

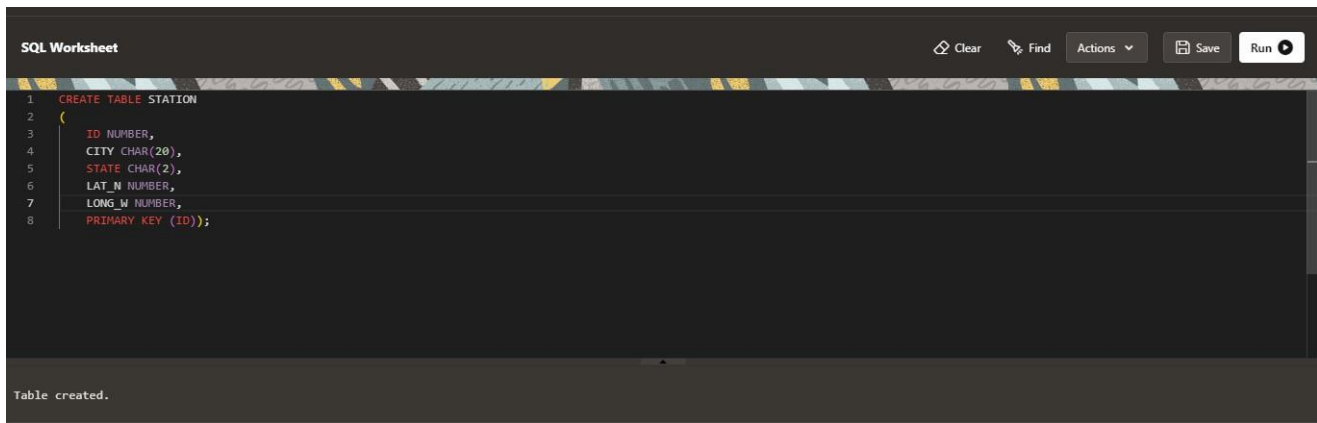
Assignment – Advance SQL [Major] by Hemant Raikwar

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

ID	Number	Primary key
CITY	CHAR(20)	
STATE	CHAR(2)	
LAT_N	Number	

QUERY -

```
CREATE TABLE STATION
(  
  ID NUMBER,  
  CITY CHAR(20),  
  STATE CHAR(2),  
  LAT_N NUMBER,  
  LONG_W NUMBER,  
  PRIMARY KEY (ID));
```



The screenshot shows a SQL Worksheet interface with a dark theme. At the top, there are buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area contains the following SQL code:

```
1 CREATE TABLE STATION
2 (
3   ID NUMBER,
4   CITY CHAR(20),
5   STATE CHAR(2),
6   LAT_N NUMBER,
7   LONG_W NUMBER,
8   PRIMARY KEY (ID));
```

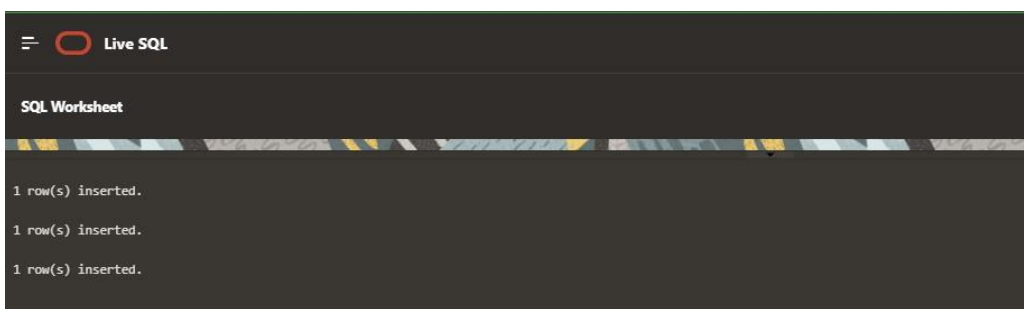
Below the code editor, a status message reads "Table created."

2. Insert the following records into the table:

ID	CITY	STATE	LAT_N	LONG_W
13	PHONEIX	AZ	33	112
44	DENVER	CO	40	105
66	CARIBOU	ME	47	68

QUERY--

```
INSERT INTO STATION (ID, CITY, STATE, LAT_N, LONG_W) VALUES (13, 'PHONEIX', 'AZ', 33, 112);  
INSERT INTO STATION (ID, CITY, STATE, LAT_N, LONG_W) VALUES (44, 'DENVER', 'CO', 40, 105);  
INSERT INTO STATION (ID, CITY, STATE, LAT_N, LONG_W) VALUES (66, 'CARIBOU', 'ME', 47, 68);
```



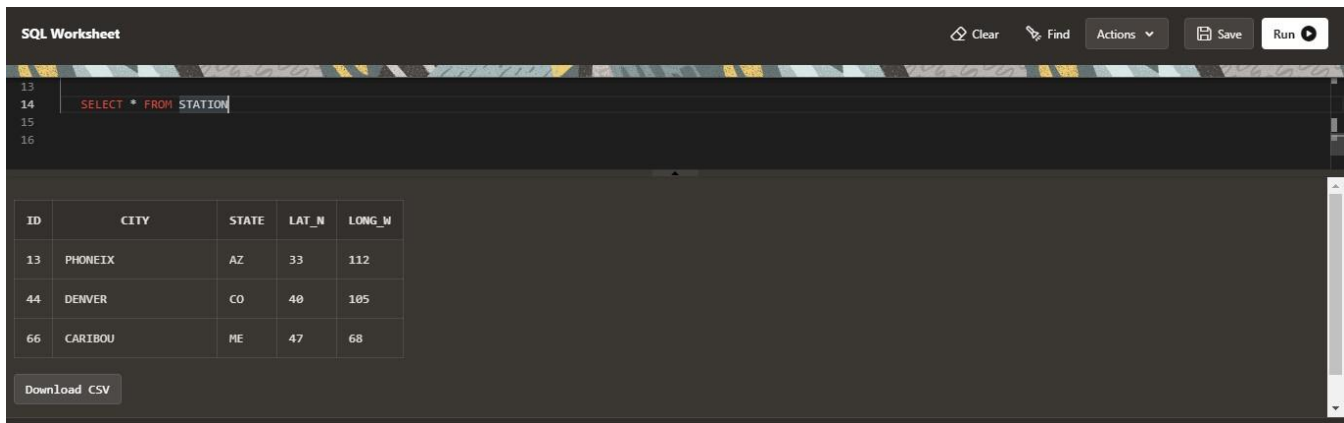
The screenshot shows a SQL Worksheet interface with a dark theme. At the top, there are buttons for 'Live SQL', 'SQL Worksheet', and 'Run'. The main area contains the following SQL code:

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

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

QUERY -

```
SELECT * FROM STATION
```



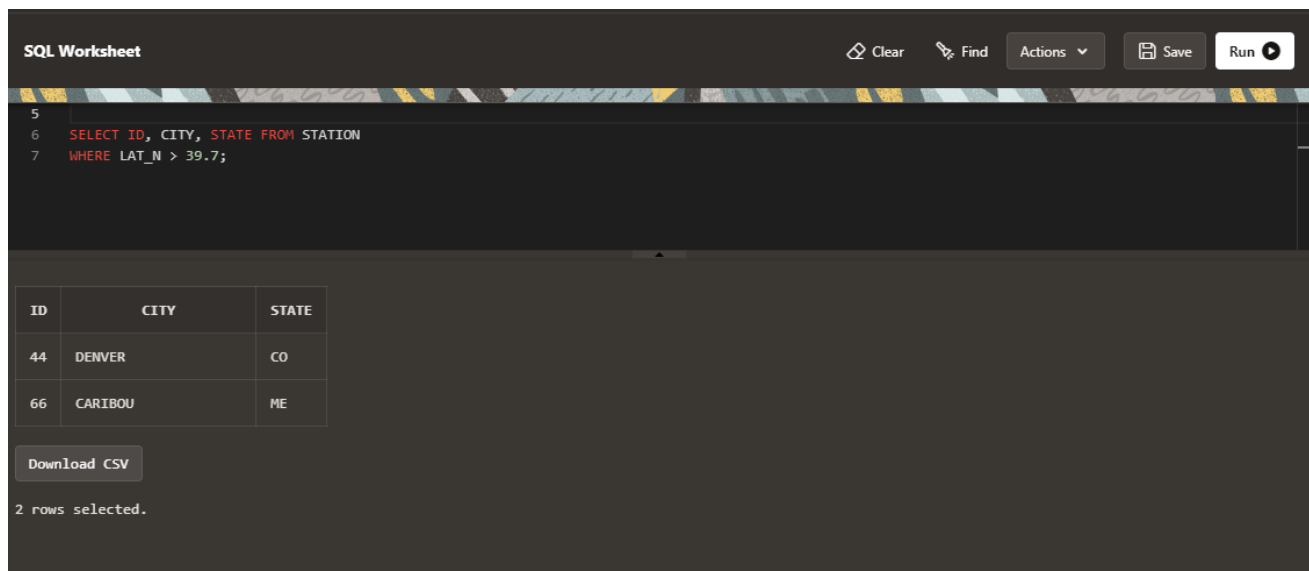
The screenshot shows an SQL Worksheet interface with a dark theme. At the top, there's a toolbar with 'Clear', 'Find', 'Actions', 'Save', and 'Run' buttons. The query editor contains the text 'SELECT * FROM STATION'. Below the editor, a table of results is displayed with 5 columns: ID, CITY, STATE, LAT_N, and LONG_W. The table has 3 rows of data. A 'Download CSV' button is located at the bottom left of the results area.

ID	CITY	STATE	LAT_N	LONG_W
13	PHONEIX	AZ	33	112
44	DENVER	CO	40	105
66	CARIBOU	ME	47	68

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

QUERY-

```
SELECT ID, CITY, STATE FROM STATION  
WHERE LAT_N > 39.7;
```



The screenshot shows an SQL Worksheet interface with a dark theme. The query editor contains the text 'SELECT ID, CITY, STATE FROM STATION WHERE LAT_N > 39.7;'. Below the editor, a table of results is displayed with 3 columns: ID, CITY, and STATE. The table has 2 rows of data. A 'Download CSV' button is located at the bottom left of the results area. Below the table, it says '2 rows selected.'.

ID	CITY	STATE
44	DENVER	CO
66	CARIBOU	ME

ID	CITY	STATE	LAT_N	LONG_W
44	DENVER	CO	40	105
66	CARIBOU	ME	47	68

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

Column	Data type	Remark
ID	Number	must match some STATION table ID(so name & location will be known).
MONTH	Number	Range between 1 and 12

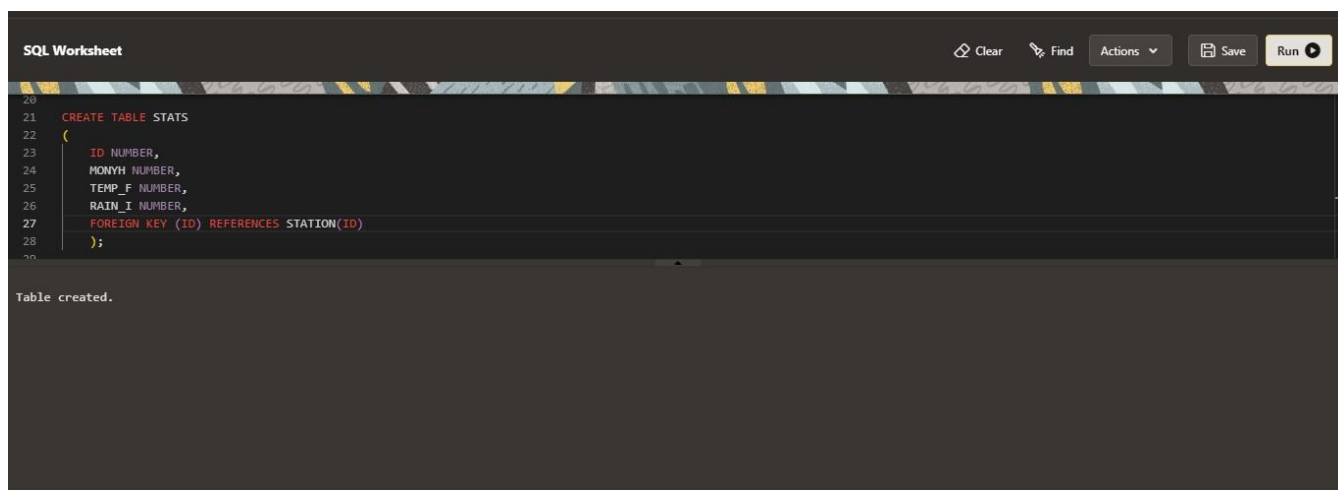
TEMP_F	Number	in Fahrenheit degrees, Range between -80 and 150
RAIN_I	Number	in inches, Range between 0 and 100

There will be no Duplicate ID and MONTH combination.

QUERY –

CREATE TABLE STATS

```
(
    ID NUMBER,
    MONTH NUMBER,
    TEMP_F NUMBER,
    RAIN_I NUMBER,
    FOREIGN KEY (ID) REFERENCES STATION(ID)
);
```



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

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

QUERY –

INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (13,1,57.4,.31);

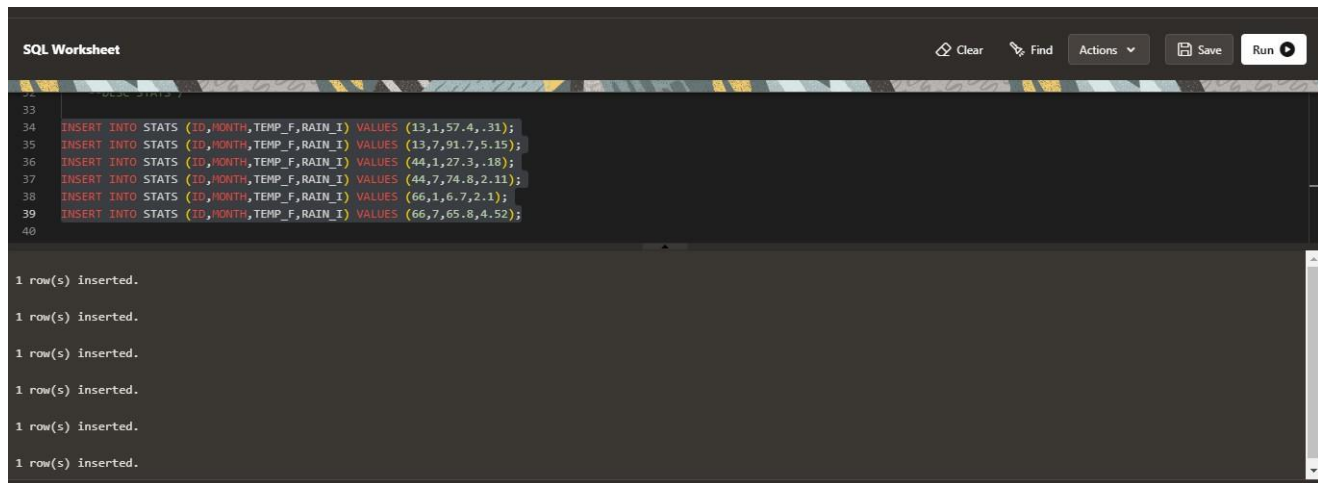
INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (13,7,91.7,5.15);

INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (44,1,27.3,.18);

INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (44,7,74.8,2.11);

INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (66,1,6.7,2.1);

INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (66,7,65.8,4.52);



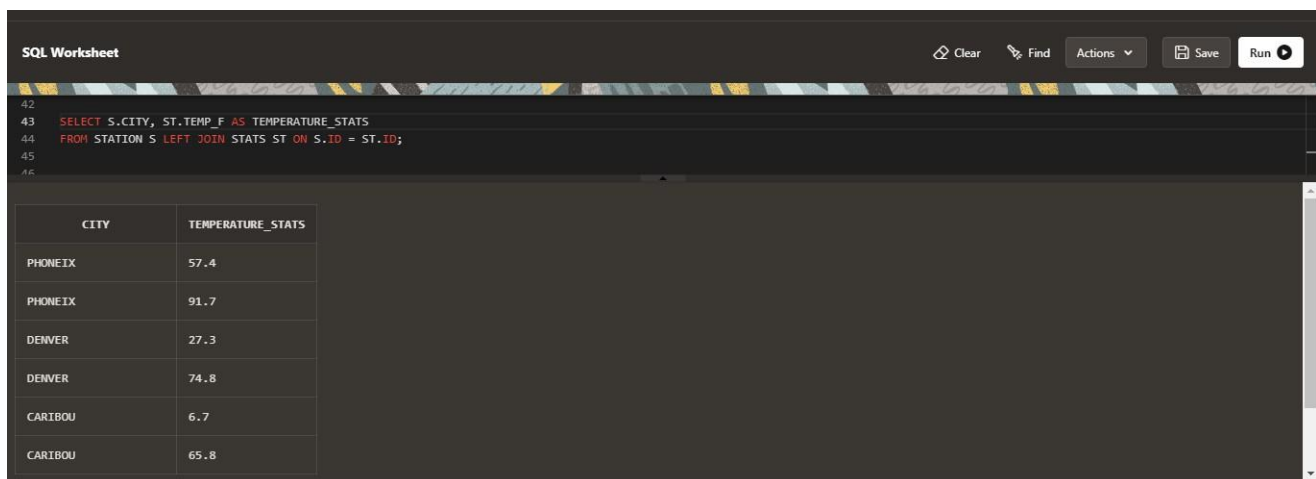
The screenshot shows a SQL Worksheet interface with a dark theme. The top bar includes 'Clear', 'Find', 'Actions', 'Save', and 'Run' buttons. The SQL editor contains six INSERT statements for the STATS table. The execution results pane shows six lines, each stating '1 row(s) inserted.'.

```
33
34 INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (13,1,57.4,.31);
35 INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (13,7,91.7,5.15);
36 INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (44,1,27.3,.18);
37 INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (44,7,74.8,2.11);
38 INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (66,1,6.7,2.1);
39 INSERT INTO STATS (ID,MONTH,TEMP_F,RAIN_I) VALUES (66,7,65.8,4.52);
40
```

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

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

```
SELECT S.CITY, ST.TEMP_F AS TEMPERATURE_STATS
FROM STATION S LEFT JOIN STATS ST ON S.ID = ST.ID;
```



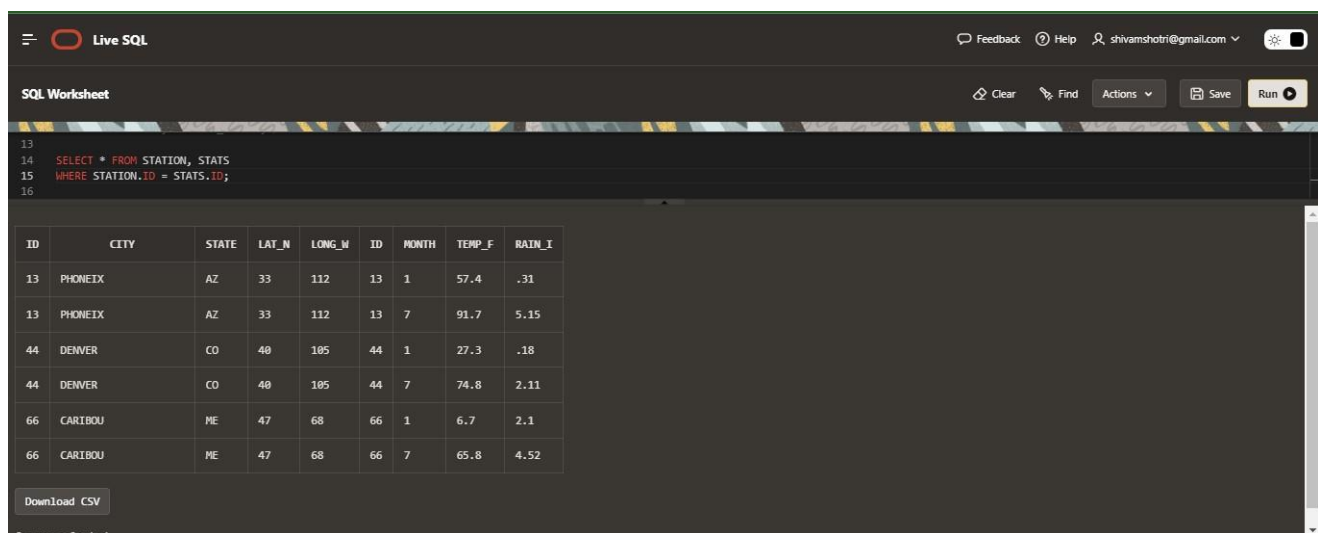
The screenshot shows a SQL Worksheet with a query that joins the STATION and STATS tables. The results are displayed in a table with two columns: CITY and TEMPERATURE_STATS.

```
42
43 SELECT S.CITY, ST.TEMP_F AS TEMPERATURE_STATS
44 FROM STATION S LEFT JOIN STATS ST ON S.ID = ST.ID;
45
46
```

CITY	TEMPERATURE_STATS
PHONEIX	57.4
PHONEIX	91.7
DENVER	27.3
DENVER	74.8
CARIBOU	6.7
CARIBOU	65.8

QUERY –

```
SELECT * FROM STATION, STATS
WHERE STATION.ID=STATS.ID
```



The screenshot shows the Live SQL interface. The top bar includes 'Feedback', 'Help', a user profile, and a settings icon. The SQL editor contains a query that joins STATION and STATS tables. The results are displayed in a table with columns: ID, CITY, STATE, LAT_N, LONG_W, ID, MONTH, TEMP_F, and RAIN_I. A 'Download CSV' button is visible at the bottom left.

```
13
14 SELECT * FROM STATION, STATS
15 WHERE STATION.ID = STATS.ID;
16
```

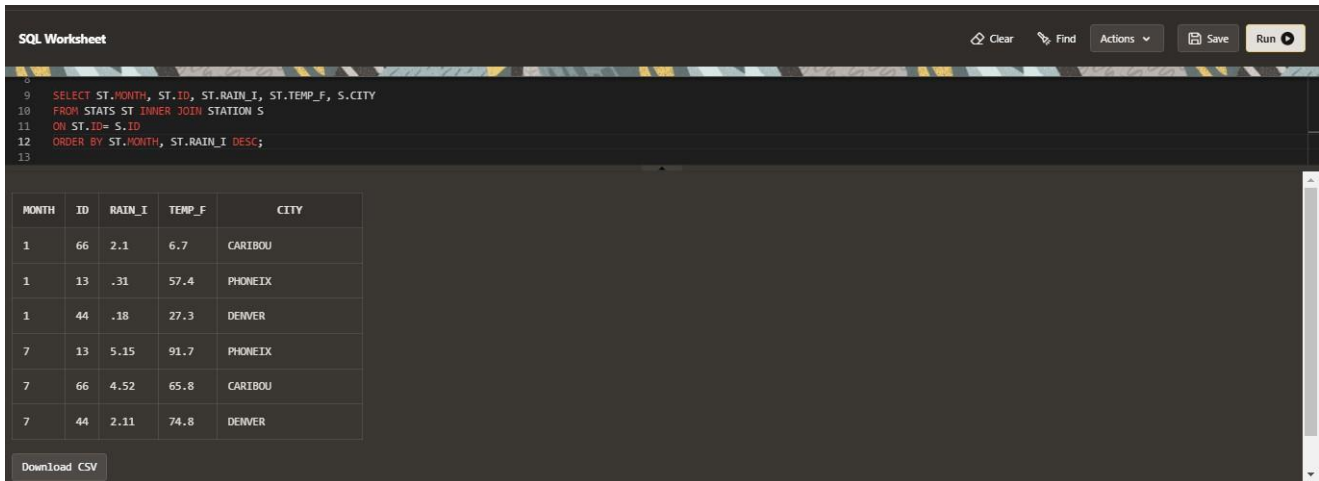
ID	CITY	STATE	LAT_N	LONG_W	ID	MONTH	TEMP_F	RAIN_I
13	PHONEIX	AZ	33	112	13	1	57.4	.31
13	PHONEIX	AZ	33	112	13	7	91.7	5.15
44	DENVER	CO	40	105	44	1	27.3	.18
44	DENVER	CO	40	105	44	7	74.8	2.11
66	CARIBOU	ME	47	68	66	1	6.7	2.1
66	CARIBOU	ME	47	68	66	7	65.8	4.52

Download CSV
6 rows selected

8. 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.

QUERY –

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



The screenshot shows a SQL Worksheet interface with a query editor at the top and a results table below. The query is: `SELECT ST.MONTH, ST.ID, ST.RAIN_I, ST.TEMP_F, S.CITY FROM STATS ST INNER JOIN STATION S ON ST.ID= S.ID ORDER BY ST.MONTH, ST.RAIN_I DESC;`. The results table has 7 rows and 5 columns: MONTH, ID, RAIN_I, TEMP_F, and CITY. The data is as follows:

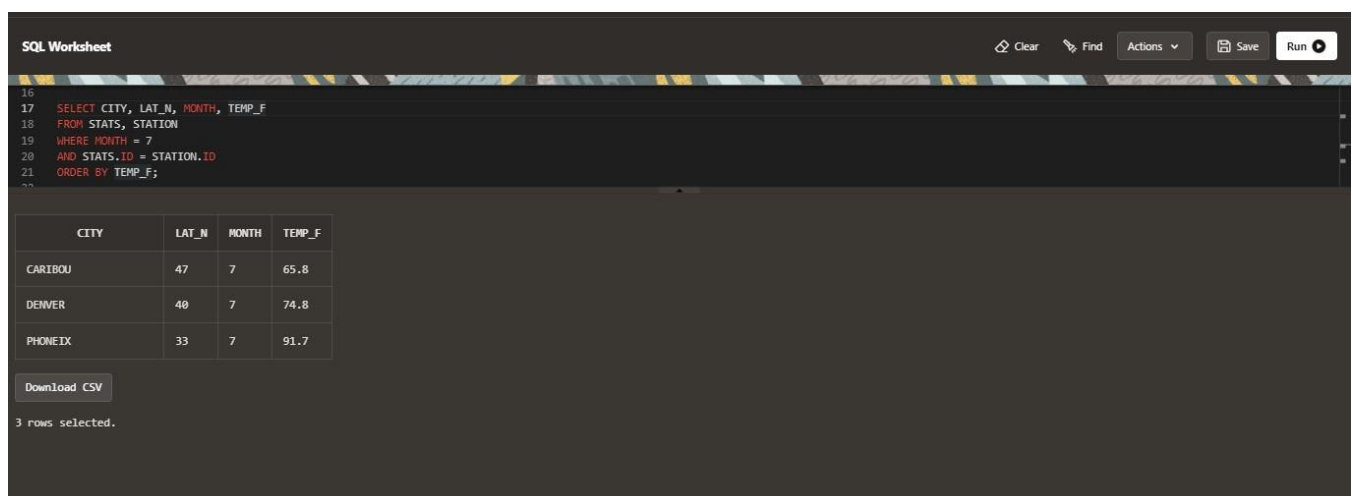
MONTH	ID	RAIN_I	TEMP_F	CITY
1	66	2.1	6.7	CARIBOU
1	13	.31	57.4	PHONEIX
1	44	.18	27.3	DENVER
7	13	5.15	91.7	PHONEIX
7	66	4.52	65.8	CARIBOU
7	44	2.11	74.8	DENVER

Below the table is a "Download CSV" button.

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

QUERY -

```
SELECT CITY, LAT_N, MONTH, TEMP_F
FROM STATS, STATION
WHERE MONTH = 7
AND STATS.ID = STATION.ID
ORDER BY TEMP_F;
```



The screenshot shows a SQL Worksheet interface with a query editor at the top and a results table below. The query is: `SELECT CITY, LAT_N, MONTH, TEMP_F FROM STATS, STATION WHERE MONTH = 7 AND STATS.ID = STATION.ID ORDER BY TEMP_F;`. The results table has 4 columns: CITY, LAT_N, MONTH, and TEMP_F. The data is as follows:

CITY	LAT_N	MONTH	TEMP_F
CARIBOU	47	7	65.8
DENVER	40	7	74.8
PHONEIX	33	7	91.7

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

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

```
SELECT ID, MAX(TEMP_F) AS MAX_TEMPERATURE,
MIN(TEMP_F) AS MIN_TEMPERATURE,
AVG(RAIN_I) AS AVG_RAINFALL
FROM STATS
GROUP BY ID
ORDER BY ID;
```

SQL Worksheet

Clear

Find

Actions

Save

Run

```
23 SELECT ID, MAX(TEMP_F) AS MAX_TEMPERATURE,
24 MIN(TEMP_F) AS MIN_TEMPERATURE,
25 AVG(RAIN_I) AS AVG_RAINFALL
26 FROM STATS
27 GROUP BY ID
28 ORDER BY ID;
```

ID	MAX_TEMPERATURE	MIN_TEMPERATURE	AVG_RAINFALL
13	91.7	57.4	2.73
44	74.8	27.3	1.145
66	65.8	6.7	3.31

Download CSV

3 rows selected.

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

```
SELECT CITY, (TEMP_F-32)*5/9 AS TEMPERATURE_CELCIUS,  
       RAIN_I*2.54 AS RAIN_CENTIMETER  
FROM STATION,STATS  
WHERE STATION.ID=STATS.ID;
```

[illegible]

NOTE – CONVERSION DETAILS OF FAHERNHEIT INTO CELCIUS

(TEMP. IN FAHERNHEIT -32) X 5/9

CONVERSION DETAILS OF INCHES TO CENTIMETRE

CENTIMETRE = INCHES X 2.54 (1 INCH = 2.54 CENTIMETRE)

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

QUERY –

```
UPDATE STATS SET RAIN_I = RAIN_I+0.01 ;  
JUST TO SEE THE UPDATE - SELECT * FROM STATS;
```

SQL Worksheet

```
36 UPDATE STATS SET RAIN_I= RAIN_I+0.01;  
37 SELECT * FROM STATS;  
38
```

6 row(s) updated.

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

Download CSV

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

QUERY –

```
UPDATE STATS SET TEMP_F = 74.9  
WHERE ID = 44  
AND MONTH = 7;  
JUST TO SEE THE UPDATE - SELECT * FROM STATS;
```

SQL Worksheet

```
39 UPDATE STATS SET TEMP_F = 74.9  
40 WHERE ID = 44  
41 AND MONTH = 7;  
42 SELECT * FROM STATS;
```

1 row(s) updated.

ID	MONTH	TEMP_F	RAIN_I
13	1	57.4	.32
13	7	91.7	5.16
44	1	27.3	.19
44	7	74.9	2.12
66	1	6.7	2.11
66	7	65.8	4.53

2022 Oracle • Live SQL 22.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 • Database Documentation • Ask Tom • Dev Gym
Built with ❤️ using Oracle APEX • Privacy • Terms of Use

29°C Sunny 04:34 PM 17-12-2022