Homework5

question1

here, I first create a python file included SQL queries in the code. which is as follows:

```
ngine.execute('CREATE TABLE IF NOT EXISTS "Place" ('
                    'code INT64 PRIMARY KEY NOT NULL,'
                    'name OBJECT NOT NULL,'
                    'latitude FLOAT64 NOT NULL,'
                    'longitude FLOAT64 NOT NULL);'
                   )

    Engine.execute('CREATE TABLE IF NOT EXISTS "Observation" ('
                    'place OBJECT NOT NULL,'
                    'date DATE PRIMARY KEY NOT NULL,'
                    'rain FLOAT64,'
                    'snow FLOAT64,'
                    'air temperature FLOAT64,'
                    'ground temperature FLOAT64);'
                   )

    Engine.execute('CREATE TABLE IF NOT EXISTS "Temperature" ('
                    'place OBJECT,'
                    'date DATE PRIMARY KEY NOT NULL,'
                    'lowest FLOAT64,'
                    'highest FLOAT64);'
                   )
```

also it includes the columns type and how many column should be created.

and the whole code for question 1 is like as follows:

```
import time
import os
import numpy as np
import pandas as pd
# Must install package matplotlib
import matplotlib.pyplot as plt
from sqlalchemy.ext.declarative import declarative_base
from datetime import datetime
from sqlalchemy import create_engine, MetaData, event, schema
from sqlalchemy import Table, Column, Date, Integer, Float, VARCHAR ,ForeignKey
# from sqlalchemy.orm import sessionmakere
from sqlalchemy_utils import database_exists,create_database


#*****************************************************************************
def read_given_file(f,use_sem_col):
    # reads CSV file into a pandas dataframe
    # return pd.read_csv(f,  sep=';',comment='#',skiprows=[1])
    if use_sem_col:
```

```
      return pd.read_csv(f,  sep=';',comment='#',dtype='unicode')
    else:
      return pd.read_csv(f,  sep=',',comment='#',dtype='unicode')

try:
    '''************************************************************************
    Main unit for Data sets
    '''
    # Create a new SQLite database
    csv_file = '//home//yuy4//weather_data_2020.csv'
    df = pd.read_csv(csv_file)
    print(list(df.columns))
    print(list(df.dtypes))
    print("Analyzed, found " +  str(df.shape[0]) + " rows; " + str(df.shape[1])
+ " cols.")
    Sqlite_Server      = 'sqlite:///'
    my_path = '//home//yuy4//'
    DB_NAME_ = 'weatherdate_.db'
    WEATHER_ = 'weather_data_2020'

    # Create an engine object
    Engine = create_engine(Sqlite_Server + my_path + DB_NAME_, echo=False)
    Sqlite_Conn = Engine.connect()
    if not Sqlite_Conn:
        print("DB connection is not OK!")
        exit()
    else:
        print("DB connection is OK.")

    # CREATE TABLE
    Engine.execute('CREATE TABLE IF NOT EXISTS "Place" ('
                   'code INT64 PRIMARY KEY NOT NULL,'
                   'name OBJECT NOT NULL,'
                   'latitude FLOAT64 NOT NULL,'
                   'longitude FLOAT64 NOT NULL);'
                  )

    Engine.execute('CREATE TABLE IF NOT EXISTS "Observation" ('
                   'place OBJECT NOT NULL,'
                   'date DATE PRIMARY KEY NOT NULL,'
                   'rain FLOAT64,'
                   'snow FLOAT64,'
                   'air temperature FLOAT64,'
                   'ground temperature FLOAT64);'
                  )

    Engine.execute('CREATE TABLE IF NOT EXISTS "Temperature" ('
                   'place OBJECT,'
                   'date DATE PRIMARY KEY NOT NULL,'
                   'lowest FLOAT64,'
                   'highest FLOAT64);'
                  )
```

question2

for question 2, the first step is that the data in the .csv file should be transformed into corresponding tables in the databases.

so the codes as follows:

```
    place_cols = ['place_code','place','latitude','longitude']
    Place = df[place_cols]
    Place.rename(columns=
{'place_code':'code','place':'name','latitude':'latitude','longitude':'longitude
'}, inplace = True)

    date = pd.to_datetime(df[['year','month','day']])
    date = date.dt.date
    df.insert(df.shape[1],'date',date)
    Observation_cols =
['place','date','rain','snow','air_temperature','ground_temperature']
    Observation = df[Observation_cols]
    Observation.rename(columns={'air_temperature':'air
temperature','ground_temperature':'ground temperature'}, inplace = True)

    Temperature_cols =
['place','date','lowest_temperature','highest_temperature']
    Temperature = df[Temperature_cols]
    Temperature.rename(columns=
{'lowest_temperature':'lowest','highest_temperature':'highest'}, inplace=True)

    Place.to_sql('Place', Sqlite_Conn, if_exists='replace', index=False)
    Observation.to_sql('Observation', Sqlite_Conn, if_exists='replace',
index=False)
    Temperature.to_sql('Temperature', Sqlite_Conn, if_exists='replace',
index=False)

    print(Engine.execute("SELECT * FROM Place").fetchall())
    print(Engine.execute("SELECT * FROM Observation").fetchall())
    print(Engine.execute("SELECT * FROM Temperature").fetchall())
```

and the second step is to sanitize the data

for task

Observation: rain and snow should have value 0, if there is no rain or snow. If the rain/snow observation is missing, value should be NULL

```
Observation['rain'].replace(-1,0,inplace=True)
Observation['snow'].replace(-1,0,inplace=True)
```

for task

If any observation value is missing, the value should be NULL.

```
Observation.fillna('NULL')
Observation.fillna('NULL')
```

question3

(a).

Find the number of snowy days on each location.

	SnowDay	place
0	8	Helsinki-Vantaa Airport
1	48	Pötsönvaara
2	218	Utsjoki

the result from the query has shown that **Utsjoki** has the most snowy days.

For this location, find the month with most snow (sum) , obvious this location is Utsjoki, but it should be picked up by code.

```
with mostsnow AS
(SELECT MAX(SnowyDay) AS maxsnowday, OVP
FROM
(SELECT count(*) AS SnowyDay, OV.place AS OVP
FROM Observation AS OV
WHERE OV.snow > 0
GROUP BY place)),

snoweachmonth AS
(SELECT SUM(snow) AS sumsnow, strftime('%m',date) as month
FROM Observation,mostsnow
WHERE Observation.place = mostsnow.OVP
GROUP BY month)

SELECT goal.goalmonth,MAX(goal.snowgoal) AS sumsnow FROM
(SELECT DISTINCT Observation.place,SEM.month AS goalmonth,SEM.sumsnow AS
snowgoal
FROM Observation,mostsnow,snoweachmonth AS SEM
WHERE Observation.place = (mostsnow.OVP) AND snow IS NOT NULL AND snow > 0
GROUP BY SEM.month) AS goal;
```

For the location with least snowy days, find the month with most snowy days. Actually the location is Helsinki-Vantaa Airport

```
with leastsnow AS
(SELECT MIN(SnowyDay) AS minsnowday, OVP
FROM
(SELECT COUNT(*) AS SnowyDay, OV.place AS OVP
FROM Observation AS OV
WHERE OV.snow > 0
GROUP BY place)),

mostsnowday AS
(SELECT COUNT(*) AS countsnowday, strftime('%m',date) as month
FROM Observation,leastsnow
WHERE Observation.place = leastsnow.OVP AND snow > 0 AND snow IS NOT NULL
GROUP BY month)

SELECT goal.goalmonth,MAX(goal.countsnowday) AS mostsnowday FROM
(SELECT DISTINCT Observation.place,MSD.month AS goalmonth,MSD.countsnowday AS
countsnowday FROM Observation,leastsnow,mostsnowday AS MSD
WHERE Observation.place = (leastsnow.OVP) AND snow IS NOT NULL AND snow > 0
GROUP BY MSD.month) AS goal;
```

(b). Inspect the rows in Temperature where both "highest" and "lowest" are not NULL. Calculate the sample correlation coefficient between these two attributes. What can you interpret from this value? Find the correlations when grouping by location.

the sample correlation coefficient is 0.9028454776893676 between two attributes.

It keeps highly linear correlation. if the correlation value tend to 1, two attributes are more relevant. here, highest temperature would change with same trend along with lowest temperature in each city if considering the place as horizontal axis.

```
8  SELECT place,date,lowest,highest FROM Temperature
9  WHERE lowest > 0 AND highest > 0
10 GROUP BY place;
11
12
13
```

Grid view | Form view

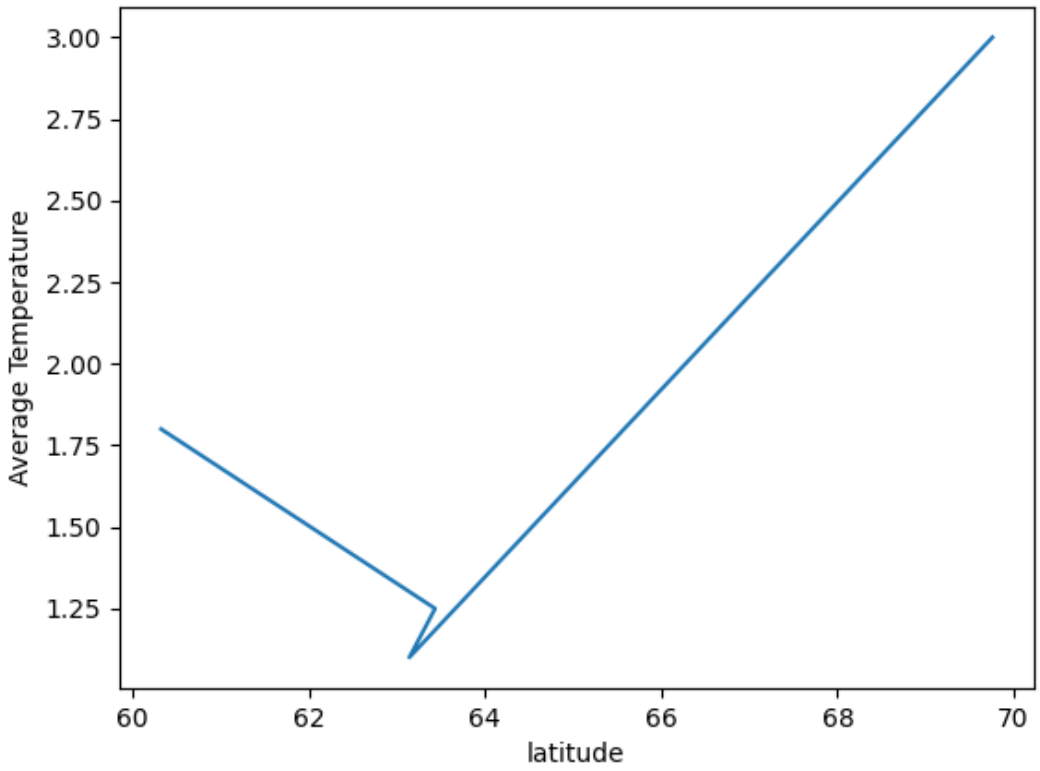
1 | Total rows lo

	place	date	lowest	highest
1	Helsinki-Vantaa Airport	2020-01-02	1	4.2
2	Mustasaari	2020-01-01	2	5.2
3	Pötsönvaara	2020-06-16	13.9	22.8
4	Utsjoki	2020-04-18	0.8	2.8

after grouping by place, each city exist highest and lowest temperature degree. For example, in Pötsönvaara, the temperature is much higher than others and the temperature keep a highest level in the lowest column. For Utsjoki, the lowest value manifested with the same trend of highest.

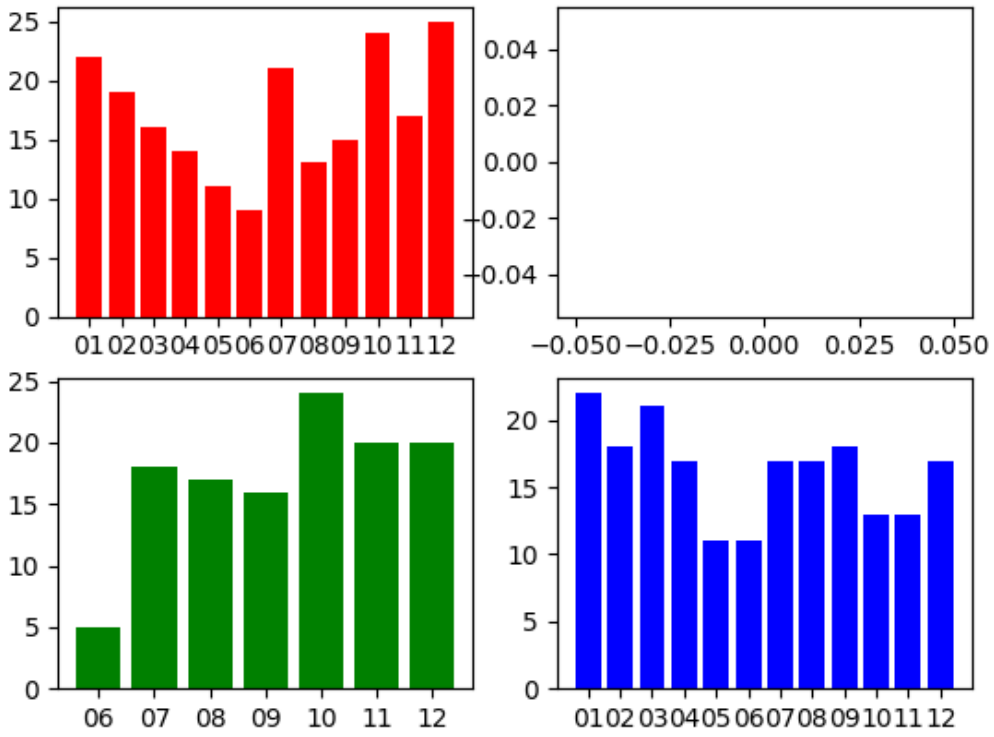
(c). Find out the correlation between average temperature and latitude of the location.

here, the correlation coefficient between these two attributes 'Latitude' and 'Average Temperature' is 0.7596095947903967. The relation is not so strong since the value is less than 0.8



(d). For each location, use myplotlib to plot the number of rainy days for each month as a bar plot.

for city Mustasaari, rainy day is 0; and for city Pötsönvaara, only from June to December belongs to be rainy.



(e). For each location, plot the average temperature throughout the year. You may plot all the graphs
into the same Figure

here is the plot figure of average temperature from January to December throughout the whole year.