tesla
- Views
- External Resources
- Synonyms
- Programmability

Results

	Date	Highest_Open
1	2017-08-07 00:00:00.0000000	181
2	2017-08-08 00:00:00.0000000	181.369995
3	2017-08-09 00:00:00.0000000	171.429993
4	2017-08-10 00:00:00.0000000	174.029999
5	2017-08-11 00:00:00.0000000	169.860001
6	2017-08-14 00:00:00.0000000	169.800003
7	2017-08-15 00:00:00.0000000	171.529999
8	2017-08-16 00:00:00.0000000	167.5
9	2017-08-17 00:00:00.0000000	169.229996
10	2017-08-18 00:00:00.0000000	165.949997
11	2017-08-21 00:00:00.0000000	166.910004
12	2017-08-22 00:00:00.0000000	167.759995
13	2017-08-23 00:00:00.0000000	168.350006
14	2017-08-24 00:00:00.0000000	169.860001
15	2017-08-25 00:00:00.0000000	168.580002
16	2017-08-28 00:00:00.0000000	166.429993
17	2017-08-29 00:00:00.0000000	165
18	2017-08-30 00:00:00.0000000	169.5
19	2017-08-31 00:00:00.0000000	175.449997
20	2017-09-01 00:00:00.0000000	175.550003
21	2017-09-05 00:00:00.0000000	173.399994
22	2017-09-06 00:00:00.0000000	175.25
23	2017-09-07 00:00:00.0000000	178.800003
24	2017-09-08 00:00:00.0000000	178.449997
25	2017-09-11 00:00:00.0000000	178.100006

Query executed successfully.

Ready En 1 Col 1 INS