

Assignment-SQL[Major]

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

SYNTAX:

CREATE TABLE STATIONS

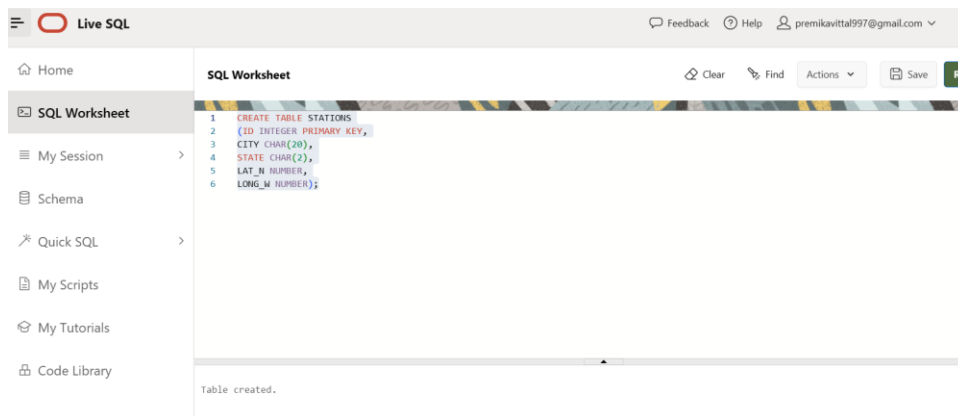
(ID INTEGER PRIMARY KEY,

CITY CHAR (20),

STATE CHAR (2),

LAT_N NUMBER,

LONG_W NUMBER);



2. Insert the following records into the table:

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

SYNTAX:

INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);

INSERT INTO STATION VALUES (44, 'Denver', 'CO', 40, 105);

INSERT INTO STATION VALUES (66, 'Caribou', 'ME', 47, 68);

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

SYNTAX:

SELECT * FROM STATION;

The screenshot shows the Live SQL interface. The SQL Worksheet contains the following code:

```
2 (ID INTEGER PRIMARY KEY,  
3 CITY CHAR(20),  
4 STATE CHAR(2),  
5 LAT_N NUMBER,  
6 LONG_W NUMBER);  
7 INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);  
8 INSERT INTO STATION VALUES (44, 'Denver', 'CO', 40, 105);  
9 INSERT INTO STATION VALUES (66, 'Caribou', 'ME', 47, 68);  
10 SELECT * FROM STATION;
```

The result is displayed as 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

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

STNTAX:

SELECT * FROM STATION

WHERE LAT_N > 39.7;

The screenshot shows the Live SQL interface. The SQL Worksheet contains the following code:

```
4 STATE CHAR(2),  
5 LAT_N NUMBER,  
6 LONG_W NUMBER);  
7 INSERT INTO STATION VALUES (13, 'Phoenix', 'AZ', 33, 112);  
8 INSERT INTO STATION VALUES (44, 'Denver', 'CO', 40, 105);  
9 INSERT INTO STATION VALUES (66, 'Caribou', 'ME', 47, 68);  
10 SELECT * FROM STATION;  
11 SELECT * FROM STATION  
12 WHERE LAT_N > 39.7;
```

The result is displayed as a table with the following data:

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

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

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.

SYNTAX:

CREATE TABLE STATS

(ID INTEGER REFERENCES STATION(ID),

MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),

TEMP_F NUMBER CHECK (TEMP_F BETWEEN -80 AND 150),

RAIN_I NUMBER CHECK (RAIN_I BETWEEN 0 AND 100),

PRIMARY KEY (ID, MONTH));

The screenshot shows a web-based SQL editor interface. On the left is a sidebar with navigation links: Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, and My Tutorials. The main area is titled 'SQL Worksheet' and contains an SQL script. The script includes a SELECT statement, a WHERE clause, and a CREATE TABLE statement for 'STATS'. The 'STATS' table has columns ID, MONTH, TEMP_F, and RAIN_I, with various constraints including foreign key references and CHECK constraints. Below the script, a message states 'Table created.'.

```
10 SELECT * FROM STATION;
11 SELECT * FROM STATION
12 WHERE LAT_N > 39.7;
13 CREATE TABLE STATS
14 (ID INTEGER REFERENCES STATION(ID),
15 MONTH INTEGER CHECK (MONTH BETWEEN 1 AND 12),
16 TEMP_F NUMBER CHECK (TEMP_F BETWEEN -80 AND 150),
17 RAIN_I NUMBER CHECK (RAIN_I BETWEEN 0 AND 100),
18 PRIMARY KEY (ID, MONTH));
```

Table created.

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

SYNTAX:

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);

INSERT INTO STATS VALUES (66, 7, 65.8, 4.52);

The screenshot shows a web-based SQL editor interface. On the left is a sidebar with navigation links: Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area is titled 'SQL Worksheet' and contains a list of SQL statements: line 16 has a check constraint for TEMP_F, line 17 has a check constraint for RAIN_I, line 18 has a primary key constraint for (ID, MONTH), and lines 19-23 contain five INSERT INTO STATS VALUES statements. Below the code editor, the execution results are displayed as a list of '1 row(s) inserted.' messages, corresponding to each insert statement.

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

SYNTAX:

SELECT * FROM STATION, STATS

WHERE STATION.ID = STATS.ID;

The screenshot shows the Live SQL interface with a SQL query entered in the worksheet:

```

25 SELECT * FROM STATION, STATS
26 WHERE STATION.ID = STATS.ID;

```

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

ID	CITY	STATE	LAT_N	LONG_W	ID	MONTH	TEMP_F	RAIN_I
13	Phoenix	AZ	33	112	13	1	57.4	.31
13	Phoenix	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

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.

SYNTAX:

```

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

```

The screenshot shows the Live SQL interface with a SQL query entered in the worksheet:

```

62
63 SELECT MONTH,RAIN_I,CITY,TEMP_F
64 FROM STATS
65 INNER JOIN STATION ON STATS.ID=STATION.ID
66 ORDER BY RAIN_I DESC;

```

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

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

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

SYNTAX:

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

The screenshot shows the Live SQL interface. The SQL Worksheet contains the following query:

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

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

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

There is a 'Download CSV' button below the table.

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

SYNTAX:

```
SELECT MAX(TEMP_F), MIN(TEMP_F), AVG(RAIN_I), ID
FROM STATS
GROUP BY ID;
```

The screenshot shows the Live SQL interface. The SQL Worksheet contains the following query:

```
SELECT MAX(TEMP_F), MIN(TEMP_F), AVG(RAIN_I), ID
FROM STATS
GROUP BY ID;
```

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

MAX(TEMP_F)	MIN(TEMP_F)	AVG(RAIN_I)	ID
74.8	27.3	1.145	44
65.8	6.7	3.31	66
91.7	57.4	2.73	13

There is a 'Download CSV' button below the table.

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

SYNTAX:

```
CREATE VIEW METRIC_STATSS (CITY, MONTH, TEMP_C, RAIN_C) AS
SELECT CITY,
MONTH,
(TEMP_F - 32) * 5 /9,
RAIN_I * 0.3937
FROM STATS,STATION;

SELECT * FROM METRIC_STATSS;
```

Live SQL

Feedback

Help

prenikavittai99@gmail.com

Home

SQL Worksheet

My Session

Schema

Quick SQL

My Scripts

My Tutorials

Code Library

SQL Worksheet

ClearFindActionsSaveRun

CREATE VIEW METRIC_STATS (CITY, MONTH, TEMP_C, RAIN_C) AS
SELECT CITY,
MONTH,
(TEMP_F - 32) * 5 / 9 AS
RAIN_1 * 0.3937
FROM STATS;STATON;

24 SELECT * FROM METRIC_STATS;

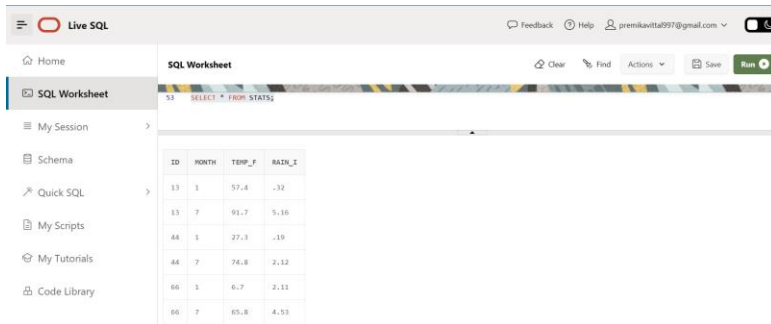
CITY	MONTH	TEMP_C	RAIN_C
Phoenix	1	14.11111111111111111111111111111111111111	-125984
Phoenix	7	33.16666666666666666666666666666666666667	2.031492
Phoenix	1	-2.61111111111111111111111111111111111111	-074803
Phoenix	7	23.83333333333333333333333333333333333333	-834644
Phoenix	1	-14.85555555555555555555555555555555555556	-830707

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

SYNTAX:

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

The screenshot shows the DBeaver SQL Worksheet interface. At the top, there's a header with 'Live SQL' and navigation links. The left sidebar has a menu with 'SQL Worksheet' selected. The main workspace shows a SQL query: `UPDATE STATS SET RAIN_T = RAIN_T * 0.01;`. Below the query, a message indicates '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	74.9	2.12
66	1	6.7	2.11
66	7	65.8	4.53

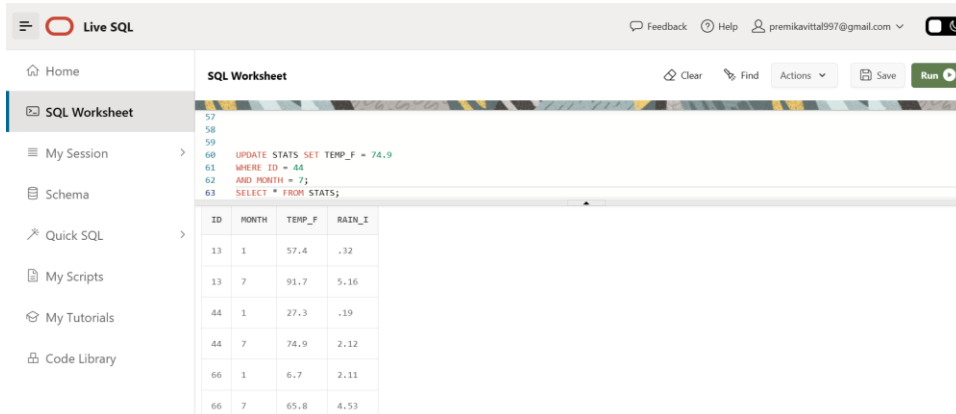
13. Update Denver's July temperature reading as 74.9

SYNTAX:

UPDATE STATS SET TEMP_F = 74.9

WHERE ID = 44

AND MONTH = 7;

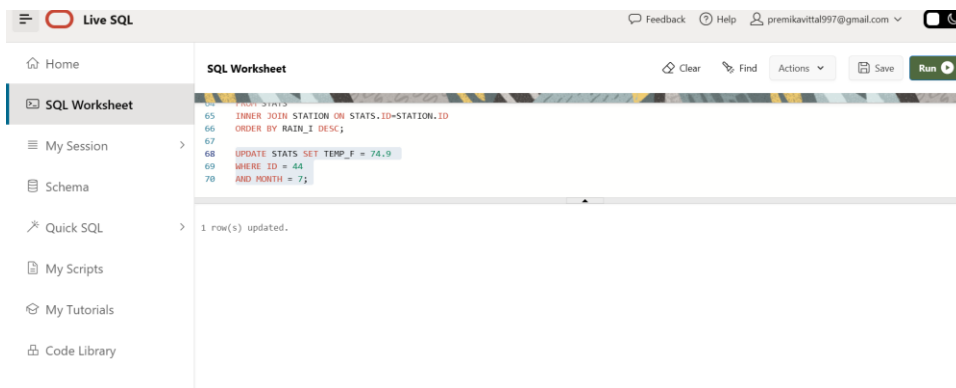


```

57
58
59
60 UPDATE STATS SET TEMP_F = 74.9
61 WHERE ID = 44
62 AND MONTH = 7;
63 SELECT * FROM STATS;

```

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



```

64
65 INNER JOIN STATION ON STATS.ID=STATION.ID
66 ORDER BY RAIN_I DESC;
67
68 UPDATE STATS SET TEMP_F = 74.9
69 WHERE ID = 44
70 AND MONTH = 7;

```

1 row(s) updated.