Data Science with Python Career Program

(ChatGPT Included)

Assignment - SQL [Major]

Gouravy86451@gmail.com

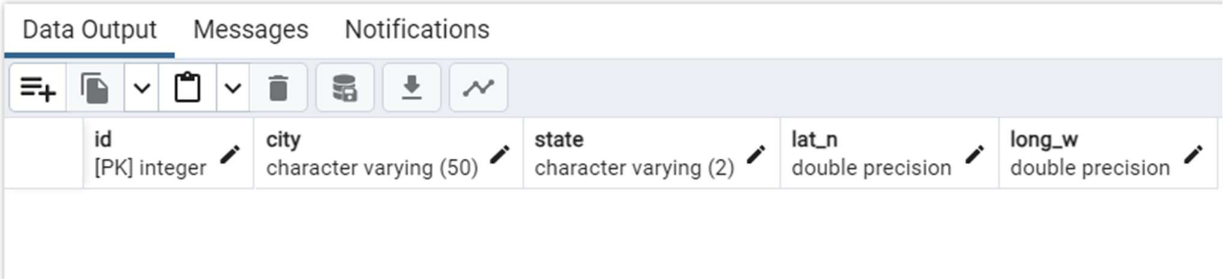
Note: I have used "**PostgreSQL**" for this assignment. Most of these codes are compatible with "**SQL Workbench**" you can alter the code per your requirements.

Q1) Create a table "**STATION**" to store information about the weather observation stations:

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

Ans.

```
CREATE TABLE IF NOT EXISTS Station (
        ID INT PRIMARY KEY,
        CITY VARCHAR(50) NOT NULL,
        STATE VARCHAR(2) NOT NULL,
        LAT_N FLOAT NOT NULL,
        LONG_W FLOAT NOT NULL
);
```



id [PK] integer	city character varying (50)	state character varying (2)	lat_n double precision	long_w double precision

Q2) 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

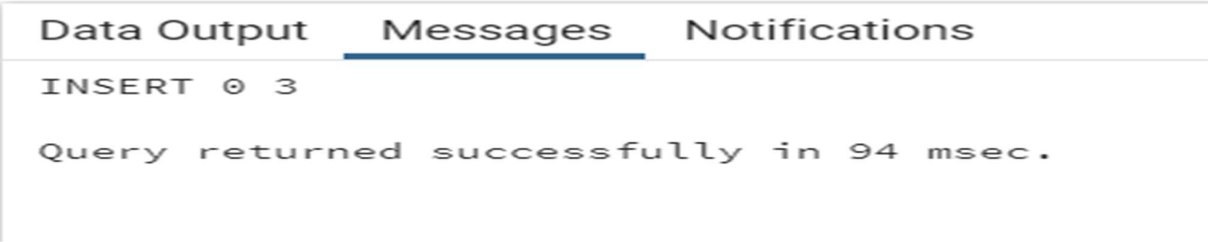
Ans. INSERT INTO Station (ID, CITY, STATE, LAT_N, LONG_W)

VALUES

(13, 'PHOENIX', 'AZ', 33, 112),

(44, 'DENVER', 'CO', 40, 105),

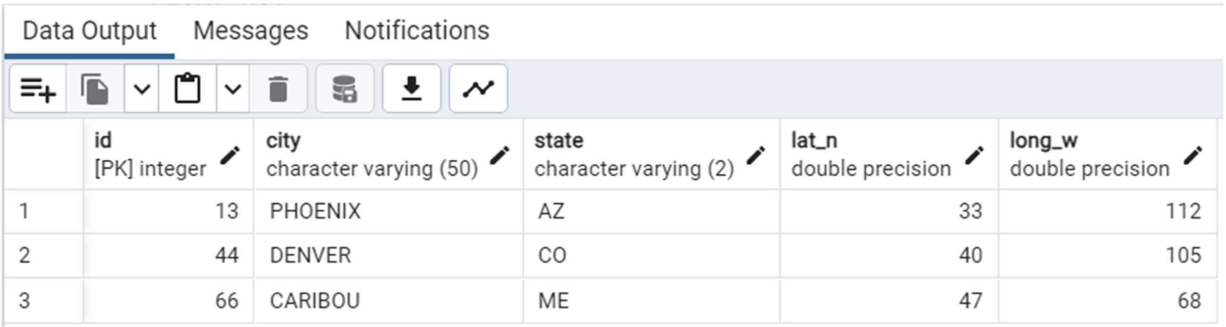
(66, 'CARIBOU', 'ME', 47, 68);



```
INSERT 0 3

Query returned successfully in 94 msec.
```

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

Ans. SELECT * FROM station;

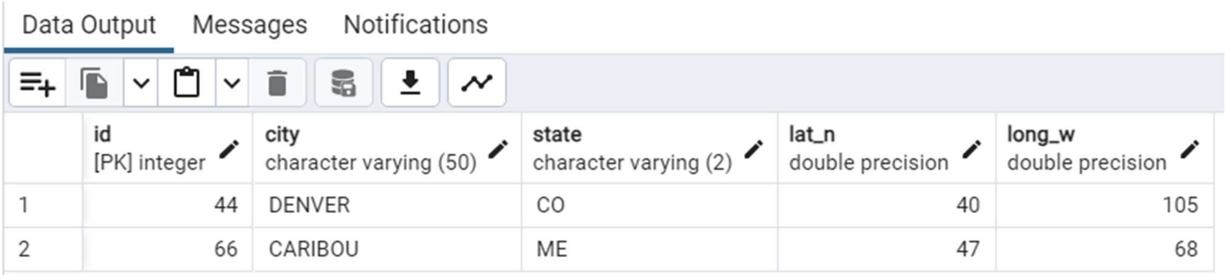


	id [PK] integer	city character varying (50)	state character varying (2)	lat_n double precision	long_w double precision
1	13	PHOENIX	AZ	33	112
2	44	DENVER	CO	40	105
3	66	CARIBOU	ME	47	68

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

Ans.

```
SELECT *
FROM station
WHERE lat_n > 39.7;
```



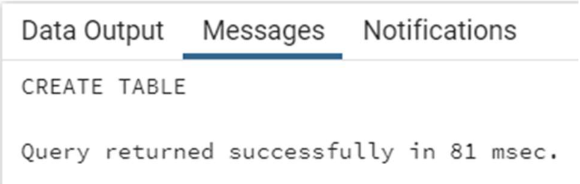
	id [PK] integer	city character varying (50)	state character varying (2)	lat_n double precision	long_w double precision
1	44	DENVER	CO	40	105
2	66	CARIBOU	ME	47	68

Q5) Create another table, **'STATS'**, to store normalized temperature and precipitation data:

Column	**Data type**	**Remark**
ID	Number	**ID** must match with some **ID** from the **STATION** table(so name & location will be known).
MONTH	Number	The range of months is between (**1 and 12**)
TEMP_F	Number	Temperature is in Fahrenheit degrees, Ranging between (**-80 and 150**)
RAIN_I	Number	Rain is in inches, Ranging between (**0 and 100**)

Ans.

```
CREATE TABLE IF NOT EXIST Stats (
 ID INT,
 MONTH INT NOT NULL,
 TEMP_F FLOAT NOT NULL,
 RAIN_I FLOAT NOT NULL,
 FOREIGN KEY (ID) REFERENCES Station(ID)
);
```



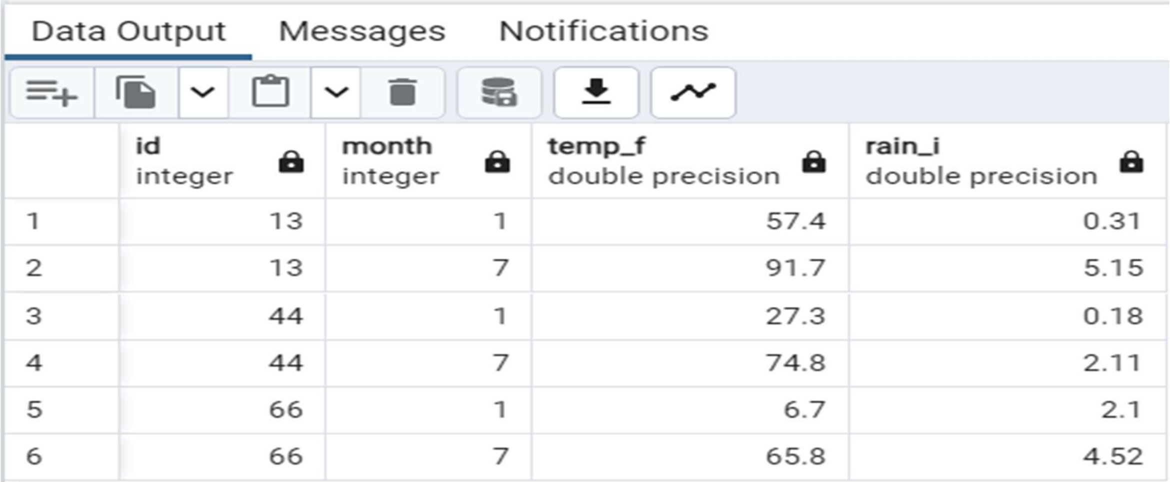
```
CREATE TABLE

Query returned successfully in 81 msec.
```

Q6) 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

Ans.

```
INSERT INTO Stats (ID, MONTH, TEMP_F, RAIN_I)
VALUES
(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);
```

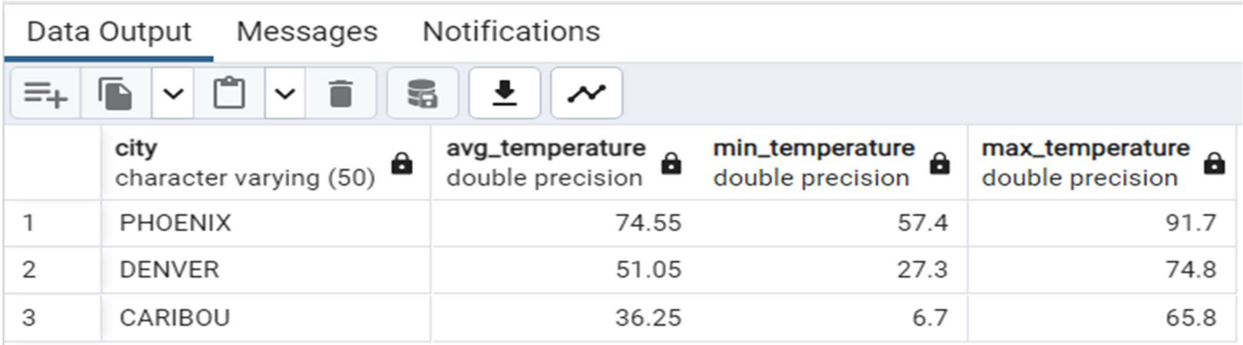


	id integer	month integer	temp_f double precision	rain_i double precision
1	13	1	57.4	0.31
2	13	7	91.7	5.15
3	44	1	27.3	0.18
4	44	7	74.8	2.11
5	66	1	6.7	2.1
6	66	7	65.8	4.52

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

Ans.

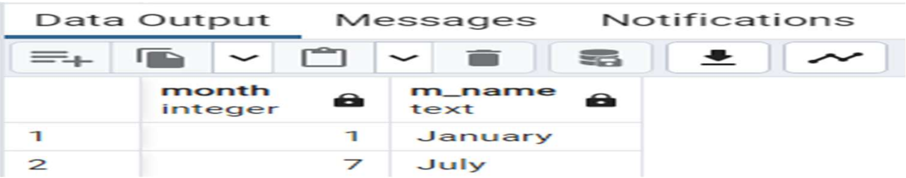
```
SELECT S.CITY, AVG(ST.TEMP_F) AS AVG_TEMPERATURE,
        MIN(ST.TEMP_F) AS MIN_TEMPERATURE,
        MAX(ST.TEMP_F) AS MAX_TEMPERATURE
FROM
Station S
JOIN
Stats ST ON S.ID = ST.ID
GROUP BY S.CITY;
```



	city character varying (50)	avg_temperature double precision	min_temperature double precision	max_temperature double precision
1	PHOENIX	74.55	57.4	91.7
2	DENVER	51.05	27.3	74.8
3	CARIBOU	36.25	6.7	65.8

- NOTE**:** I have created a temporary table Using CTE named "**Temp_month_name**" for month names to show the names correctly. Here is the code for that.

```
CREATE TEMPORARY TABLE IF NOT EXISTS temp_month_name AS
WITH month_name AS(
        SELECT DISTINCT month, TO_CHAR(TO_DATE(CONCAT('2023-',
month, '-01'), 'YYYY-MM-DD'), 'Month') AS M_name
 FROM stats
)
SELECT * FROM month_name;
SELECT * FROM temp_month_name;
```

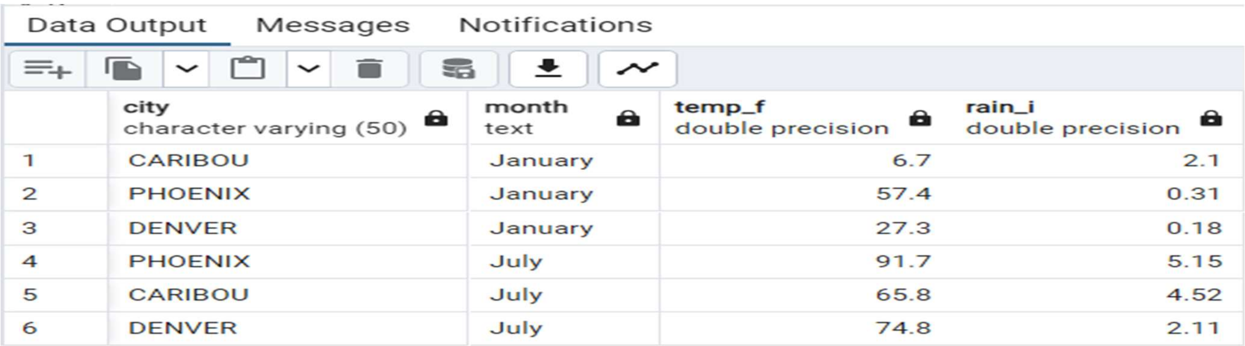


	month integer	m_name text
1	1	January
2	7	July

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.

Ans.

```
SELECT s.city, m.m_name AS month, st.temp_f, st.rain_i
FROM stats AS st
JOIN station AS s
ON st.id = s.id
JOIN temp_month_name AS m
ON st.month = m.month
ORDER BY month, rain_i DESC;
```

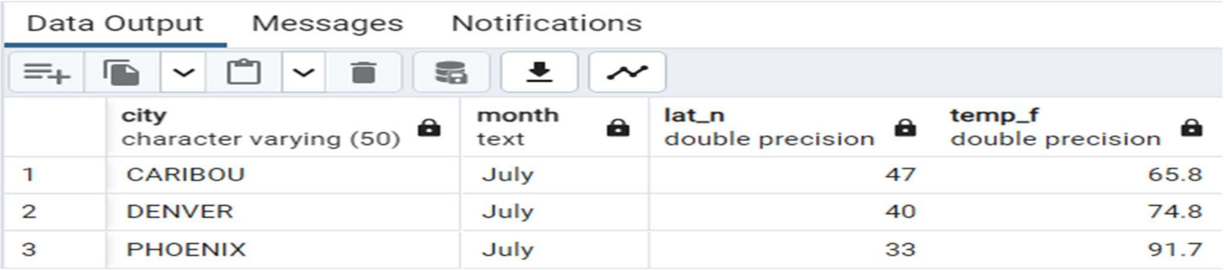


	city character varying (50)	month text	temp_f double precision	rain_i double precision
1	CARIBOU	January	6.7	2.1
2	PHOENIX	January	57.4	0.31
3	DENVER	January	27.3	0.18
4	PHOENIX	July	91.7	5.15
5	CARIBOU	July	65.8	4.52
6	DENVER	July	74.8	2.11

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

Ans.

```
SELECT S.CITY, M.m_name AS month, S.LAT_N, ST.TEMP_F
FROM Stats AS ST
JOIN Station AS S
ON ST.ID = S.ID
JOIN temp_month_name AS M
ON ST.month = M.month
WHERE ST.MONTH = 7
ORDER BY ST.TEMP_F;
```

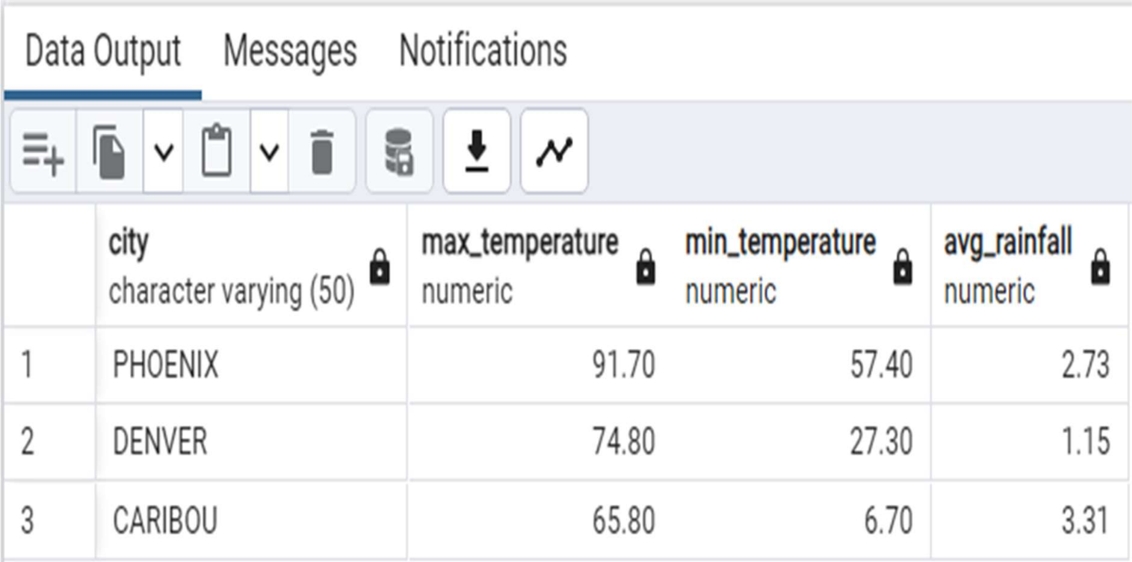


	city character varying (50)	month text	lat_n double precision	temp_f double precision
1	CARIBOU	July	47	65.8
2	DENVER	July	40	74.8
3	PHOENIX	July	33	91.7

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

Ans.

```
S.CITY,
ROUND(MAX(ST.TEMP_F)::numeric, 2) AS MAX_TEMPERATURE,
ROUND(MIN(ST.TEMP_F)::numeric, 2) AS MIN_TEMPERATURE,
ROUND(AVG(ST.RAIN_I)::numeric, 2) AS AVG_RAINFALL
FROM Station AS S
JOIN Stats AS  ST ON S.ID = ST.ID
GROUP BY S.CITY;

-- suitable for workbench
SELECT
        S.CITY,
        ROUND(MAX(ST.TEMP_F), 2) AS MAX_TEMPERATURE,
        ROUND(MIN(ST.TEMP_F), 2) AS MIN_TEMPERATURE,
        ROUND(AVG(ST.RAIN_I), 2) AS AVG_RAINFALL
FROM Station AS S
JOIN Stats ST ON S.ID = ST.ID
GROUP BY S.CITY;
```

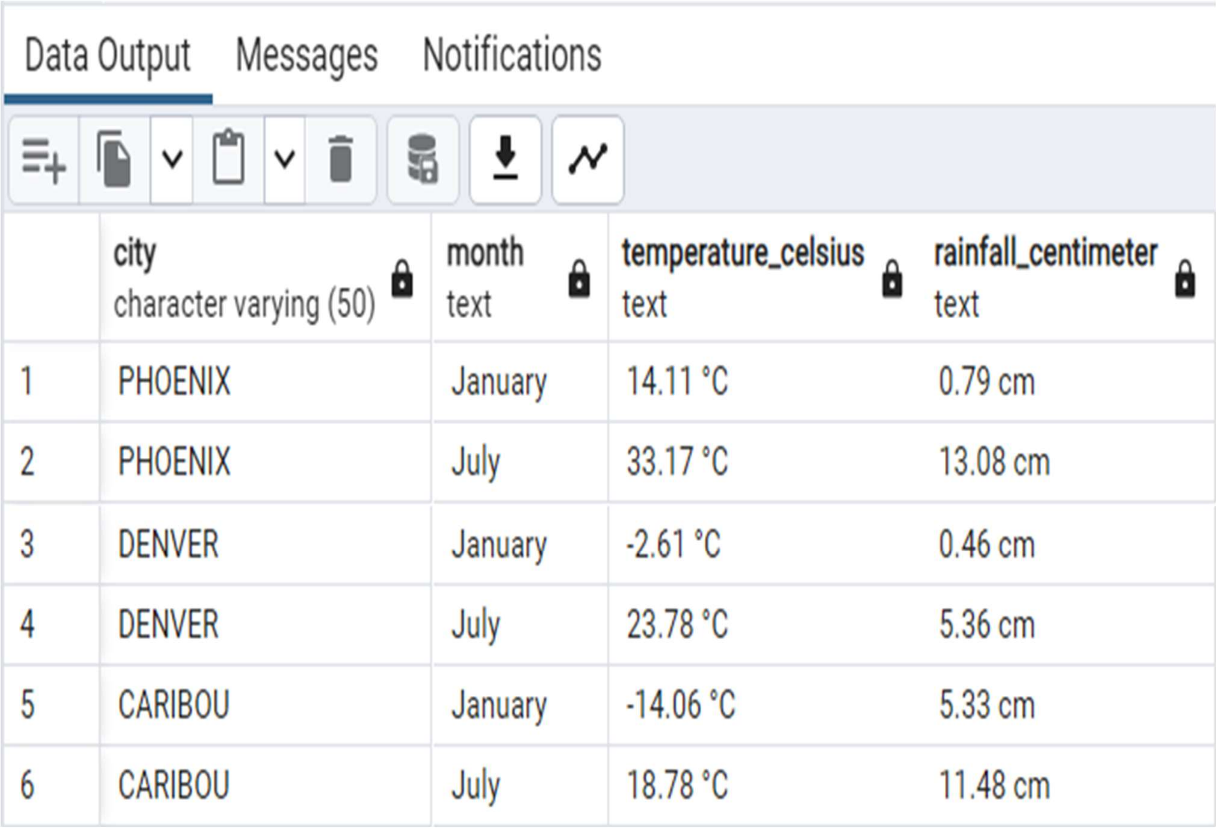


	city character varying (50)	max_temperature numeric	min_temperature numeric	avg_rainfall numeric
1	PHOENIX	91.70	57.40	2.73
2	DENVER	74.80	27.30	1.15
3	CARIBOU	65.80	6.70	3.31

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

Ans.

```
SELECT
   S.CITY,
   M.m_name AS MONTH,
   CONCAT(ROUND(((ST.TEMP_F - 32) * 5/9)::numeric, 2), ' °C') AS TEMPERATURE_CELSIUS,
   CONCAT(ROUND((ST.RAIN_I * 2.54)::numeric, 2), ' cm') AS RAINFALL_CENTIMETER
FROM Stats ST
JOIN Station S ON ST.ID = S.ID
JOIN temp_month_name M ON ST.MONTH = M.month
ORDER BY s.id;
```

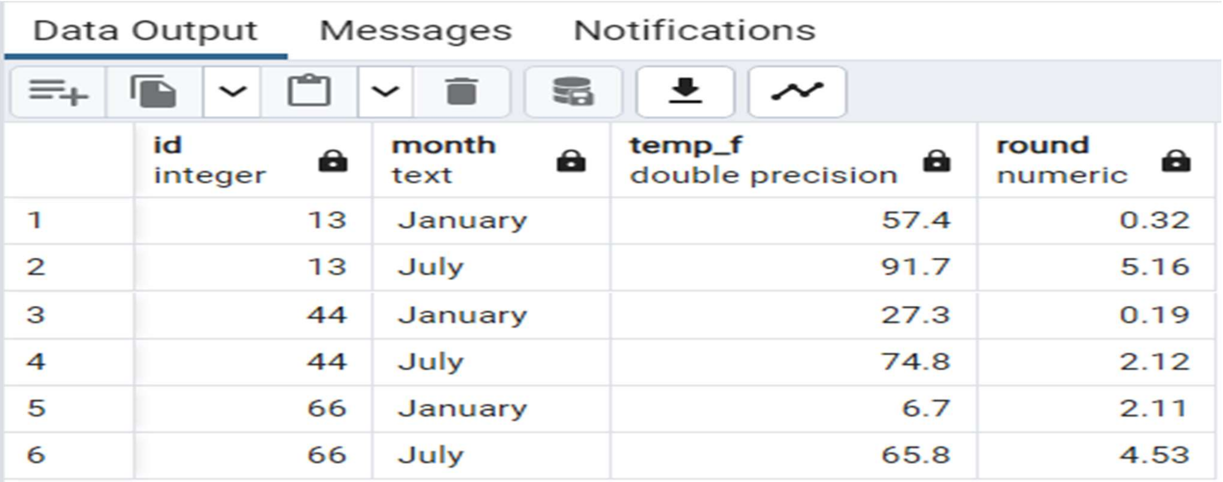


	city character varying (50)	month text	temperature_celsius text	rainfall_centimeter text
1	PHOENIX	January	14.11 °C	0.79 cm
2	PHOENIX	July	33.17 °C	13.08 cm
3	DENVER	January	-2.61 °C	0.46 cm
4	DENVER	July	23.78 °C	5.36 cm
5	CARIBOU	January	-14.06 °C	5.33 cm
6	CARIBOU	July	18.78 °C	11.48 cm

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

Ans.

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

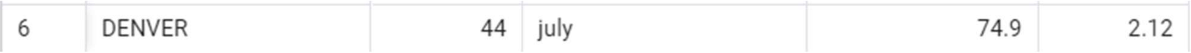


	id integer	month text	temp_f double precision	round numeric
1	13	January	57.4	0.32
2	13	July	91.7	5.16
3	44	January	27.3	0.19
4	44	July	74.8	2.12
5	66	January	6.7	2.11
6	66	July	65.8	4.53

Q13) Update **Denver's July** temperature reading as **74.9**.

Ans.

```
UPDATE Stats
SET TEMP_F = 74.9
WHERE ID = (SELECT ID FROM Station WHERE CITY = 'DENVER')
          AND MONTH = 7;
```



6	DENVER	44	july	74.9	2.12