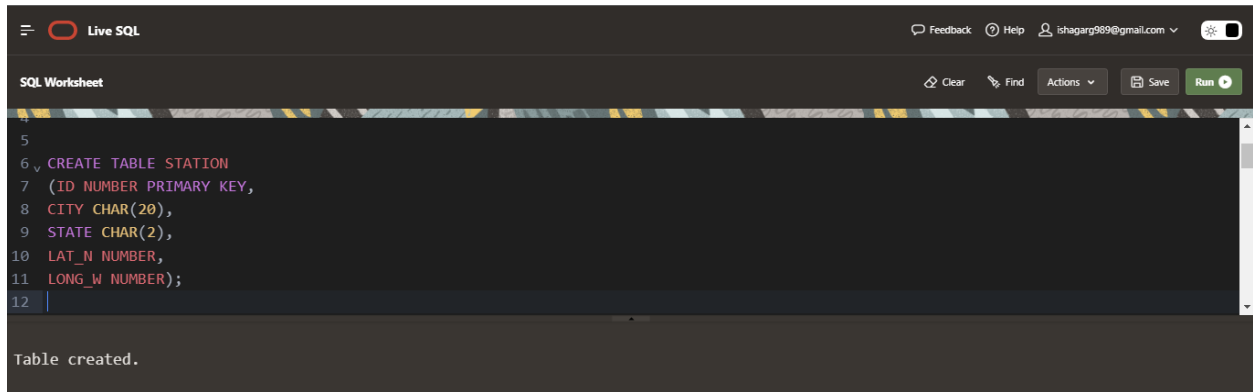


1. Create a table "Station" to store information about weather observation stations:

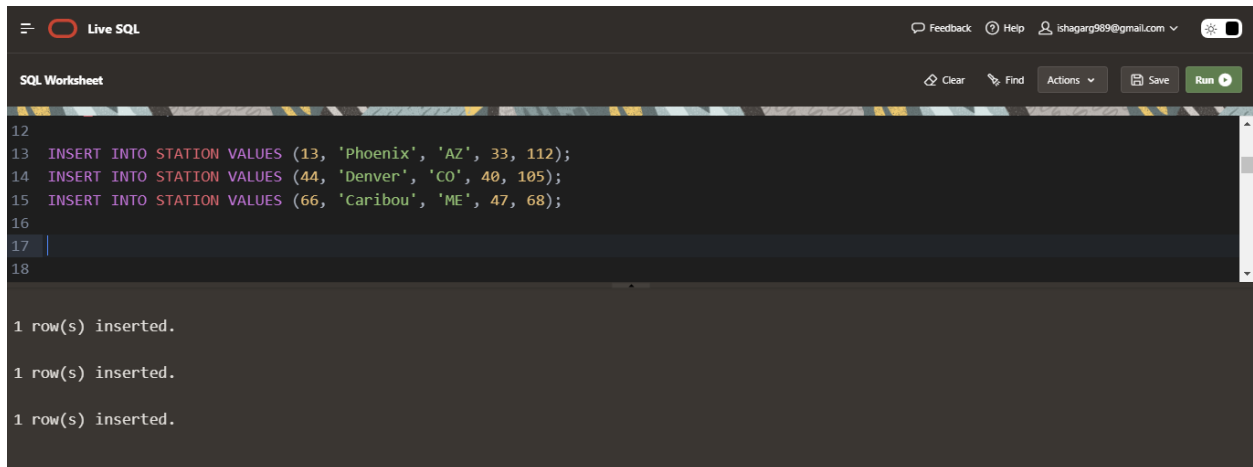


The screenshot shows the Live SQL interface with a dark theme. The SQL Worksheet area contains the following SQL code:

```
5  
6 CREATE TABLE STATION  
7 (ID NUMBER PRIMARY KEY,  
8 CITY CHAR(20),  
9 STATE CHAR(2),  
10 LAT_N NUMBER,  
11 LONG_W NUMBER);  
12
```

Below the code editor, the output area displays the message: "Table created."

2. Insert the following records into the table:



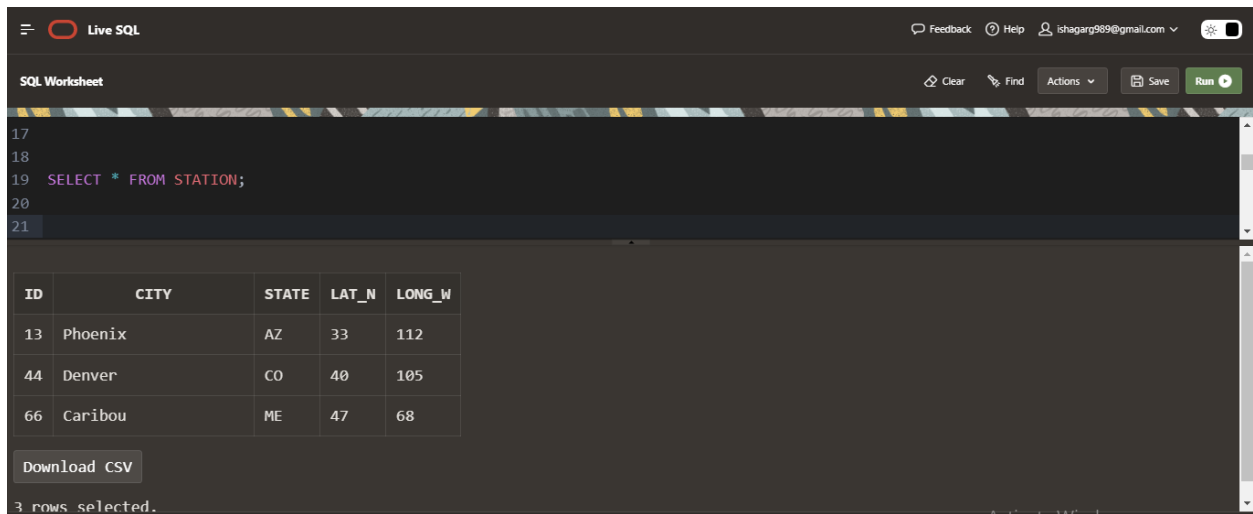
The screenshot shows the Live SQL interface with the following SQL code in the worksheet:

```
12  
13 INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);  
14 INSERT INTO STATION VALUES (44, 'Denver', 'CO', 40, 105);  
15 INSERT INTO STATION VALUES (66, 'Caribou', 'ME', 47, 68);  
16  
17  
18
```

The output area shows the results of the three insert statements:

```
1 row(s) inserted.  
1 row(s) inserted.  
1 row(s) inserted.
```

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



The screenshot shows the Live SQL interface with the following SQL code in the worksheet:

```
17  
18  
19 SELECT * FROM STATION;  
20  
21
```

The output area displays the results of the query in a table format:

ID	CITY	STATE	LAT_N	LONG_W
13	Phoenix	AZ	33	112
44	Denver	CO	40	105
66	Caribou	ME	47	68

Below the table, there is a "Download CSV" button and the message: "3 rows selected."

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

The screenshot shows the Live SQL interface with a query executed. The query is: `SELECT * FROM STATION WHERE LAT_N > 39.7;`. The result is a table with 2 rows selected.

ID	CITY	STATE	LAT_N	LONG_W
44	Denver	CO	40	105
66	Caribou	ME	47	68

Download CSV

2 rows selected.

5. Create another table, 'STATS', to store normalized temperature and precipitation data:

The screenshot shows the Live SQL interface with a query executed to create a table named 'STATS'. The query is: `CREATE TABLE STATS (ID NUMBER REFERENCES STATION(ID), MONTH NUMBER CHECK (MONTH BETWEEN 1 AND 12), TEMP_F REAL CHECK (TEMP_F BETWEEN -80 AND 150), RAIN_I REAL CHECK (RAIN_I BETWEEN 0 AND 100), PRIMARY KEY (ID, MONTH));`. The result is: Table created.

6. Populate the table STATS with some statistics for January and July:

The screenshot shows the Live SQL interface with a query executed to insert data into the 'STATS' table. The query is: `INSERT INTO STATS VALUES (13, 1, 57.4, 0.31);`, `INSERT INTO STATS VALUES (13, 7, 91.7, 5.15);`, `INSERT INTO STATS VALUES (44, 1, 27.3, 0.18);`, `INSERT INTO STATS VALUES (44, 7, 74.8, 2.11);`, `INSERT INTO STATS VALUES (66, 1, 6.7, 2.10);`, and `INSERT INTO STATS VALUES (66, 7, 65.8, 4.52);`. The result is: 1 row(s) inserted. (repeated 5 times).

- Execute a query to display temperature stats (from STATS table) for each city (from Station table).

Live SQL

SQL Worksheet

```
SELECT S.ID, ST.CITY, S.MONTH, S.TEMP_F, S.RAIN_I FROM STATS S INNER JOIN STATION ST ON ST.ID = S.ID;
```

ID	CITY	MONTH	TEMP_F	RAIN_I
13	Phoenix	1	57.4	.31
13	Phoenix	7	91.7	5.15
44	Denver	1	27.3	.18
44	Denver	7	74.8	2.11
66	Caribou	1	6.7	2.1
66	Caribou	7	65.8	4.52

- 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.

Live SQL

SQL Worksheet

```
SELECT ST.CITY, S.MONTH, S.RAIN_I, S.TEMP_F FROM STATION ST INNER JOIN STATS S ON ST.ID = S.ID  
ORDER BY MONTH, RAIN_I DESC;
```

CITY	MONTH	RAIN_I	TEMP_F
Caribou	1	2.1	6.7
Phoenix	1	.31	57.4
Denver	1	.18	27.3
Phoenix	7	5.15	91.7
Caribou	7	4.52	65.8
Denver	7	2.11	74.8

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

The screenshot shows the Live SQL interface with a query executed. The query is: `SELECT ST.CITY, S.MONTH, ST.LAT_N, S.TEMP_F FROM STATION ST INNER JOIN STATS S ON ST.ID = S.ID WHERE S.MONTH = 7 ORDER BY S.TEMP_F;` The result is a table with 4 columns: CITY, MONTH, LAT\_N, and TEMP\_F. The data rows are: Caribou (7, 47, 65.8), Denver (7, 40, 74.8), and Phoenix (7, 33, 91.7). A 'Download CSV' button is visible below the table, and the status '3 rows selected.' is shown at the bottom.

CITY	MONTH	LAT_N	TEMP_F
Caribou	7	47	65.8
Denver	7	40	74.8
Phoenix	7	33	91.7

Download CSV

3 rows selected.

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

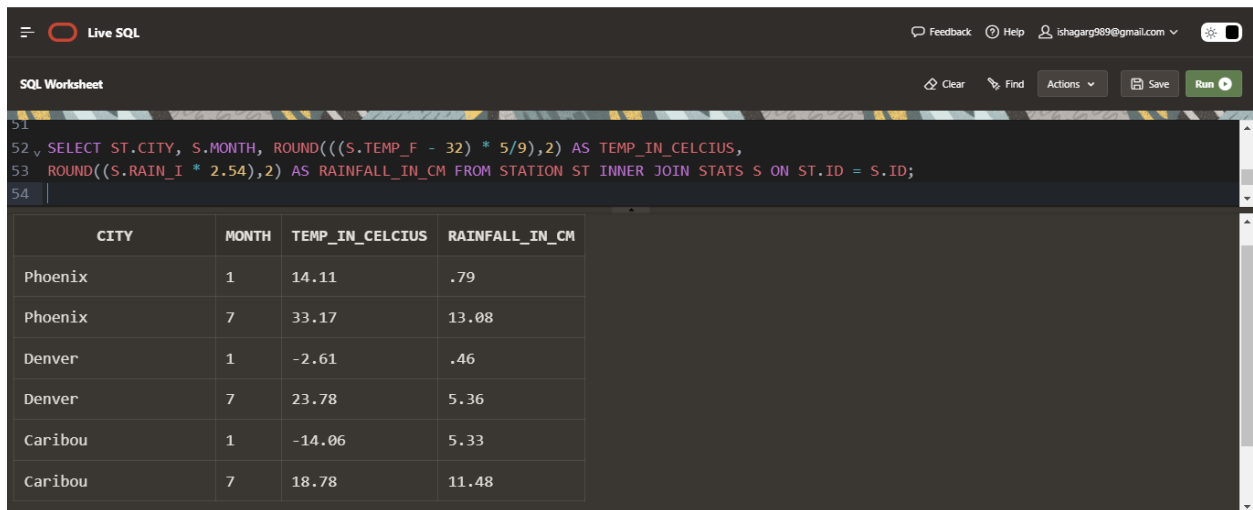
The screenshot shows the Live SQL interface with a query executed. The query is: `SELECT ST.CITY, MAX(S.TEMP_F) AS MAX_TEMP_F, MIN(S.TEMP_F) AS MIN_TEMP_F, ROUND(AVG(S.RAIN_I),2) AS AVG_RAIN_I FROM STATS S INNER JOIN STATION ST ON S.ID = ST.ID GROUP BY ST.CITY;` The result is a table with 4 columns: CITY, MAX\_TEMP\_F, MIN\_TEMP\_F, and AVG\_RAIN\_I. The data rows are: Caribou (65.8, 6.7, 3.31), Denver (74.8, 27.3, 1.15), and Phoenix (91.7, 57.4, 2.73). A 'Download CSV' button is visible below the table, and the status '3 rows selected.' is shown at the bottom.

CITY	MAX_TEMP_F	MIN_TEMP_F	AVG_RAIN_I
Caribou	65.8	6.7	3.31
Denver	74.8	27.3	1.15
Phoenix	91.7	57.4	2.73

Download CSV

3 rows selected.

11. Execute a query to display each city's monthly temperature in Celcius and rainfall in Centimeter.



The screenshot shows the Live SQL interface with the following SQL query:

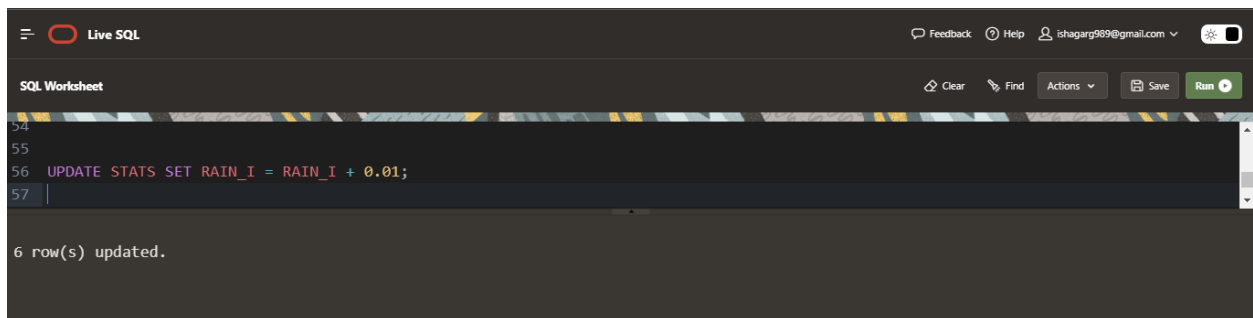
```
52 SELECT ST.CITY, S.MONTH, ROUND(((S.TEMP_F - 32) * 5/9),2) AS TEMP_IN_CELCIUS,  
53 ROUND((S.RAIN_I * 2.54),2) AS RAINFALL_IN_CM FROM STATION ST INNER JOIN STATS S ON ST.ID = S.ID;
```

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

CITY	MONTH	TEMP_IN_CELCIUS	RAINFALL_IN_CM
Phoenix	1	14.11	.79
Phoenix	7	33.17	13.08
Denver	1	-2.61	.46
Denver	7	23.78	5.36
Caribou	1	-14.06	5.33
Caribou	7	18.78	11.48

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

Updating rain:



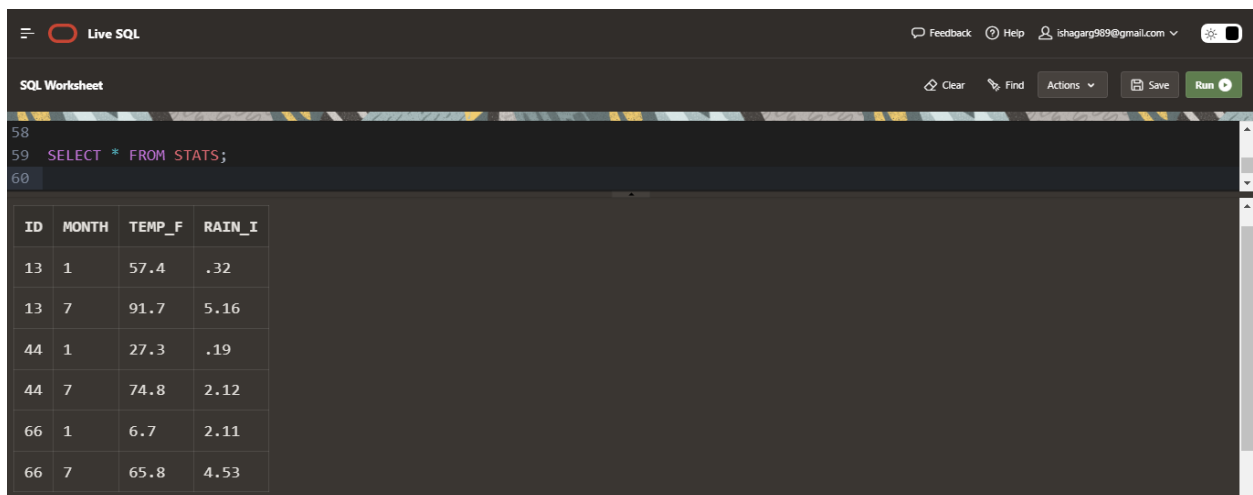
The screenshot shows the Live SQL interface with the following SQL query:

```
56 UPDATE STATS SET RAIN_I = RAIN_I + 0.01;
```

The execution result is displayed as:

6 row(s) updated.

Updated data:



The screenshot shows the Live SQL interface with the following SQL query:

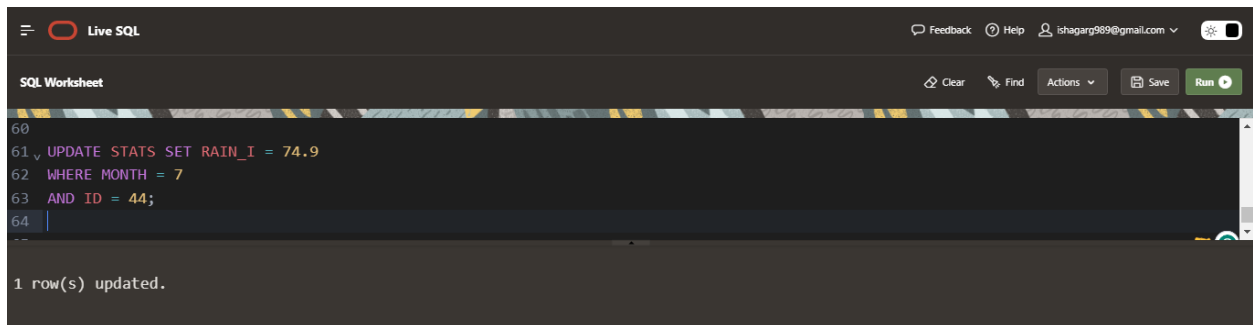
```
59 SELECT * FROM STATS;
```

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

ID	MONTH	TEMP_F	RAIN_I
13	1	57.4	.32
13	7	91.7	5.16
44	1	27.3	.19
44	7	74.8	2.12
66	1	6.7	2.11
66	7	65.8	4.53

### 13. Update Denver's July temperature reading as 74.9

Updating data:



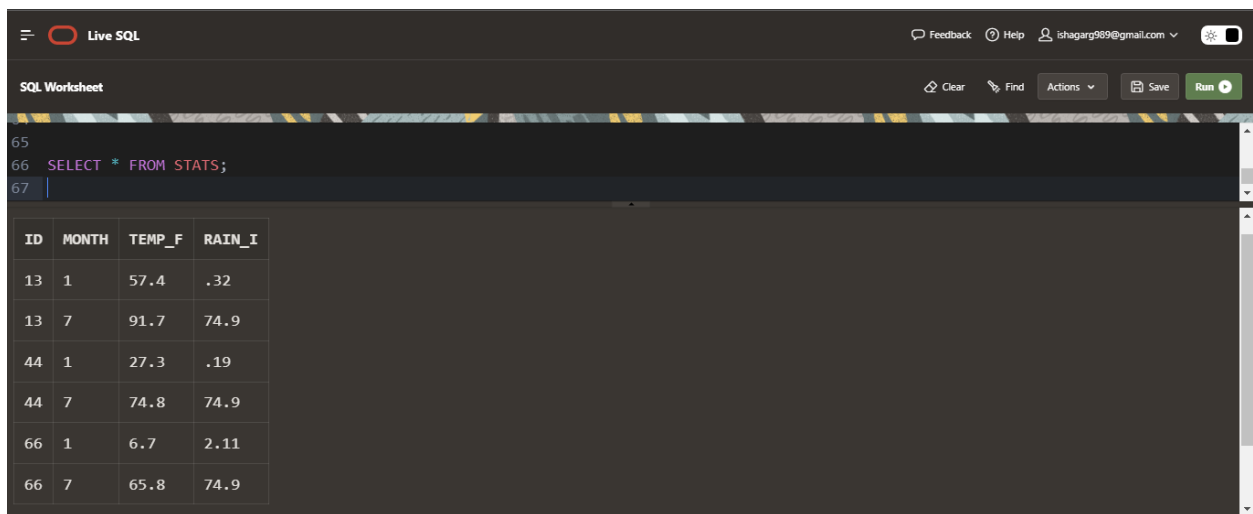
The screenshot shows the Live SQL interface with the following SQL statement entered in the editor:

```
60  
61 UPDATE STATS SET RAIN_I = 74.9  
62 WHERE MONTH = 7  
63 AND ID = 44;  
64
```

Below the editor, the result of the query is displayed:

```
1 row(s) updated.
```

Updated data:



The screenshot shows the Live SQL interface with the following SQL statement entered in the editor:

```
65  
66 SELECT * FROM STATS;  
67
```

Below the editor, the result of the query is displayed as a table:

ID	MONTH	TEMP_F	RAIN_I
13	1	57.4	.32
13	7	91.7	74.9
44	1	27.3	.19
44	7	74.8	74.9
66	1	6.7	2.11
66	7	65.8	74.9