

Assignment 3 - Gathering, Scraping, Munging and Cleaning Data

Team name : errorFree

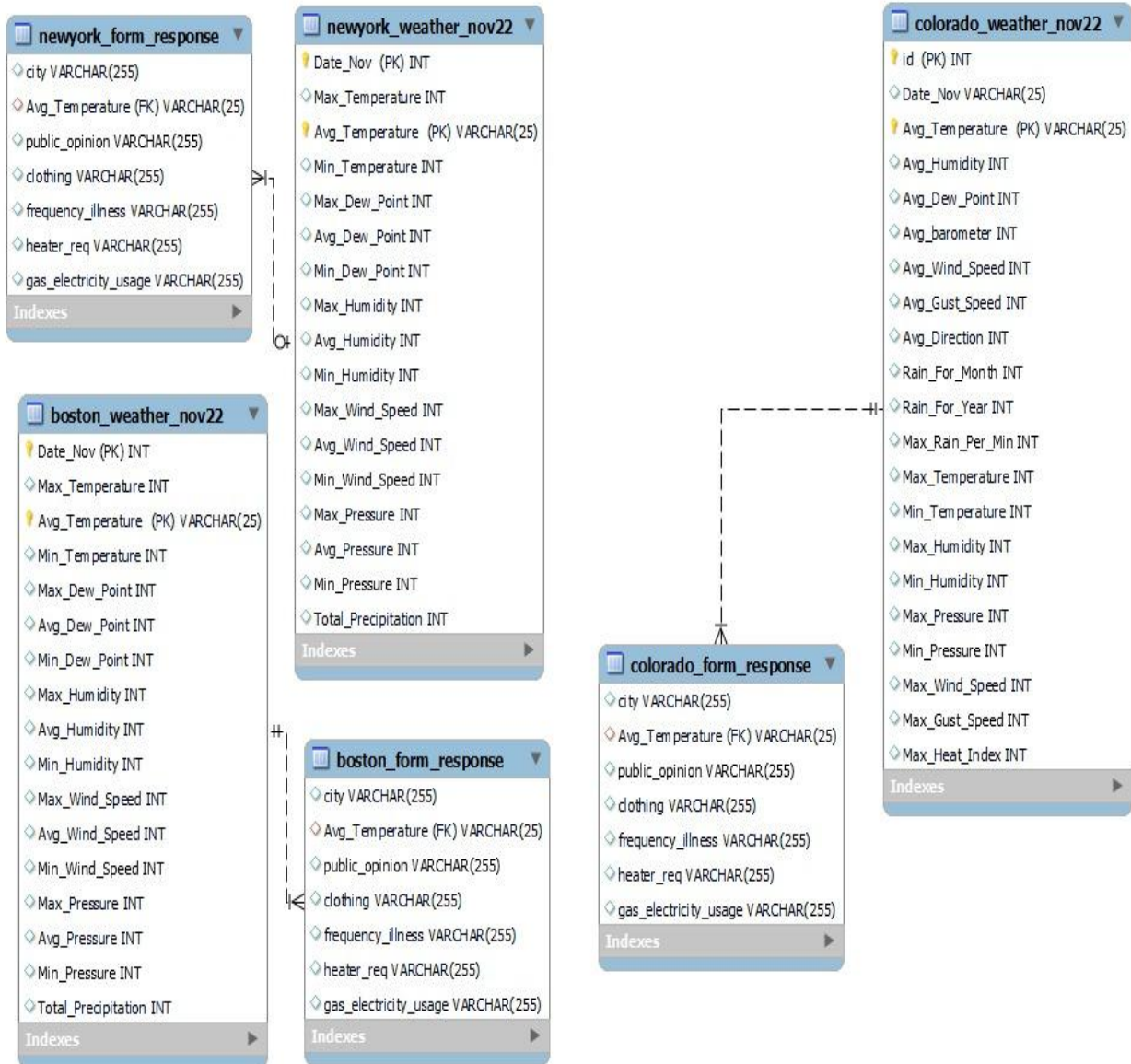
Group Members:

Ritik Bhandari(002738904)

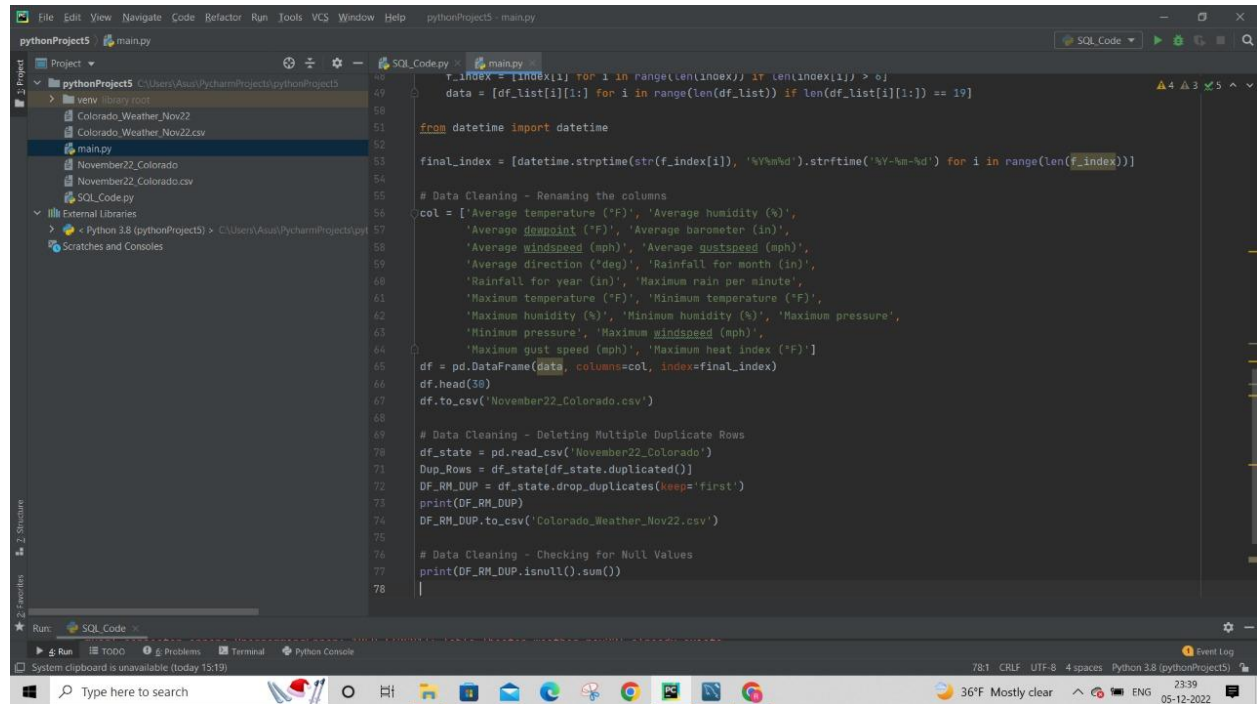
Rishi Shelly(002771020)

Dhiral Mayavanshi(002706357)

Database Schema: ER Diagram

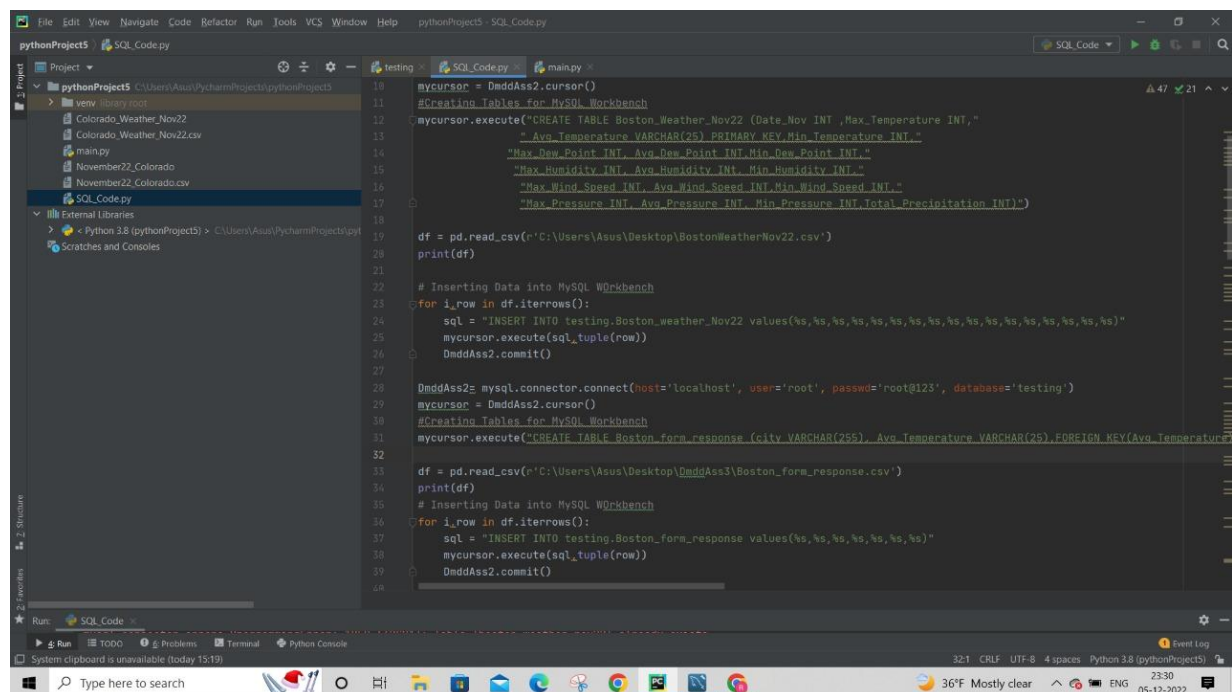


Data Cleaning using Python Code:



```
49 r_index = [index[i] for i in range(len(index)) if len(index[i]) > 0]
50 data = [df_list[i][1:] for i in range(len(df_list)) if len(df_list[i][1:]) == 19]
51
52 from datetime import datetime
53
54 final_index = [datetime.strptime(str(f_index[i]), '%Y-%m-%d').strftime('%Y-%m-%d') for i in range(len(f_index))]
55
56 # Data Cleaning - Renaming the columns
57 col = ['Average temperature (°F)', 'Average humidity (%)',
58       'Average dewpoint (°F)', 'Average barometer (in)',
59       'Average windspeed (mph)', 'Average gustspeed (mph)',
60       'Average direction (deg)', 'Rainfall for month (in)',
61       'Rainfall for year (in)', 'Maximum rain per minute',
62       'Maximum temperature (°F)', 'Minimum temperature (°F)',
63       'Maximum humidity (%)', 'Minimum humidity (%)', 'Maximum pressure',
64       'Minimum pressure', 'Maximum windspeed (mph)',
65       'Maximum gust speed (mph)', 'Maximum heat index (°F)']
66
67 df = pd.DataFrame(data, columns=col, index=final_index)
68 df.head(30)
69 df.to_csv('November22_Colorado.csv')
70
71 # Data Cleaning - Deleting Multiple Duplicate Rows
72 df_state = pd.read_csv('November22_Colorado')
73 Dup_Rows = df_state[df_state.duplicated()]
74 DF_RM_DUP = df_state.drop_duplicates(keep='first')
75 print(DF_RM_DUP)
76 DF_RM_DUP.to_csv('Colorado_Weather_Nov22.csv')
77
78 # Data Cleaning - Checking for Null Values
79 print(DF_RM_DUP.isnull().sum())
80
```

Code For Creating tables and inserting the data into MySQL Workbench:



```
10 mycursor = DmddAss2.cursor()
11 #Creating Tables for MySQL Workbench
12 mycursor.execute("CREATE TABLE Boston_Weather_Nov22 (Date_Nov INT, Max_Temperature INT,
13               "Avg_Temperature VARCHAR(255) PRIMARY KEY, Min_Temperature INT,
14               "Max_Dew_Point INT, Avg_Dew_Point INT, Min_Dew_Point INT,
15               "Max_Humidity INT, Avg_Humidity INT, Min_Humidity INT,
16               "Max_Wind_Speed INT, Avg_Wind_Speed INT, Min_Wind_Speed INT,
17               "Max_Pressure INT, Avg_Pressure INT, Min_Pressure INT, Total_Precipitation INT)")
18
19 df = pd.read_csv(r"C:\Users\Asus\Desktop\BostonWeatherNov22.csv")
20 print(df)
21
22 # Inserting Data into MySQL Workbench
23 for i, row in df.iterrows():
24     sql = "INSERT INTO testing.Boston_weather_Nov22 values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
25     mycursor.execute(sql_tuple(row))
26     DmddAss2.commit()
27
28 DmddAss2 = pymysql.connect(host='localhost', user='root', passwd='root@123', database='testing')
29 mycursor = DmddAss2.cursor()
30 #Creating Tables for MySQL Workbench
31 mycursor.execute("CREATE TABLE Boston_form_response (city VARCHAR(255), Avg_Temperature VARCHAR(255), FOREIGN KEY(Avg_Temperature)
32               REFERENCES testing.Boston_weather_Nov22 (Avg_Temperature))")
33
34 df = pd.read_csv(r"C:\Users\Asus\Desktop\DmddAss3\Boston_form_response.csv")
35 print(df)
36 # Inserting Data into MySQL Workbench
37 for i, row in df.iterrows():
38     sql = "INSERT INTO testing.Boston_form_response values(%s,%s,%s,%s,%s,%s)"
39     mycursor.execute(sql_tuple(row))
40     DmddAss2.commit()
41
```

Description :

Weather Analysis for 3 cities ,NewYork , Boston and Colorado was done. The data for Colorado was Scrapped from the website - [Estes Park Weather - Home/Forecasts](#) and data for Boston and New York was collected from - [East Boston, MA Weather History | Weather Underground \(wunderground.com\)](#). Public Opinions and responses was collected from the people of these cities and then Analysis was performed using various MySQL Workbench. Data Visulaization of the collected data was also done to present the analyzed data in an effective manner.

Use Cases:

Use Case 1: Clothing preference of people in Boston according to it's weather conditions

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view containing 'testing' and 'weather' databases. The 'testing' database is selected, showing tables like 'boston_form_response', 'boston_weather_nov22', 'colorado_form_response', 'newyork_form_response', and 'newyork_weather_nov22'. The 'weather' database is also visible. The main editor window shows a SQL query:

```
1 use testing;
2 select bw.avg_temperature, bw.avg_humidity, bw.Avg_Wind_Speed, bf.clothing
3 from boston_weather_nov22 bw
4 right join boston_form_response bf
5 on bw.avg_temperature = bf.avg_temperature;
6
```

The 'Result Grid' shows 13 rows of data with columns: avg_temperature, avg_humidity, Avg_Wind_Speed, and clothing. The data is as follows:

avg_temperature	avg_humidity	Avg_Wind_Speed	clothing
61.0	88	8	t-shirt
61.0	88	8	full-sleeves
54.1	67	7	full-sleeves
33.7	39	17	jackets
66.2	82	12	t-shirt
69.9	79	16	t-shirt
67.4	54	15	t-shirt
48.8	36	14	full-sleeves
43.0	60	8	jackets
56.4	66	15	full-sleeves
66.2	82	12	t-shirt
41.0	59	15	jackets
50.3	77	12	t-shirt
39.5	46	11	jackets
38.2	53	6	jackets
46.9	85	16	full-sleeves
41.0	59	15	jackets
38.5	49	12	jackets

The bottom panel shows the 'Action Output' with three rows of execution logs, each indicating a successful query execution with 39 rows returned.

Use Case 2: Which is the best city for generating win energy?

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

sys

testing

boston_form_response

Columns

Indexes

Foreign Keys

Triggers

boston_weather_nov22

colorado_form_response

colorado_weather_nov22

Columns

Indexes

Foreign Keys

Triggers

Administration Schemas

Information

Table: boston_form_response

Columns:

city varchar(255)

Avg_Temperature varchar(255)

public_opinion varchar(255)

clothing varchar(255)

frequency_illness varchar(255)

heater_req varchar(255)

gas_electricity_usage varchar(255)

1 use testing;

2 select bw.Max_Wind_Speed, count(bw.Max_Wind_Speed),bf.city

3 from boston_weather_nov22 bw

4 join boston_form_response bf

5 on bw.avg_temperature = bf.avg_temperature

6 where Max_Wind_Speed > '28' group by Max_Wind_Speed

7 UNION

8 select nw.Max_Wind_Speed, count(nw.Max_Wind_Speed),nf.city

9 from newyork_weather_nov22 nw

10 join newyork_form_response nf

11 on nw.avg_temperature = nf.avg_temperature

12 where Max_Wind_Speed > '28' group by Max_Wind_Speed

13 UNION

14 select cw.Max_Wind_Speed, count(cw.Max_Wind_Speed),cf.city

15 from colorado_weather_nov22 cw

16 join colorado_form_response cf

17 on cw.avg_temperature = cf.avg_temperature

18 where Max_Wind_Speed > '28' group by Max_Wind_Speed;

19

Result Grid

Max_Wind_Speed	count(bw.Max_Wind_Speed)	city
23	4	Boston
26	4	Boston
25	5	Boston
21	2	Boston
29	1	Boston

Result 48 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
75	22:55:31	select nw.Max_Wind_Speed, count(nw.Max_Wind_Speed),nf.city from Newyork_weather_nov22 nw join ...	7 row(s) returned	0.000 sec / 0.000 sec
76	22:56:31	select bw.Max_Wind_Speed, count(bw.Max_Wind_Speed),bf.city from boston_weather_nov22 bw join bo...	21 row(s) returned	0.219 sec / 0.000 sec

Object Info Session

Type here to search

37°F Partly cloudy

22:57 05-12-2022

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

sys

testing

boston_form_response

Columns

Indexes

Foreign Keys

Triggers

boston_weather_nov22

colorado_form_response

colorado_weather_nov22

Columns

Indexes

Foreign Keys

Triggers

Administration Schemas

Information

Table: boston_form_response

Columns:

city varchar(255)

Avg_Temperature varchar(255)

public_opinion varchar(255)

clothing varchar(255)

frequency_illness varchar(255)

heater_req varchar(255)

gas_electricity_usage varchar(255)

1 use testing;

2 select bw.Max_Wind_Speed, count(bw.Max_Wind_Speed),bf.city

3 from boston_weather_nov22 bw

4 join boston_form_response bf

5 on bw.avg_temperature = bf.avg_temperature

6 where Max_Wind_Speed > '28' group by Max_Wind_Speed

7 UNION

8 select nw.Max_Wind_Speed, count(nw.Max_Wind_Speed),nf.city

9 from newyork_weather_nov22 nw

10 join newyork_form_response nf

11 on nw.avg_temperature = nf.avg_temperature

12 where Max_Wind_Speed > '28' group by Max_Wind_Speed

13 UNION

14 select cw.Max_Wind_Speed, count(cw.Max_Wind_Speed),cf.city

15 from colorado_weather_nov22 cw

16 join colorado_form_response cf

17 on cw.avg_temperature = cf.avg_temperature

18 where Max_Wind_Speed > '28' group by Max_Wind_Speed;

19

Result Grid

Max_Wind_Speed	count(bw.Max_Wind_Speed)	city
23	4	Boston
26	4	Boston
25	5	Boston
21	2	Boston
29	1	Boston
22	1	Boston
24	2	Boston
26	1	NewYork
23	7	NewYork
22	1	NewYork
28	3	NewYork
24	3	NewYork
29	3	NewYork
21	3	NewYork
21	2	Colorado
29	1	Colorado
32	2	Colorado
30	1	Colorado
25	2	Colorado
36	1	Colorado

Result 48 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
75	22:55:31	select nw.Max_Wind_Speed, count(nw.Max_Wind_Speed),nf.city from Newyork_weather_nov22 nw join ...	7 row(s) returned	0.000 sec / 0.000 sec
76	22:56:31	select bw.Max_Wind_Speed, count(bw.Max_Wind_Speed),bf.city from boston_weather_nov22 bw join bo...	21 row(s) returned	0.219 sec / 0.000 sec

Object Info Session

Type here to search

37°F Partly cloudy

22:57 05-12-2022

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Use Case 3: Rate of illness in NewYork due to low temperature and Dew Point

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

testing

boston_form_response

boston_weather_nov22

colorado_form_response

colorado_weather_nov22

newyork_form_response

newyork_weather_nov22

Views

Stored Procedures

Functions

tp

weather

Administration Schemas

Information

Table: newyork_form_response

Columns:

city varchar(255)

Avg_Temperature varchar(255)

public_opinion varchar(255)

clothing varchar(255)

frequency_illness varchar(255)

heater_req varchar(255)

gas_electricity_usage varchar(255)

abc boston_weather_nov22

Limit to 1000 rows

1 use testing;

2 select nw.Min_Temperature, nw.Avg_Dew_Point,nf.frequency_illness

3 from newyork_weather_nov22 nw

4 right join newyork_form_response nf

5 on nw.avg_temperature = nf.avg_temperature;

6

7

Result Grid

Filter Rows:

Export: Wrap Cell Contents

Export recordset to an external file

Min_Temperature Avg_Dew_Point frequency_illness

39 24 normal

41 36 normal

33 15 extreme cough ...

40 21 feverish

47 38 feverish

33 15 extreme cough ...

29 12 extreme cough ...

44 26 normal

40 42 normal

39 36 normal

63 61 normal

Result 15 x

Output

Action Output

Time Action Message Duration / Fetch

29 20:47:08 select bw.avg_temperature, bw.Total_Precipitation,bw.Avg_Wind_Speed,bf.heater_req from boston_w... 39 row(s) returned 0.000 sec / 0.000 sec

31 20:51:48 select nw.Min_Temperature, nw.Avg_Dew_Point,nf.frequency_illness from newyork_weather_nov22 nw e... 44 row(s) returned 1.828 sec / 0.000 sec

Object Info Session

Type here to search

40°F Mostly cloudy

20:52 05-12-2022

Use Case 4 : Heater Requirement by the people of Boston according to it's weather conditions

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

sys

testing

boston_form_response

boston_weather_nov22

colorado_form_response

colorado_weather_nov22

newyork_form_response

newyork_weather_nov22

Views

Stored Procedures

Functions

tp

weather

Administration Schemas

Information

Table: boston_form_response

Columns:

city varchar(255)

Avg_Temperature varchar(255)

public_opinion varchar(255)

clothing varchar(255)

frequency_illness varchar(255)

heater_req varchar(255)

gas_electricity_usage varchar(255)

abc boston_weather_nov22

Limit to 1000 rows

1 use testing;

2 select bw.avg_temperature, bw.Total_Precipitation,bw.Avg_Wind_Speed,bf.heater_req

3 from boston_weather_nov22 bw

4 right join boston_form_response bf

5 on bw.avg_temperature = bf.avg_temperature;

6

Result Grid

Filter Rows:

Export: Wrap Cell Contents

avg_temperature Total_Precipitation Avg_Wind_Speed heater_req

44.1 0 10 no

39.5 0 11 yes

47.2 0 12 yes

46.1 0 13 no

50.0 0 10 yes

49.0 1 16 no

38.6 0 7 yes

50.4 0 15 no

44.1 0 10 yes

33.7 0 17 yes

46.9 0 16 no

38.5 0 12 yes

33.7 0 17 yes

41.0 0 15 no

50.3 0 12 no

67.4 0 15 yes

69.9 0 16 no

Result 14 x

Output

Action Output

Time Action Message Duration / Fetch

29 20:42:10 select bw.avg_temperature, bw.avg_humidity,bw.Avg_Wind_Speed,bf.clothing from boston_weather_nov... 39 row(s) returned 0.000 sec / 0.000 sec

30 20:47:08 select bw.avg_temperature, bw.Total_Precipitation,bw.Avg_Wind_Speed,bf.heater_req from boston_w... 39 row(s) returned 0.000 sec / 0.000 sec

Object Info Session

Type here to search

40°F Mostly cloudy

20:48 05-12-2022

Use Case 5: Public Opinion about the weather conditions in NewYork

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'testing' database selected. The 'Tables' list includes 'boston_form_response', 'boston_weather_nov22', 'colorado_form_response', 'colorado_weather_nov22', 'newyork_form_response', and 'newyork_weather_nov22'. The 'Table: newyork_form_response' is expanded, showing columns: city, Avg_Temperature, public_opinion, frequency_illness, heater_req, and gas_electricity_usage. The main editor window contains the following SQL query:

```
1 use testing;
2 select nw.Max_Temperature, nw.Max_Humidity, nw.Max_Wind_Speed, nf.public_opinion
3 from newyork_weather_nov22 nw
4 join newyork_form_response nf
5 on nw.avg_temperature = nf.avg_temperature;
```

The 'Result Grid' shows 16 rows of data with columns: Max_Temperature, Max_Humidity, Max_Wind_Speed, and public_opinion. The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
31	20:51:48	select nw.Min_Temperature, nw.Avg_Dew_Point, nf.frequency_illness from newyork_weather_nov22 nw...	44 row(s) returned	1.828 sec / 0.000 sec
32	20:56:14	select nw.Max_Temperature, nw.Max_Humidity, nw.Max_Wind_Speed, nf.public_opinion from newyork_weather_nov22 nw...	44 row(s) returned	0.703 sec / 0.000 sec

Use Case 6: Heater, gas and electricity usage/requirement by the people of Colorado

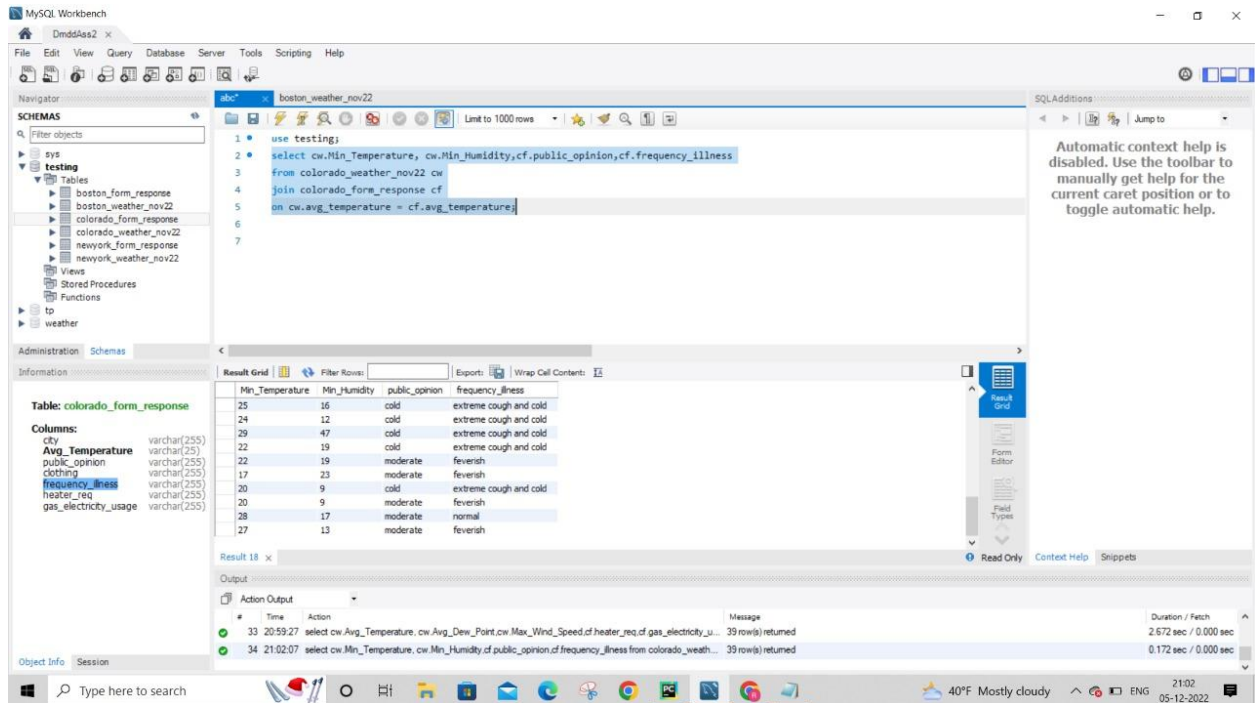
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'testing' database selected. The 'Tables' list includes 'boston_form_response', 'boston_weather_nov22', 'colorado_form_response', 'colorado_weather_nov22', 'newyork_form_response', and 'newyork_weather_nov22'. The 'Table: colorado_form_response' is expanded, showing columns: city, Avg_Temperature, public_opinion, frequency_illness, heater_req, and gas_electricity_usage. The main editor window contains the following SQL query:

```
1 use testing;
2 select cw.Avg_Temperature, cw.Avg_Dew_Point, cw.Max_Wind_Speed, cf.heater_req, cf.gas_electricity_usage
3 from colorado_weather_nov22 cw
4 join colorado_form_response cf
5 on cw.avg_temperature = cf.avg_temperature;
```

The 'Result Grid' shows 17 rows of data with columns: Avg_Temperature, Avg_Dew_Point, Max_Wind_Speed, heater_req, and gas_electricity_usage. The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
32	20:56:14	select nw.Max_Temperature, nw.Max_Humidity, nw.Max_Wind_Speed, nf.public_opinion from newyork_weather_nov22 nw...	44 row(s) returned	0.703 sec / 0.000 sec
33	20:59:27	select cw.Avg_Temperature, cw.Avg_Dew_Point, cw.Max_Wind_Speed, cf.heater_req, cf.gas_electricity_usage from colorado_weather_nov22 cw...	39 row(s) returned	2.672 sec / 0.000 sec

Use Case 7: Colorado's public opinion and rate of illness during different weather conditions



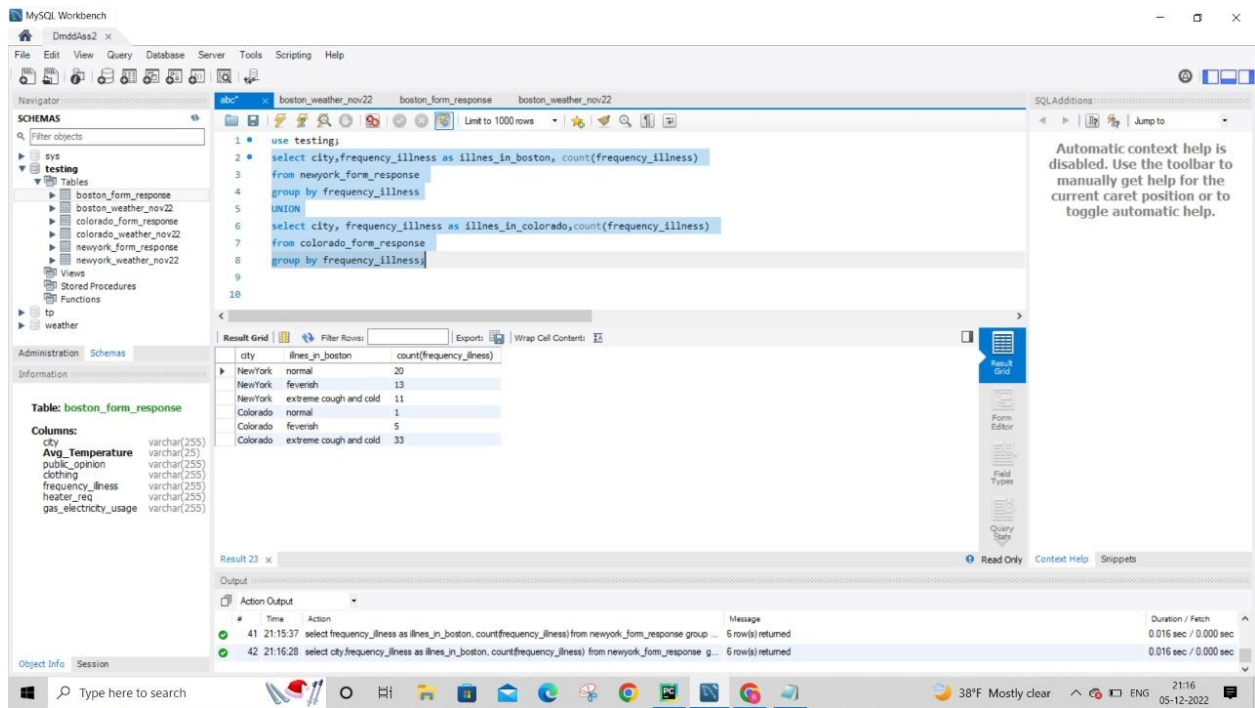
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of databases including 'testing'. The 'testing' database is selected, showing tables like 'boston_form_response', 'boston_weather_nov22', 'colorado_form_response', 'colorado_weather_nov22', 'newyork_form_response', and 'newyork_weather_nov22'. The main editor window contains a SQL query:

```
1 use testing;
2 select cw.Min_Temperature, cw.Min_Humidity, cf.public_opinion, cf.frequency_illness
3 from colorado_weather_nov22 cw
4 join colorado_form_response cf
5 on cw.avg_temperature = cf.avg_temperature;
```

The 'Result Grid' shows the output of the query, displaying columns: Min_Temperature, Min_Humidity, public_opinion, and frequency_illness. The data includes rows for various weather conditions and public opinions.

The bottom status bar shows the system clock as 21:02 on 05-12-2022, and the weather as 40°F Mostly cloudy.

Use Case 8: Comparison of Frequency of illness of NewYork and Colorado



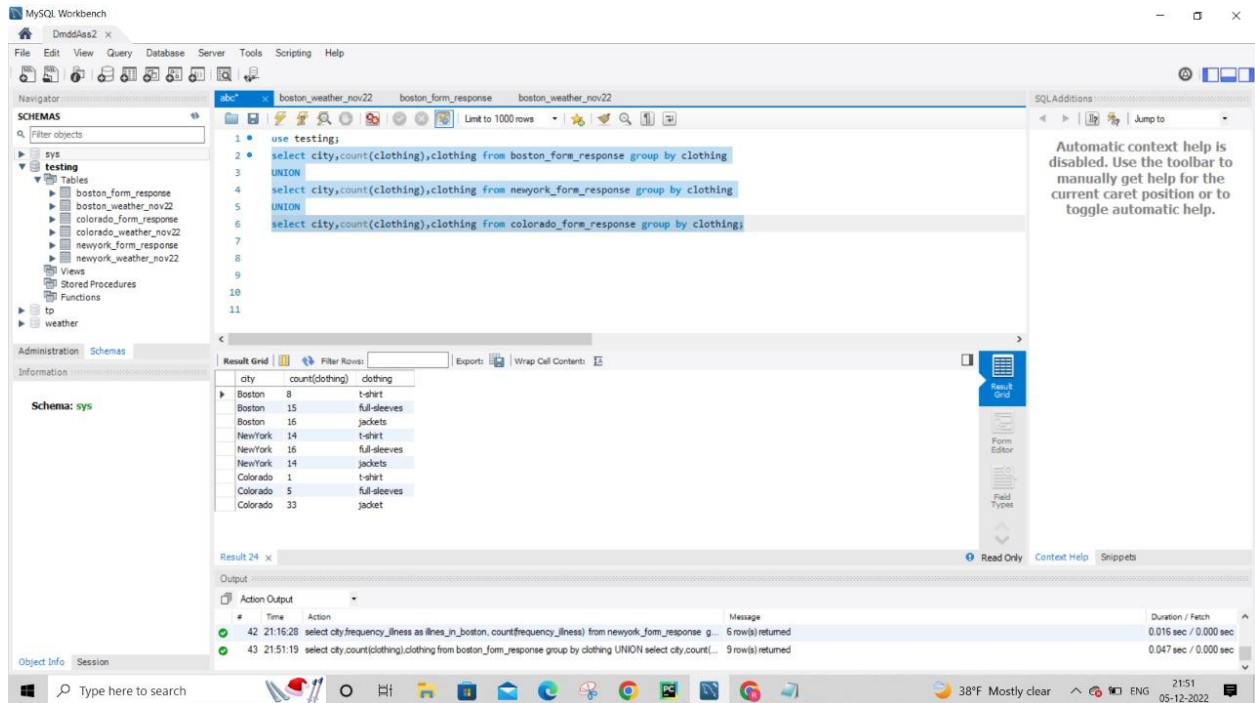
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of databases including 'testing'. The 'testing' database is selected, showing tables like 'boston_form_response', 'boston_weather_nov22', 'colorado_form_response', 'colorado_weather_nov22', 'newyork_form_response', and 'newyork_weather_nov22'. The main editor window contains a SQL query:

```
1 use testing;
2 select city, frequency_illness as illness_in_boston, count(frequency_illness)
3 from newyork_form_response
4 group by frequency_illness
5 UNION
6 select city, frequency_illness as illness_in_colorado, count(frequency_illness)
7 from colorado_form_response
8 group by frequency_illness;
```

The 'Result Grid' shows the output of the query, displaying columns: city, illness_in_boston, and count(frequency_illness). The data includes rows for NewYork and Colorado, comparing the frequency of illness across different weather conditions.

The bottom status bar shows the system clock as 21:16 on 05-12-2022, and the weather as 38°F Mostly clear.

Use Case 9: Comparison between the preferred clothing in NewYork, Boston and Colorado



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that uses a UNION to compare clothing preferences across three datasets: boston_form_response, newyork_form_response, and colorado_form_response. The query is as follows:

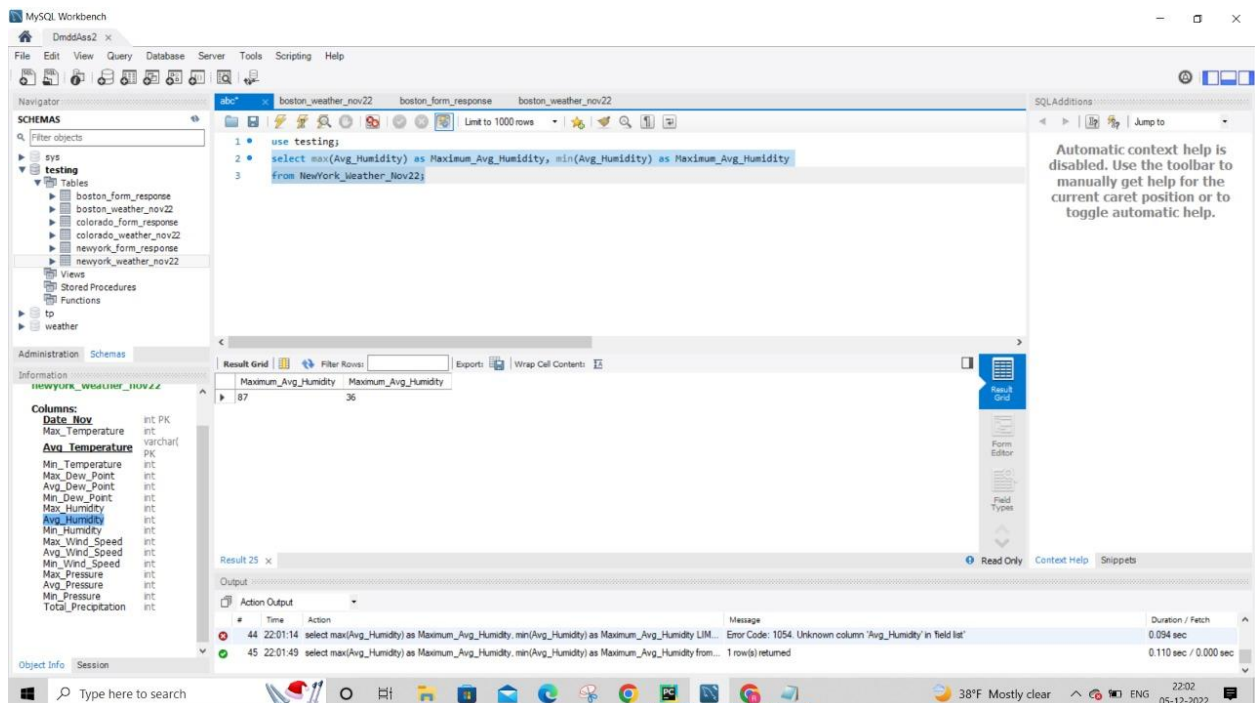
```
1 use testing;
2 select city,count(clothing),clothing from boston_form_response group by clothing
3 UNION
4 select city,count(clothing),clothing from newyork_form_response group by clothing
5 UNION
6 select city,count(clothing),clothing from colorado_form_response group by clothing;
```

The Results window displays the output of the query, showing the count of each clothing item for each city. The data is as follows:

city	count(clothing)	clothing
Boston	8	t-shirt
Boston	15	full-sleeves
Boston	16	jackets
NewYork	14	t-shirt
NewYork	16	full-sleeves
NewYork	14	jackets
Colorado	1	t-shirt
Colorado	5	full-sleeves
Colorado	33	jacket

The SQL Additions window on the right displays a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Use Case 10: What is the Maximum and Minimum Humidity in NewYork during the month of Nov 22



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that uses the MAX and MIN functions to find the maximum and minimum humidity in New York during the month of November 2022. The query is as follows:

```
1 use testing;
2 select max(Avg_Humidity) as Maximum_Avg_Humidity, min(Avg_Humidity) as Minimum_Avg_Humidity
3 from NewYork_Weather_Nov22;
```

The Results window displays the output of the query, showing the maximum and minimum average humidity. The data is as follows:

Maximum_Avg_Humidity	Minimum_Avg_Humidity
87	36

The SQL Additions window on the right displays a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The Object Info window on the left shows the schema for the NewYork_Weather_Nov22 table, including columns such as Date_Nov, Max_Temperature, Avg_Temperature, Mn_Temperature, Max_Dew_Point, Avg_Dew_Point, Mn_Dew_Point, Avg_Humidity, Mn_Humidity, Max_Wind_Speed, Avg_Wind_Speed, Mn_Wind_Speed, Max_Pressure, Avg_Pressure, Mn_Pressure, and Total_Precipitation.

Use Case 11: How many times Maximum Temperature was greater than 70F in Boston

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 use testing;
2 select count(Max_Temperature) from Boston_Weather_Nov22
3 where Max_Temperature > 70;
4
```

The Results tab shows a single row with the value 5.

Columns: Date_Nov int PK, Max_Temperature int, Avg_Temperature varchar PK, Mn_Temperature int, Max_Dew_Point int, Avg_Dew_Point int, Mn_Dew_Point int, Max_Humidity int, Avg_Humidity int, Mn_Humidity int, Max_Wind_Speed int, Avg_Wind_Speed int, Mn_Wind_Speed int, Max_Pressure int, Avg_Pressure int, Mn_Pressure int, Total_Precipitation int.

Output:

#	Time	Action	Message	Duration / Fetch
48	22:10:20	select Max_Temperature.count(Max_Temperature) from Boston_Weather_Nov22 where Max_Temperature > 70 LIMIT 0, 10...	1 row(s) returned	0.000 sec / 0.000 sec
49	22:10:39	select count(Max_Temperature) from Boston_Weather_Nov22 where Max_Temperature > 70 LIMIT 0, 10...	1 row(s) returned	0.000 sec / 0.000 sec

Use Case 12: Which is the best city to visit in the month of November

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 use testing;
2 select city, count(public_opinion), public_opinion, frequency_illness
3 from newyork_form_response
4 group by public_opinion
5 union
6 select city, count(public_opinion), public_opinion, frequency_illness
7 from colorado_form_response
8 group by public_opinion;
9
```

The Results tab shows the following data:

city	count(public_opinion)	public_opinion	frequency_illness
NewYork	16	warm	normal
NewYork	15	moderate	normal
NewYork	13	cold	normal
Colorado	6	moderate	normal
Colorado	33	cold	extreme cough and cold

Columns: city varchar(255), Avg_Temperature varchar(255), public_opinion varchar(255), clothing varchar(255), frequency_illness varchar(255), heater_req varchar(255), gas_electricity_usage varchar(255).

Output:

#	Time	Action	Message	Duration / Fetch
52	22:19:00	SELECT * FROM testing.boston_form_response LIMIT 0, 1000	39 row(s) returned	0.000 sec / 0.000 sec
53	22:19:55	Select city, count(public_opinion), public_opinion, frequency_illness from newyork_form_response group by public_opinion	5 row(s) returned	0.078 sec / 0.000 sec

Use Case 13: How does the Average Temperature of Boston in month of Nov affect the public

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables like `boston_form_response`, `boston_weather_nov22`, `colorado_form_response`, `colorado_weather_nov22`, `newyork_form_response`, and `newyork_weather_nov22`. The main editor contains a SQL query:

```
1 use testing;
2 select bw.Date_Nov,bw.avg_temperature,bf.city,bf.public_opinion,bf.clothing,bf.frequency_illness,bf.heater_req,bf.gas_electricity_usage
3 from boston_weather_nov22 bw
4 left join boston_form_response bf
5 on bw.avg_temperature = bf.avg_temperature
6 order by Date_Nov;
```

The 'Result Grid' shows 35 rows of data. The columns are: `Date_Nov`, `avg_temperature`, `city`, `public_opinion`, `clothing`, `frequency_illness`, `heater_req`, and `gas_electricity_usage`. The data shows various weather conditions and public responses for different dates in November.

On the right, a sidebar contains a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Use Case 14: On which date did the city experience this particular weather conditions

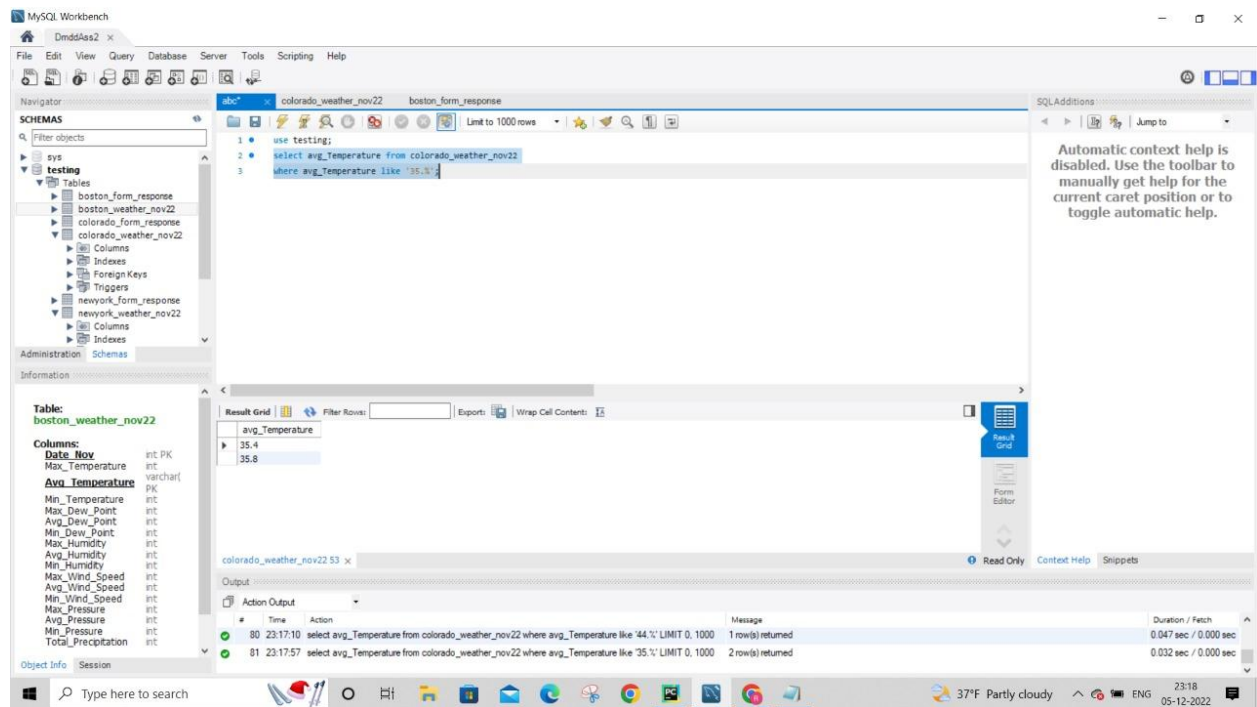
The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables like `boston_form_response`, `boston_weather_nov22`, `colorado_form_response`, `colorado_weather_nov22`, `newyork_form_response`, and `newyork_weather_nov22`. The main editor contains a SQL query:

```
1 use testing;
2 select * from newyork_weather_nov22
3 where max_Temperature = '63' and max_dew_point = '65' and max_humidity = '93' and max_wind_speed = '31';
```

The 'Result Grid' shows 1 row of data. The columns are: `Date_Nov`, `Max_Temperature`, `Avg_Temperature`, `Min_Temperature`, `Max_Dew_Point`, `Avg_Dew_Point`, `Min_Dew_Point`, `Max_Humidity`, `Avg_Humidity`, `Min_Humidity`, `Max_Wind_Speed`, `Avg_Wind_Speed`, `Min_Wind_Speed`, `Max_Pressure`, `Avg_Pressure`, `Min_Pressure`, and `Total_Precipitation`.

On the right, a sidebar contains a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Use Case 15: how many times the Average Temperature of Colorado was around 33?



The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'testing' selected, containing tables 'boston_form_response', 'boston_weather_nov22', 'colorado_form_response', and 'colorado_weather_nov22'. The main editor shows a SQL query:

```
1 use testing;
2 select avg_Temperature from colorado_weather_nov22
3 where avg_Temperature like '35.3';
```

The 'Result Grid' shows the following data:

avg_Temperature
35.4
35.8

The bottom panel shows the 'Action Output' and 'Message' log:

#	Time	Action	Message	Duration / Fetch
80	23:17:10	select avg_Temperature from colorado_weather_nov22 where avg_Temperature like '44.1%' LIMIT 0, 1000	1 row(s) returned	0.047 sec / 0.000 sec
81	23:17:57	select avg_Temperature from colorado_weather_nov22 where avg_Temperature like '35.1%' LIMIT 0, 1000	2 row(s) returned	0.032 sec / 0.000 sec

Data Visualization and Representation:

Colorado URL: [Assignment3_datavisualizationColorado.ipynb - Colaboratory \(google.com\)](#)

New York URL: [Assignment3_datavisualizationNewYork.ipynb - Colaboratory \(google.com\)](#)

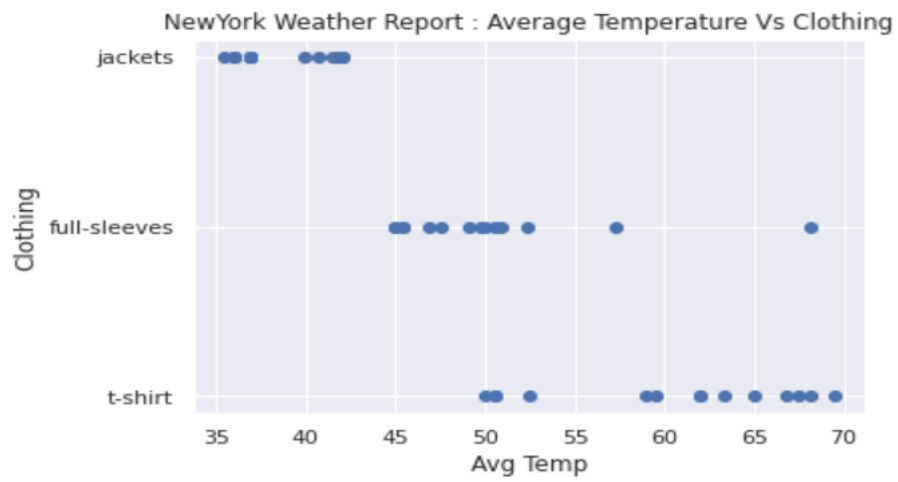
Boston URL: [Copy of Assignment3_datavisualizationBoston.ipynb - Colaboratory \(google.com\)](#)

New York Weather and Response Form Dataset Visualization:

```
[ ] data = pd.read_csv("/content/NewYork_form_response.csv")
```

```
[ ] import matplotlib.pyplot as plt
```

```
plt.scatter(df.Avg_Temperature, df.clothing)  
plt.title(' NewYork Weather Report : Average Temperature Vs Clothing ')  
plt.xlabel('Avg Temp')  
plt.ylabel('Clothing')  
plt.show()
```

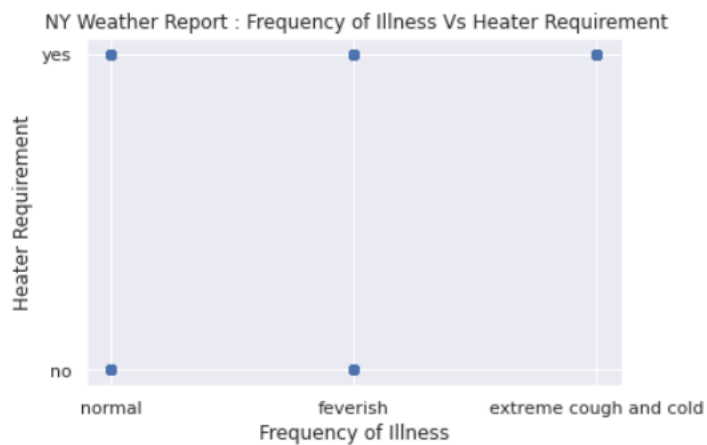



```

import matplotlib.pyplot as plt

plt.scatter(df.frequency_illness, df.heater_req)
plt.title(' NY Weather Report : Frequency of Illness Vs Heater Requirement')
plt.xlabel('Frequency of Illness')
plt.ylabel('Heater Requirement')
plt.show()

```

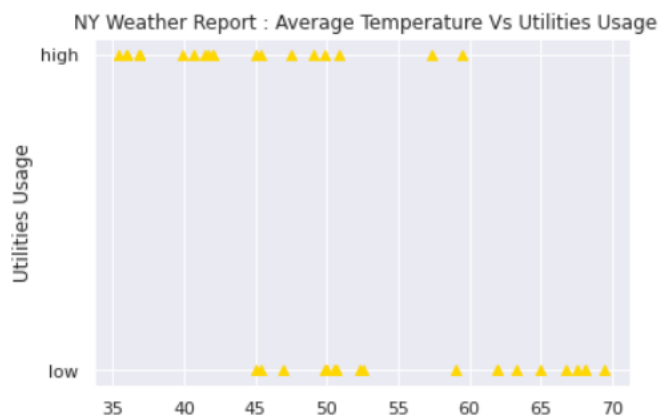


```

[ ] import matplotlib.pyplot as plt

plt.scatter(df.Avg_Temperature, df.gas_electricity_usage, color='gold', marker='^')
plt.title(' NY Weather Report : Average Temperature Vs Utilities Usage')
plt.xlabel('Avg Temp')
plt.ylabel('Utilities Usage')
plt.show()

```





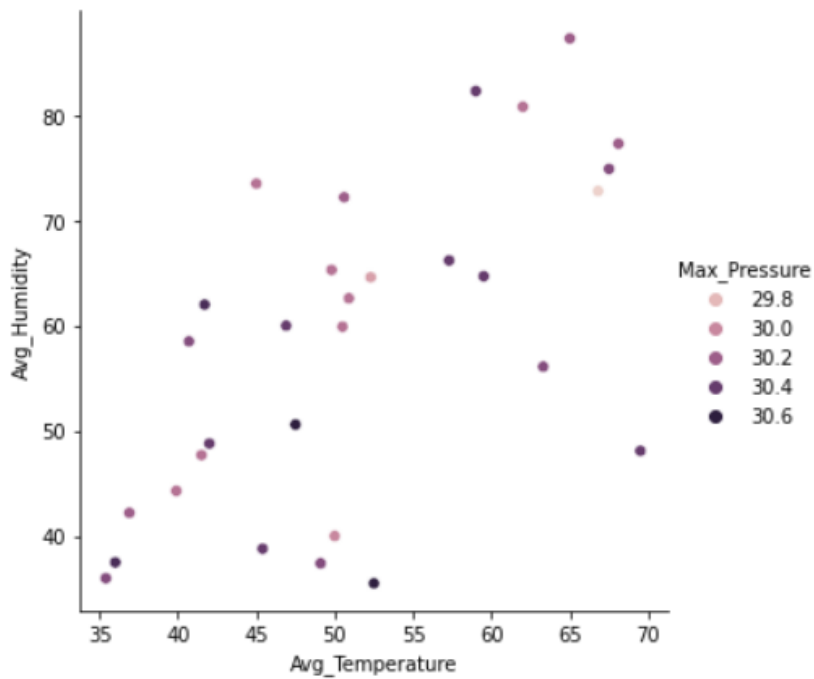
```
df_fullsleeve = df[df['clothing'] == 't-shirt']
df_tshirt = df[df['clothing'] == 'full-sleeve']
df_jacket = df[df['clothing'] == 'jacket']

plt.scatter(df_fullsleeve.clothing, df_fullsleeve.Avg_Temperature, label='Full-sleeve Clothing', color='gold', marker= '^')
plt.scatter(df_tshirt.clothing, df_tshirt.Avg_Temperature, label='T-shirt Clothing', color='silver', marker= '*')
plt.scatter(df_jacket.clothing, df_jacket.Avg_Temperature, label='Jacket Clothing', color='black', marker= 'o')
plt.title('NY Weather Report : Clothing Vs Avg Temperature')
plt.xlabel('clothing')
plt.ylabel('Avg Temo')
plt.legend()
plt.show()
```



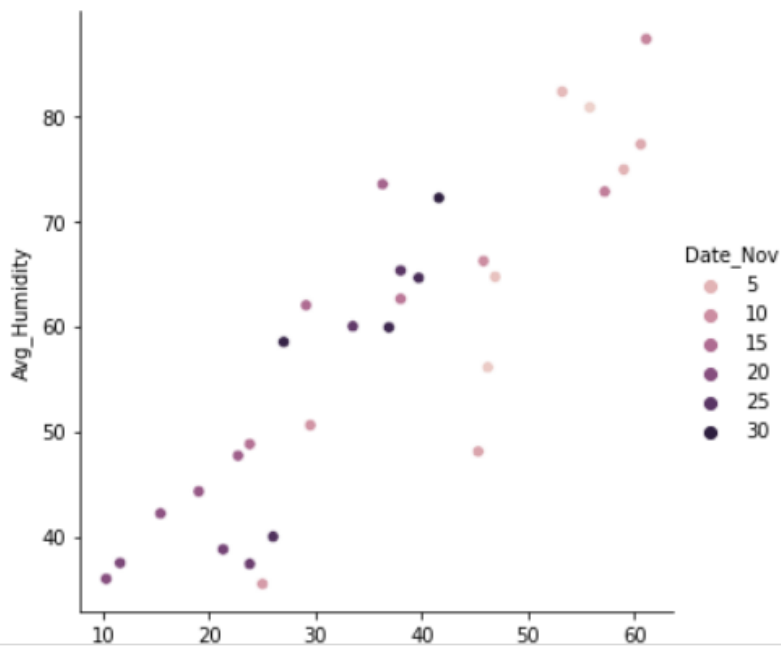
```
sns.relplot(x= 'Avg_Temperature',y='Avg_Humidity', hue='Max_Pressure',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb0c28ba040>
```



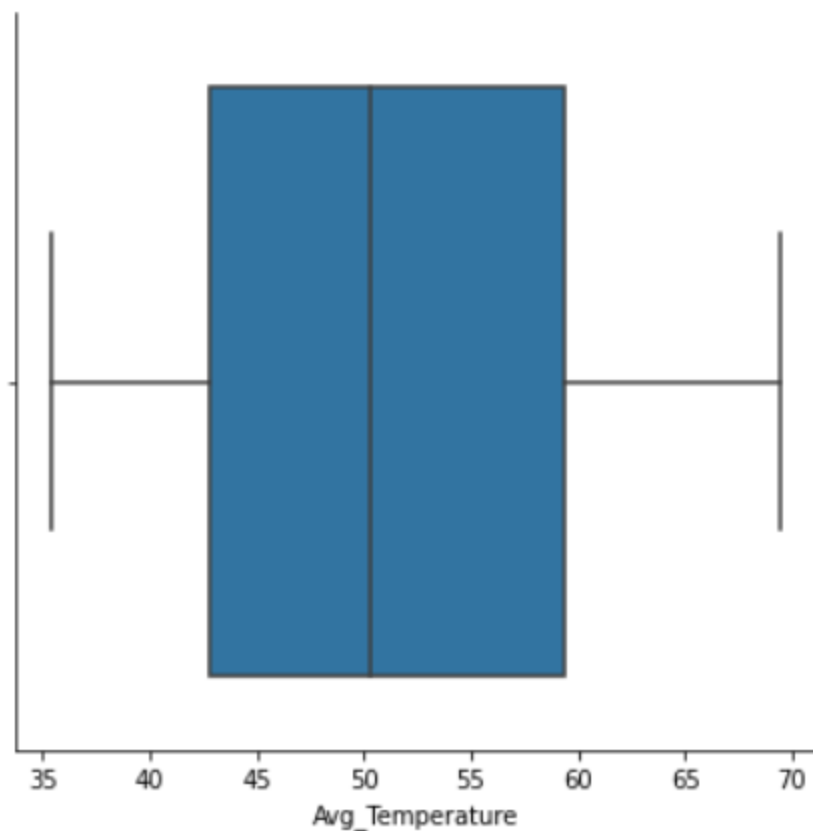
```
sns.relplot(x= 'Avg_Dew Point',y='Avg_Humidity', hue='Date_Nov',data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb0b8f8eee0>
```



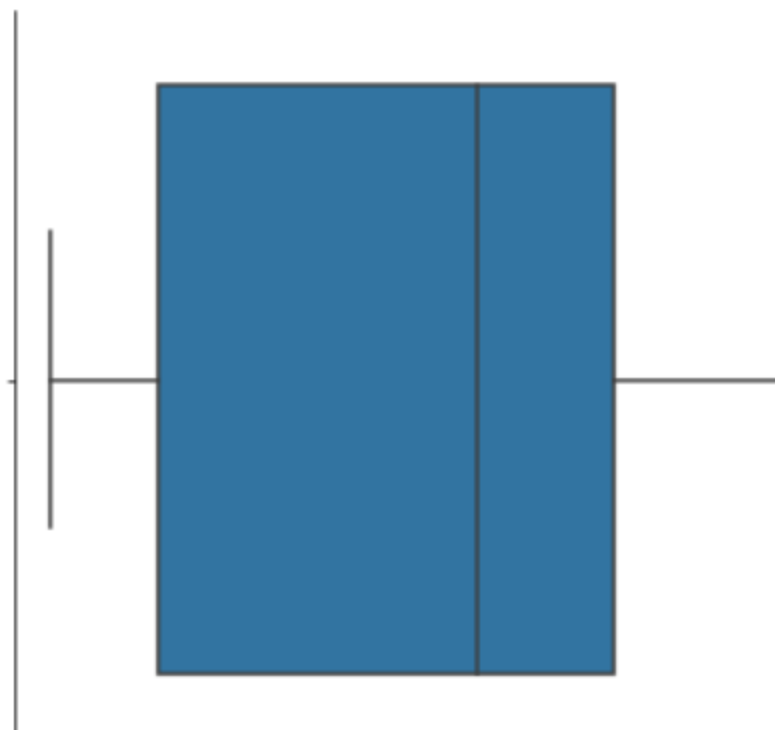
```
▶ sns.catplot(x='Avg_Temperature', kind='box', data=df)
```

```
⦿ <seaborn.axisgrid.FacetGrid at 0x7fb0b88106d0>
```



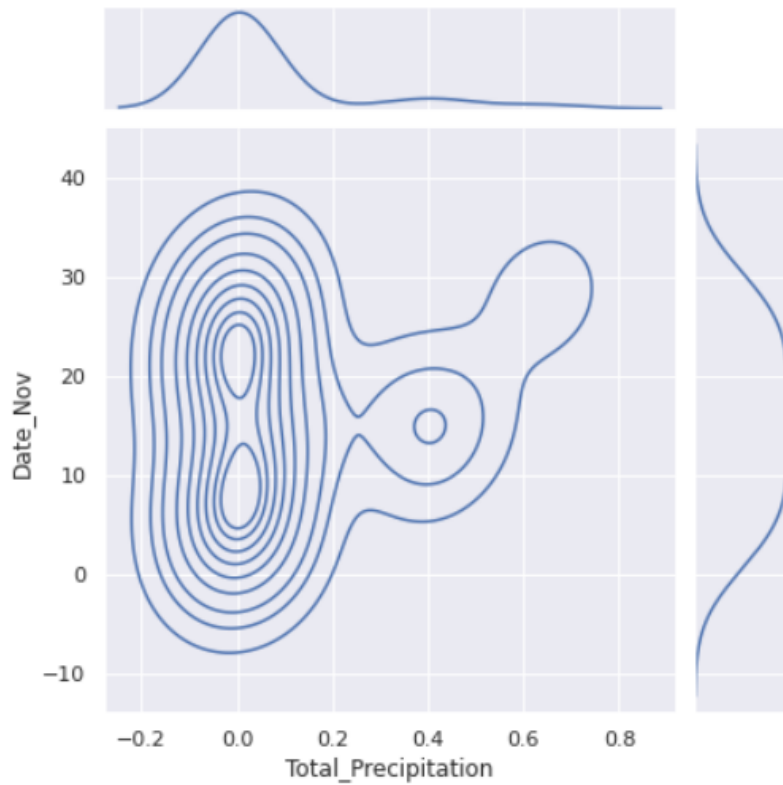
```
[ ] sns.catplot(x='Max_Humidity', kind='box', data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb0b847b340>
```



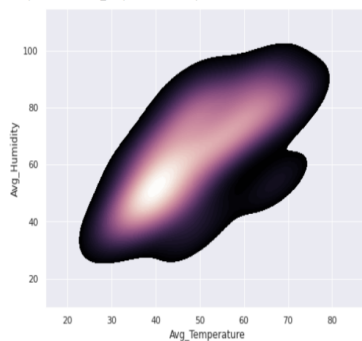
Linear Regression:

```
] import seaborn as sns
sns.set(color_codes=True)
sns.jointplot(x= "Total_Precipitation", y= "Date_Nov", data=df, kind= 'kde');
```



```
f, ax = plt.subplots(figsize=(8, 6))
cmap = sns.cubehelix_palette(as_cmap=True, dark=0, light=1, reverse=True)
sns.kdeplot(df.Avg_Temperature, df.Avg_Humidity, cmap=cmap, n_levels=60, shade=True)
```

`/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: y. From version 0.12, the only valid positional argument will be `data`, and passing other`
`warnings.warn(`
`<matplotlib.axes._subplots.AxesSubplot at 0x7f46ced628e0>`



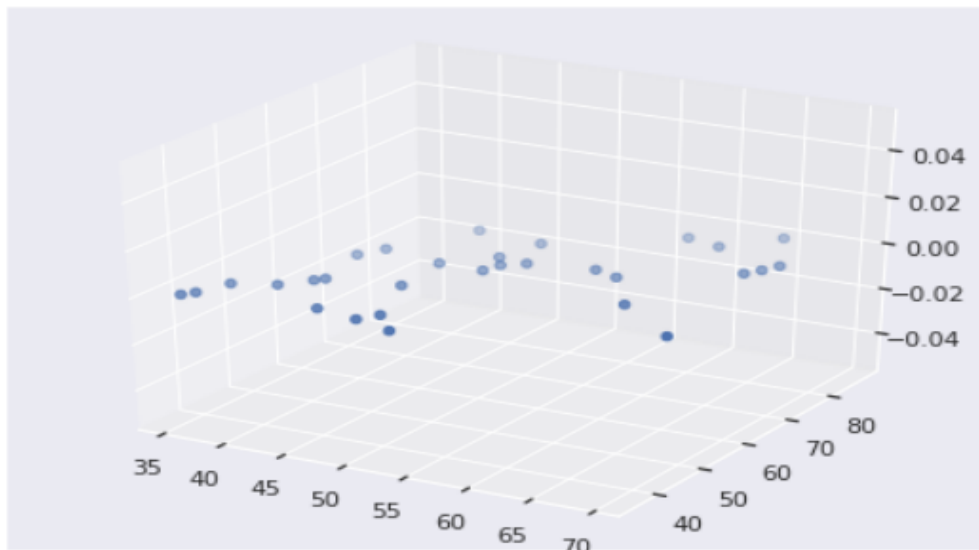
```

▶ from matplotlib import pyplot as plt
  from mpl_toolkits.mplot3d import Axes3D

  font = {'size': 8}
  plt.rc('font', **font)

  fig = plt.figure()
  three_d_plot = Axes3D(fig)
  three_d_plot.scatter(df.Avg_Temperature, df.Avg_Humidity)
  plt.show()

```



Boston Weather and Response Form Dataset Visualization:

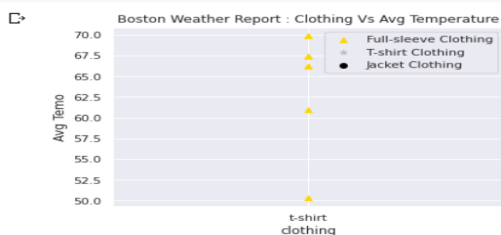
Scatter Plot:

```

▶ df_fullsleeve = df[df['clothing'] == 't-shirt']
  df_tshirt = df[df['clothing'] == 'full-sleeve']
  df_jacket = df[df['clothing'] == 'jacket']

  plt.scatter(df_fullsleeve.clothing, df_fullsleeve.Avg_Temperature, label='Full-sleeve Clothing', color='gold', marker='^')
  plt.scatter(df_tshirt.clothing, df_tshirt.Avg_Temperature, label='T-shirt Clothing', color='silver', marker='*')
  plt.scatter(df_jacket.clothing, df_jacket.Avg_Temperature, label='Jacket Clothing', color='black', marker='o')
  plt.title('Boston Weather Report : Clothing Vs Avg Temperature')
  plt.xlabel('clothing')
  plt.ylabel('Avg Temo')
  plt.legend()
  plt.show()

```

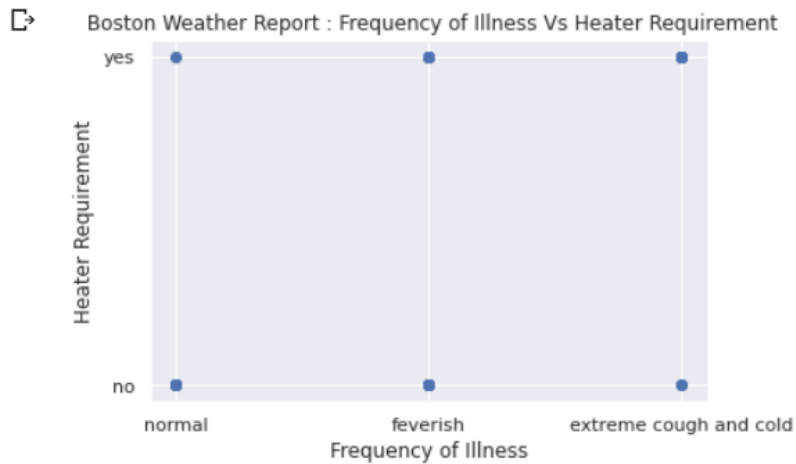


```

import matplotlib.pyplot as plt

plt.scatter(df.frequency_illness, df.heater_req)
plt.title(' Boston Weather Report : Frequency of Illness Vs Heater Requirement')
plt.xlabel('Frequency of Illness')
plt.ylabel('Heater Requirement')
plt.show()

```

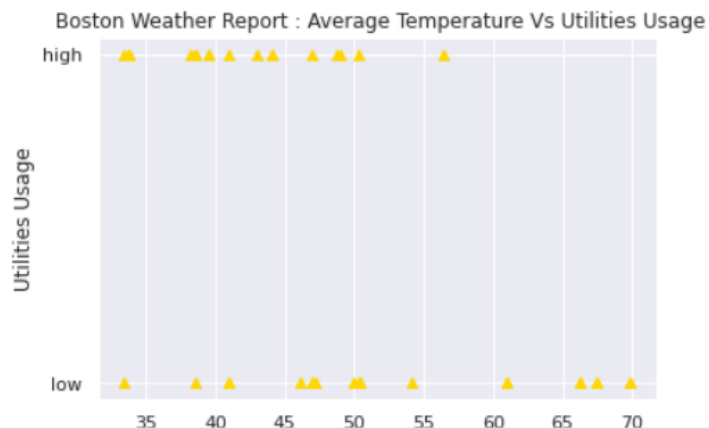


```

[ ] import matplotlib.pyplot as plt

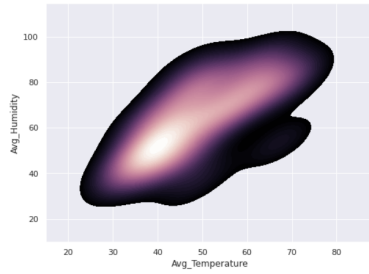
plt.scatter(df.Avg_Temperature, df.gas_electricity_usage, color='gold', marker='^')
plt.title(' Boston Weather Report : Average Temperature Vs Utilities Usage')
plt.xlabel('Avg Temp')
plt.ylabel('Utilities Usage')
plt.show()

```



```
f, ax = plt.subplots(figsize=(8, 6))
cmap = sns.cubehelix_palette(as_cmap=True, dark=0, light=1, reverse=True)
sns.kdeplot(df.Avg_Temperature, df.Avg_Humidity, cmap=cmap, n_levels=60, shade=True)

/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: y. From version 0.12, the only valid positional argument will be `data`, and passing other
warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f365a832880>
```



```
[ ] from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

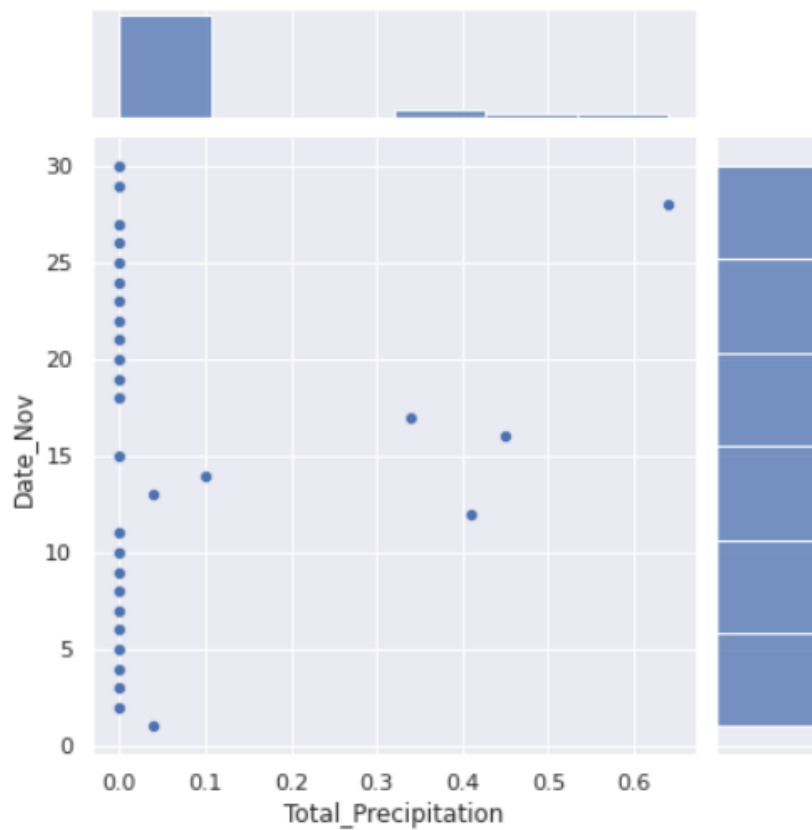
font = {'size': 8}
plt.rc('font', **font)

fig = plt.figure()
three_d_plot = Axes3D(fig)
three_d_plot.scatter(df.Avg_Temperature, df.Avg_Humidity)
plt.show()
```

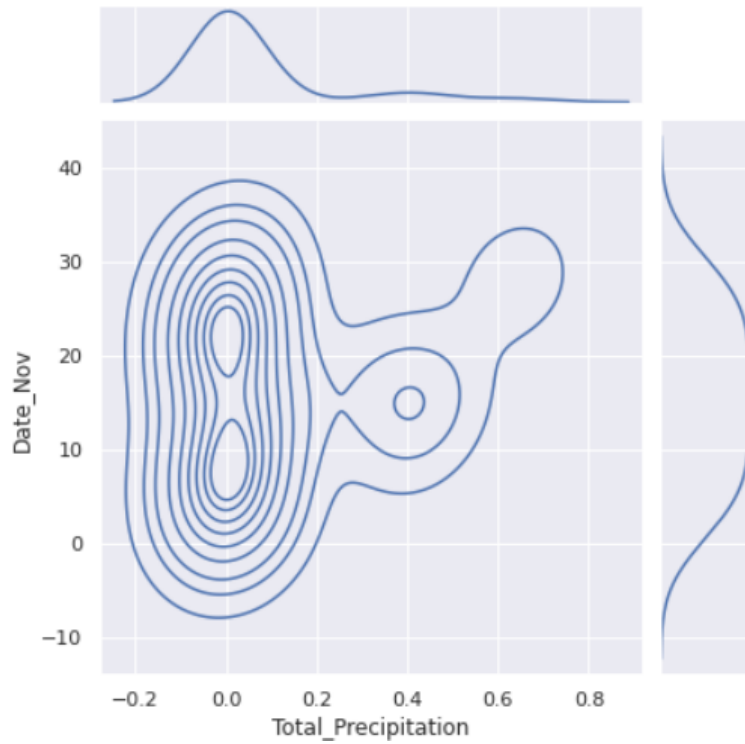



```
import seaborn as sns
sns.set(color_codes=True)
sns.jointplot(x= "Total_Precipitation", y= "Date_Nov", data=df)
```

<seaborn.axisgrid.JointGrid at 0x7f3661fbf7f0>



```
[ ] import seaborn as sns
sns.set(color_codes=True)
sns.jointplot(x= "Total_Precipitation", y= "Date_Nov", data=df, kind= 'kde');
```



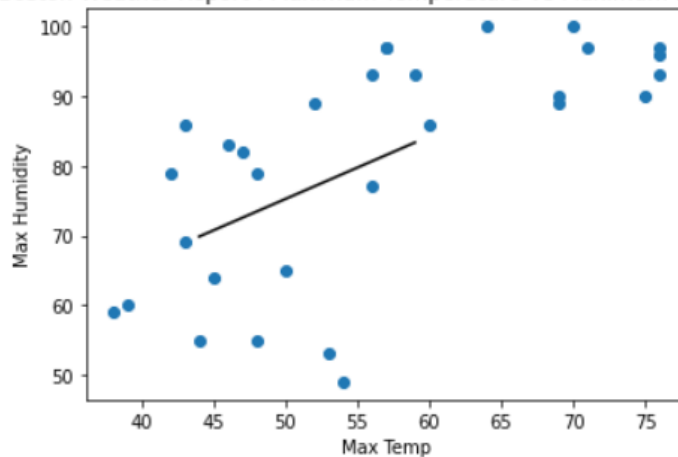
```

# y = mx + b
# y = slope(x) + intercept

m, b, r, p, err = stats.linregress(df.Max_Temperature, df.Max_Humidity)
x = range(44, 60)
y = m * x + b
plt.plot(x, y, color='black')
plt.scatter(df.Max_Temperature, df.Max_Humidity)
plt.title('Boston Weather Report : Maximum Temperature Vs Maximum Humidity')
plt.xlabel('Max Temp')
plt.ylabel('Max Humidity')
plt.show()

```

☐ Boston Weather Report : Maximum Temperature Vs Maximum Humidity



```

[ ] from statsmodels.stats.diagnostic import het_breuschpagan
    from statsmodels.stats.diagnostic import het_white
    from statsmodels.formula.api import ols

model = ols(formula='Max_Temperature~Max_Humidity', data=df).fit()

white_test = het_white(model.resid, model.model.exog)
breuschpagan_test = het_breuschpagan(model.resid, model.model.exog)

output_df = pd.DataFrame(columns=['LM stat', 'LM p', 'F stat', 'F stat p'])
output_df.loc['White'] = white_test
output_df.loc['Breusch-Pagan'] = breuschpagan_test

output_df

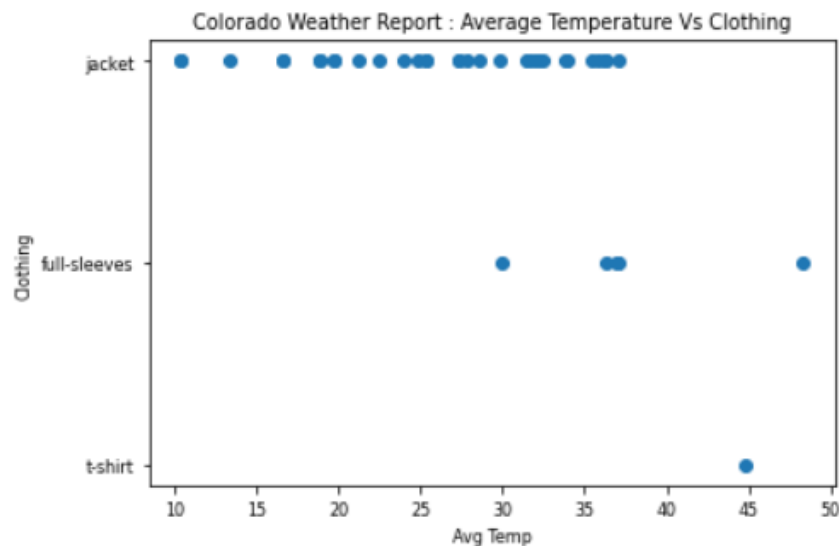
```

	LM stat	LM p	F stat	F stat p
White	0.200923	0.904420	0.091025	0.913274
Breusch-Pagan	0.192211	0.661083	0.180553	0.674146

Colorado Weather Data and Response Form : Data Visualization

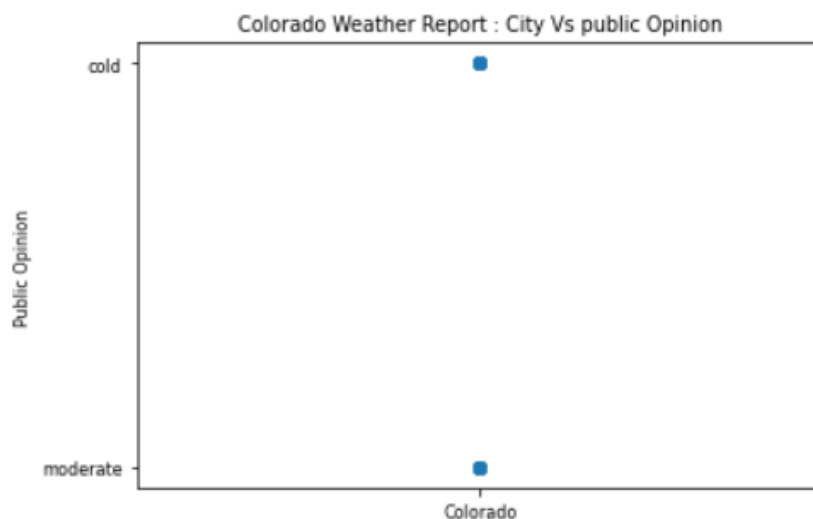

```
[ ] import matplotlib.pyplot as plt

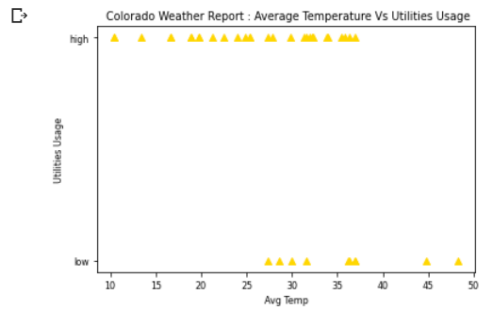
plt.scatter(df.Avg_Temperature, df.clothing)
plt.title(' Colorado Weather Report : Average Temperature Vs Clothing ')
plt.xlabel('Avg Temp')
plt.ylabel('Clothing')
plt.show()
```



```
[ ] import matplotlib.pyplot as plt

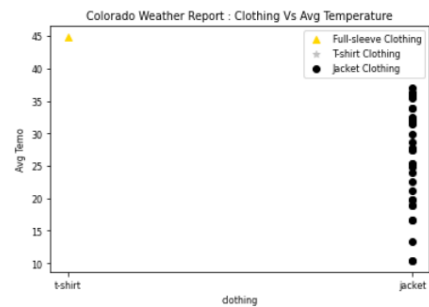
plt.scatter(df.City, df.public_opinion)
plt.title(' Colorado Weather Report : City Vs public Opinion ')
plt.xlabel('City')
plt.ylabel('Public Opinion')
plt.show()
```





```
[ ] df_fullsleeve = df[df['clothing'] == 't-shirt']
df_tshirt = df[df['clothing'] == 'full-sleeve']
df_jacket = df[df['clothing'] == 'jacket']

plt.scatter(df_fullsleeve.clothing, df_fullsleeve.Avg_Temperature, label='Full-sleeve Clothing', color='gold', marker= '^')
plt.scatter(df_tshirt.clothing, df_tshirt.Avg_Temperature, label='T-shirt Clothing', color='silver', marker= '*')
plt.scatter(df_jacket.clothing, df_jacket.Avg_Temperature, label='Jacket Clothing', color='black', marker= 'o')
plt.title('Colorado Weather Report : Clothing Vs Avg Temperature')
plt.xlabel('clothing')
plt.ylabel('Avg Temo')
plt.legend()
plt.show()
```



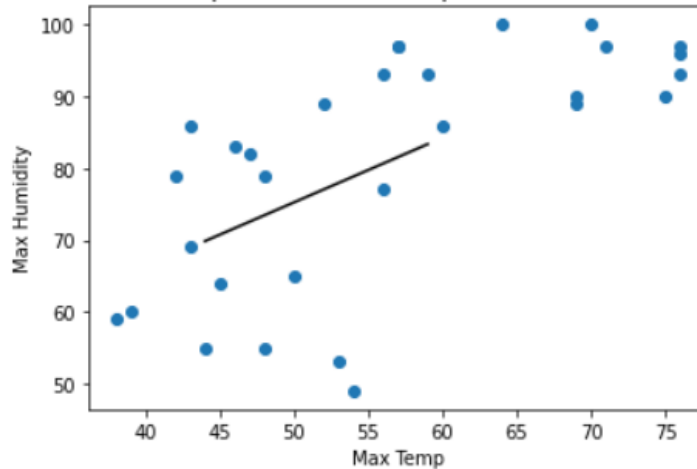


```
# y = mx + b
# y = slope(x) + intercept

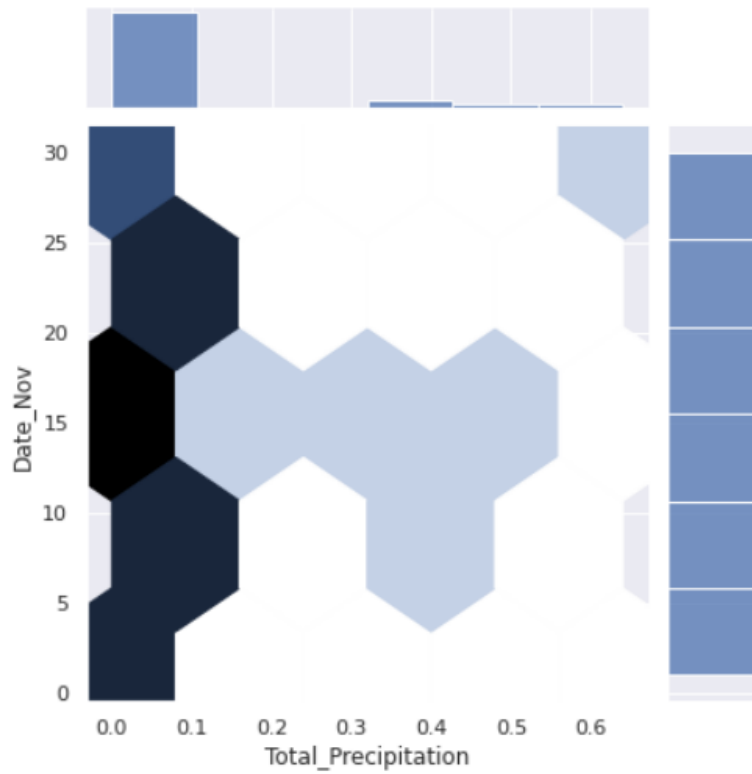
m, b, r, p, err = stats.linregress(df.Max_Temperature, df.Max_Humidity)
x = range(44, 60)
y = m * x + b
plt.plot(x, y, color='black')
plt.scatter(df.Max_Temperature, df.Max_Humidity)
plt.title('Colorado Weather Report : Maximum Temperature Vs Maximum Humidity')
plt.xlabel('Max Temp')
plt.ylabel('Max Humidity')
plt.show()
```



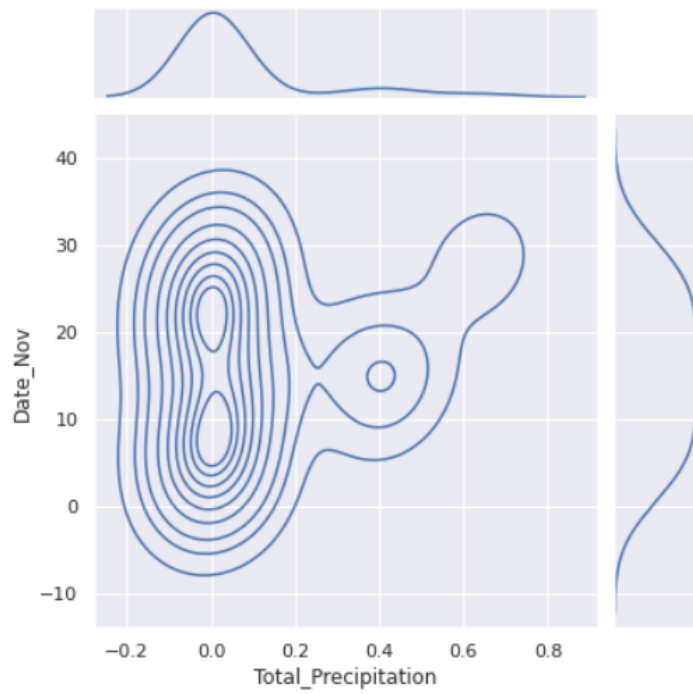
Boston Weather Report : Maximum Temperature Vs Maximum Humidity



```
[ ] import seaborn as sns
sns.set(color_codes=True)
sns.jointplot(x= "Total_Precipitation", y= "Date_Nov", data=df, kind= 'hex');
```



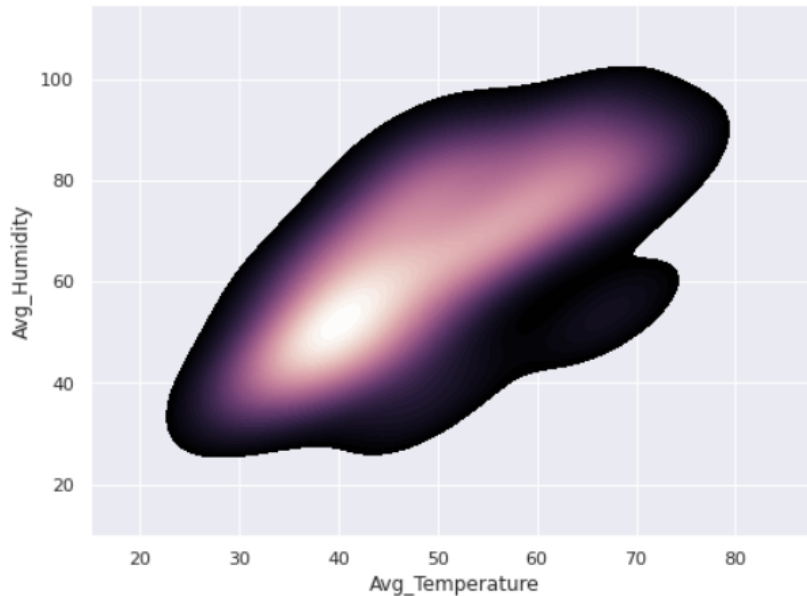
```
[ ] import seaborn as sns
sns.set(color_codes=True)
sns.jointplot(x= "Total_Precipitation", y= "Date_Nov", data=df, kind= 'kde');
```



```
] f, ax = plt.subplots(figsize=(8, 6))
cmap = sns.cubehelix_palette(as_cmap=True, dark=0, light=1, reverse=True)
sns.kdeplot(df.Avg_Temperature, df.Avg_Humidity, cmap=cmap, n_levels=60, shade=True)
```

/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following warnings as a list of dicts. Use the 'warn_kwarg' keyword as an example. (Please see the 'warn' keyword in the documentation.)

<matplotlib.axes._subplots.AxesSubplot at 0x7f46ced628e0>



Data Visualization Collabs:

Colorado URL: [Assignment3_datavisualizationColorado.ipynb - Colaboratory \(google.com\)](#)

New York URL: [Assignment3_datavisualizationNewYork.ipynb - Colaboratory \(google.com\)](#)

Boston URL: [Copy of Assignment3_datavisualizationBoston.ipynb - Colaboratory \(google.com\)](#)

GitHub URL: [Dhiral25/Weather-Analysis \(github.com\)](#)