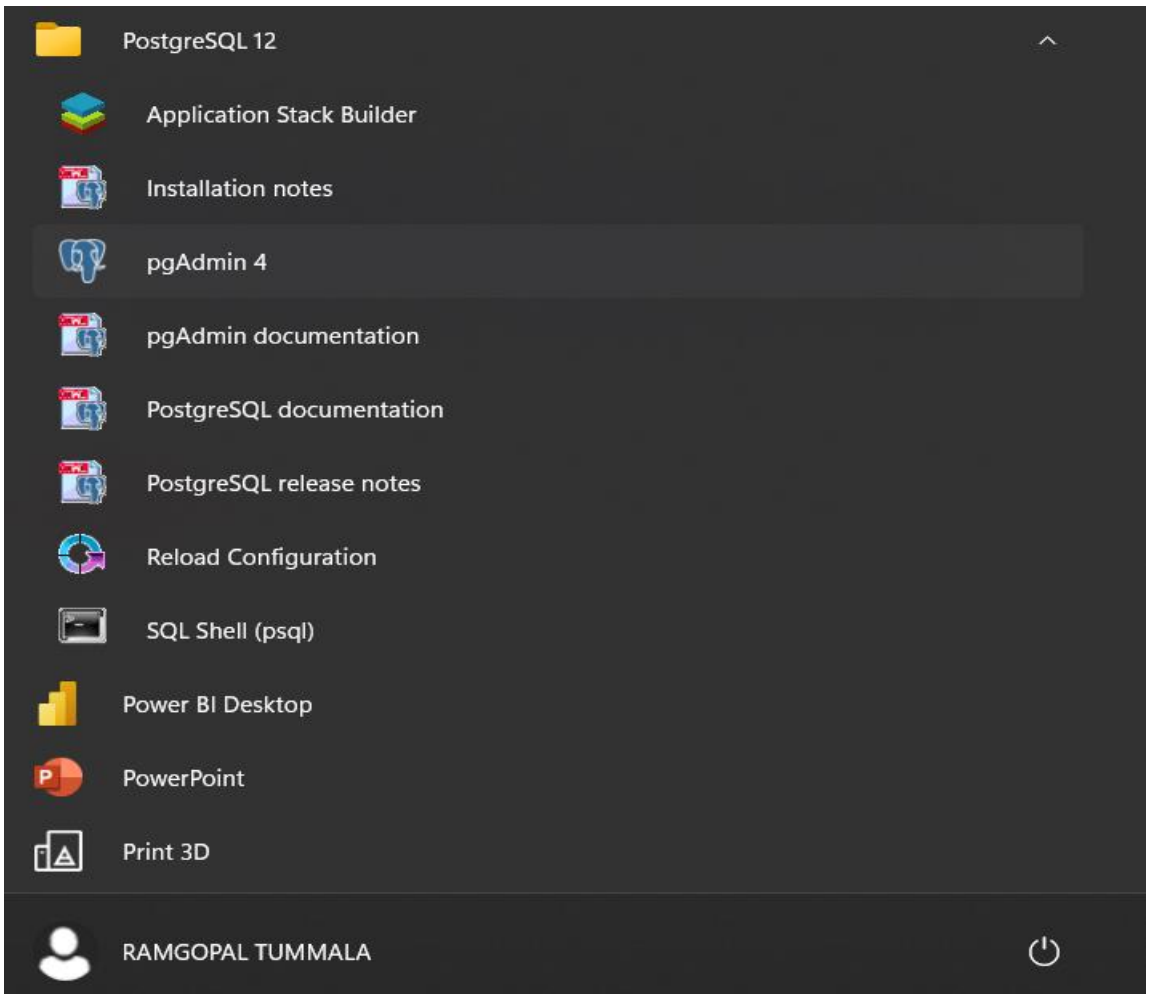
CS 486/586 Introduction to Databases Project - Part II

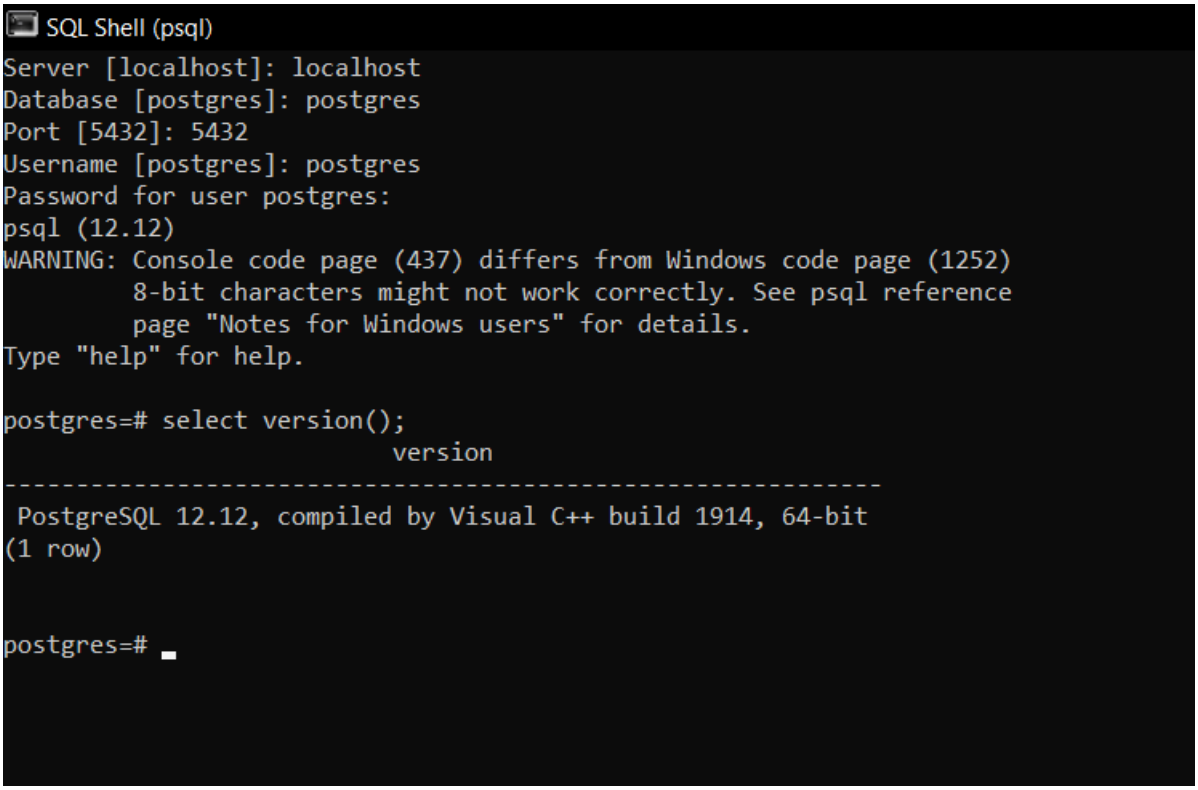
Database Project Implementation:

- **Part 1: Project proposal (Due date - 3rd week, Oct 14th) - 20 points**
- **Part 2: Mid project (Due date - 7th week, Nov 10th) - 200 points**
- **Part 3: Progress report (Due date - 8th week, Nov 18th) - 30 points**
- **Part 4: Final project (Due date - Finals week, Dec 7th) - 150 points**

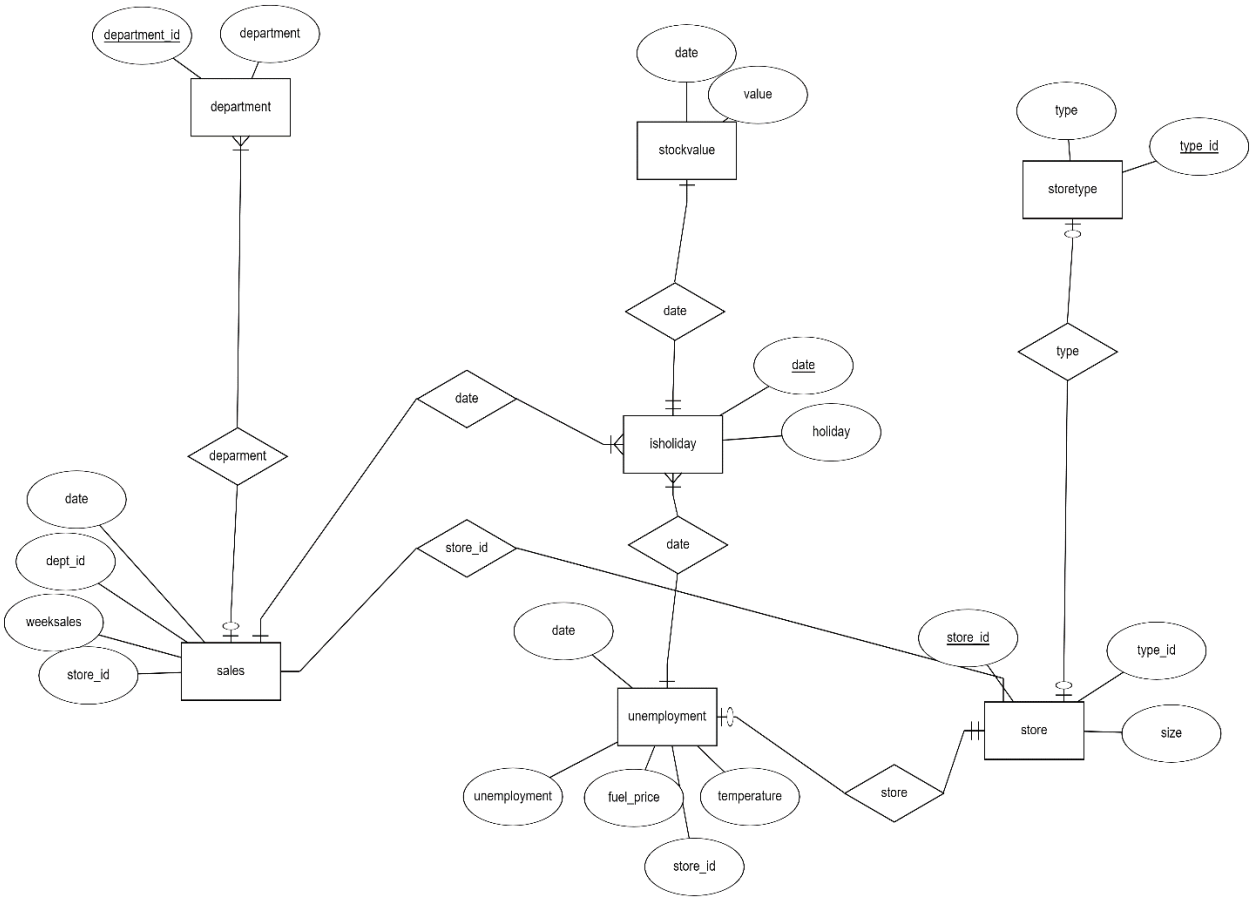
Name:RAMGOPAL PSUID:918450087

- Show that you have successfully installed Postgres and you can use pgAdmin or psql on your local machine or virtual machine (you can provide a screenshot).




SQL Shell (psql)
Server [localhost]: localhost
Database [postgres]: postgres
Port [5432]: 5432
Username [postgres]: postgres
Password for user postgres:
psql (12.12)
WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.
postgres=# select version();
version
PostgreSQL 12.12, compiled by Visual C++ build 1914, 64-bit
(1 row)
postgres=#

- **ER diagram.**


department_id
department
department
date
value
stockvalue
type
type_id
storetype
date
type
date
holiday
isholiday
deparment
date
date
dept_id
weeksales
store_id
sales
store_id
date
unemployment
store_id
type_id
store
size
unemployment
fuel_price
temperature
store_id
store

● Database schema designed with your designed tables, attribute declaration, primary/foreign keys, views, or temporary tables etc. (You should demonstrate a variety of schema and SQL features such as a variety of data types, keys, foreign keys, different cardinalities).

Table: Department

Department(department_id INTEGER,department TEXT)

```
postgres=# \d department
                Table "public.department"
    Column     |  Type   | Collation | Nullable | Default
---------------+---------+-----------+----------+---------
 department_id | integer |           | not null |
 department    | text    |           |          |
Indexes:
    "department_pkey" PRIMARY KEY, btree (department_id)
Referenced by:
    TABLE "sales" CONSTRAINT "fk_sales_dept_id" FOREIGN KEY (dept_id) REFERENCES department(department_id)
```

Data of Department table :

```
postgres=# select * from department;
 department_id |             department
---------------+-------------------------------------
             1 | Animal
             2 |  pet
             3 | equestrian
             4 | livestock and animal products
             5 | Animal housing
             6 | carriers
             7 | equestrian riding gear
             8 | professional veterinary supplies
             9 | Animal Accessories
```

Table: isholiday

Isholiday(date DATE,holiday BOOLEAN)

```
postgres=# \d isholiday
              Table "public.isholiday"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 date    | date    |           | not null |
 holiday | boolean |           |          |
Indexes:
    "isholiday_pkey" PRIMARY KEY, btree (date)
Referenced by:
    TABLE "sales" CONSTRAINT "fk_sales_date" FOREIGN KEY (date) REFERENCES isholiday(date)
```

Data

```
postgres=# select * from isholiday;
    date    | holiday
------------+---------
 2010-02-05 | f
 2010-02-12 | t
 2010-02-19 | f
 2010-02-26 | f
 2010-03-05 | f
 2010-03-12 | f
 2010-03-19 | f
```

Table: sales

```
postgres=# \d sales
                Table "public.sales"
   Column   |  Type   | Collation | Nullable | Default
------------+---------+-----------+----------+---------
 store_id   | integer |           |          |
 dept_id    | integer |           |          |
 date       | date    |           |          |
 week_sales | numeric |           |          |
Foreign-key constraints:
    "fk_sales_date" FOREIGN KEY (date) REFERENCES isholiday(date)
    "fk_sales_dept_id" FOREIGN KEY (dept_id) REFERENCES department(department_id)
    "fk_sales_store_id" FOREIGN KEY (store_id) REFERENCES store(store_id)
```

```
postgres=# select * from sales;
 store_id | dept_id |    date    | week_sales
----------+---------+------------+------------
        1 |       1 | 2010-02-05 |    24924.5
        1 |       1 | 2010-02-12 |   46039.49
        1 |       1 | 2010-02-19 |   41595.55
        1 |       1 | 2010-02-26 |   19403.54
        1 |       1 | 2010-03-05 |    21827.9
        1 |       1 | 2010-03-12 |   21043.39
        1 |       1 | 2010-03-19 |   22136.64
        1 |       1 | 2010-03-26 |   26229.21
        1 |       1 | 2010-04-02 |   57258.43
```

Table: stock value

```
postgres=# \d stockvalue;
            Table "public.stockvalue"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 date   | date    |           |          |
 value  | numeric |           |          |
```

```
postgres=# select * from stockvalue;
    date    |    value
------------+-------------
 2010-02-05 | 53.45000076
 2010-02-12 | 52.90000153
 2010-02-19 | 53.49000168
 2010-02-26 | 54.06999969
 2010-03-05 | 54.13999939
 2010-03-12 | 53.90000153
 2010-03-19 | 55.34000015
 2010-03-26 | 55.50999832
 2010-04-02 | 55.49000168
```

Table: store_type

sl

```
postgres=# \d store_type
           Table "public.store_type"
 Column  | Type | Collation | Nullable | Default
---------+------+-----------+----------+---------
 type_id | text |           | not null |
 type    | text |           |          |
Indexes:
    "store_type_pkey" PRIMARY KEY, btree (type_id)
Referenced by:
    TABLE "store" CONSTRAINT "fk_store_type_id" FOREIGN KEY (type_id) REFERENCES store_type(type_id)
```

```
 type_id |  type
---------+---------
 A       | city
 B       | town
 C       | village
(3 rows)
```

Table: unemployment

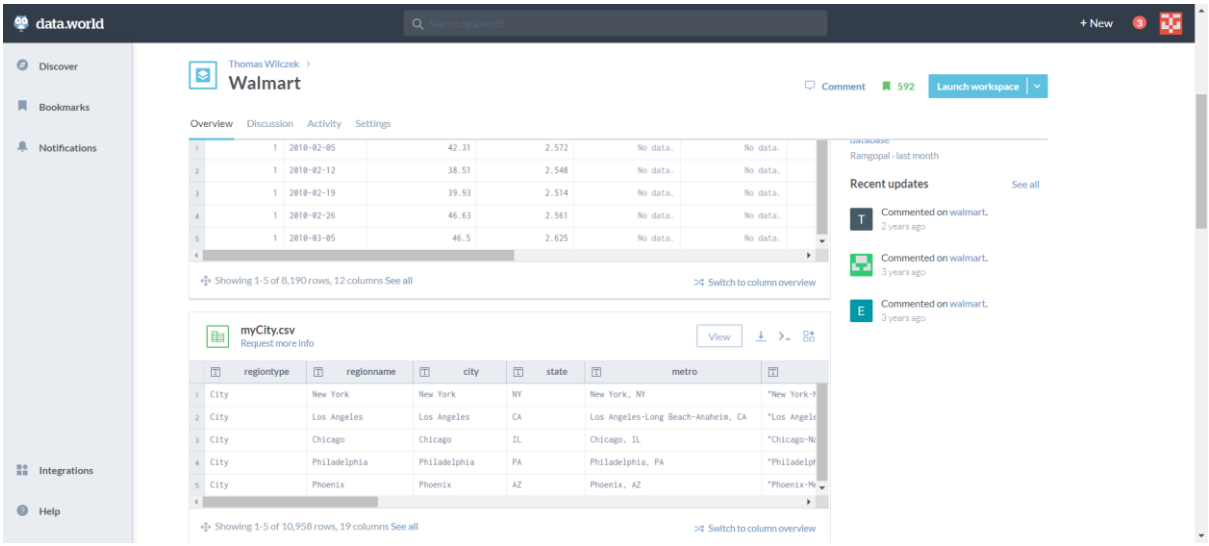
```
postgres=# \d unemployment
              Table "public.unemployment"
    Column     |  Type   | Collation | Nullable | Default
---------------+---------+-----------+----------+---------
 store_id      | integer |           |          |
 date          | date    |           |          |
 temperature   | numeric |           |          |
 fuel_price    | numeric |           |          |
 unemployment  | numeric |           |          |
Foreign-key constraints:
    "fk_unemployment_store_id" FOREIGN KEY (store_id) REFERENCES store(store_id)
```

```
postgres=# select * from unemployment;
 store_id |    date    | temperature | fuel_price | unemployment
----------+------------+-------------+------------+--------------
        1 | 2010-02-05 |       42.31 |      2.572 |        8.106
        1 | 2010-02-12 |       38.51 |      2.548 |        8.106
        1 | 2010-02-19 |       39.93 |      2.514 |        8.106
        1 | 2010-02-26 |       46.63 |      2.561 |        8.106
        1 | 2010-03-05 |        46.5 |      2.625 |        8.106
        1 | 2010-03-12 |       57.79 |      2.667 |        8.106
        1 | 2010-03-19 |       54.58 |       2.72 |        8.106
        1 | 2010-03-26 |       51.45 |      2.732 |        8.106
        1 | 2010-04-02 |       62.27 |      2.719 |        7.808
        1 | 2010-04-09 |       65.86 |       2.77 |        7.808
```

- Data loading process and data preprocessing and cleaning. You should have effective data entry and avoid large amounts of manual data entry. Please describe what you have done to clean your data fully, what you have considered in cleaning, and how you have chosen to load and import your data in your tables.

Data Processing

Collecting data: Initially Data is collected from website called data.world: https://data.world/tommywilczek/walmart the data is in the form of structured but the data is not-normalized and also the data is having noise data(noise data: Data which is inaccurate and disturb the results Ex: data type is numerical but data present is string, which will cause the problem during the processing)



Data Cleaning: It is also known as scrubbing. This task involves filling of missing values, smoothing, or removing noisy data and outliers along with resolving inconsistencies

Data cleaned steps for this data:

1) Mismatched data types-Checking the data type within each column
2) Missing data-Initially checking with null values

Noisy data

Data cleaning also includes fixing "noisy" data. This is data that includes unnecessary data points, irrelevant data, and data that's more difficult to group together.

Removing the unwanted data or replacing the unwanted data

Google Co-lab link : https://colab.research.google.com/drive/1QyN5RnQB6m5MAUHtXOTOe5r3Zbbc4L10?usp=sharing

Importing libraries for processing

```
#Name:Ramgopal Tummala #PSUID:918450087
#Preprocessing the data with some steps
#importing libraries
import pandas as pd
import numpy as np
```

Reading data From CSV to Data Frame

```
#Reading the csv File  from local machine
df = pd.read_csv('/content/project/features.csv')
df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False

Checking the columns

```
for col in df.columns:
    print(col)
```

```
Store
Date
Temperature
Fuel_Price
MarkDown1
MarkDown2
MarkDown3
MarkDown4
MarkDown5
CPI
Unemployment
IsHoliday
```

Removing Unwanted Columns

```
#we dont requried the column MarkDown1,Markdown2....,CPI

del df["MarkDown1"] #removes the column from the Data Frame
del df["MarkDown2"]
del df["MarkDown3"]
del df["MarkDown4"]
del df["MarkDown5"]
del df["CPI"]
```

Finding null values

```
#After removing the columns
for col in df.columns:
    print(col)
#sum NUll values for each column
df.isnull().sum()
```

```
Store
Date
Temperature
Fuel_Price
Unemployment
IsHoliday
Store             0
Date              0
Temperature       0
Fuel_Price        0
Unemployment    585
IsHoliday         0
dtype: int64
```

Filling Null values with Zero

```
[ ]  #Filling NUll values with zero
     df['Unemployment'] = df['Unemployment'].fillna(0)
```

After Filing with zero

```
[ ]  #sum of NUll values after filling zero in null values for each column
     df.isnull().sum()
```

```
Store           0
Date            0
Temperature     0
Fuel_Price      0
Unemployment    0
IsHoliday       0
dtype: int64
```

Checking the column data type

```
[ ]  #check the data type for each coulmn
     df.dtypes
```

```
Store            int64
Date            object
Temperature    float64
Fuel_Price     float64
Unemployment   float64
IsHoliday         bool
dtype: object
```

Pulling data from stocks

Want to add another table with Walmart stock market values at particular dates, for that I've used the python and yfinance(Yahoo Finance) library

Google colab code link:
https://colab.research.google.com/drive/1Vlkm8ixbZRrYUiOlENZabbOMuGLjxdOI#scrollTo=nkfEGflle1NP

Importing Needed libraries

```
#importing supporting libraries
import datetime as dt
import matplotlib.pyplot as plt
from matplotlib import style
import pandas as pd
import pandas_datareader.data as web
```

Importing dates from Isholiday data table for pulling stock value

```
[ ]  #importing particular dates for stock value
     import pandas as pd

     df = pd.read_csv('Isholiday.csv')

     #print(df.to_string())
     df3= pd.DataFrame({'Date': [],'Value': []})
```

Code to pull the stock data on particular date and storing into data frame

```
#code to pull data from Yahoo walmart stock data at particular date
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
temp=0
df2=pd.DataFrame({"Date","Close"})
for ind in df.index:
    if temp==142:
      continue
    else:
      d1 = today.strftime(df['Date'][temp+1])
      end_date = d1
      d2 = date.today() - timedelta(days=360)
      d2 = d2.strftime(df['Date'][temp])
      start_date = d2
      data = pd.DataFrame(yf.download('WMT', start=start_date, end=end_date, progress=False))
      df3=df3.append({'Date':df['Date'][temp],'Value':data['Close'][0]},ignore_index=True)
      temp+=1
print(df3)
```

Results

```
df3
```

	Date	Value
0	2010-02-05	6.980714
1	2010-02-12	7.156429
2	2010-02-19	7.202500
3	2010-02-26	7.307857
4	2010-03-05	7.819643
5	2010-03-12	8.092857
6	2010-03-19	7.937500
7	2010-03-26	8.246429
8	2010-04-02	8.517500
9	2010-04-09	8.635357
10	2010-04-16	8.835714
11	2010-04-23	9.672500
12	2010-04-30	9.324643

Converting data frame into csv file

```
#converting data frame into csv
df3.to_csv('/content/stock.csv')
```

Normalizing_Data

Unemployment data TABLE

Store	Date	Temperature	Fuel_Price	Unemploym	IsHoliday
1	05-02-2010	42.31	2.572	8.106	FALSE
1	12-02-2010	38.51	2.548	8.106	TRUE
1	19-02-2010	39.93	2.514	8.106	FALSE
1	26-02-2010	46.63	2.561	8.106	FALSE
1	05-03-2010	46.5	2.625	8.106	FALSE
1	12-03-2010	57.79	2.667	8.106	FALSE

Week_sales table

Store	Dept	Date	Weekly_Sale	IsHoliday
1	1	05-02-2010	24924.5	FALSE
1	1	12-02-2010	46039.49	TRUE
1	1	19-02-2010	41595.55	FALSE
1	1	26-02-2010	19403.54	FALSE
1	1	05-03-2010	21827.9	FALSE
1	1	12-03-2010	21043.39	FALSE
1	1	19-03-2010	22136.64	FALSE
1	1	26-03-2010	26229.21	FALSE
1	1	02-04-2010	57258.43	FALSE
1	1	09-04-2010	42960.91	FALSE
1	1	16-04-2010	17596.96	FALSE
1	1	23-04-2010	16145.35	FALSE

As we can see the holidays is common for every store at particular date so we can just create a new table for the holiday, so that we can easily access the data from new table created.

ISHOLIDAY TABLE

Date	IsHoliday
05-02-2010	FALSE
12-02-2010	TRUE
19-02-2010	FALSE
26-02-2010	FALSE
05-03-2010	FALSE
12-03-2010	FALSE
19-03-2010	FALSE

New Table for the Isholiday

Loading data to Database from csv files

Importing libraries for connecting to database using python

```
In [1]: #RAMGOPAL PUSID:918450087
        import psycopg2
```

Connecting to psql

```
In [2]: #connection to sql
conn =psycopg2.connect(database="postgres",user="postgres",password="**********",host="localhost",port="5432")
```

Checking connections

```
In [19]: sql='''SELECT datname FROM pg_database'''
conn.autocommit=True
cursor=conn.cursor()
cursor.execute(sql)
```

Print results

```
In [20]: for i in cursor.fetchall():
             print(i)

('postgres',)
('template1',)
('template0',)
```

Loading tables from file path to database

In [16]:
```
#Loadin unemployment table
f=open(r'R:\Ram\PDX\Database\Project\unemployment.csv','r')
cursor.copy_from(f,'unemployment',sep=',')
```

In [10]:
```
#Loading store table
f=open(r'R:\Ram\PDX\Database\Project\store.csv','r')
cursor.copy_from(f,'store',sep=',')
```

In [30]:
```
#Loadin Department table
f=open(r'R:\Ram\PDX\Database\Project\department4.csv','r')
cursor.copy_from(f,'department',sep=',')
```

In [33]:
```
#Loadin Isholiday table
f=open(r'R:\Ram\PDX\Database\Project\Isholiday.csv','r')
cursor.copy_from(f,'isholiday',sep=',')
```

In [8]:
```
#Loadin Isholiday table
f=open(r'R:\Ram\PDX\Database\Project\sales.csv','r')
cursor.copy_from(f,'sales',sep=',')
```

In [17]:
```
#Loadin Isholiday table
f=open(r'R:\Ram\PDX\Database\Project\store_type1.csv','r')
cursor.copy_from(f,'store_type',sep=',')
```

In [16]:
```
#Loadin Isholiday table
f=open(r'R:\Ram\PDX\Database\Project\walstock.csv','r')
cursor.copy_from(f,'stockvalue',sep=',')
```

```
postgres=# \d
            List of relations
 Schema |     Name     | Type  |  Owner
--------+--------------+-------+----------
 public | department   | table | postgres
 public | isholiday    | table | postgres
 public | sales        | table | postgres
 public | stockvalue   | table | postgres
 public | store        | table | postgres
 public | store_type   | table | postgres
 public | unemployment | table | postgres
(7 rows)
```

1)Get the total sales of Baby Toys department from store 1 during holidays

```
SELECT s.week_sales,d.department,s.date,da.holiday

FROM department d

INNER JOIN sales s ON d.department_id=s.dept_id

INNER JOIN isholiday da ON s.date=da.date

WHERE  da.holiday='t' AND d.department_id=(select department_id from department where department ='Baby Toys');
```

week_sales	department	date	holiday
3450.68	Baby Toys	2010-02-12	t
2939.98	Baby Toys	2010-09-10	t
2827.25	Baby Toys	2010-11-26	t
2681.71	Baby Toys	2010-12-31	t
3672.78	Baby Toys	2011-02-11	t
3015.32	Baby Toys	2011-09-09	t
2786.48	Baby Toys	2011-11-25	t

Here used the concept of joins and subquery for retrieving the data, Joined the 3 tables sales, department and the date and used subquery to find the ‘Baby Toys’ id

2)Avg unemployment during holidays for each store

```
SELECT AVG(unemployment),store_id

FROM unemployment e JOIN isholiday da

ON e.date=da.date

WHERE da.holiday='t'

GROUP BY store_id

ORDER BY store_id;
```

avg	store_id
7.7261000000000000	1
7.7133000000000000	2
7.2521000000000000	3
6.1373000000000000	4
6.3907000000000000	5
6.6931000000000000	6
8.6899000000000000	7
6.1639000000000000	8
6.1849000000000000	9
8.4434000000000000	10

Used AVG function for easy way to get the avg of results and grouped the results with the store_id so that we can get individual results of each.

3)what is the stock price and fuel_price near store 5 during the peak sales

SELECT st.value,e.fuel_price

FROM stockvalue st JOIN unemployment e

ON st.date=e.date

WHERE st.date=(SELECT date FROM sales ORDER BY week_sales LIMIT 1) AND e.store_id=5;

```
    value    | fuel_price
-------------+------------
 54.40999985 |      2.633
(1 row)
```

4) What is the difference between stock value between start date and ending of your data

SELECT (SELECT value FROM stockvalue ORDER by date desc limit 1)-(SELECT value FROM stockvalue ORDER BY date ASC LIMIT 1) AS “DIFFERENCE”;

```
 Difference
-------------
 22.20999924
(1 row)
```

5) Get the sale of store 5 with department starting with letter P on 2010-07-02

SELECT s.week_sales,s.date,d.department

From sales s join department d

On s.dept_id =d.department_id

Where s.date=’2010-07-02’ and s.store_id=5 and d.department LIKE ‘p%’;

```
 week_sales |    date    |            department
------------+------------+----------------------------------
   12588.91 | 2010-07-02 | professional veterinary supplies
    7554.09 | 2010-07-02 | pet clothing
     3600.8 | 2010-07-02 | professional cleaning
      334.4 | 2010-07-02 | personal safety equipment
      569.5 | 2010-07-02 | protective clothing
(5 rows)
```

6)Avg Sales During Unemplyment days for each store

SELECT AVG(s. week.sales),s.store_id

From sales s join isholiday i

On s.date=i.date

Where i.holiday='t'

Group by store_id;

Order by store_id;

```
         avg          | store_id
----------------------+----------
   23039.386666666667 |        1
   28798.710526315789 |        2
6916.4462875197472354 |        3
   30854.231416781293 |        4
5617.2126093750000000 |        5
   23313.563481276006 |        6
9730.8287264833574530 |        7
   14013.374425287356 |        8
9423.2131360000000000 |        9
   29195.524157458564 |       10
   20486.484936350778 |       11
   16663.842166910688 |       12
```

7) Get the total count of individual store type

SELECT type_id,count(type_id)

From store

Group by type_id;

```
postgres=# SELECT type_id,count(type_id)
postgres-# from store
postgres-# group by type_id;
 type_id | count
---------+-------
 B       |    17
 C       |     6
 A       |    22
(3 rows)
```

8) what is the fuel price during the Peak sales

SELECT DISTINCT un.fuel_price,un.date

From unemployment un joins sales s

On un.store_id= s.store_id

Where un.date=(sales date from sales order by week_sales limit 1);

```
fuel_price |    date
-----------+------------
     2.645 | 2010-10-08
```

9) Temperature during the low fuel price

SELECT temperature,fuel_price

From unemployment

ORDER BY fuel_price ASC limit 1;

```
 temperature | fuel_price
-------------+------------
       45.66 |      2.472
(1 row)
```

10) Get the sales of department ‘cameras’

SELECT s.store_id,d.department,sum(s.week_sales)

From sales s join department d

On s.dept_id=d.department_id

Where department=’cameras’

Group by s.store_id,department;

```
store_id | department |     sum
---------+------------+-------------
       1 | cameras    |  9186168.91
       2 | cameras    | 11527284.39
       3 | cameras    |    45572.04
       4 | cameras    |  9514563.23
       5 | cameras    |   208382.70
       6 | cameras    |  6473667.94
       7 | cameras    |  1467734.84
       8 | cameras    |  4508870.21
       9 | cameras    |   124306.08
      10 | cameras    |  1816608.28
      11 | cameras    |  6010343.01
      12 | cameras    |   969517.33
      13 | cameras    | 11622037.71
      14 | cameras    | 13071120.15
```