

# E09-ASSIGNMENT 9

**NAME: HRISHIKESH S**

**ROLL NO: 22B4217**

**Q1)**

**Do a manual review of the table nusedata and describe its contents (no SQL to be executed for this task):**

The given CSV file contains a huge amount of data organised into 13 columns and 1893059 rows!

The given data consists of the Stock price details like the opening Stock price,closing stock price,highest stock price,lowest stock price,etc which are all self explanatory.

**Q2)**

**Select the database stockdata using SQL:**

[illegible]

Q3)

Get a schema dump of the table nsedata:

The screenshot shows a SQL IDE interface. The top panel contains the following SQL code:

```
use stockdata;  
show create table nsedata;
```

The bottom panel displays the results of the query. The first column, labeled 'Table', shows 'nsedata'. The second column, labeled 'Create Table', shows the following SQL statement:

```
CREATE TABLE `nsedata` (`symbol` varchar(10) NOT NULL,  
`series` varchar(10) NOT NULL,  
`open` decimal(10,4) NOT NULL,  
`high` decimal(10,4) NOT NULL,  
`low` decimal(10,4) NOT NULL,  
`close` decimal(10,4) NOT NULL,  
`last` decimal(10,4) NOT NULL,  
`prevclose` decimal(10,4) NOT NULL,  
`tottrdqty` int(11) NOT NULL)
```

The IDE also shows a 'Value' column with a preview of the table structure and a 'Content...5 Bytes' indicator. The bottom status bar includes buttons for 'Refresh', 'Save', 'Cancel', and navigation icons.

Q4)

1893059

The screenshot shows a database query results window titled "Results 1". The SQL query entered is "select count(\*) from i". The results are displayed in a grid view with one row and one column. The first row has a value of 1,893,059. The column header is "count(\*)".

	count(*)
1	1,893,059

Q5)

Get the total count of the records for the month "October 2012":

```
select count(*) from nsedata where timestamp like "__-OCT-2012";
```

The screenshot shows a SQL IDE with two tabs: 'nsedata' and '\*<localhost> Script-2'. The script in the 'Script-2' tab contains the following SQL commands:

```
use stockdata;
show create table nsedata;
select count(*) from nsedata;
select count(*) from nsedata where timestamp like "__-OCT-
```

The 'Results 1' tab is active, displaying the results of the last query. The results are shown in a grid format with the following data:

	count(*)
1	33,244

On the right side of the results grid, there is a 'Value' panel showing the value '33244'. The interface also includes a sidebar with various icons and a bottom panel with additional icons.

**Q6)**

**Repeat '4', but only for the stock with symbol "GEOMETRIC":**

The screenshot shows a SQL IDE with a script editor and a results pane. The script editor contains the following SQL code:

```
use stockdata;  
show create table nsedata;  
select count(*) from nsedata;  
select count(*) from nsedata where timestamp like "__-OCT-"  
select count(*) from nsedata where symbol="GEOMETRIC";
```

The results pane shows the output of the first query, `select count(*) from nsedata;`. It displays a single row with the value 1,237.

Grid	123 count(*)
1	1,237

A tooltip shows the value 1237.

Q7)

Repeat '6', but only display the first 10 records:

nsedata \* <localhost> Script-2 x

```

use stockdata;
show create table nsedata;
select count(*) from nsedata;
select count(*) from nsedata where timestamp like "__-OCT-";
select count(*) from nsedata where symbol="GEOMETRIC";

select * from nsedata where symbol="GEOMETRIC" limit 10;

```

nsedata 1 x

select \* from nsedata Enter a SQL expression to filter results (u. > |

	ABC symbol	ABC series	123 open	123 high
1	GEOMETRIC	EQ	62.35	
2	GEOMETRIC	EQ	100.7	
3	GEOMETRIC	EQ	116	
4	GEOMETRIC	EQ	166.5	
5	GEOMETRIC	EQ	49.8	
6	GEOMETRIC	EQ	94.4	
7	GEOMETRIC	EQ	69.45	
8	GEOMETRIC	EQ	141.2	
9	GEOMETRIC	EQ	73.3	
10	GEOMETRIC	EQ	45.9	

Value x  
GEOMETRIC

Grid  
Text  
cord

Panels

**Q8)Totally, how many records of “INFY” does the table contain?:**

The screenshot shows a SQL IDE with two tabs: 'nsedata' and '\*<localhost> Script-2'. The script in the 'Script-2' tab contains the following SQL commands:

```
use stockdata;
show create table nsedata;
select count(*) from nsedata;
select count(*) from nsedata where timestamp like "__-OCT-";
select count(*) from nsedata where symbol="GEOMETRIC";

select * from nsedata where symbol="GEOMETRIC" limit 10;
select count(*) from nsedata where symbol="INFY";
```

The 'Results 1' tab is active, displaying the results of the first query: 'select count(\*) from nsedata;'. The results are shown in a grid with one row and one column, where the value is 1,023. A 'Value' panel on the right also shows the value 1023.

Grid	123 count(*)
1	1,023

Q9)

Get a listing of the first 10 records of "3IINFOTECH", but the listing should contain only the following columns: symbol, open, high, low, close, and timestamp:

nsedata \* <localhost> Script-2 x

```

use stockdata;
show create table nsedata;
select count(*) from nsedata;
select count(*) from nsedata where timestamp like "__-OCT-";
select count(*) from nsedata where symbol="GEOMETRIC";

select * from nsedata where symbol="GEOMETRIC" limit 10;
select count(*) from nsedata where symbol="INFY";

select symbol,open,high,low,close,timestamp
from nsedata where symbol ="3IINFOTECH" limit 10;

```

nsedata 1 x

select symbol,open,h Enter a SQL expression to filter results (u. ▶ |

	symbol	open	high	low
1	3IINFOTECH	43.75	45.3	43
2	3IINFOTECH	5.65	6.1	5
3	3IINFOTECH	7.85	7.9	7
4	3IINFOTECH	5.9	6.3	
5	3IINFOTECH	41.6	42.45	4
6	3IINFOTECH	10.8	10.8	1
7	3IINFOTECH	3.95	4.15	3
8	3IINFOTECH	8.75	9.1	
9	3IINFOTECH	55.9	59.4	55
10	3IINFOTECH	20	20	1

Value x

3IINFOTECH

Refresh Save Cancel

**Q10)**

**Repeat '9', but this time use the results to create a temporary table t1:**



nsedata \* <localhost> Script-2 x

```

use stockdata;
show create table nsedata;
select count(*) from nsedata;
select count(*) from nsedata where timestamp like "__-OCT-";
select count(*) from nsedata where symbol="GEOMETRIC";

select * from nsedata where symbol="GEOMETRIC" limit 10;
select count(*) from nsedata where symbol="INFY";

select symbol,open,high,low,close,timestamp
from nsedata where symbol ="3IINFOTECH" limit 10;

create temporary table t1 as
select symbol,open,high,low,close,timestamp
from nsedata where symbol="3IINFOTECH" limit 10;

select * from t1;

```

t1 1 x

select \* from t1 Enter a SQL expression to filter results (use Ctrl)

	symbol	open	high	low
1	3IINFOTECH	43.75	45.3	43.75
2	3IINFOTECH	5.65	6.1	5.65
3	3IINFOTECH	7.85	7.9	7.85
4	3IINFOTECH	5.9	6.3	5.9
5	3IINFOTECH	41.6	42.45	41.6
6	3IINFOTECH	10.8	10.8	10.8
7	3IINFOTECH	3.95	4.15	3.95
8	3IINFOTECH	8.75	9.1	8.75
9	3IINFOTECH	55.9	59.4	55.9

Value x

3IINFOTECH

Refresh Save Cancel

Q11)

Using t1 find out the following for the column close: max, min, mean. standard deviation and variance:



nsedata \* <localhost> Script-2

```

select * from t1;

SELECT
    MIN(CLOSE),
    AVG(CLOSE),
    STDDEV(CLOSE),
    MAX(CLOSE),
    VAR_POP(CLOSE) AS VARIANCE
from t1;

SELECT AVG(close) AS median
FROM (
    SELECT close
    FROM t1
    ORDER BY close ASC
    LIMIT 2 OFFSET 4
) AS t2;

```

Results 1

SELECT AVG(close) A Enter a SQL expression to filter results (u)

Grid	123 median	Value
1	9.725	9.725000000

Text

Value

Q13)

Delete table t1:

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```

select * from t1;

-- SELECT
MIN(CLOSE),
AVG(CLOSE),
STDDEV(CLOSE),
MAX(CLOSE),
VAR_POP(CLOSE) AS VARIANCE
from t1;

-- SELECT AVG(close) AS median
FROM (
    SELECT close
    FROM t1
    ORDER BY close ASC
    LIMIT 2 OFFSET 4
) AS t2;

drop temporary table t1;

```

Below the query editor, the 'Statistics 1' panel is visible, showing the following data:

Name	Value
Updated Rows	0
Query	drop temporary table t1
Start time	Sun Oct 22 16:19:37 IST 2023
Finish time	Sun Oct 22 16:19:38 IST 2023

**Q14)**

Use nstdata. Using the GROUP BY functionality of SQL create a table t2 containing the average value of close for each and every symbol in the table. Hint: the table will have the columns: symbol, average:

```

SELECT AVG(close) AS median
FROM (
    SELECT close
    FROM t1
    ORDER BY close ASC
    LIMIT 2 OFFSET 4

```

```

) AS t2;

drop temporary table t1;

create table t2 as select symbol,avg(close) as average
from nsedata n group by symbol;

select * from t2;

```

The screenshot shows a SQL IDE interface. The script editor contains the following SQL code:

```

SELECT AVG(close) AS media
FROM (
    SELECT close
    FROM t1
    ORDER BY close ASC
    LIMIT 2 OFFSET 4
) AS t2;

drop temporary table t1;

create table t2 as select
from nsedata n group by s
select * from t2;

-- SELECT AVG(close) AS me
-- from(
--     SELECT close
--     from t1
--     ORDER BY close ASC

```

The output window displays the following message:

```

Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av
Data truncated for column 'av

```

The results grid shows the following data:

	symbol	average
1	20MICRONS	53.0041228779
2	3IINFOTECH	18.038803557
3	3MINDIA	4,520.3439773646
4	3RDROCK	173.2137755102
5	8KMILES	480.7362204724
6	A2ZINFRA	18.6094339623
7	A2ZMES	89.6938950555
8	AANJANEYA	441.8403024911
9	AARTIDRUGS	312.9444624091
10	AARTIIND	127 7027081649

The Value window shows the value 20MICRONS.

Q15)

Create a table t3 such that it contains the following columns: symbol, open, close, "average of open and close". Fill up this table for the company GEOMETRIC, for the month of October:

```

) AS t2;

drop temporary table t1;

create table t2 as select symbol, avg(close) as average
from nsedata n group by symbol;
select * from t2;

create table t3 as
select symbol,
       open,
       close
from nsedata where symbol="GEOMETRIC"
and timestamp like "__-OCT-2012";

ALTER TABLE t3 add average float;
update t3 set average=(open+close)/2;
select * from t3;

```

symbol	open	close	average
GEOMETRIC	117	120.25	
GEOMETRIC	121.45	120.3	
GEOMETRIC	121.55	117.05	
GEOMETRIC	117.1	117.45	
GEOMETRIC	121	122.55	
GEOMETRIC	124.8	120.85	
GEOMETRIC	120.45	117.3	
GEOMETRIC	118	116.05	
GEOMETRIC	116.2	115.65	

Q16)

It is required to create a table t4 such that it contains the data for two companies

GEOMETRIC and TCS. The columns of this table should be as follows:

timestamp, close\_tcs,

close\_geometric. Hint: use JOIN:

The screenshot shows a database management tool interface. The top pane displays a series of SQL queries. The bottom pane shows a result grid for a query executed on table t4.

**SQL Queries:**

```

open,
close
from nsedata where symbol="GEOMETRIC"
and timestamp like "__-OCT-2012";

ALTER TABLE t3 add average float;
update t3 set average=(open+close)/2;
select * from t3;

create table table1 as select close as TCS_CLOSE,
timestamp from nsedata where symbol="TCS";
create table table2 as select close as GEOMETRIC_CLOSE,
timestamp from nsedata where symbol="GEOMETRIC";

create table t4 as select table1.timestamp,
TCS_CLOSE,table2.GEOMETRIC_CLOSE
from table1 inner join table2 on
table1.timestamp=table2.timestamp;
select * from t4;

```

**Result Grid:**

	timestamp	TCS_CLOSE	GEOMETRIC_CLOSE
1	01-APR-2011	1,180.15	63.2
2	01-APR-2013	1,556.85	103.
3	01-APR-2014	2,176.7	12
4	01-APR-2015	2,542.65	177.5
5	01-AUG-2011	1,135.25	49.
6	01-AUG-2012	1,224.65	91.
7	01-AUG-2013	1,815.4	64.
8	01-AUG-2014	2,516.4	140.
9	01-DEC-2010	1,081.9	72.3

Q17)

Find out the maximum and minimum difference in the daily closing prices of these two companies.:

The screenshot shows a SQL IDE with a script editor and a results pane. The script in the editor is as follows:

```

-- from nssdata where symbol = GEOMETRIC
and timestamp like "__-OCT-2012";

ALTER TABLE t3 add average float;
update t3 set average=(open+close)/2;
select * from t3;

-- create table table1 as select close as TCS_CLOSE,
timestamp from nssdata where symbol="TCS";
-- create table table2 as select close as GEOMETRIC_CLOSE,
timestamp from nssdata where symbol="GEOMETRIC";

-- create table t4 as select table1.timestamp,
TCS_CLOSE,table2.GEOMETRIC_CLOSE
from table1 inner join table2 on
table1.timestamp=table2.timestamp;
select * from t4;

-- select MIN(TCS_CLOSE-GEOMETRIC_CLOSE),
MAX(TCS_CLOSE-GEOMETRIC_CLOSE) from t4;

```

The results pane shows the execution of the final query. The query bar contains:

```
select MIN(TCS_CLOSE-GEOMETRIC_CLOSE), MAX(TCS_CLOSE-GEOMETRIC_CLOSE) from t4;
```

The results table has two columns: `MIN(TCS_CLOSE-GEOMETRIC_CLOSE)` and `MAX(TCS_CLOSE-GEOMETRIC_CLOSE)`. The first column has a value of 123, and the second column has a value of 123. The data type for the second column is DECIMAL(21,6) (Read-only: No corresponding table column).

id	MIN(TCS_CLOSE-GEOMETRIC_CLOSE)	MAX(TCS_CLOSE-GEOMETRIC_CLOSE)
123	123	123

The interface includes a toolbar with various icons for editing, running, and saving the script, and a status bar at the bottom showing the number of records (200) and the current page (1).

Q18)



Based on t4 can you identify those days on which the difference in their closing price was more than the average of the minimum and maximum difference.:

```

create table table2 as select close as GEOMETRIC_CLOSE,
timestamp from nsedata where symbol="GEOMETRIC";

create table t4 as select table1.timestamp,
TCS_CLOSE,table2.GEOMETRIC_CLOSE
from table1 inner join table2 on
table1.timestamp=table2.timestamp;
select * from t4;

select MIN(TCS_CLOSE-GEOMETRIC_CLOSE),
MAX(TCS_CLOSE-GEOMETRIC_CLOSE) from t4;

SET @average=0;
select (min(TCS_CLOSE-GEOMETRIC_CLOSE)+
MAX(TCS_CLOSE-GEOMETRIC_CLOSE))/2
into @average from t4;

select * from t4 where
|(TCS_CLOSE-GEOMETRIC_CLOSE)|>@average;

```

1 x

lect \* from t4 wher Enter a SQL expression to filter results (u. ▶ |

timestamp	TCS_CLOSE	GEOMETRIC_CLOSE
01-APR-2014	2,176.7	12
01-APR-2015	2,542.65	177.5
01-AUG-2013	1,815.4	64.
01-AUG-2014	2,516.4	140.
01-DEC-2014	2,692.95	130.2
01-JAN-2014	2,153.3	103.
01-JAN-2015	2,545.55	128.
01-JUL-2014	2,390.75	145.
01-JUL-2015	2,593.1	117.

Value x 01-APR-2014

Refresh Save Cancel

Q19)

**Based on nsedata, create table t5 such that it contains the average close price of each company traded in the month of April 2012. The table should be sorted in descending order of the average close price.:**

```
SET @average=0;  
select (min(TCS_CLOSE-GEOMETRIC_CLOSE)+  
MAX(TCS_CLOSE-GEOMETRIC_CLOSE))/2  
into @average from t4;  
select * from t4 where  
(TCS_CLOSE-GEOMETRIC_CLOSE)>@average;  
create table t5 as  
select symbol,  
avg(close) as  
CLOSE_PRICE_AVERAGE from  
nsedata where timestamp  
like "___-APR-2012" group  
by symbol order by  
avg(close) desc;  
select * from t5;
```

The screenshot shows a SQL IDE with a script editor, an output window, and a table view.

**Script Editor:**

```

SET @average=0;
select (min(TCS_CLOSE-GEOM
MAX(TCS_CLOSE-GEOMETRIC_C
into @average from t4;

select * from t4 where
(TCS_CLOSE-GEOMETRIC_CLOS

create table t5 as
select symbol,
avg(close) as
CLOSE_PRICE_AVERAGE from
nsedata where timestamp
like "__-APR-2012" group
by symbol order by
avg(close) desc;
select * from t5;

```

**Output Window:**

Enter a part of a message to search for here

Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI  
Data truncated for column 'CI

**Table View:**

Grid	symbol	CLOSE_PRICE_AVERAGE
1	ORISSAMINE	34,041.3868421053
2	MRF	10,993.7
3	SBIN	8,620.1764646465
4	BOSCHLTD	8,504.1
5	TIDEWATER	7,691.1236842105
6	NESTLEIND	4,813.3921052632
7	3MINDIA	4,157.8105263158
8	ALFALAVAL	3,935.85
9	GODFRYPHLP	3,574.0394736842
10	ASIANPAINT	3,359.1184210526

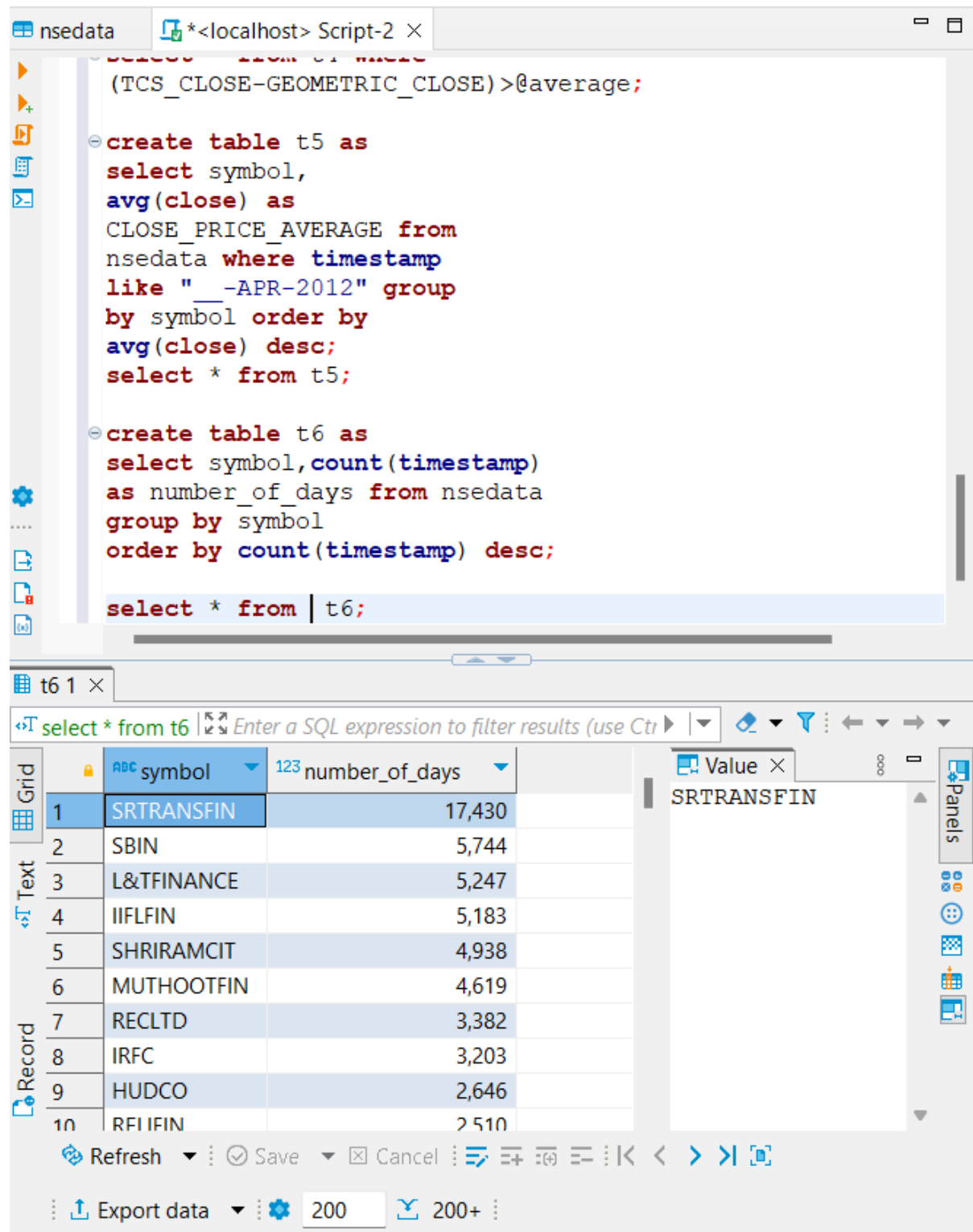
Value x  
ORISSAMINE

Refresh Save Cancel Export data 200 200+ 200 row(s) fetched - 261ms, on 2023-10-22 at 16:53:34

Q20)

Not all companies are traded every day. It is required to create a table that contains a count of the days each company has been traded. The table should be sorted in

descending order  
of the count.:



The screenshot shows a SQL IDE window titled "nsedata" and "Script-2". The script contains the following SQL code:

```

-- (TCS_CLOSE-GEOMETRIC_CLOSE)>@average;

create table t5 as
select symbol,
avg(close) as
CLOSE_PRICE_AVERAGE from
nsedata where timestamp
like "__-APR-2012" group
by symbol order by
avg(close) desc;
select * from t5;

create table t6 as
select symbol, count(timestamp)
as number_of_days from nsedata
group by symbol
order by count(timestamp) desc;

select * from t6;

```

Below the script, the results of the query "select \* from t6" are displayed in a grid. The grid has two columns: "symbol" and "number\_of\_days". The results are sorted in descending order of "number\_of\_days".

	symbol	number_of_days
1	SRTRANSFIN	17,430
2	SBIN	5,744
3	L&TFINANCE	5,247
4	IIFLFIN	5,183
5	SHRIRAMCIT	4,938
6	MUTHOOTFIN	4,619
7	RECLTD	3,382
8	IRFC	3,203
9	HUDCO	2,646
10	RFI IFIN	2,510

The IDE also shows a "Value" panel on the right with the text "SRTRANSFIN". At the bottom, there are buttons for "Refresh", "Save", "Cancel", and "Export data".

Code used(for reference):

```

use stockdata;
show create table nsedata;
select count(*) from nsedata;
select count(*) from nsedata where timestamp like "__-OCT-2012";
select count(*) from nsedata where symbol="GEOMETRIC";
select * from nsedata where symbol="GEOMETRIC" limit 10;
select count(*) from nsedata where symbol="INFY";
select symbol,open,high,low,close,timestamp
from nsedata where symbol ="3IINFOTECH" limit 10;
create temporary table t1 as
select symbol,open,high,low,close,timestamp
from nsedata where symbol="3IINFOTECH" limit 10;
select * from t1;
SELECT
MIN(CLOSE),
AVG(CLOSE),
STDDEV(CLOSE),
MAX(CLOSE),
VAR_POP(CLOSE) AS VARIANCE
from t1;
SELECT AVG(close) AS median
FROM (
SELECT close
FROM t1
ORDER BY close ASC
LIMIT 2 OFFSET 4
) AS t2;
drop temporary table t1;
create table t2 as select symbol,avg(close) as average

```

```

from nsedata n group by symbol;

select * from t2;

create table t3 as

select symbol,

open,

close

from nsedata where symbol="GEOMETRIC"

and timestamp like "__-OCT-2012";

ALTER TABLE t3 add average float;

update t3 set average=(open+close)/2;

select * from t3;

create table table1 as select close as TCS_CLOSE,

timestamp from nsedata where symbol="TCS";

create table table2 as select close as GEOMETRIC_CLOSE,

timestamp from nsedata where symbol="GEOMETRIC";

create table t4 as select table1.timestamp,

TCS_CLOSE,table2.GEOMETRIC_CLOSE

from table1 inner join table2 on

table1.timestamp=table2.timestamp;

select * from t4;

select MIN(TCS_CLOSE-GEOMETRIC_CLOSE),

MAX(TCS_CLOSE-GEOMETRIC_CLOSE) from t4;

SET @average=0;

select (min(TCS_CLOSE-GEOMETRIC_CLOSE)+

MAX(TCS_CLOSE-GEOMETRIC_CLOSE))/2

into @average from t4;

select * from t4 where

(TCS_CLOSE-GEOMETRIC_CLOSE)>@average;

create table t5 as

```

```
select symbol,  
avg(close) as  
CLOSE_PRICE_AVERAGE from  
nsedata where timestamp  
like "___-APR-2012" group  
by symbol order by  
avg(close) desc;  
select * from t5;  
create table t6 as  
select symbol,count(timestamp)  
as number_of_days from nsedata  
group by symbol  
order by count(timestamp) desc;  
select * from t6;
```

The above code has been executed line by line,one line at a time.