DATA ENGINEER QUESTIONS

This assignment has 10 questions, you have 1 week to complete as many questions as possible. You need to create a repository in github to submit an assignment. How well you construct the repository is a plus for you. Good luck to you!!!

SQL question: Using SQL language you feel comfortable with to solve

Q.1 (15 points) Assume that we have a table storing scores of athletes in a competition: Performance(AthleteId, Gender, Country, Score).
Write an SQL to find the second highest score of athletes.

Answer Q.1 (MySQL):

```
-- Create a table
CREATE TABLE Performance (
  AthleteId INTEGER PRIMARY KEY,
  Gender BOOLEAN NOT NULL,
  Country TEXT NOT NULL,
  Score INTEGER NOT NULL
);
GO

-- Insert some values
INSERT INTO Performance VALUES (1, 0, 'VietNam', 100),  (2, 1, 'Singapore', 120), (3, 0, 'America', 90), (4, 0, 'China', 100);

-- Fetch some values
SELECT * FROM Performance
WHERE Score = (
    SELECT DISTINCT(Score)
    FROM Performance
    ORDER BY Score DESC LIMIT 2-1, 1
);
```

Q.2 (10 points) Assume that we have Customers(id, name) and Orders(id, customerId) tables.

Write an SQL query to report all customers who never order anything.

Answer Q.2 (MySQL):

```
-- Create a table
CREATE TABLE Customers (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL
);
GO

CREATE TABLE Orders (
  id INTEGER PRIMARY KEY,
  customerId INTEGER NOT NULL,
  FOREIGN KEY (customerId) REFERENCES Customers(id)
);
GO

-- Insert some values
INSERT INTO Customers VALUES (1, 'Paul'),  (2, 'Jacques'), (3, 'Luc');
INSERT INTO Orders VALUES (1, 2), (2, 3), (3, 2);

-- Fetch some values
SELECT name FROM
```

```
Customers
WHERE Customers.id NOT IN (
    SELECT Customers.id FROM
    Customers INNER JOIN Orders
    ON Customers.id = Orders.customerId
);
```

Q.3 (20 points) Assume that we have Employee(id, name, salary, departmentId) and Department(id, name). A company's executives are interested in seeing who earns the most money in each of the company's departments. A high earner in a department is an employee who has a salary in the top three unique salaries for that department.

Write an SQL query to find the employees who are high earners in each of the departments.

Answer Q.3 (MySQL):

```
-- Create a table
CREATE TABLE Department (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL
);
GO

CREATE TABLE Employee (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  salary INTEGER,
  deparmentId INTEGER,
  FOREIGN KEY (deparmentId) REFERENCES Department(id)
);
GO

-- Insert some values
INSERT INTO Department VALUES (1, 'Marketing'), (2, 'Sales'), (3, 'HR'), (4,
'Finance');
INSERT INTO Employee VALUES (1, 'A1', 100, 2), (2, 'A2', 120, 4), (3, 'A3',
150, 1), (4, 'A4', 90, 3),
(5, 'B1', 80, 3), (6, 'B2', 100, 2), (7, 'B3', 90, 4), (8, 'B4', 90, 1),
(9, 'C1', 110, 1), (10, 'C2', 100, 3), (11, 'C3', 80, 2), (12, 'C4', 150, 4),
(13, 'D1', 110, 2), (14, 'D2', 100, 1), (15, 'D3', 80, 4), (16, 'D4', 150, 3);

-- Fetch some values
SELECT *
FROM Employee INNER JOIN Department
ON Employee.deparmentId = Department.id
WHERE (
    SELECT COUNT(DISTINCT(salary))
    FROM Employee AS f
    WHERE f.deparmentId = Employee.deparmentId and f.salary >=
Employee.salary
) <= 3
ORDER BY Department.id ASC, Employee.salary DESC
;
```

Coding question: You can use any language that you feel confident to solve the problem. Each question has many solutions, find the best solution and explain your solution, find the complexity of this solution (BigO)?

Q.4 (15 points) We have an array of n elements A[1..n]. This array contains n different numbers from 0 to n. Given that there are totally n + 1 numbers from 0 to n, there is a missing number.

Example:

- array: [0, 1, 2, 4]. The missing number is 3.
- array: [1, 4, 2, 3]. The missing number is 0.

Answer Q.4 (Python):

```
import numpy as np
import time

A = np.array([33, 29, 43, 9, 10, 26, 24, 25, 48, 7, 20, 44, 8, 30, 0, 13, 38, 39, 40, 35, 18, 45, 46, 42, 16,
              37, 27, 50, 17, 15, 11, 19, 41, 47, 34, 21, 5, 36, 4, 14, 32, 12, 1, 22, 2, 6, 3, 31, 23, 49])

# Solution 1: Mathematical solution
start = time.time()

n = len(A)
S = np.sum(A)
print(f'Solution 1: The missing number is '+str(n*(n+1)/2-S))

end = time.time()
print(end - start)
```

```
# Solution 2:
start = time.time()

n = len(A)
B = np.arange(n+1)
for k in A:
    B[k] = 0
print(f'Solution 2: The missing number is '+str(sum(B)))

end = time.time()
```

Q.5 (20 points) Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

Example:

- nums1 = [1, 3, 6]; nums2 = [1, 2, 10] => Merged array: [1, 1, 2, 3, 6, 10]. Median is (2 + 3)/2 = 2.5.
- nums1 = [1, 3]; nums2 = [1, 2, 10] => Merged array: [1, 1, 2, 6, 10]. Median is 2.

Answer Q.5 (Python):

```
import numpy as np

# A = np.empty(0)
# B = np.empty(0)
A = np.array([])
B = np.array([])

# med: median, cnt: count
n1 = len(A)
```

```
n2 = len(B)
if (n1 == 0) and (n2 == 0):
    print('There is no data!')
else:
    if (n1 == 0) or (n2 == 0):
        if n2 == 0:
            # med = med(A)
            m1 = n1 // 2
            if n1 % 2 == 0:
                med = (A[m1-1] + A[m1]) / 2
                print(f'median = (A[' + str(m1-1) + ']+A['+str(m1)+'])/2 = ', med)
            else:
                med = A[m1]
                print(f'median = A['+str(m1)+'] = ', med)
        else:
            # med = med(B)
            m2 = n2 // 2
            if n2 % 2 == 0:
                med = (B[m2-1] + B[m2]) / 2
                print(f'median = (B['+str(m2-1)+']+B['+str(m2)+'])/2 = ', med)
            else:
                med = B[m2]
                print(f'median = B['+str(m2)+'] = ', med)
    else:
        n = n1 + n2
        cnt = 0
        cnt1 = 0
        cnt2 = 0
    med = A[0]
    if n % 2 == 1:  # n is odd
        m = n // 2
        while cnt <= m:
            # print(f'count1=', cnt1, 'count2=', cnt2, 'count=', cnt, 'median=', med)
            if (cnt1 <= n1-1) and (cnt2 <= n2-1):
                if A[cnt1] <= B[cnt2]:
                    if cnt == m:
                        med = A[cnt1]
                        print(f'median = A['+str(cnt1)+'] = ', med)
                    cnt1 += 1
                else:
                    if cnt == m:
                        med = B[cnt2]
                        print(f'median = B['+str(cnt2)+'] = ', med)
                    cnt2 += 1
                cnt = cnt + 1
            else:
                if cnt2 > n2-1:
                    cnt = m+1
                    med = A[m+1-cnt1-cnt2]
                    print(f'median = A['+str(m+1-cnt1-cnt2)+'] = '+str(med))
                else:
                    cnt = m+1
                    med = B[m+1-cnt1-cnt2]
                    print(f'median = B['+str(m+1-cnt1-cnt2)+'] = '+str(med))
        # print(f'count1=', cnt1, 'count2=', cnt2, 'count=', cnt, 'median=', med)
    else:   # n is even
    m = n // 2
    med1 = A[0]
    med2 = B[0]
    while cnt <= m:
        # print(f'count1=', cnt1, 'count2=', cnt2, 'count=', cnt, 'median=', med)
        if (cnt1 <= n1-1) and (cnt2 <= n2-1):
            if A[cnt1] <= B[cnt2]:
                if cnt == m-1:
                    med1 = A[cnt1]
                    print(f'median = (A[' + str(cnt1) + '] + ', end='')
                if cnt == m:
                    med2 = A[cnt1]
                    print(f'A[' + str(cnt1) + '])/2 = '+str((med1+med2)/2))
                cnt1 += 1
            else:
                if cnt == m-1:
                    med1 = B[cnt2]
                    print(f'median = (B[' + str(cnt2) + ']+ ', end='')
                if cnt == m:
                    med2 = B[cnt2]
                    print(f'B[' + str(cnt2) + '])/2 = '+str((med1+med2)/2))
                cnt2 += 1
            cnt = cnt + 1
        else:
            if cnt2 > n2-1:
                cnt = m+1
                if cnt != m-1:
                    med1 = A[m-cnt1-cnt2]
                    print(f'median = (A[' + str(m-cnt1-cnt2) + '] + ', end='')
                med2 = A[m+1-cnt1-cnt2]
                print(f'A[' + str(m+1-cnt1-cnt2) + '])/2 = '+str((med1+med2)/2))
            else:
                cnt = m+1
                if cnt != m-1:
                    med1 = B[m-cnt1-cnt2]
                    print(f'median = (B[' + str(m-cnt1-cnt2) + '] + ', end='')
                med2 = B[m+1-cnt1-cnt2]
                print(f'B[' + str(m+1-cnt1-cnt2) + '])/2 = '+str((med1+med2)/2))
    # print(f'count1=', cnt1, 'count2=', cnt2, 'count=', cnt, 'median=', med)
    med = (med1+med2)/2

if (n1 != 0) or (n2 != 0):
    print(f'The median is: ' + str(med))
```

Q.6 (20 points) Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Example:

- head = [2, 2, 5] => [2, 5]
- head = [1, 1, 3, 4, 4, 5, 6, 6] => [1, 3, 4, 5, 6]

Answer Q.6 (Python):

```
import numpy as np


Head = np.array([1, 1, 3, 4, 4, 5, 5, 6, 6, 6, 6])


# Soltuon 1:
n = len(Head)
```

```
Index = np.arange(n)
for k in range(1, n-1):
    if Head[k] != Head[k-1]:
        Index[k] = 0
Index = np.delete(Index, np.where(Index == 0))
Head = np.delete(Head, Index)
print(Head)
```

Design question (Optional):

Q.7 (Max 10 points) We have a crawler that crawls websites in a list to find sensitive information (e.g., people talk or have opinions about our products). Our list initially contains 100 websites called seeds. When a crawler visits a website, it can find several links to other websites. Depending on the information of the linking websites, they can be added in the list to revisit later (e.g., if they are related to the seeds or contain valuable information about what we want to know, i.e., sensitive information). It means that with the initial of 100 seeds, our list can be updated to include more websites. However, since our resources are limited, we want to maintain only up to 1000 websites. It means that in addition to 100 seeds, we can only maintain a maximum of 900 other websites. Design a solution to maintain this list of websites. Also justify your solution.

Hint: before the list grows to 1000 websites, you can freely add new websites to the list. However, when the list reaches 1000 websites, you need an algorithm to rank websites according to the usefulness of their information to our wanted information (i.e., sensitive information) and keep only those with high scores while removing those with low scores. E.g., when a new website is discovered and its score is higher than an existing one in the list, the new website will be replaced by the lower score in the list.

Answer Q.7:

I have no idea!

Mathematical statistics + other question (Bonus question):

Q.8 (Max 2 points) Assume that you are a rector of a university and you want to show to the public a statistics report for examinees in your university entrance exam, what is the best graph to use. Please justify your decision.

Answer Q.8:

My best graph is bell curve graph but a little right skewed. Because this can make the people think my student at my university is so smart (or the test is easy!).

Q.9 (2 points) We have three identical six-sided dice. We roll one dice first and the remaining two dice after that. What is the probability that the point obtained in the first roll is greater than the sum of the points obtained in the second roll.

Answer Q.9:

$$P = \frac{20}{6^3} = \frac{5}{54} = 9.259\%$$

1st die	2nd die	3rd die
1	✗	✗
2	✗	✗
3	1	1
4	1	2
	1	1
	2	1

1st die	2nd die	3rd die
5	1	3
	1	2
	1	1
	2	2
	2	1
	3	1

1st die	2nd die	3rd die
6	1	4
	1	3
	1	2
	1	1
	2	3
	2	2
	2	1
	3	2
	3	1
	4	1

Q.10 (Max 6 points) Obtain the subway data set of NYC from this link:

https://github.com/andynganle/Data-for-Assignment

This data set records statistics of NYC subway riders in different weather conditions. Study and discuss any interesting features about the data set that you find.

Answer Q.10:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Adding data
path = "D:\DISKP\Bai tap lam gium\Tuyen dung PrimeData"
os.chdir(path)  # Change path
filename = 'Data.csv'
Data = pd.read_csv(filename)
# Data.describe().to_csv("DataDescribe.csv")
# print(Data.describe())

y1 = Data['ENTRIESn_hourly']
y2 = Data['EXITSn_hourly']
print(f'   The number of entries is ' + str(len(y1>=y2)/len(y1)*100) + '% greater
than the number of exits!')

plt.plot(y1, label='Entries')
plt.plot(y2, label='Exits')
plt.xlabel('Time')
plt.ylabel('Number of people')
plt.legend()
plt.show()

C = Data['conds']
print(f'There is ' + str(len(np.unique(C))) + ' weather conditions. There are:')
Clist = np.unique(C).tolist()
print(type(Clist))

print('\nSummary conditions with entries!')
print(Data.groupby('conds')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
plt.pie(Data.groupby('conds')['ENTRIESn_hourly'].mean(), labels=Clist)
plt.show()

print('\nSummary conditions with exits!')
plt.pie(Data.groupby('conds')['EXITSn_hourly'].mean(), labels=Clist)
plt.show()

print(f'    The people don''t want to go on Fog or Mist condition. But Haze is Ok.')
print(f'    Other conditions are not effect to the people')

S = Data['station']
print(f'There is ' + str(len(np.unique(S))) + ' stations.')
plt.plot(Data.groupby('station')['ENTRIESn_hourly'].mean(), label='Entries')
plt.plot(Data.groupby('station')['EXITSn_hourly'].mean(), label='Exits')
plt.xlabel('Time')
plt.ylabel('Number of people')
plt.xticks(rotation=90)
plt.legend()
plt.show()

F = Data['fog']
print(f'There is ' + str(np.sum(F)) + '/' + str(len(F)) + ' days are foggy (' +
f'{(np.sum(F)/len(F)*100):.2f}' + '%).')
print(Data.groupby('fog')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
print(f'    Fog can reduce the number of people')

R = Data['rain']
print(f'There is ' + str(np.sum(R)) + '/' + str(len(R)) + ' days are rainy (' +
f'{(np.sum(R)/len(R)*100):.2f}' + '%).')
print(f'There is ' + str(np.sum(F)) + '/' + str(len(F)) + ' days are foggy (' +
f'{(np.sum(F)/len(F)*100):.2f}' + '%).')
print(Data.groupby('rain')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
print(f'    Rain can reduce the number of people')

T = Data['tempi']
print(f'The temperature in range [' + str(np.min(T)) + '°F, ' + str(max(T)) + '°F]')
print(Data.groupby('tempi')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
print(f'    If the temperature is so cold, it can reduce the number of people')

W = Data['wspdi']
print(f'The wind speed in range [' + str(np.min(W)) + 'mph, ' + str(max(W)) +
'mph]')
print(Data.groupby('wspdi')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
print(f'    If the wind speed is so fast (>=20.7 mph), it can reduce the number of
people.')
```

```
P1 = Data['precipi']
print(f'The precipitation in range [' + str(np.min(P1)) + ' in, ' + str(max(P1)) + ' in]')
print(Data.groupby('precipi')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
print(f'   The precipitation is not effect to the number of people.')

P2 = Data['pressurei']
print(f'The barometric pressure in range [' + str(np.min(P2)) + ' in Hg, ' + str(max(P2)) + ' in Hg]')
print(f'   The barometric pressure is almost the same! It is not effect to the number of people.')
print(Data.groupby('pressurei')['ENTRIESn_hourly', 'EXITSn_hourly'].mean())
```

```
The number of entries is 100.0% greater than the number of exits!
There is 12 weather conditions. There are:
['Clear', 'Fog', 'Haze', 'Heavy Rain', 'Light Drizzle', 'Light Rain', 'Mist', 'Mostly Cloudy', 'Overcast', 'Partly Cloudy', 'Rain', 'Scattered Clouds']


   The people dont want to go on Fog or Mist condition. But Haze is Ok.
   Other conditions are not effect to the people
There is 207 stations.
There is 419/42649 days are foggy (0.98%).
   Fog can reduce the number of people
There is 9585/42649 days are rainy (22.47%).
   Rain can reduce the number of people
The temperature in range [46.9°F, 86.0°F]
   If the temperature is so cold, it can reduce the number of people
```

The wind speed in range [0.0mph, 23.0mph]

If the wind speed is so fast (>=20.7 mph), it can reduce the number of people.

The precipitation in range [0.0 in, 0.3 in]

The precipitation is not effect to the people.

The barometric pressure in range [29.55 in Hg, 30.32 in Hg]

The barometric pressure is almost the same!