Walmart version b

August 14, 2019

1 Important Note:

- This assigement requires at least PostgreSQL 9.5 or higher release since this assignments uses **ROLLUP** and **CUBE** grouping operations
- Curently the DSCC has a PostgreSQL 10.5 that we will use for this assignment

2 Deliverables:

- Submit files that have the name: YourLastName_Walmart:

1. Your **PDF document** that has your Source code and output
2. Your **ipynb script** that has your Source code and output
3. You may zip these files and submit

3 Objectives:

- Experiment with SQL grouping operations like CUBE and ROLLUP to retrieve, group and cluster data from Walmart dataset
- Use Ecnomic data to analyze and visualize the weekly total sales per Walmart-store

4 Author: Atef Bader

1. Last Edit 10-28-2018

image.png

5 Descriptions and Requirement Specifications

5.1 Walmart Weekly Sales

Walmart is the world's largest company by revenue, it has over US$500 billion; Walmart has 11,718 stores and clubs in 28 countries.

You can read more about Walmart by visitin the following page **More Info**

5.2 Walmart total weekly sales

Walmart tracks the total weekly sales per store and there is a sample dataset that is published on **Kaggle**, you can read more about this dataset **here**

In this assignment we will not discuss the store sales **prediction**, rather we will focus on store sales **description**

Note: Unlike the employment weekly numbers, the CPI is published monthly by LBS even though it is listed on a weekly basis teh given dataset.

The walmart database is composed of 3 tables that are populated on PosgeSQL10.5 on DSCC. The tables are listed below:

Stores:

- Store: The store number. Range from 1-45.
- Type: Three types of stores 'A', 'B' or 'C'.
- Size: Sets the size of a Store would be calculated by the no. of products available in the particular store ranging from 34,000 to 210,000.

Weekly_Sales:

- Store: The store which observation in recorded 1-45.
- Dept: One of 1-99 that shows the department.
- IsHoliday: Boolean value representing a holiday week or not.

Features:

- Temperature:Temperature of the region during that week.
- Fuel_Price: Fuel Price in that region during that week.
- MarkDown1:5 : Represents the Type of markdown and what quantity was available during that week.
- CPI: Consumer Price Index during that week.
- Unemployment: The unemployment rate during that week in the region of the store.

6 Bureau of Labor Statistics

The Bureau of Labor Statistics **(BLS)** of the U.S. Department of Labor publishes many of the monthly and weekly MAJOR ECONOMIC INDICATORS that are used to measure the labor market, inflation and price changes in the economy. You can read more about these indicators **here**

Examples of these indicators that you will see in the database for this assignment are unemployement and CPI

image.png

7 PostgreSQL 10.5

Postgres provides few programming language constructs for complex grouping operations like ROLLUP and CUBE. You can read more about these grouping operations **here**

image.png

7.1 Import the packages needed

[1]:
```
import psycopg2
import csv
import pandas as pd
import numpy as np
from datetime import datetime,date
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

%matplotlib inline
```

[2]:
```
#setup the format for the pretty print of Total
pd.options.display.float_format = '{:20,.2f}'.format
```

7.1.1 First, we need to connect to the postgresql10 database server

- make sure you are already connected to VPN before you execute the following command

[3]:
```
db_connection = psycopg2.connect(host='postgresql10.sps.northwestern.
 ↪edu',dbname="walmart", user="aso4098" , password="")


cursor = db_connection.cursor()
```

[11]:
```
cursor.execute("SELECT * from weekly_sales")
rows=cursor.fetchall()
colnames = [desc[0] for desc in cursor.description]
df = pd.DataFrame(rows, columns = colnames)
#df.to_csv("C:\\Users\\asidd\\Desktop\\MSDS\\420 Database Systems\\Lecture␣
 ↪7\\walmart.csv")
```

7.2 Query #1:

- Get the total weekly sales by store

[4]:
```
cursor.execute("SELECT Store, sum(Weekly_Sales) from weekly_sales GROUP BY␣
 ↪Store")
rows=cursor.fetchall()
```

[6]:
```
colnames = [desc[0] for desc in cursor.description]

#total_weekly_sales_by_store = pd.DataFrame(rows, columns=['Store','Total'])
total_weekly_sales_by_store = pd.DataFrame(rows, columns = colnames)
```

```
total_weekly_sales_by_store.tail()
```

[6]:

	store	sum
40	41	181,341,934.89
41	42	79,565,752.43
42	43	90,565,435.41
43	44	43,293,087.84
44	45	112,395,341.42

7.3 Query #2:

- Get the total weekly sales by store and department

[8]:
```
cursor.execute("SELECT Store, Dept, sum(Weekly_Sales) from weekly_sales GROUP␣
 ↪BY Store, Dept")
rows=cursor.fetchall()
```

[10]:
```
colnames = [desc[0] for desc in cursor.description]

#total_weekly_sales_by_store_dept = pd.DataFrame(rows,␣
 ↪columns=['Store','Department','Total'])
total_weekly_sales_by_store_dept = pd.DataFrame(rows, columns = colnames)

total_weekly_sales_by_store_dept = total_weekly_sales_by_store_dept.
 ↪sort_values(by=['store', 'dept'])

total_weekly_sales_by_store_dept
```

[10]:

	store	dept	sum
11	1	1	3,219,405.18
2015	1	2	6,592,598.93
672	1	3	1,880,518.36
303	1	4	5,285,874.09
35	1	5	3,468,885.58
940	1	6	686,654.56
1408	1	7	3,513,007.70
2084	1	8	5,107,710.84
2107	1	9	4,012,873.47
328	1	10	4,437,774.25
25	1	11	3,563,455.70
2214	1	12	1,511,015.98
2974	1	13	5,533,081.91
1020	1	14	2,183,402.78
2025	1	16	3,453,601.77
1596	1	17	1,315,107.78
1439	1	18	877,479.40
1793	1	19	180,039.65

1436	1	20	585,094.73
255	1	21	1,116,608.43
2046	1	22	1,151,446.89
850	1	23	3,092,115.41
1912	1	24	884,796.73
252	1	25	1,451,784.16
2976	1	26	967,823.61
516	1	27	196,574.90
20	1	28	84,815.30
859	1	29	665,098.75
967	1	30	488,387.19
1425	1	31	344,420.26
...	...	...	...
2780	45	51	104.52
2605	45	52	209,675.92
2750	45	54	5,784.62
2573	45	55	870,254.51
1714	45	56	519,397.43
3076	45	58	241,179.85
1842	45	59	85,912.45
3121	45	60	25,313.20
898	45	67	972,546.81
1413	45	71	628,686.82
2810	45	72	5,357,682.10
207	45	74	1,531,547.32
2150	45	77	1,525.96
3183	45	78	88.00
2851	45	79	2,180,038.31
540	45	80	72,622.47
1023	45	81	2,095,392.73
3219	45	82	1,920,560.27
2904	45	83	117,915.34
802	45	85	286,197.78
430	45	87	901,911.85
3046	45	90	3,385,387.04
629	45	91	2,379,795.61
924	45	92	6,882,003.38
2606	45	93	390,193.68
1032	45	94	494,496.46
1447	45	95	7,564,151.83
749	45	96	5.94
56	45	97	924,775.55
2461	45	98	75,767.27

[3331 rows x 3 columns]

7.4 Query #3:

- Get the total weekly sales by store and week

[8]:
```
cursor.execute("SELECT Store, Date, sum(Weekly_Sales) from weekly_sales GROUP␣
 ↪BY Store, Date")
rows=cursor.fetchall()
```

[9]:
```
total_weekly_sales_by_store_dept = pd.DataFrame(rows,␣
 ↪columns=['Store','Date','Total'])

total_weekly_sales_by_store_dept = total_weekly_sales_by_store_dept.
 ↪sort_values(by=['Store', 'Date'])


total_weekly_sales_by_store_dept
```

[9]:

	Store	Date	Total
3874	1	2010-02-05	1,643,690.90
2271	1	2010-02-12	1,641,957.44
4071	1	2010-02-19	1,611,968.17
855	1	2010-02-26	1,409,727.59
2331	1	2010-03-05	1,554,806.68
4027	1	2010-03-12	1,439,541.59
4302	1	2010-03-19	1,472,515.79
2925	1	2010-03-26	1,404,429.92
5899	1	2010-04-02	1,594,968.28
2009	1	2010-04-09	1,545,418.53
3162	1	2010-04-16	1,466,058.28
442	1	2010-04-23	1,391,256.12
2556	1	2010-04-30	1,425,100.71
1608	1	2010-05-07	1,603,955.12
2849	1	2010-05-14	1,494,251.50
1874	1	2010-05-21	1,399,662.07
1035	1	2010-05-28	1,432,069.95
6002	1	2010-06-04	1,615,524.71
2163	1	2010-06-11	1,542,561.09
4310	1	2010-06-18	1,503,284.06
1135	1	2010-06-25	1,422,711.60
602	1	2010-07-02	1,492,418.14
3056	1	2010-07-09	1,546,074.18
4960	1	2010-07-16	1,448,938.92
2038	1	2010-07-23	1,385,065.20
4054	1	2010-07-30	1,371,986.60
4969	1	2010-08-06	1,605,491.78
2525	1	2010-08-13	1,508,237.76
6075	1	2010-08-20	1,513,080.49
5004	1	2010-08-27	1,449,142.92
...	...	...	...

```
3310     45 2012-04-06          899,479.43
5738     45 2012-04-13          781,970.60
4526     45 2012-04-20          776,661.74
4959     45 2012-04-27          711,571.88
691      45 2012-05-04          782,300.68
5338     45 2012-05-11          770,487.37
2475     45 2012-05-18          800,842.28
1284     45 2012-05-25          817,741.17
3270     45 2012-06-01          837,144.63
1718     45 2012-06-08          795,133.00
207      45 2012-06-15          821,498.18
1807     45 2012-06-22          822,569.16
4303     45 2012-06-29          773,367.71
5519     45 2012-07-06          843,361.10
6151     45 2012-07-13          749,817.08
1453     45 2012-07-20          737,613.65
4402     45 2012-07-27          711,671.58
5304     45 2012-08-03          725,729.51
3814     45 2012-08-10          733,037.32
3416     45 2012-08-17          722,496.93
3840     45 2012-08-24          718,232.26
5159     45 2012-08-31          734,297.87
6109     45 2012-09-07          766,512.66
3771     45 2012-09-14          702,238.27
1716     45 2012-09-21          723,086.20
2566     45 2012-09-28          713,173.95
3423     45 2012-10-05          733,455.07
5940     45 2012-10-12          734,464.36
5850     45 2012-10-19          718,125.53
3439     45 2012-10-26          760,281.43

[6435 rows x 3 columns]
```

7.5 Query #4:

- Get the total weekly sales by department and week

[10]:
```
cursor.execute("SELECT Dept, Date, sum(Weekly_Sales) from weekly_sales GROUP BY␣
 ↪Dept, Date")
rows=cursor.fetchall()
```

[11]:
```
total_weekly_sales_by_dept_date = pd.DataFrame(rows,␣
 ↪columns=['Dept','Date','Total'])

total_weekly_sales_by_dept_date = total_weekly_sales_by_dept_date.
 ↪sort_values(by=['Dept', 'Date'])
```

```
total_weekly_sales_by_dept_date
```

[11]:

	Dept	Date	Total
8954	1	2010-02-05	881,833.41
7577	1	2010-02-12	1,457,182.40
9110	1	2010-02-19	1,118,257.36
6345	1	2010-02-26	681,391.58
2020	1	2010-03-05	762,652.57
9090	1	2010-03-12	803,886.93
9338	1	2010-03-19	846,686.47
8139	1	2010-03-26	1,045,724.42
5118	1	2010-04-02	2,451,952.54
7341	1	2010-04-09	1,518,946.82
2731	1	2010-04-16	609,719.72
391	1	2010-04-23	588,496.89
2228	1	2010-04-30	600,156.00
7004	1	2010-05-07	669,362.80
2472	1	2010-05-14	666,912.20
7226	1	2010-05-21	633,105.07
869	1	2010-05-28	643,528.83
10762	1	2010-06-04	648,393.81
1870	1	2010-06-11	648,890.42
9346	1	2010-06-18	665,184.13
6583	1	2010-06-25	648,196.08
6108	1	2010-07-02	678,962.23
2655	1	2010-07-09	637,733.18
4323	1	2010-07-16	648,002.69
1782	1	2010-07-23	631,305.04
3493	1	2010-07-30	634,299.47
4334	1	2010-08-06	642,139.20
7809	1	2010-08-13	624,871.72
10814	1	2010-08-20	608,719.73
9917	1	2010-08-27	609,494.16
...	...	...	...
7845	99	2012-03-23	1.01
6959	99	2012-04-06	1,553.18
286	99	2012-04-13	386.06
911	99	2012-04-20	1.90
1899	99	2012-05-04	7,400.01
8334	99	2012-05-11	1,105.00
1009	99	2012-05-18	-11.99
795	99	2012-05-25	32.44
5931	99	2012-06-01	29.90
7462	99	2012-06-08	2,404.76
39	99	2012-06-15	4,134.76
2471	99	2012-06-22	800.00

```
1036     99 2012-06-29                0.00
7284     99 2012-07-06              210.29
2265     99 2012-07-13               61.43
10903    99 2012-07-20              150.97
3663     99 2012-07-27              120.54
11011    99 2012-08-03            1,820.76
1165     99 2012-08-10              510.32
9585     99 2012-08-17              411.25
8639     99 2012-08-24            3,091.09
8631     99 2012-08-31              611.11
7085     99 2012-09-07              234.04
5069     99 2012-09-14               31.16
9086     99 2012-09-21              119.81
4854     99 2012-09-28                0.01
8752     99 2012-10-05           11,587.17
2196     99 2012-10-12            2,024.93
9054     99 2012-10-19                0.03
7563     99 2012-10-26               41.89

[11090 rows x 3 columns]
```

7.6 Query #5:

- Get the total weekly sales using **ROLLUP** clause for the hierarchical data: store, department and week

[12]:
```
cursor.execute("SELECT Store, Dept, Date, sum(Weekly_Sales) from weekly_sales␣
 ↪GROUP BY ROLLUP(Store, Dept, Date)")
rows=cursor.fetchall()
```

[13]:
```
rollup_by_store_dept_date = pd.DataFrame(rows,␣
 ↪columns=['Store','Dept','Date','Total'])

#For the pretty print : Replace NaN by -1

rollup_by_store_dept_date['Store'] = rollup_by_store_dept_date['Store'].
 ↪replace(np.nan,-1)
rollup_by_store_dept_date['Dept'] = rollup_by_store_dept_date['Dept'].
 ↪replace(np.nan,-1)

#type conversion of values returned by ROLLUP

rollup_by_store_dept_date['Store'] = rollup_by_store_dept_date['Store'].
 ↪astype(int)
rollup_by_store_dept_date['Dept'] = rollup_by_store_dept_date['Dept'].
 ↪astype(int)
```

```
rollup_by_store_dept_date['Total'] = rollup_by_store_dept_date['Total'].
 ↪astype(np.float64)


#For the pretty print: Replace -1  by BLANK for Store and department
rollup_by_store_dept_date['Store'].replace(-1,'',inplace=True)
rollup_by_store_dept_date['Dept'].replace(-1,'',inplace=True)

#For the pretty print: Replace NaN  by BLANK for date
rollup_by_store_dept_date['Date'] = rollup_by_store_dept_date['Date'].dt.date
rollup_by_store_dept_date['Date'].replace(np.nan,'',inplace=True)


rollup_by_store_dept_date = rollup_by_store_dept_date.sort_values(by=['Store',␣
 ↪'Dept', 'Date'])


rollup_by_store_dept_date
```

[13]:

	Store	Dept	Date	Total
56093	1	1	2010-02-05	24,924.50
267400	1	1	2010-02-12	46,039.49
10751	1	1	2010-02-19	41,595.55
295442	1	1	2010-02-26	19,403.54
162717	1	1	2010-03-05	21,827.90
287383	1	1	2010-03-12	21,043.39
248692	1	1	2010-03-19	22,136.64
317889	1	1	2010-03-26	26,229.21
157684	1	1	2010-04-02	57,258.43
219234	1	1	2010-04-09	42,960.91
409083	1	1	2010-04-16	17,596.96
344205	1	1	2010-04-23	16,145.35
50692	1	1	2010-04-30	16,555.11
38051	1	1	2010-05-07	17,413.94
377780	1	1	2010-05-14	18,926.74
356067	1	1	2010-05-21	14,773.04
318343	1	1	2010-05-28	15,580.43
258991	1	1	2010-06-04	17,558.09
304988	1	1	2010-06-11	16,637.62
298146	1	1	2010-06-18	16,216.27
66777	1	1	2010-06-25	16,328.72
49534	1	1	2010-07-02	16,333.14
169403	1	1	2010-07-09	17,688.76
190943	1	1	2010-07-16	17,150.84
366965	1	1	2010-07-23	15,360.45
211566	1	1	2010-07-30	15,381.82
399671	1	1	2010-08-06	17,508.41

```
251978     1    1  2010-08-13           15,536.40
325142     1    1  2010-08-20           15,740.13
321811     1    1  2010-08-27           15,793.87
...      ...  ...         ...                 ...
13095     45   98  2012-04-27              619.41
36605     45   98  2012-05-04              694.25
206934    45   98  2012-05-11              893.60
3776      45   98  2012-05-18              745.44
135480    45   98  2012-05-25              795.94
31131     45   98  2012-06-01              874.64
211184    45   98  2012-06-08              713.50
237333    45   98  2012-06-15              856.35
198669    45   98  2012-06-22              622.62
223119    45   98  2012-06-29              690.52
291750    45   98  2012-07-06              659.65
31624     45   98  2012-07-13              695.21
134526    45   98  2012-07-20              845.30
227689    45   98  2012-07-27              657.63
253329    45   98  2012-08-03              516.46
313596    45   98  2012-08-10              727.49
168367    45   98  2012-08-17              500.16
71731     45   98  2012-08-24              415.40
6841      45   98  2012-08-31              346.04
35985     45   98  2012-09-07              352.44
124491    45   98  2012-09-14              605.96
243787    45   98  2012-09-21              467.30
349445    45   98  2012-09-28              508.37
29738     45   98  2012-10-05              628.10
358054    45   98  2012-10-12            1,061.02
41037     45   98  2012-10-19              760.01
380735    45   98  2012-10-26            1,076.80
424031    45   98                       75,767.27
424914    45                       112,395,341.42
0                                 6,737,218,987.11

[424947 rows x 4 columns]
```

7.7 Query #6:

- Get the total weekly sales using **ROLLUP** clause for the hierarchical data: store, department and week

[12]:
```
cursor.execute("SELECT Store, Dept, sum(Weekly_Sales) from weekly_sales GROUP␣
 ↪BY ROLLUP(Store, Dept)")
rows=cursor.fetchall()
```

[15]:
```
pd.DataFrame(rows, columns=['Store','Dept','Total']).sort_values(by=['Store',␣
 ↪'Dept'])
```

[15]:

	Store	Dept	Total
13	1.00	1.00	3,219,405.18
2017	1.00	2.00	6,592,598.93
674	1.00	3.00	1,880,518.36
305	1.00	4.00	5,285,874.09
37	1.00	5.00	3,468,885.58
941	1.00	6.00	686,654.56
1412	1.00	7.00	3,513,007.70
2086	1.00	8.00	5,107,710.84
2108	1.00	9.00	4,012,873.47
329	1.00	10.00	4,437,774.25
26	1.00	11.00	3,563,455.70
2216	1.00	12.00	1,511,015.98
2976	1.00	13.00	5,533,081.91
1023	1.00	14.00	2,183,402.78
2026	1.00	16.00	3,453,601.77
1598	1.00	17.00	1,315,107.78
1441	1.00	18.00	877,479.40
1795	1.00	19.00	180,039.65
1438	1.00	20.00	585,094.73
257	1.00	21.00	1,116,608.43
2047	1.00	22.00	1,151,446.89
851	1.00	23.00	3,092,115.41
1913	1.00	24.00	884,796.73
255	1.00	25.00	1,451,784.16
2979	1.00	26.00	967,823.61
518	1.00	27.00	196,574.90
21	1.00	28.00	84,815.30
861	1.00	29.00	665,098.75
968	1.00	30.00	488,387.19
1426	1.00	31.00	344,420.26
...	...	...	...
2751	45.00	54.00	5,784.62
2573	45.00	55.00	870,254.51
1715	45.00	56.00	519,397.43
3076	45.00	58.00	241,179.85
1842	45.00	59.00	85,912.45
3122	45.00	60.00	25,313.20
899	45.00	67.00	972,546.81
1414	45.00	71.00	628,686.82
2809	45.00	72.00	5,357,682.10
208	45.00	74.00	1,531,547.32
2151	45.00	77.00	1,525.96
3184	45.00	78.00	88.00
2852	45.00	79.00	2,180,038.31
541	45.00	80.00	72,622.47
1024	45.00	81.00	2,095,392.73

```
3220                45.00                82.00         1,920,560.27
2904                45.00                83.00           117,915.34
803                 45.00                85.00           286,197.78
430                 45.00                87.00           901,911.85
3047                45.00                90.00         3,385,387.04
629                 45.00                91.00         2,379,795.61
925                 45.00                92.00         6,882,003.38
2607                45.00                93.00           390,193.68
1032                45.00                94.00           494,496.46
1448                45.00                95.00         7,564,151.83
749                 45.00                96.00                5.94
56                  45.00                97.00           924,775.55
2461                45.00                98.00            75,767.27
3344                45.00                 nan       112,395,341.42
0                     nan                 nan     6,737,218,987.11

[3377 rows x 3 columns]
```

[15]:
```
rollup_by_store_dept = pd.DataFrame(rows, columns=['Store','Dept','Total'])

#For the pretty print : Replace NaN by -1

rollup_by_store_dept['Store'] = rollup_by_store_dept['Store'].replace(np.nan,-1)
rollup_by_store_dept['Dept'] = rollup_by_store_dept['Dept'].replace(np.nan,-1)

#type conversion of values returned by ROLLUP

rollup_by_store_dept['Store'] = rollup_by_store_dept['Store'].astype(int)
rollup_by_store_dept['Dept'] = rollup_by_store_dept['Dept'].astype(int)
rollup_by_store_dept['Total'] = rollup_by_store_dept['Total'].astype(np.float64)


#For the pretty print: Replace -1  by BLANK for Store and department
rollup_by_store_dept['Store'].replace(-1,'',inplace=True)
rollup_by_store_dept['Dept'].replace(-1,'',inplace=True)


rollup_by_store_dept = rollup_by_store_dept.sort_values(by=['Store', 'Dept'])


rollup_by_store_dept
```

[15]:
```
      Store Dept                Total
13        1    1         3,219,405.18
2017      1    2         6,592,598.93
674       1    3         1,880,518.36
305       1    4         5,285,874.09
37        1    5         3,468,885.58
```

941	1	6	686,654.56
1412	1	7	3,513,007.70
2086	1	8	5,107,710.84
2108	1	9	4,012,873.47
329	1	10	4,437,774.25
26	1	11	3,563,455.70
2216	1	12	1,511,015.98
2976	1	13	5,533,081.91
1023	1	14	2,183,402.78
2026	1	16	3,453,601.77
1598	1	17	1,315,107.78
1441	1	18	877,479.40
1795	1	19	180,039.65
1438	1	20	585,094.73
257	1	21	1,116,608.43
2047	1	22	1,151,446.89
851	1	23	3,092,115.41
1913	1	24	884,796.73
255	1	25	1,451,784.16
2979	1	26	967,823.61
518	1	27	196,574.90
21	1	28	84,815.30
861	1	29	665,098.75
968	1	30	488,387.19
1426	1	31	344,420.26
...	...	...	...
2751	45	54	5,784.62
2573	45	55	870,254.51
1715	45	56	519,397.43
3076	45	58	241,179.85
1842	45	59	85,912.45
3122	45	60	25,313.20
899	45	67	972,546.81
1414	45	71	628,686.82
2809	45	72	5,357,682.10
208	45	74	1,531,547.32
2151	45	77	1,525.96
3184	45	78	88.00
2852	45	79	2,180,038.31
541	45	80	72,622.47
1024	45	81	2,095,392.73
3220	45	82	1,920,560.27
2904	45	83	117,915.34
803	45	85	286,197.78
430	45	87	901,911.85
3047	45	90	3,385,387.04
629	45	91	2,379,795.61

```
925     45   92          6,882,003.38
2607    45   93            390,193.68
1032    45   94            494,496.46
1448    45   95          7,564,151.83
749     45   96                  5.94
56      45   97            924,775.55
2461    45   98             75,767.27
3344    45              112,395,341.42
0                     6,737,218,987.11

[3377 rows x 3 columns]
```

7.8 Query #7:

- Get the total weekly sales using **CUBE** clause for the hierarchical data: store, department and week

[16]:
```
cursor.execute("SELECT Store, Dept, Date, sum(Weekly_Sales) from weekly_sales␣
 ↪GROUP BY CUBE(Store, Dept, Date)")
rows=cursor.fetchall()
```

[17]:
```
pd.DataFrame(rows, columns=['Store','Dept','Date','Total']).
 ↪sort_values(by=['Store', 'Dept', 'Date'])
```

[17]:
```
                      Store                Dept       Date  \
0                      1.00                1.00 2010-02-05
1                      1.00                1.00 2010-02-12
2                      1.00                1.00 2010-02-19
3                      1.00                1.00 2010-02-26
4                      1.00                1.00 2010-03-05
5                      1.00                1.00 2010-03-12
6                      1.00                1.00 2010-03-19
7                      1.00                1.00 2010-03-26
8                      1.00                1.00 2010-04-02
9                      1.00                1.00 2010-04-09
10                     1.00                1.00 2010-04-16
11                     1.00                1.00 2010-04-23
12                     1.00                1.00 2010-04-30
13                     1.00                1.00 2010-05-07
14                     1.00                1.00 2010-05-14
15                     1.00                1.00 2010-05-21
16                     1.00                1.00 2010-05-28
17                     1.00                1.00 2010-06-04
18                     1.00                1.00 2010-06-11
19                     1.00                1.00 2010-06-18
20                     1.00                1.00 2010-06-25
21                     1.00                1.00 2010-07-02
22                     1.00                1.00 2010-07-09
```

23	1.00	1.00	2010-07-16
24	1.00	1.00	2010-07-23
25	1.00	1.00	2010-07-30
26	1.00	1.00	2010-08-06
27	1.00	1.00	2010-08-13
28	1.00	1.00	2010-08-20
29	1.00	1.00	2010-08-27
...	...	...	...
424959	nan	nan	2012-04-13
424997	nan	nan	2012-04-20
424981	nan	nan	2012-04-27
424952	nan	nan	2012-05-04
425008	nan	nan	2012-05-11
425078	nan	nan	2012-05-18
425010	nan	nan	2012-05-25
424990	nan	nan	2012-06-01
425066	nan	nan	2012-06-08
425025	nan	nan	2012-06-15
425083	nan	nan	2012-06-22
424973	nan	nan	2012-06-29
425005	nan	nan	2012-07-06
425079	nan	nan	2012-07-13
424967	nan	nan	2012-07-20
424965	nan	nan	2012-07-27
425071	nan	nan	2012-08-03
424995	nan	nan	2012-08-10
424977	nan	nan	2012-08-17
425052	nan	nan	2012-08-24
424999	nan	nan	2012-08-31
424985	nan	nan	2012-09-07
424963	nan	nan	2012-09-14
425050	nan	nan	2012-09-21
425040	nan	nan	2012-09-28
424964	nan	nan	2012-10-05
425074	nan	nan	2012-10-12
425043	nan	nan	2012-10-19
425063	nan	nan	2012-10-26
424946	nan	nan	NaT

	Total
0	24,924.50
1	46,039.49
2	41,595.55
3	19,403.54
4	21,827.90
5	21,043.39
6	22,136.64

7	26,229.21
8	57,258.43
9	42,960.91
10	17,596.96
11	16,145.35
12	16,555.11
13	17,413.94
14	18,926.74
15	14,773.04
16	15,580.43
17	17,558.09
18	16,637.62
19	16,216.27
20	16,328.72
21	16,333.14
22	17,688.76
23	17,150.84
24	15,360.45
25	15,381.82
26	17,508.41
27	15,536.40
28	15,740.13
29	15,793.87
...	...
424959	46,629,261.41
424997	45,072,529.78
424981	43,716,798.89
424952	47,124,197.93
425008	46,925,878.99
425078	46,823,939.22
425010	47,892,463.31
424990	48,281,649.72
425066	49,651,171.78
425025	48,412,110.70
425083	47,668,284.97
424973	46,597,112.12
425005	51,253,021.88
425079	46,099,732.10
424967	46,059,543.45
424965	44,097,154.97
425071	47,485,899.56
424995	47,403,451.04
424977	47,354,452.05
425052	47,447,323.60
424999	47,159,639.43
424985	48,330,059.31
424963	44,226,038.65

```
425050        44,354,547.11
425040        43,734,899.40
424964        47,566,639.31
425074        46,128,514.25
425043        45,122,410.57
425063        45,544,116.29
424946     6,737,218,987.11

[442696 rows x 4 columns]
```

[17]:
```
cube_by_store_dept_date = pd.DataFrame(rows,␣
 ↪columns=['Store','Dept','Date','Total'])

#For the pretty print : Replace NaN by -1

cube_by_store_dept_date['Store'] = cube_by_store_dept_date['Store'].replace(np.
 ↪nan,-1)
cube_by_store_dept_date['Dept'] = cube_by_store_dept_date['Dept'].replace(np.
 ↪nan,-1)

#type conversion of values returned by ROLLUP

cube_by_store_dept_date['Store'] = cube_by_store_dept_date['Store'].astype(int)
cube_by_store_dept_date['Dept'] = cube_by_store_dept_date['Dept'].astype(int)
cube_by_store_dept_date['Total'] = cube_by_store_dept_date['Total'].astype(np.
 ↪float64)


#For the pretty print: Replace -1  by BLANK for Store and department
cube_by_store_dept_date['Store'].replace(-1,'',inplace=True)
cube_by_store_dept_date['Dept'].replace(-1,'',inplace=True)

#For the pretty print: Replace NaN  by BLANK for date
cube_by_store_dept_date['Date'] = cube_by_store_dept_date['Date'].dt.date
cube_by_store_dept_date['Date'].replace(np.nan,'',inplace=True)


cube_by_store_dept_date = cube_by_store_dept_date.sort_values(by=['Store',␣
 ↪'Dept', 'Date'])


cube_by_store_dept_date
```

[17]:

	Store	Dept	Date	Total
0	1	1	2010-02-05	24,924.50
1	1	1	2010-02-12	46,039.49
2	1	1	2010-02-19	41,595.55
3	1	1	2010-02-26	19,403.54

4	1	1	2010-03-05	21,827.90
5	1	1	2010-03-12	21,043.39
6	1	1	2010-03-19	22,136.64
7	1	1	2010-03-26	26,229.21
8	1	1	2010-04-02	57,258.43
9	1	1	2010-04-09	42,960.91
10	1	1	2010-04-16	17,596.96
11	1	1	2010-04-23	16,145.35
12	1	1	2010-04-30	16,555.11
13	1	1	2010-05-07	17,413.94
14	1	1	2010-05-14	18,926.74
15	1	1	2010-05-21	14,773.04
16	1	1	2010-05-28	15,580.43
17	1	1	2010-06-04	17,558.09
18	1	1	2010-06-11	16,637.62
19	1	1	2010-06-18	16,216.27
20	1	1	2010-06-25	16,328.72
21	1	1	2010-07-02	16,333.14
22	1	1	2010-07-09	17,688.76
23	1	1	2010-07-16	17,150.84
24	1	1	2010-07-23	15,360.45
25	1	1	2010-07-30	15,381.82
26	1	1	2010-08-06	17,508.41
27	1	1	2010-08-13	15,536.40
28	1	1	2010-08-20	15,740.13
29	1	1	2010-08-27	15,793.87
...	...	...	...	...
424959			2012-04-13	46,629,261.41
424997			2012-04-20	45,072,529.78
424981			2012-04-27	43,716,798.89
424952			2012-05-04	47,124,197.93
425008			2012-05-11	46,925,878.99
425078			2012-05-18	46,823,939.22
425010			2012-05-25	47,892,463.31
424990			2012-06-01	48,281,649.72
425066			2012-06-08	49,651,171.78
425025			2012-06-15	48,412,110.70
425083			2012-06-22	47,668,284.97
424973			2012-06-29	46,597,112.12
425005			2012-07-06	51,253,021.88
425079			2012-07-13	46,099,732.10
424967			2012-07-20	46,059,543.45
424965			2012-07-27	44,097,154.97
425071			2012-08-03	47,485,899.56
424995			2012-08-10	47,403,451.04
424977			2012-08-17	47,354,452.05
425052			2012-08-24	47,447,323.60

```
424999             2012-08-31       47,159,639.43
424985             2012-09-07       48,330,059.31
424963             2012-09-14       44,226,038.65
425050             2012-09-21       44,354,547.11
425040             2012-09-28       43,734,899.40
424964             2012-10-05       47,566,639.31
425074             2012-10-12       46,128,514.25
425043             2012-10-19       45,122,410.57
425063             2012-10-26       45,544,116.29
424946                              6,737,218,987.11

[442696 rows x 4 columns]
```

7.9 Query #8:

- Get the **descriptive statisitics** per store

[18]:
```
cursor.execute("SELECT * from weekly_sales")
rows=cursor.fetchall()
```

[19]:
```
weekly_sales = pd.DataFrame(rows, columns=['Store','Dept',␣
↪'Date','Weekly_Sales','IsHoliday'])

#For the pretty print: format date with no time-field
weekly_sales['Date'] = weekly_sales['Date'].dt.date
```

DataFrame has few good methods for descriptive statistics, grouping, and heirarchical clusterning:

- Count only non-null values, use **count**
- Count total values including null values, use **size** attribute
- Count distinct values, use **nunique**
- **stack()** on the **groupby** result will produce the pretty print

[20]:
```
weekly_sales.groupby('Store').agg(['count', 'size', 'nunique', 'max']).stack()
```

[20]:
```
                Dept        Date         Weekly_Sales IsHoliday
Store
1      count    10244       10244           10,244.00     10244
       size     10244       10244           10,244.00     10244
       nunique     77         143           10,042.00         2
       max         99  2012-10-26          203,670.47      True
2      count    10238       10238           10,238.00     10238
       size     10238       10238           10,238.00     10238
       nunique     78         143           10,088.00         2
       max         99  2012-10-26          285,353.53      True
3      count     9036        9036            9,036.00      9036
       size      9036        9036            9,036.00      9036
```

	nunique	72	143	8,688.00	2
	max	98	2012-10-26	155,897.94	True
4	count	10272	10272	10,272.00	10272
	size	10272	10272	10,272.00	10272
	nunique	78	143	10,098.00	2
	max	99	2012-10-26	385,051.04	True
5	count	8999	8999	8,999.00	8999
	size	8999	8999	8,999.00	8999
	nunique	72	143	8,594.00	2
	max	98	2012-10-26	93,517.72	True
6	count	10211	10211	10,211.00	10211
	size	10211	10211	10,211.00	10211
	nunique	77	143	10,076.00	2
	max	99	2012-10-26	342,578.65	True
7	count	9762	9762	9,762.00	9762
	size	9762	9762	9,762.00	9762
	nunique	76	143	9,391.00	2
	max	99	2012-10-26	222,921.09	True
8	count	9895	9895	9,895.00	9895
	size	9895	9895	9,895.00	9895
...		...	...	...	...
38	nunique	63	143	6,715.00	2
	max	99	2012-10-26	100,618.04	True
39	count	9878	9878	9,878.00	9878
	size	9878	9878	9,878.00	9878
	nunique	75	143	9,713.00	2
	max	99	2012-10-26	351,553.98	True
40	count	10017	10017	10,017.00	10017
	size	10017	10017	10,017.00	10017
	nunique	77	143	9,724.00	2
	max	99	2012-10-26	145,504.24	True
41	count	10088	10088	10,088.00	10088
	size	10088	10088	10,088.00	10088
	nunique	77	143	9,864.00	2
	max	99	2012-10-26	290,809.17	True
42	count	6953	6953	6,953.00	6953
	size	6953	6953	6,953.00	6953
	nunique	62	143	6,452.00	2
	max	98	2012-10-26	112,152.35	True
43	count	6751	6751	6,751.00	6751
	size	6751	6751	6,751.00	6751
	nunique	61	143	6,292.00	2
	max	99	2012-10-26	108,517.42	True
44	count	7169	7169	7,169.00	7169
	size	7169	7169	7,169.00	7169
	nunique	62	143	6,548.00	2
	max	99	2012-10-26	66,629.98	True

```
45    count      9637         9637              9,637.00      9637
      size       9637         9637              9,637.00      9637
      nunique      74          143              9,381.00         2
      max          98   2012-10-26           240,758.86      True

[180 rows x 4 columns]
```

7.10 Query #9:

- Get the top 500 weekly sales and plot them (Red color for Holiday and Blue color for Not Holiday) against the weekly Unemployment and Temperature

[21]:
```
cursor.execute("SELECT * from weekly_sales")
rows=cursor.fetchall()
```

[22]:
```
weekly_sales = pd.DataFrame(rows, columns=['Store','Dept',␣
 ↪'Date','Weekly_Sales','IsHoliday'])

weekly_sales = weekly_sales.sort_values(by=['Weekly_Sales'], ascending=False)
```

[23]:
```
top_ten_weekly_sales = weekly_sales[:500]

top_ten_weekly_sales.head()
```

[23]:
```
        Store  Dept       Date         Weekly_Sales  IsHoliday
95373      10    72 2010-11-26           693,099.36       True
338013     35    72 2011-11-25           649,770.18       True
95425      10    72 2011-11-25           630,999.19       True
337961     35    72 2010-11-26           627,962.93       True
135665     14    72 2010-11-26           474,330.10       True
```

[24]:
```
cursor.execute("SELECT * from features")
rows=cursor.fetchall()
```

[25]:
```
weekly_sales_temp_unemp = pd.DataFrame(rows, columns=['Store', 'Date',␣
 ↪'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3',␣
 ↪'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment', 'IsHoliday'])
```

[26]:
```
# Sanity test that we got good data from db server

weekly_sales_temp_unemp.head()
```

[26]:
```
   Store       Date          Temperature           Fuel_Price  \
0      1 2010-02-05                42.31                 2.57
1      1 2010-02-12                38.51                 2.55
2      1 2010-02-19                39.93                 2.51
3      1 2010-02-26                46.63                 2.56
4      1 2010-03-05                46.50                 2.62

             MarkDown1           MarkDown2           MarkDown3  \
```

```
0                0.00                0.00                0.00
1                0.00                0.00                0.00
2                0.00                0.00                0.00
3                0.00                0.00                0.00
4                0.00                0.00                0.00

             MarkDown4           MarkDown5                 CPI  \
0                 0.00                0.00              211.10
1                 0.00                0.00              211.24
2                 0.00                0.00              211.29
3                 0.00                0.00              211.32
4                 0.00                0.00              211.35

          Unemployment  IsHoliday
0                 8.11      False
1                 8.11       True
2                 8.11      False
3                 8.11      False
4                 8.11      False
```

[27]:
```
weekly_sales_temp_unemp.groupby('Store').agg(['count', 'size', 'nunique',␣
↪'max']).stack()
```

[27]:
```
                                Date          Temperature          Fuel_Price  \
Store
1     count                      182               182.00              182.00
      size                       182               182.00              182.00
      nunique                    182               181.00              169.00
      max      2013-07-26 00:00:00                91.65                3.91
2     count                      182               182.00              182.00
      size                       182               182.00              182.00
      nunique                    182               179.00              169.00
      max      2013-07-26 00:00:00                93.34                3.91
3     count                      182               182.00              182.00
      size                       182               182.00              182.00
      nunique                    182               180.00              169.00
      max      2013-07-26 00:00:00                89.12                3.91
4     count                      182               182.00              182.00
      size                       182               182.00              182.00
      nunique                    182               177.00              168.00
      max      2013-07-26 00:00:00                86.29                3.88
5     count                      182               182.00              182.00
      size                       182               182.00              182.00
      nunique                    182               182.00              169.00
      max      2013-07-26 00:00:00                91.07                3.91
6     count                      182               182.00              182.00
      size                       182               182.00              182.00
      nunique                    182               182.00              169.00
```

```
      max      2013-07-26 00:00:00              91.46              3.91
7     count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            177.00            165.00
      max      2013-07-26 00:00:00              68.84              3.94
8     count                   182            182.00            182.00
      size                    182            182.00            182.00
...                           ...               ...               ...
38    nunique                 182            180.00            166.00
      max      2013-07-26 00:00:00             101.95              4.47
39    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            180.00            169.00
      max      2013-07-26 00:00:00              88.65              3.91
40    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            178.00            165.00
      max      2013-07-26 00:00:00              76.67              4.10
41    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            175.00            165.00
      max      2013-07-26 00:00:00              76.54              3.94
42    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            176.00            174.00
      max      2013-07-26 00:00:00              95.36              4.47
43    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            179.00            169.00
      max      2013-07-26 00:00:00              91.36              3.91
44    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            177.00            165.00
      max      2013-07-26 00:00:00              85.58              3.85
45    count                   182            182.00            182.00
      size                    182            182.00            182.00
      nunique                 182            179.00            161.00
      max      2013-07-26 00:00:00              82.99              4.07

                         MarkDown1          MarkDown2          MarkDown3  \
Store
1     count                182.00             182.00             182.00
      size                 182.00             182.00             182.00
      nunique               91.00              73.00              89.00
      max                72,937.29          46,011.38          74,910.32
2     count                182.00             182.00             182.00
      size                 182.00             182.00             182.00
```

	nunique	91.00	76.00	87.00
	max	75,149.79	92,523.94	105,146.30
3	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	70.00	77.00
	max	29,091.04	14,356.07	54,466.91
4	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	75.00	89.00
	max	56,705.09	72,413.71	93,310.30
5	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	66.00	74.00
	max	23,811.41	17,079.76	43,319.43
6	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	76.00	88.00
	max	64,733.80	82,881.16	112,255.67
7	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	66.00	87.00
	max	56,917.70	17,021.30	44,081.25
8	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
...		...	...	...
38	nunique	91.00	31.00	70.00
	max	5,631.57	2,765.45	265.53
39	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	75.00	87.00
	max	66,099.34	53,918.62	109,976.14
40	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	71.00	85.00
	max	39,257.29	31,425.65	60,367.16
41	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	76.00	90.00
	max	65,739.82	56,106.20	97,533.40
42	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	86.00	32.00	66.00
	max	8,063.80	2,655.21	220.60
43	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	90.00	30.00	72.00
	max	1,952.91	3,350.20	163.92

44	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	89.00	28.00	65.00	
	max	3,297.48	1,821.61	106.31	
45	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	91.00	71.00	89.00	
	max	53,311.88	43,941.56	72,542.01	

		MarkDown4	MarkDown5	CPI	\
Store					
1	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	91.00	91.00	168.00	
	max	32,403.87	20,475.32	225.17	
2	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	91.00	91.00	168.00	
	max	48,159.86	36,430.33	224.80	
3	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	88.00	91.00	168.00	
	max	8,368.15	7,297.10	228.73	
4	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	89.00	91.00	168.00	
	max	48,086.64	28,604.20	132.72	
5	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	85.00	91.00	168.00	
	max	14,928.42	24,751.93	225.77	
6	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	91.00	91.00	168.00	
	max	34,049.73	27,272.53	226.80	
7	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	91.00	91.00	168.00	
	max	16,519.53	57,029.78	201.24	
8	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
...		...	...	...	
38	nunique	29.00	91.00	168.00	
	max	334.12	3,186.15	132.72	
39	count	182.00	182.00	182.00	
	size	182.00	182.00	182.00	
	nunique	91.00	91.00	168.00	

	max	32,731.31	108,519.28	223.84
40	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	89.00	91.00	168.00
	max	25,676.37	15,828.62	139.12
41	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	91.00	168.00
	max	63,830.91	21,739.26	201.24
42	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	17.00	91.00	168.00
	max	313.04	4,143.90	132.72
43	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	16.00	91.00	168.00
	max	615.93	7,035.07	216.44
44	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	21.00	91.00	168.00
	max	89.76	2,583.41	132.72
45	count	182.00	182.00	182.00
	size	182.00	182.00	182.00
	nunique	91.00	91.00	168.00
	max	38,157.91	17,861.50	193.59

		Unemployment	IsHoliday
Store			
1	count	182.00	182
	size	182.00	182
	nunique	15.00	2
	max	8.11	True
2	count	182.00	182
	size	182.00	182
	nunique	15.00	2
	max	8.32	True
3	count	182.00	182
	size	182.00	182
	nunique	15.00	2
	max	7.57	True
4	count	182.00	182
	size	182.00	182
	nunique	15.00	2
	max	8.62	True
5	count	182.00	182
	size	182.00	182
	nunique	15.00	2

```
      max                   6.77      True
6     count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   7.26      True
7     count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   9.14      True
8     count               182.00       182
      size                182.00       182
...                          ...       ...
38    nunique              15.00         2
      max                  14.31      True
39    count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   8.55      True
40    count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   5.89      True
41    count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   7.54      True
42    count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   9.77      True
43    count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                  10.64      True
44    count               182.00       182
      size                182.00       182
      nunique              15.00         2
      max                   8.12      True
45    count               182.00       182
      size                182.00       182
      nunique              14.00         2
      max                   8.99      True

[180 rows x 11 columns]
```

[28]:
```
unemployment = []
temperature = []
```

```
for row in top_ten_weekly_sales.itertuples():
    temperature.append(weekly_sales_temp_unemp[(weekly_sales_temp_unemp.Store␣
 ↪== row.Store) & (weekly_sales_temp_unemp.Date == row.Date)]['Temperature'].
 ↪values[0])
    unemployment.append(weekly_sales_temp_unemp[(weekly_sales_temp_unemp.Store␣
 ↪== row.Store) & (weekly_sales_temp_unemp.Date == row.Date)]['Unemployment'].
 ↪values[0])
```

[29]:
```
se_unemployment = pd.Series(unemployment)
se_temperature = pd.Series(temperature)
```

[30]:
```
top_ten_weekly_sales.insert(loc=5, column='Unemployment', value=se_unemployment.
 ↪values)
top_ten_weekly_sales.insert(loc=6, column='Temperature', value=se_temperature.
 ↪values)
```

[31]:
```
top_ten_weekly_sales.head()
```

[31]:
```
        Store  Dept       Date          Weekly_Sales  IsHoliday  \
95373      10    72 2010-11-26          693,099.36       True
338013     35    72 2011-11-25          649,770.18       True
95425      10    72 2011-11-25          630,999.19       True
337961     35    72 2010-11-26          627,962.93       True
135665     14    72 2010-11-26          474,330.10       True

               Unemployment          Temperature
95373                  9.00                55.33
338013                 8.74                47.88
95425                  7.87                60.68
337961                 8.76                46.67
135665                 8.72                46.15
```

[32]:
```
top_ten_weekly_sales.describe()
```

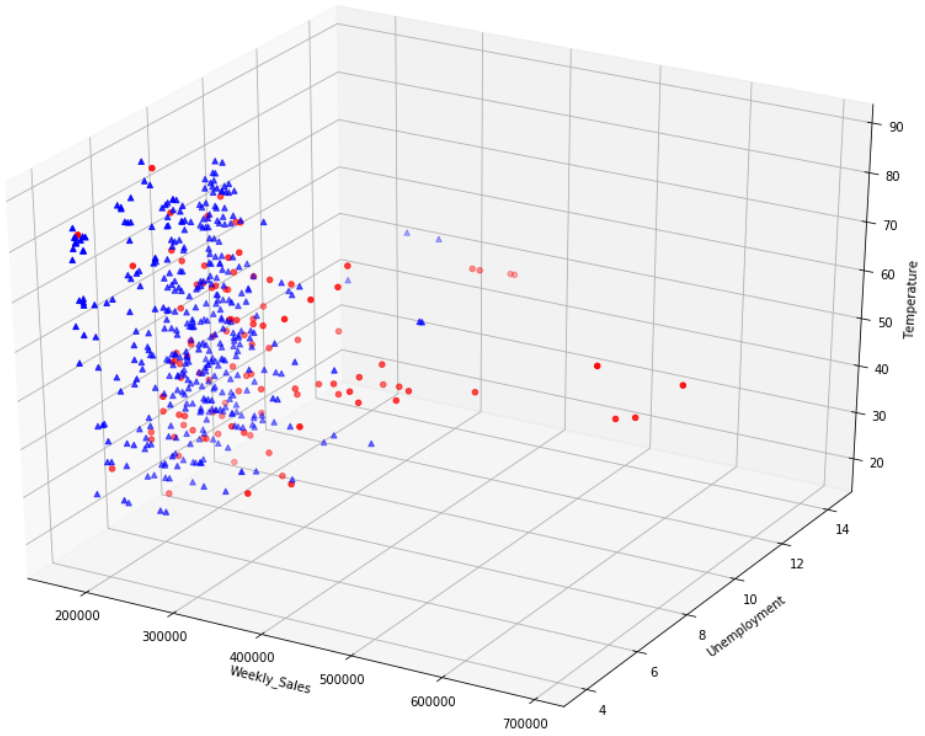
[32]:
```
                    Store                 Dept         Weekly_Sales  \
count              500.00               500.00               500.00
mean               14.10                77.46           208,667.25
std                 8.61                27.57            62,695.45
min                 1.00                 1.00           170,170.92
25%                10.00                72.00           177,431.23
50%                14.00                92.00           186,580.04
75%                20.00                92.00           207,864.04
max                45.00                95.00           693,099.36

              Unemployment          Temperature
count               500.00               500.00
mean                 7.59                51.54
std                  1.52                16.37
min                  3.88                17.95
```

```
25%                6.96               39.37
50%                7.82               49.91
75%                8.57               64.24
max               14.31               88.55
```

[33]:
```
feature0_holiday = top_ten_weekly_sales[top_ten_weekly_sales.IsHoliday ==␣
 ↪True]['Weekly_Sales'].values
feature0_not_holiday = top_ten_weekly_sales[top_ten_weekly_sales.IsHoliday ==␣
 ↪False]['Weekly_Sales'].values

feature1_holiday = top_ten_weekly_sales[top_ten_weekly_sales.IsHoliday ==␣
 ↪True]['Unemployment'].values
feature1_not_holiday = top_ten_weekly_sales[top_ten_weekly_sales.IsHoliday ==␣
 ↪False]['Unemployment'].values

feature2_holiday = top_ten_weekly_sales[top_ten_weekly_sales.IsHoliday ==␣
 ↪True]['Temperature'].values
feature2_not_holiday = top_ten_weekly_sales[top_ten_weekly_sales.IsHoliday ==␣
 ↪False]['Temperature'].values
```

[34]:
```
# Plot the raw data

fig = plt.figure(figsize=(16, 12))
ax = fig.add_subplot(111, projection='3d')



x_ax_holiday = np.array(feature0_holiday)
y_ax_holiday = np.array(feature1_holiday)
z_ax_holiday = np.array(feature2_holiday)

x_ax_not_holiday = np.array(feature0_not_holiday)
y_ax_not_holiday = np.array(feature1_not_holiday)
z_ax_not_holiday = np.array(feature2_not_holiday)


ax.scatter(x_ax_holiday, y_ax_holiday, z_ax_holiday, marker='o', c = 'red')
ax.scatter(x_ax_not_holiday, y_ax_not_holiday, z_ax_not_holiday, marker='^', c␣
 ↪= 'blue')



ax.set_xlabel('Weekly_Sales')
ax.set_ylabel('Unemployment')
ax.set_zlabel('Temperature')
```

```
plt.show()
```



8 Requirements

** The PDF document and you rIPYNB script that you are submitting on Canvas must have the source code and the output for the following requirements **

8.0.1 Requirement #1:

- Discuss the difference between **ROLLUP** and the **CUBE** that can be used in the **Group By** clause of the **SQL-SELECT** statement in PostgreSQL 10.5.

According to the chapter 33 of the textbook, ROLLUP supports calculations using aggregations such as SUM, COUNT, MAX, MIN, and AVG at increasing levels of aggregation, from the most detailed up to a grand total. CUBE is similar to ROLLUP, enabling a single statement to calculate all possible combinations of aggregations. CUBE can generate the information needed in cross-tabulation reports with a single query.

So, ROLLUP operators let you extend the functionality of GROUP BY clauses by calculating subtotals and grand totals for a set of columns. The CUBE operator is similar in functionality to

the ROLLUP operator; however, the CUBE operator can calculate subtotals and grand totals for all permutations of the columns specified in it.

8.0.2 Requirement #2:

- Get the descriptive statisitics per department

[4]:
```
cursor.execute("SELECT * FROM weekly_sales")
rows=cursor.fetchall()
colnames = [desc[0] for desc in cursor.description]
df = pd.DataFrame(rows, columns = colnames)

#For the pretty print: format date with no time-field
df['date'] = df['date'].dt.date
```

[7]:
```
df.groupby('dept').agg(['count', 'size', 'nunique', 'max']).stack()
```

[7]:

dept		store	date	weekly_sales	isholiday
1	count	6435	6435	6,435.00	6435
	size	6435	6435	6,435.00	6435
	nunique	45	143	6,427.00	2
	max	45	2012-10-26	172,225.55	True
2	count	6435	6435	6,435.00	6435
	size	6435	6435	6,435.00	6435
	nunique	45	143	6,433.00	2
	max	45	2012-10-26	151,090.50	True
3	count	6435	6435	6,435.00	6435
	size	6435	6435	6,435.00	6435
	nunique	45	143	6,419.00	2
	max	45	2012-10-26	131,564.25	True
4	count	6435	6435	6,435.00	6435
	size	6435	6435	6,435.00	6435
	nunique	45	143	6,432.00	2
	max	45	2012-10-26	72,179.92	True
5	count	6347	6347	6,347.00	6347
	size	6347	6347	6,347.00	6347
	nunique	45	143	6,207.00	2
	max	45	2012-10-26	259,955.82	True
6	count	5986	5986	5,986.00	5986
	size	5986	5986	5,986.00	5986
	nunique	45	143	5,756.00	2
	max	45	2012-10-26	54,700.40	True
7	count	6435	6435	6,435.00	6435
	size	6435	6435	6,435.00	6435
	nunique	45	143	6,426.00	2
	max	45	2012-10-26	406,988.63	True
8	count	6435	6435	6,435.00	6435
	size	6435	6435	6,435.00	6435

```
...              ...         ...                  ...       ...
92   nunique      45         143             6,430.00         2
     max          45  2012-10-26           293,966.05      True
93   count      5913        5913             5,913.00      5913
     size       5913        5913             5,913.00      5913
     nunique      43         143             5,897.00         2
     max          45  2012-10-26           103,827.72      True
94   count      5685        5685             5,685.00      5685
     size       5685        5685             5,685.00      5685
     nunique      45         143             5,395.00         2
     max          45  2012-10-26           103,929.39      True
95   count      6435        6435             6,435.00      6435
     size       6435        6435             6,435.00      6435
     nunique      45         143             6,434.00         2
     max          45  2012-10-26           213,042.66      True
96   count      4854        4854             4,854.00      4854
     size       4854        4854             4,854.00      4854
     nunique      45         143             4,834.00         2
     max          45  2012-10-26            63,978.78      True
97   count      6278        6278             6,278.00      6278
     size       6278        6278             6,278.00      6278
     nunique      45         143             6,270.00         2
     max          45  2012-10-26            49,034.16      True
98   count      5836        5836             5,836.00      5836
     size       5836        5836             5,836.00      5836
     nunique      45         143             5,612.00         2
     max          45  2012-10-26            33,759.90      True
99   count       862         862               862.00       862
     size        862         862               862.00       862
     nunique      37          94               311.00         2
     max          44  2012-10-26            12,550.00      True

[324 rows x 4 columns]
```

8.0.3 Requirement #3:

- Use the **CUBE** and dataframe to produce a listing of total sales per department across the entire list of stores

[13]:
```
cursor.execute("SELECT Dept, Store, sum(Weekly_Sales) FROM weekly_sales GROUP␣
 ↪BY CUBE(Dept, Store)")
rows=cursor.fetchall()
cube_by_dept = pd.DataFrame(rows, columns=['Dept','Store','Total'])
```

[14]:
```
cube_by_dept
```

[14]:
```
   Dept  Store                Total
0   nan    nan     6,737,218,987.11
```

1	25.00	10.00	3,023,251.13
2	92.00	31.00	18,162,446.96
3	93.00	34.00	4,231,385.92
4	18.00	28.00	957,897.82
5	18.00	36.00	24,258.04
6	82.00	11.00	3,285,972.04
7	78.00	45.00	88.00
8	32.00	5.00	350,478.36
9	85.00	13.00	687,343.65
10	83.00	26.00	412,269.59
11	29.00	12.00	997,204.55
12	1.00	1.00	3,219,405.18
13	83.00	4.00	1,310,054.08
14	38.00	34.00	6,981,158.73
15	80.00	32.00	2,395,249.11
16	1.00	41.00	3,318,352.17
17	92.00	18.00	7,161,386.18
18	85.00	17.00	601,917.35
19	94.00	30.00	3,506,735.03
20	19.00	12.00	421,729.32
21	1.00	28.00	2,885,804.92
22	85.00	32.00	325,013.26
23	60.00	20.00	83,894.30
24	17.00	8.00	1,071,704.78
25	4.00	13.00	6,086,548.39
26	1.00	11.00	2,697,110.41
27	92.00	35.00	4,829,972.58
28	51.00	18.00	295.44
29	81.00	10.00	1,516,807.88
...	...	...	...
3428	nan	6.00	223,756,130.64
3429	nan	26.00	143,416,393.79
3430	nan	12.00	144,287,230.15
3431	nan	39.00	207,445,542.47
3432	nan	24.00	194,016,021.28
3433	nan	19.00	206,634,862.10
3434	nan	36.00	53,412,214.97
3435	nan	25.00	101,061,179.17
3436	nan	31.00	199,613,905.50
3437	nan	30.00	62,716,885.12
3438	nan	21.00	108,117,878.92
3439	nan	14.00	288,999,911.34
3440	nan	3.00	57,586,735.07
3441	nan	17.00	127,782,138.83
3442	nan	37.00	74,202,740.32
3443	nan	28.00	189,263,680.58
3444	nan	22.00	147,075,648.57

```
3445                 nan              20.00       301,397,792.46
3446                 nan              33.00        37,160,221.96
3447                 nan              13.00       286,517,703.80
3448                 nan               1.00       222,402,808.85
3449                 nan               5.00        45,475,688.90
3450                 nan              18.00       155,114,734.21
3451                 nan               2.00       275,382,440.98
3452                 nan              16.00        74,252,425.40
3453                 nan              27.00       253,855,916.88
3454                 nan              23.00       198,750,617.85
3455                 nan              44.00        43,293,087.84
3456                 nan              11.00       193,962,786.80
3457                 nan               8.00       129,951,181.13

[3458 rows x 3 columns]
```

[15]:
```
#For the pretty print : Replace NaN by -1
cube_by_dept['Store'] = cube_by_dept['Store'].replace(np.nan,-1)
cube_by_dept['Dept'] = cube_by_dept['Dept'].replace(np.nan,-1)

#type conversion of values returned by ROLLUP
cube_by_dept['Store'] = cube_by_dept['Store'].astype(int)
cube_by_dept['Dept'] = cube_by_dept['Dept'].astype(int)
cube_by_dept['Total'] = cube_by_dept['Total'].astype(np.float64)

#For the pretty print: Replace -1  by BLANK for Store and department
cube_by_dept['Store'].replace(-1,'',inplace=True)
cube_by_dept['Dept'].replace(-1,'',inplace=True)

#For the pretty print: Replace NaN  by BLANK for date
#cube_by_dept['Date'] = cube_by_dept['Date'].dt.date
#cube_by_dept['Date'].replace(np.nan,'',inplace=True)

cube_by_dept= cube_by_dept.sort_values(by=['Dept', 'Store'])
cube_by_dept
```

[15]:

	Dept	Store	Total
12	1	1	3,219,405.18
2070	1	2	4,401,251.25
677	1	3	1,047,992.81
304	1	4	5,288,131.43
36	1	5	1,397,761.09
960	1	6	3,413,060.19
1461	1	7	1,364,620.58
2138	1	8	2,114,945.31
2158	1	9	1,694,057.83
324	1	10	5,709,294.87
26	1	11	2,697,110.41

2280	1	12	2,478,202.53
2987	1	13	6,723,925.13
1064	1	14	4,377,485.02
519	1	15	1,979,941.94
2081	1	16	1,623,404.55
1647	1	17	3,260,630.11
1492	1	18	3,144,334.94
1860	1	19	3,075,076.17
1490	1	20	5,798,002.67
264	1	21	2,137,857.04
2104	1	22	3,073,537.77
863	1	23	4,745,663.86
1966	1	24	2,696,840.34
263	1	25	2,880,863.34
2988	1	26	2,774,595.10
521	1	27	4,352,630.60
21	1	28	2,885,804.92
873	1	29	2,217,172.04
997	1	30	1,399,737.86
...	...	...	...
3441		17	127,782,138.83
3450		18	155,114,734.21
3433		19	206,634,862.10
3445		20	301,397,792.46
3438		21	108,117,878.92
3444		22	147,075,648.57
3454		23	198,750,617.85
3432		24	194,016,021.28
3435		25	101,061,179.17
3429		26	143,416,393.79
3453		27	253,855,916.88
3443		28	189,263,680.58
3414		29	77,141,554.31
3437		30	62,716,885.12
3436		31	199,613,905.50
3420		32	166,819,246.16
3446		33	37,160,221.96
3416		34	138,249,763.00
3424		35	131,520,672.08
3434		36	53,412,214.97
3442		37	74,202,740.32
3426		38	55,159,626.42
3431		39	207,445,542.47
3418		40	137,870,309.79
3417		41	181,341,934.89
3413		42	79,565,752.43
3419		43	90,565,435.41

```
3455          44         43,293,087.84
3425          45        112,395,341.42
0                     6,737,218,987.11

[3458 rows x 3 columns]
```

8.1 Requirement #4:

- Use the **CUBE** and dataframe to produce a listing of total sales per store across the entire list of departments for the week of 2010-02-19

[27]:
```
cursor.execute("SELECT Store, Dept, sum(Weekly_Sales) FROM weekly_sales WHERE␣
 ↪Date = '2010-02-19' GROUP BY CUBE(Store, Dept)")
rows=cursor.fetchall()
cube_by_store = pd.DataFrame(rows, columns=['Store', 'Dept', 'Total'])
```

[28]:
```
#For the pretty print : Replace NaN by -1
cube_by_store ['Store'] = cube_by_store ['Store'].replace(np.nan,-1)
cube_by_store ['Dept'] = cube_by_store ['Dept'].replace(np.nan,-1)

#type conversion of values returned by ROLLUP
cube_by_store ['Store'] = cube_by_store ['Store'].astype(int)
cube_by_store ['Dept'] = cube_by_store ['Dept'].astype(int)
cube_by_store ['Total'] = cube_by_store ['Total'].astype(np.float64)

#For the pretty print: Replace -1  by BLANK for Store and department
cube_by_store ['Store'].replace(-1,'',inplace=True)
cube_by_store ['Dept'].replace(-1,'',inplace=True)

#For the pretty print: Replace NaN  by BLANK for date
#cube_by_store ['Date'] = cube_by_store ['Date'].dt.date
#cube_by_store ['Date'].replace(np.nan,'',inplace=True)

cube_by_store = cube_by_store .sort_values(by=['Store', 'Dept'])
cube_by_store
```

[28]:

	Store	Dept	Total
11	1	1	41,595.55
1799	1	2	47,928.89
594	1	3	11,523.47
264	1	4	36,826.95
30	1	5	26,468.27
837	1	6	6,060.26
1258	1	7	19,985.20
1862	1	8	38,717.60
1882	1	9	15,880.85
284	1	10	29,634.13
21	1	11	18,706.21

1976	1	12	9,165.98
2658	1	13	37,857.68
906	1	14	17,491.36
1808	1	16	13,855.54
1423	1	17	13,485.61
1284	1	18	17,623.72
1596	1	19	1,722.17
1281	1	20	4,719.89
221	1	21	8,949.67
1827	1	22	12,290.16
757	1	23	21,381.85
1701	1	24	7,366.70
220	1	25	10,271.25
2660	1	26	10,547.60
454	1	27	2,494.50
17	1	28	1,012.09
763	1	29	7,966.55
860	1	30	5,735.00
1270	1	31	2,652.00
...	...	...	...
3025		54	15,599.47
3095		55	666,698.06
3098		56	77,826.84
3100		58	119,888.82
3051		59	37,329.66
3047		60	12,537.40
3053		65	42,908.44
3035		67	531,250.85
3027		71	323,734.01
3074		72	3,034,799.98
3024		74	661,960.78
3076		78	41.00
3069		79	1,089,782.13
3032		80	476,861.18
3078		81	657,129.60
3061		82	599,874.14
3034		83	203,166.20
3073		85	88,368.19
3023		87	518,886.95
3036		90	1,976,215.64
3099		91	1,546,117.67
3042		92	3,312,991.71
3043		93	1,005,051.42
3081		94	1,513,229.84
3075		95	2,814,037.79
3031		96	395,923.92
3048		97	569,748.82

```
3056          98           304,997.85
3059          99                8.99
0                        48,276,993.78

[3102 rows x 3 columns]
```

8.2 Requirement #5:

- Use the **CUBE** and dataframe to produce a listing of total sales per department across the entire list of stores for the week of 2010-02-19

[29]:
```
cursor.execute("SELECT Dept, Store, sum(Weekly_Sales) FROM weekly_sales  WHERE␣
 ↪Date = '2010-02-19' GROUP BY CUBE(Dept, Store)")
rows=cursor.fetchall()
cube_by_dept = pd.DataFrame(rows, columns=['Dept','Store','Total'])
```

[30]:
```
#For the pretty print : Replace NaN by -1
cube_by_dept['Store'] = cube_by_dept['Store'].replace(np.nan,-1)
cube_by_dept['Dept'] = cube_by_dept['Dept'].replace(np.nan,-1)

#type conversion of values returned by ROLLUP
cube_by_dept['Store'] = cube_by_dept['Store'].astype(int)
cube_by_dept['Dept'] = cube_by_dept['Dept'].astype(int)
cube_by_dept['Total'] = cube_by_dept['Total'].astype(np.float64)

#For the pretty print: Replace -1  by BLANK for Store and department
cube_by_dept['Store'].replace(-1,'',inplace=True)
cube_by_dept['Dept'].replace(-1,'',inplace=True)

#For the pretty print: Replace NaN  by BLANK for date
#cube_by_dept['Date'] = cube_by_dept['Date'].dt.date
#cube_by_dept['Date'].replace(np.nan,'',inplace=True)

cube_by_dept= cube_by_dept.sort_values(by=['Dept', 'Store'])
cube_by_dept
```

[30]:
```
      Dept Store                Total
11       1     1            41,595.55
1839     1     2            58,221.52
607      1     3             8,918.31
268      1     4            49,937.09
35       1     5            11,417.67
865      1     6            34,750.82
1307     1     7            12,477.79
1902     1     8            22,319.25
1921     1     9            14,819.97
287      1    10            49,748.33
25       1    11            25,294.18
```

2040	1	12	22,135.29
2660	1	13	44,042.19
960	1	14	35,163.18
463	1	15	17,391.31
1850	1	16	13,714.98
1470	1	17	23,375.58
1335	1	18	26,068.39
1656	1	19	32,365.74
1333	1	20	55,649.79
234	1	21	26,567.03
1873	1	22	23,035.70
776	1	23	28,504.70
1744	1	24	24,879.99
233	1	25	21,305.96
2661	1	26	27,534.54
465	1	27	42,607.96
20	1	28	30,763.19
786	1	29	18,411.25
900	1	30	12,321.18
...	...	...	...
3085		17	800,714.00
3094		18	1,150,663.42
3077		19	1,515,976.11
3089		20	2,161,549.76
3082		21	867,283.25
3088		22	988,467.61
3098		23	1,319,588.04
3076		24	1,385,362.49
3079		25	676,260.67
3073		26	999,348.55
3097		27	1,945,070.33
3087		28	1,491,300.42
3058		29	542,399.07
3081		30	463,513.26
3080		31	1,473,386.75
3064		32	1,082,559.06
3090		33	296,850.83
3060		34	983,963.07
3068		35	1,270,658.64
3078		36	470,281.03
3086		37	510,382.50
3070		38	327,237.92
3075		39	1,230,591.97
3062		40	916,289.20
3061		41	1,052,034.74
3057		42	508,794.87
3063		43	658,997.55

```
3099          44            267,956.30
3069          45            841,264.04
0                         48,276,993.78

[3102 rows x 3 columns]
```

8.3 Requirement #6:

- Get the top 500 weekly sales and plot them (Red color for Unemployment greater than or equal to 5 and Green color if it is less than 5) against the weekly Unemployment and Temperature

[31]:
```
cursor.execute("SELECT * from weekly_sales")
rows=cursor.fetchall()

weekly_sales = pd.DataFrame(rows, columns=['Store','Dept',␣
 ↪'Date','Weekly_Sales','IsHoliday'])
weekly_sales = weekly_sales.sort_values(by=['Weekly_Sales'], ascending=False)
```

[32]:
```
top_ten_weekly_sales = weekly_sales[:500]

# Sanity test that we got good data from db server
top_ten_weekly_sales.head()
```

[32]:
```
        Store  Dept       Date         Weekly_Sales  IsHoliday
95373      10    72 2010-11-26           693,099.36       True
338013     35    72 2011-11-25           649,770.18       True
95425      10    72 2011-11-25           630,999.19       True
337961     35    72 2010-11-26           627,962.93       True
135665     14    72 2010-11-26           474,330.10       True
```

[33]:
```
cursor.execute("SELECT * from features")
rows = cursor.fetchall()

weekly_sales_temp_unemp = pd.DataFrame(rows,
                                       columns=['Store', 'Date', 'Temperature',␣
 ↪'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4',␣
 ↪'MarkDown5', 'CPI', 'Unemployment', 'IsHoliday'])
# Sanity test that we got good data from db server
weekly_sales_temp_unemp.head()
```

[33]:
```
   Store       Date          Temperature           Fuel_Price  \
0      1 2010-02-05                42.31                 2.57
1      1 2010-02-12                38.51                 2.55
2      1 2010-02-19                39.93                 2.51
3      1 2010-02-26                46.63                 2.56
4      1 2010-03-05                46.50                 2.62

              MarkDown1            MarkDown2            MarkDown3  \
```

```
0                0.00                0.00                0.00
1                0.00                0.00                0.00
2                0.00                0.00                0.00
3                0.00                0.00                0.00
4                0.00                0.00                0.00

             MarkDown4           MarkDown5                 CPI  \
0                0.00                0.00              211.10
1                0.00                0.00              211.24
2                0.00                0.00              211.29
3                0.00                0.00              211.32
4                0.00                0.00              211.35

          Unemployment  IsHoliday
0                 8.11      False
1                 8.11       True
2                 8.11      False
3                 8.11      False
4                 8.11      False
```

[36]:
```
unemployment = []
temperature = []

for row in top_ten_weekly_sales.itertuples():
    temperature.append(weekly_sales_temp_unemp[(weekly_sales_temp_unemp.Store␣
 ↪== row.Store) & (weekly_sales_temp_unemp.Date == row.Date)]['Temperature'].
 ↪values[0])
    unemployment.append(weekly_sales_temp_unemp[(weekly_sales_temp_unemp.Store␣
 ↪== row.Store) & (weekly_sales_temp_unemp.Date == row.Date)]['Unemployment'].
 ↪values[0])
```

[37]:
```
se_unemployment = pd.Series(unemployment)
se_temperature = pd.Series(temperature)
```

[39]:
```
top_ten_weekly_sales.insert(loc=5, column='Unemployment', value=se_unemployment.
 ↪values)
top_ten_weekly_sales.insert(loc=6, column='Temperature', value=se_temperature.
 ↪values)

top_ten_weekly_sales.head()
```

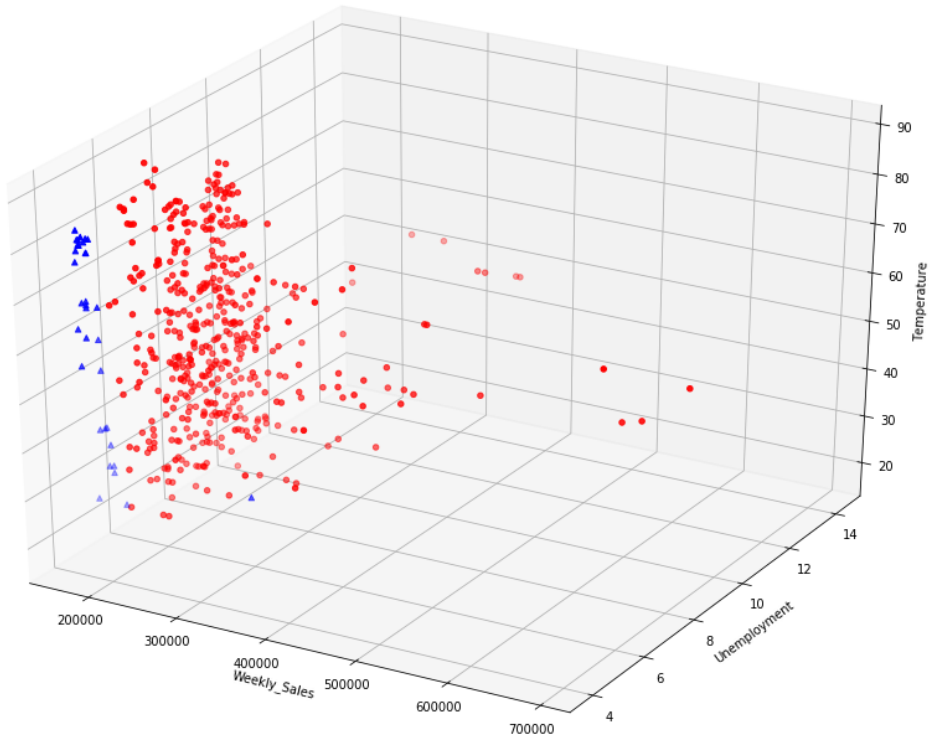
[39]:
```
        Store  Dept       Date         Weekly_Sales  IsHoliday  \
95373      10    72 2010-11-26           693,099.36       True
338013     35    72 2011-11-25           649,770.18       True
95425      10    72 2011-11-25           630,999.19       True
337961     35    72 2010-11-26           627,962.93       True
135665     14    72 2010-11-26           474,330.10       True

               Unemployment          Temperature
```

```
95373                9.00                55.33
338013               8.74                47.88
95425                7.87                60.68
337961               8.76                46.67
135665               8.72                46.15
```

[40]:
```
feature0_unemployment_g5 = top_ten_weekly_sales[top_ten_weekly_sales.
 ↪Unemployment >= 5]['Weekly_Sales'].values
feature0_unemployment_l5 = top_ten_weekly_sales[top_ten_weekly_sales.
 ↪Unemployment <  5]['Weekly_Sales'].values

feature1_unemployment_g5 = top_ten_weekly_sales[top_ten_weekly_sales.
 ↪Unemployment >= 5]['Unemployment'].values
feature1_unemployment_l5 = top_ten_weekly_sales[top_ten_weekly_sales.
 ↪Unemployment <  5]['Unemployment'].values

feature2_unemployment_g5 = top_ten_weekly_sales[top_ten_weekly_sales.
 ↪Unemployment >= 5]['Temperature'].values
feature2_unemployment_l5 = top_ten_weekly_sales[top_ten_weekly_sales.
 ↪Unemployment <  5]['Temperature'].values
```

[41]:
```
# Plot the raw data

fig = plt.figure(figsize=(16, 12))
ax = fig.add_subplot(111, projection='3d')


x_ax_unemployment_g5 = np.array(feature0_unemployment_g5)
y_ax_unemployment_g5 = np.array(feature1_unemployment_g5)
z_ax_unemployment_g5 = np.array(feature2_unemployment_g5)

x_ax_unemployment_l5 = np.array(feature0_unemployment_l5)
y_ax_unemployment_l5 = np.array(feature1_unemployment_l5)
z_ax_unemployment_l5 = np.array(feature2_unemployment_l5)


ax.scatter(x_ax_unemployment_g5, y_ax_unemployment_g5, z_ax_unemployment_g5,␣
 ↪marker='o', c = 'red')
ax.scatter(x_ax_unemployment_l5, y_ax_unemployment_l5, z_ax_unemployment_l5,␣
 ↪marker='^', c = 'blue')


ax.set_xlabel('Weekly_Sales')
ax.set_ylabel('Unemployment')
ax.set_zlabel('Temperature')
```

```
plt.show();
```


Weekly_Sales
200000
300000
400000
500000
600000
700000
Unemployment
4
6
8
10
12
14
Temperature
20
30
40
50
60
70
80
90