**Data Engineer Questions**

**Q.1**: 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:**

The below SQL steps (Posgres SQL syntax) are described as below:

1. Get the highest score (1st sub-query)
2. Get the second highest score (2nd sub-query)
3. Get the Athletes with score equal to second highest score

|  |
| --- |
| SELECT AthleteId FROM Performance WHERE Score =  ( SELECT Score FROM Performance WHERE Score < ( SELECT MAX(Score) FROM Performance) ORDER BY Score DESC LIMIT 1 ); |

**Q.2**: Assume that we have a table storing order information: **Order(OrderId, ItemId, Price, Quantity, Description)** in which **OrderId** is the Primary Key of the table and there is no constraint.

We have two SQL queries.

· **Select Count(\*) from Order**. The returned result is 2015.

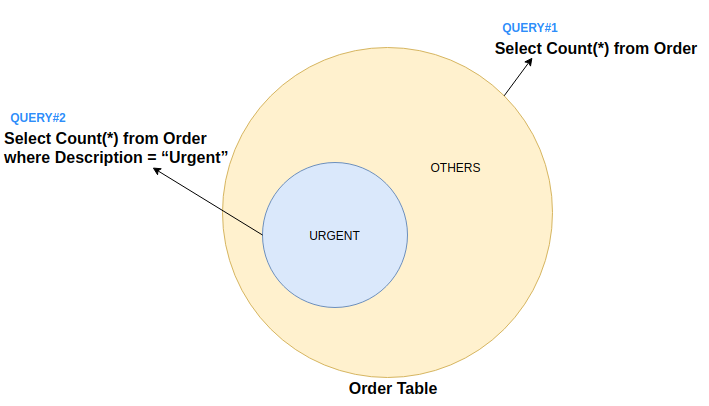
· **Select Count(\*) from Order where Description = “Urgent”**. The returned result is 115.

What is the returned result of the query **Select Count(\*) from Order where Description != “Urgent”**. Please explain the answer.

**Answer:**

The returned result is: 2015 - 115 = 1900

The explanation can be illustrated as below Figure. The result of Query#2 with filter condition Description = 'Urgent' is part of Query#1 OrderId collection. Then the result of Select Count(\*) from Order where Description != ‘Urgent’is just the deviation of result between QUERY#1 & 2.



**Q.3:** 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. Write a function to find this missing number in O(n) time.

**Answer:**

* This solution relies on below formula:

|  |
| --- |
| 1 + 2 + 3 + ... + n = n(n + 1)/2 |

* The time complexity of to implement **findMissingNo** is O(n) which is to iterate array of n elements for the sum calculation.

|  |
| --- |
| def findMissingNo(arr):  # get the length of input array  n = len(arr)    # calculate the total of n+1 consecutive numbers from 0-> n  total = n\*(n+1)/2    # calculate sum of all elements of the input array  sumArr = 0  for i in arr:  sumArr += i    # return missing one  return total - sumArr # Test numbers = [1,2,3,5,6] print(findMissingNo(numbers)) |

**Q.4:** We have a crawler that craws websites in a **list** to find sensitive information (e.g., people talk or have opinion about our products.). Our list initially contains 100 websites called **seeds**. When a crawler visits a website, it can finds 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 resource is 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 the lower score in the list.

**Answer:**

To maintain the to list that maintains up to 1000 websites and each website possesses a score, we will build a linked list as below (python code). The linked list structure is chosen because it is easier to insert and remove element within a list compared to array, list, stack or queue.

|  |
| --- |
| class Node():  def \_\_init\_\_(self, score):  self.score = score  self.next = None  class LinkList():  def \_\_init\_\_(self, max\_length = 1000):  self.head = None  self.length = 0  self.max\_length = max\_length   def insert(self, new\_node):  # First node  if self.head is None:  new\_node.next = self.head  self.head = new\_node    # Insert before head node  elif self.head.score < new\_node.score:  new\_node.next = self.head  self.head = new\_node   # Insert after prior node having lower score   else:  current = self.head  while (current.next is not None and \  current.next.score >= new\_node.score):  current = current.next    new\_node.next = current.next  current.next = new\_node  # Link list length count  self.length += 1   # If any node exceed the max\_length,   # the last node will be removed to maintain the max\_length  if self.length > self.max\_length:  current = self.head  for i in range(self.max\_length - 1):  current = current.next  current.next = None    def printLinkList(self):  temp = self.head  while(temp):  print(temp.score)  temp = temp.next |

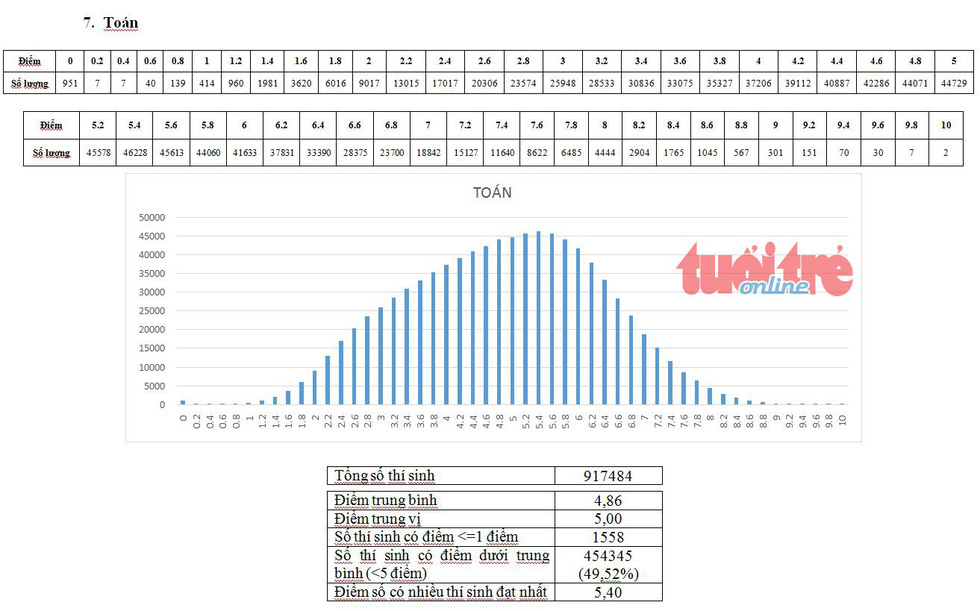
Test with below sample code, the max\_length is set to 5. The list linked list only maintain the top 5 node with highest score (11, 9, 8, 7, 6).

|  |
| --- |
| ll = LinkList(5) ll.insert(Node(3)) ll.insert(Node(9)) ll.insert(Node(7)) ll.insert(Node(6)) ll.insert(Node(11)) ll.insert(Node(8)) ll.printLinkList() |

**Q.5**: Assume that you are a rector of a university and you want to show to public a statistics report for examinees in your university entrance exam, what is the best graph to use. Please justify your decision.

**Answer:**

The best graph to use is the histogram chart because:

* It can demonstrate the distribution of the score where people can easily identify where their score compared with others.
* Moreover, the abnormal outliers are also easily detected at the two most edges.  
  

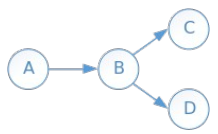
**Q.6**: We have three identical six-sided dices. We role one dice first and the remaining two dices after that. What is the probability that the point obtained in the first roll is greater than sum of the points obtained in the second roll.

**Answer:**

* The result is derived from below python code: **9.26%**

|  |
| --- |
| count = 0 # count number of cases that roll1 get the value greater that roll2 & roll3 for roll1 in range(1, 7):  for roll2 in range(1, 7):  for roll3 in range(1, 7):  if roll1 > roll2 + roll3:  count += 1   # total possibilities (3 six-sided dices) total = 6\*\*3  # probability result print(f"{round(count/total\*100, 2)}%") |

**Q. 7**: Given a Bayesian Network as in Figure 1:

 Figure 1: a Bayesian Network

The above Bayesian Network has the following probabilities

• P(A) = 0.6

• P(B|A) = 0.7, P(B|¬A) = 0.3

• P(C|B) = 0.2, P(C|¬B) = 0.8

• P(D|B) = 0.1, P(D|¬B) = 0.4

Calculate the probability of P(¬C)?

**Answer:**

P(¬C) = 1- P(C) = 1- (P(A).P(B|A).P(C|B) + P(A).P(B|A).P(C|¬B) + P(A).P(B|¬A).P(C|B)+ P(A).P(B|¬A).P(C|¬B)) = 1 – ( 0.6\*0.7\*0.2 + 0.6\*0.7\*0.8 + 0.6\*0.3\*0.2 + 0.6\*0.3\*0.8)

= 0.4

**Q. 8**: Obtain the subway data set of N0.4YC from this link

https://github.com/quanghieu-vu/assignments

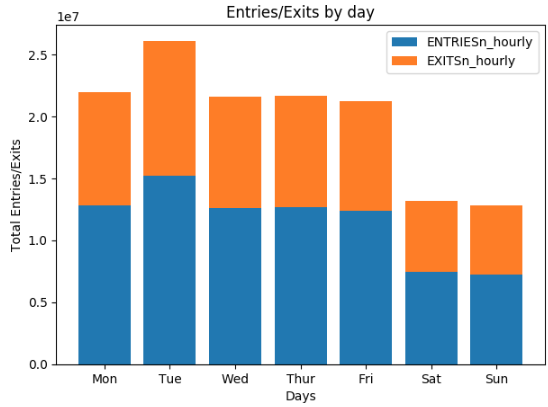
This data set records statistics of NYC subway riders in different weather conditions. Study and discuss any interesting feature about the data set that you find.

**Answer: (refer attached notebook for details)**

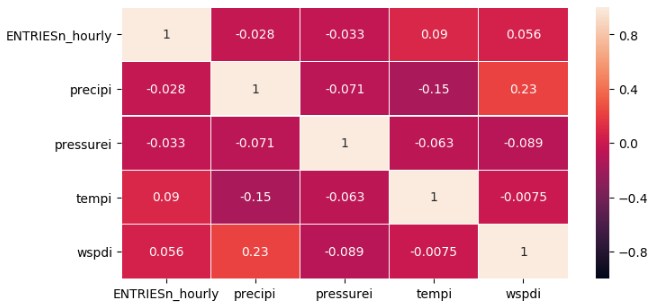
There are 3 interesting features found from the dataset:

1. People use subway much more in weekday (Mon -> Friday). It is understandable that they are going for work.

And total number of exits is less than total number of entries (supposed to be equal). There is one potentital reason is that some people use emergency exit way instead of turnstile. Ref here: [Link](https://streeteasy.com/blog/nyc-subway-emergency-exit-can-i-use-it/)



1. No strong correlation between precipitation, pressure, temperature, wspd with the demand of subway riders.



1. When it is raining, subway gets busier with more people using the subway. It is understandable, people may like to use a bicycle or walking when it is not raining.

