

Name = Sakshi Suhas Shinde

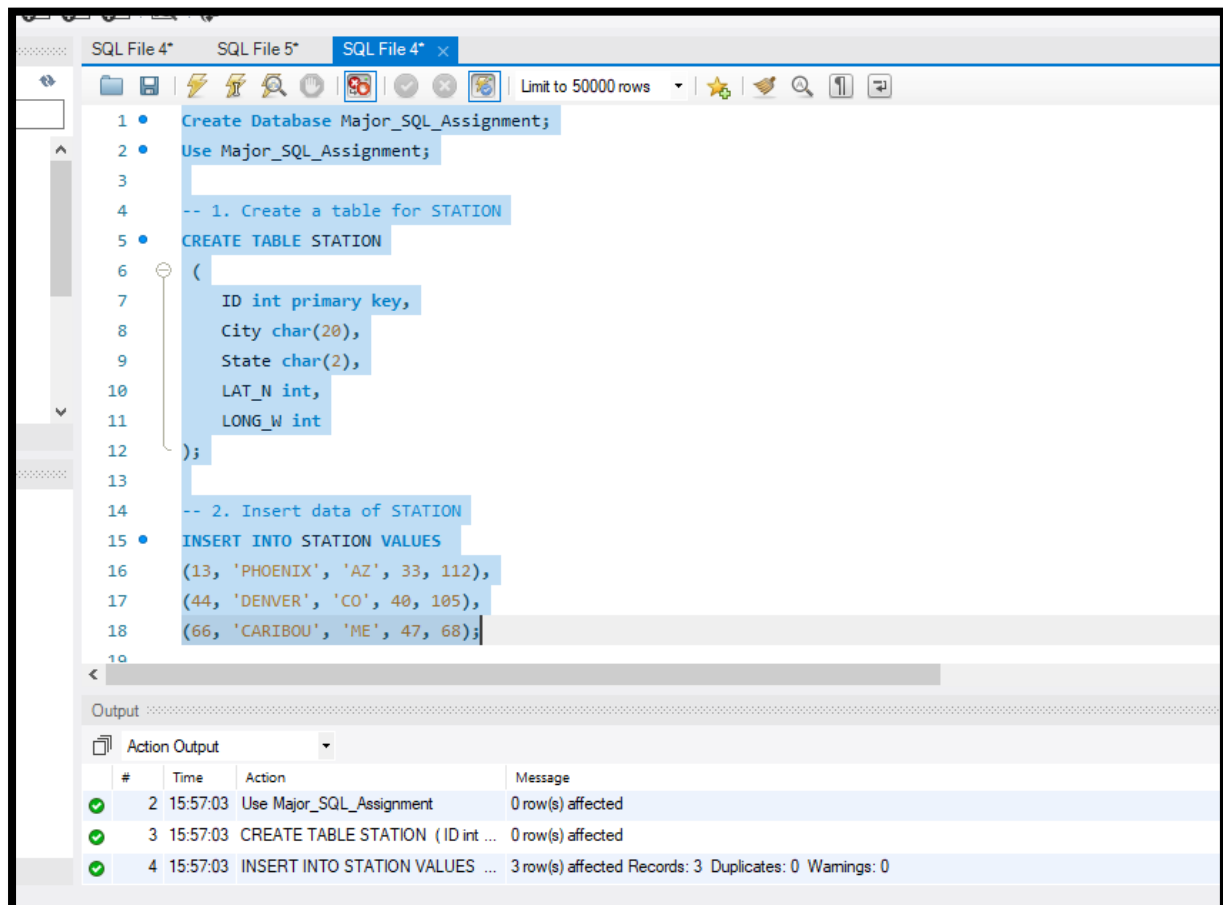
Email = sakshishinde209@gmail.com

Course = Business Analyst

Major SQL Assignment

Q1) Create a table "STATION "

Q2) Insert the following records into the table:



The screenshot shows a SQL IDE with a script editor and an output window. The script editor contains the following SQL code:

```
1 • Create Database Major_SQL_Assignment;
2 • Use Major_SQL_Assignment;
3
4 -- 1. Create a table for STATION
5 • CREATE TABLE STATION
6 (
7     ID int primary key,
8     City char(20),
9     State char(2),
10    LAT_N int,
11    LONG_W int
12 );
13
14 -- 2. Insert data of STATION
15 • INSERT INTO STATION VALUES
16 (13, 'PHOENIX', 'AZ', 33, 112),
17 (44, 'DENVER', 'CO', 40, 105),
18 (66, 'CARIBOU', 'ME', 47, 68);
```

The output window shows the execution results:

#	Time	Action	Message
✓ 2	15:57:03	Use Major_SQL_Assignment	0 row(s) affected
✓ 3	15:57:03	CREATE TABLE STATION (ID int ...	0 row(s) affected
✓ 4	15:57:03	INSERT INTO STATION VALUES ...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0

Q3) Execute a query to look at table STATION in undefined order.

The screenshot shows a SQL IDE interface with a query editor and a results panel. The query editor contains the following SQL code:

```
12 );
13
14 -- 2. Insert data of STATION
15 • INSERT INTO STATION VALUES
16 (13, 'PHOENIX', 'AZ', 33, 112),
17 (44, 'DENVER', 'CO', 40, 105),
18 (66, 'CARIBOU', 'ME', 47, 68);
19
20 -- 3. Execute a query to look at table STATION in undefined order.
21 • SELECT * FROM STATION;
22
```

The results panel displays a table with the following data:

ID	City	State	LAT_N	LONG_W
13	PHOENIX	AZ	33	112
44	DENVER	CO	40	105
66	CARIBOU	ME	47	68
NULL	NULL	NULL	NULL	NULL

The output panel shows the following actions and messages:

#	Time	Action	Message
3	15:57:03	CREATE TABLE STATION (ID int...	0 row(s) affected
4	15:57:03	INSERT INTO STATION VALUES ...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
5	15:57:52	SELECT * FROM STATION LIMIT ...	3 row(s) returned

Q4) Execute a query to select Northern stations (Northern latitude > 39.7).

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

```
15 • INSERT INTO STATION VALUES
16 (13, 'PHOENIX', 'AZ', 33, 112),
17 (44, 'DENVER', 'CO', 40, 105),
18 (66, 'CARIBOU', 'ME', 47, 68);
19
20 -- 3. Execute a query to look at table STATION in undefined order.
21 • SELECT * FROM STATION;
22
23 -- 4. Execute a query to select Northern stations (Northern latitude > 39.7).
24 • SELECT * FROM STATION WHERE LAT_N > 39.7;
25
```

The results pane displays the 'Result Grid' for the query 'SELECT * FROM STATION WHERE LAT_N > 39.7'. The grid shows the following data:

ID	City	State	LAT_N	LONG_W
44	DENVER	CO	40	105
66	CARIBOU	ME	47	68
NULL	NULL	NULL	NULL	NULL

Below the result grid, the 'Output' pane shows the 'Action Output' for the executed queries:

#	Time	Action	Message
4	15:57:03	INSERT INTO STATION VALUES ...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
5	15:57:52	SELECT * FROM STATION LIMIT ...	3 row(s) returned
6	15:58:57	SELECT * FROM STATION WHERE...	2 row(s) returned

Q5) Create another table, 'STATS' Q6) Populate the table

The screenshot shows a SQL IDE with two tabs: 'SQL File 4*' and 'SQL File 5*'. The active tab 'SQL File 4*' contains the following SQL code:

```

26 -- 5. Create a table for STATS
27 CREATE TABLE STATS
28 (
29     ID int references STATION(ID),
30     MONTH int check (MONTH between 1 AND 12),
31     TEMP_F real check (TEMP_F between -80 AND 150),
32     RAIN_I real check (RAIN_I between 0 AND 100),
33     primary key (ID, MONTH)
34 );
35
36 -- 6. Insert data of STATS
37 INSERT INTO STATS VALUES
38 (13,1,57.4,.31),
39 (13,7,91.7,5.15),
40 (44,1,27.3,.18),
41 (44,7,74.8,2.11),
42 (66,1,6.7,2.1),
43 (66,7,65.8,4.52);
44
45

```

The Output pane shows the execution results:

#	Time	Action	Message
7	16:03:14	CREATE TABLE STATS (ID int ...	0 row(s) affected
8	16:03:14	INSERT INTO STATS VALUES (1...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0

The screenshot shows the same SQL IDE with the following SQL code:

```

36 -- 6. Insert data of STATS
37 INSERT INTO STATS VALUES
38 (13,1,57.4,.31),
39 (13,7,91.7,5.15),
40 (44,1,27.3,.18),
41 (44,7,74.8,2.11),
42 (66,1,6.7,2.1),
43 (66,7,65.8,4.52);
44
45 SELECT * FROM STATS;
46

```

The Result Grid shows the data retrieved from the STATS table:

ID	MONTH	TEMP_F	RAIN_I
13	1	57.4	0.31
13	7	91.7	5.15
44	1	27.3	0.18
44	7	74.8	2.11
66	1	6.7	2.1
66	7	65.8	4.52

The Output pane shows the execution results:

#	Time	Action	Message
1	16:14:54	SELECT * FROM STATS LIMIT 0, 5...	6 row(s) returned

Q7) Execute a query to display temperature stats (from the STATS table) for each city (from the STATION table).

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

```
39 (13,7,91.7,5.15),
40 (44,1,27.3,.18),
41 (44,7,74.8,2.11),
42 (66,1,6.7,2.1),
43 (66,7,65.8,4.52);
44
45 • SELECT * FROM STATS;
46
47 -- 7. Execute a query to display temperature stats (from the STATS table) for each city (from the STATION table).
48 • SELECT STATION.City, STATS.MONTH, STATS.TEMP_F
49 FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID;
50
```

The results pane displays a table with the following data:

City	MONTH	TEMP_F
PHOENIX	1	57.4
PHOENIX	7	91.7
DENVER	1	27.3
DENVER	7	74.8
CARIBOU	1	6.7
CARIBOU	7	65.8

The results pane also shows a message: "Result 9 x" and "Output". The Action Output section shows a successful execution of the query: "SELECT * FROM STATS LIMIT 0, 5... 6 row(s) returned".

Q8) Execute a query to look at the table STATS, ordered by month and greatest rainfall, with columns rearranged. It should also show the corresponding cities.

SQL File 4* SQL File 5* SQL File 4* x

```

44
45 • SELECT * FROM STATS;
46
47 -- 7. Execute a query to display temperature stats (from the STATS table) for each city (from the STATION table).
48 • SELECT STATION.City, STATS.MONTH, STATS.TEMP_F
49 FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID;
50
51 -- 8. Execute a query to look at the table STATS, ordered by month and greatest rainfall, with columns rearranged. I
52 • SELECT STATION.City, STATS.MONTH, STATS.RAIN_I, STATS.TEMP_F
53 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID ORDER BY STATS.MONTH, STATS.RAIN_I DESC;

```

Result Grid | Filter Rows: | Exports: | Wrap Cell Content: IA

City	MONTH	RAIN_I	TEMP_F
CARIBOU	1	2.1	6.7
PHOENIX	1	0.31	57.4
DENVER	1	0.18	27.3
PHOENIX	7	5.15	91.7
CARIBOU	7	4.52	65.8
DENVER	7	2.11	74.8

Result 10 x

Output

Action Output

#	Time	Action	Message
2	16:19:54	SELECT STATION.City, STATS.M...	6 row(s) returned

SQL File 4* SQL File 5* SQL File 4* x SQL File 5* SQL File 6*

```

49 FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID;
50
51 -- 8. Execute a query to look at the table STATS, ordered by month and greatest rainfall, with columns rearran
52 • SELECT STATION.City, STATS.MONTH, STATS.RAIN_I
53 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID ORDER BY STATS.MONTH, STATS.RAIN_I DESC;
54
55 -- 9. Execute a query to look at temperatures for July from table STATS. Lowest temperatures first, picking up

```

Result Grid | Filter Rows: | Exports: | Wrap Cell Content: IA

City	MONTH	RAIN_I
CARIBOU	1	2.11
PHOENIX	1	0.32
DENVER	1	0.19
PHOENIX	7	5.16
CARIBOU	7	4.529999999999999
DENVER	7	2.1199999999999997

Result 4 x

Output

Action Output

#	Time	Action	Message
3	18:29:27	SELECT STATION.City, STATS.MONTH, STATS.RAIN_I, STATS.TEMP_F FROM STATS I...	6 row(s) returned
4	18:33:14	SELECT STATION.City, STATS.MONTH, STATS.RAIN_I FROM STATS INNER JOIN STA...	6 row(s) returned

Q9) Execute a query to look at temperatures for July from table STATS, lowest temperatures first, picking up city name and latitude.

The screenshot shows a SQL Developer window with a query executed. The query is as follows:

```

49 FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID;
50
51 -- 8. Execute a query to look at the table STATS, ordered by month and greatest rainfall, with columns rearranged
52 • SELECT STATION.City, STATS.MONTH, STATS.RAIN_I, STATS.TEMP_F
53 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID ORDER BY STATS.MONTH, STATS.RAIN_I DESC;
54
55 -- 9. Execute a query to look at temperatures for July from table STATS, lowest temperatures first, picking up city name and latitude
56 • SELECT STATION.City, STATION.LAT_N, STATS.TEMP_F
57 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;
58

```

The results are displayed in a table with the following data:

City	LAT_N	TEMP_F
CARIBOU	47	65.8
DENVER	40	74.8
PHOENIX	33	91.7

The bottom of the window shows the 'Action Output' pane with the following message:

#	Time	Action	Message
1	16:33:32	SELECT STATION.City, STATION.LAT_N, STATS.TEMP_F FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;	3 row(s) returned

The screenshot shows a SQL Developer window with a query executed. The query is as follows:

```

53 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID ORDER BY STATS.MONTH, STATS.RAIN_I DESC;
54
55 -- 9. Execute a query to look at temperatures for July from table STATS, lowest temperatures first, picking up city name and latitude
56 • SELECT STATION.City, STATION.LAT_N, STATS.TEMP_F, STATS.MONTH
57 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;
58
59 -- 10. Execute a query to show MAY and MTN temperatures as well as average rainfall for each city

```

The results are displayed in a table with the following data:

City	LAT_N	TEMP_F	MONTH
CARIBOU	47	65.8	7
DENVER	40	74.9	7
PHOENIX	33	91.7	7

The bottom of the window shows the 'Action Output' pane with the following messages:

#	Time	Action	Message
5	18:53:54	SELECT STATION.City, STATION.LAT_N, STATS.TEMP_F FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;	3 row(s) returned
6	18:54:34	SELECT STATION.City, STATION.LAT_N, STATS.TEMP_F, STATS.MONTH FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;	3 row(s) returned

Q10) Execute a query to show MAX and MIN temperatures as well as average rainfall for each city.

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

```
53 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID ORDER BY STATS.MONTH, STATS.RAIN_I DESC;
54
55 -- 9. Execute a query to look at temperatures for July from table STATS, lowest temperatures first, picking up city n
56 • SELECT STATION.City, STATION.LAT_N, STATS.TEMP_F
57 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;
58
59 -- 10. Execute a query to show MAX and MIN temperatures as well as average rainfall for each city.
60 • SELECT STATION.City, max(TEMP_F) as Max_Temperature, min(TEMP_F) as Min_Temperature, avg(RAIN_I) as Average_Rainfall
61 FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID group by STATION.City;
62
```

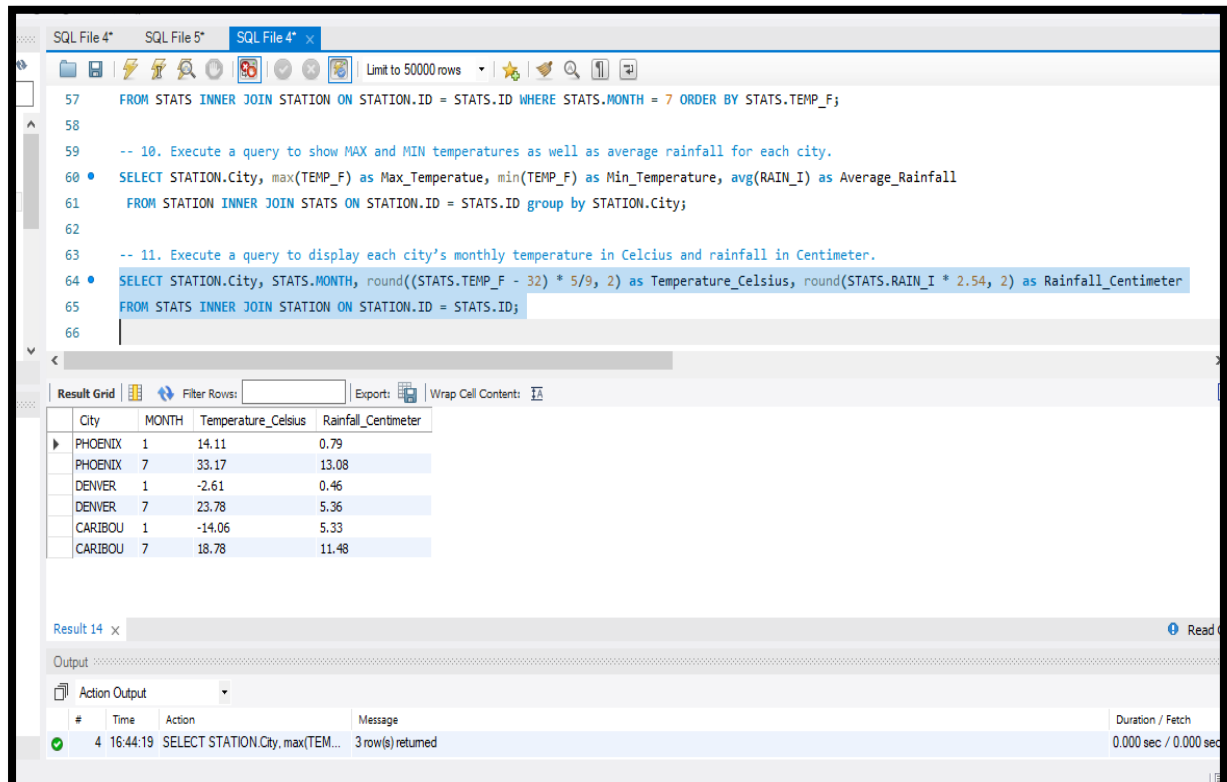
The results grid displays the following data:

City	Max_Temperature	Min_Temperature	Average_Rainfall
PHOENIX	91.7	57.4	2.73
DENVER	74.8	27.3	1.145
CARIBOU	65.8	6.7	3.3099999999999996

Below the results grid, the 'Output' section shows the execution details for 'Result 13':

#	Time	Action	Message
3	16:43:30	SELECT STATION.City, max(TEMP_F) as Max_Temperature, min(TEMP_F) as Min_Temperature, avg(RAIN_I) as Average_Rainfall FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID group by STATION.City;	1 row(s) returned

Q11) Execute a query to display each city's monthly temperature in Celsius and rainfall in Centimetre.



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

```
57 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID WHERE STATS.MONTH = 7 ORDER BY STATS.TEMP_F;
58
59 -- 10. Execute a query to show MAX and MIN temperatures as well as average rainfall for each city.
60 • SELECT STATION.City, max(TEMP_F) as Max_Temperature, min(TEMP_F) as Min_Temperature, avg(RAIN_I) as Average_Rainfall
61 FROM STATION INNER JOIN STATS ON STATION.ID = STATS.ID group by STATION.City;
62
63 -- 11. Execute a query to display each city's monthly temperature in Celcius and rainfall in Centimeter.
64 • SELECT STATION.City, STATS.MONTH, round((STATS.TEMP_F - 32) * 5/9, 2) as Temperature_Celsius, round(STATS.RAIN_I * 2.54, 2) as Rainfall_Centimeter
65 FROM STATS INNER JOIN STATION ON STATION.ID = STATS.ID;
66
```

The results grid displays the following data:

City	MONTH	Temperature_Celsius	Rainfall_Centimeter
PHOENIX	1	14.11	0.79
PHOENIX	7	33.17	13.08
DENVER	1	-2.61	0.46
DENVER	7	23.78	5.36
CARIBOU	1	-14.06	5.33
CARIBOU	7	18.78	11.48

The output section shows the following message:

```
4 16:44:19 SELECT STATION.City, max(TEM... 3 row(s) returned 0.000 sec / 0.000 sec
```

Q12) Update all rows of table STATS to compensate for faulty rain gauges known to read 0.01 inches low.

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

```
66
67 -- 12. Update all rows of table STATS to compensate for faulty rain gauges known to read 0.01 inches low.
68 • UPDATE STATS SET RAIN_I = RAIN_I + 0.01 WHERE ID > 0;
69
70 • SELECT * FROM STATS;
71
```

The results pane displays a table with the following data:

ID	MONTH	TEMP_F	RAIN_I
13	1	57.4	0.32
13	7	91.7	5.16
44	1	27.3	0.19
44	7	74.8	2.1199999999999997
66	1	6.7	2.11
66	7	65.8	4.5299999999999999
NULL	NULL	NULL	NULL

The results pane also shows an "Output" section with the following action output:

#	Time	Action	Message
1	16:55:50	UPDATE STATS SET RAIN_I = RAIN_I + 0.01 WHERE ID > 0;	6 row(s) affected Rows matched: 6 Changed: 6 Warnings: 0
2	16:57:00	SELECT * FROM STATS LIMIT 0, 50000	6 row(s) returned

Q13) Update Denver's July temperature reading as 74.9

SQL File 4* SQL File 5* SQL File 4* x

Limit to 50000 rows

```

69 • SELECT * FROM STATS;
70
71 -- 13. Update Denver's July temperature reading as 74.9.
72 • UPDATE STATS SET TEMP_F = 74.9 WHERE ID = (SELECT ID FROM STATION WHERE City = 'DENVER') AND MONTH = 7;
73 • SELECT * FROM STATION;
74 • SELECT * FROM STATS;

```

Result Grid

ID	City	State	LAT_N	LONG_W
13	PHOENIX	AZ	33	112
44	DENVER	CO	40	105
66	CARIBOU	ME	47	68
NULL	NULL	NULL	NULL	NULL

STATION 10 x

Output

Action Output

#	Time	Action	Message
1	16:55:50	UPDATE STATS SET RAIN_I = RAIN_I + 0...	6 row(s) affected Rows matched: 6 Changed: 6 Warnings: 0
2	16:57:00	SELECT * FROM STATS LIMIT 0, 50000	6 row(s) returned
3	16:59:57	UPDATE STATS SET TEMP_F = 74.9 WHE...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
4	17:01:14	SELECT * FROM STATION LIMIT 0, 50000	3 row(s) returned

SQL File 4* SQL File 5* SQL File 4* x

Limit to 50000 rows

```

69 • SELECT * FROM STATS;
70
71 -- 13. Update Denver's July temperature reading as 74.9.
72 • UPDATE STATS SET TEMP_F = 74.9 WHERE ID = (SELECT ID FROM STATION WHERE City = 'DENVER') AND MONTH = 7;
73 • SELECT * FROM STATION;
74 • SELECT * FROM STATS;

```

Result Grid

ID	MONTH	TEMP_F	RAIN_I
13	1	57.4	0.32
13	7	91.7	5.16
44	1	27.3	0.19
44	7	74.9	2.1199999999999997
66	1	6.7	2.11
66	7	65.8	4.5299999999999999
NULL	NULL	NULL	NULL

STATS 11 x

Output

Action Output

#	Time	Action	Message
1	16:55:50	UPDATE STATS SET RAIN_I = RAIN_I + 0...	6 row(s) affected Rows matched: 6 Changed: 6 Warnings: 0
2	16:57:00	SELECT * FROM STATS LIMIT 0, 50000	6 row(s) returned
3	16:59:57	UPDATE STATS SET TEMP_F = 74.9 WHE...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
4	17:01:14	SELECT * FROM STATION LIMIT 0, 50000	3 row(s) returned
5	17:01:51	SELECT * FROM STATS LIMIT 0, 50000	6 row(s) returned