Asad Tariq's Solution to the Liqid's Case Study.

1. SQL

Suppose you have a table named *transactions* with the following columns:

- **id:** the transaction ID (integer)
- **timestamp:** the time the transaction occurred (timestamp)
- **user_id:** the ID of the user who made the transaction (integer)
- **amount:** the amount of the transaction (float)

Write a SQL query to find the second highest transaction amount for each user.

Solution:

I have created a sample table with the provided information.

	ID	USER_ID	AMOUNT
1	1	1	100
2	2	1	200
3	3	2	1000
4	4	2	500
5	5	2	300
6	6	3	1
7	7	3	0
8	8	3	10

Here is the query

```
SELECT user_id, amount AS second_highest_amount
FROM (
  SELECT user_id, amount,
  ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY amount DESC) AS row_number
  FROM transactions
) WHERE row_number = 2;
```

- This inner query selects the columns "user_id" and "amount" from the table "transactions".

- The ROW_NUMBER() function assigns a row number to each row within a user's partition based on the amount in descending order. The "user_id" column defines the partition.
- The result of this inner query is a result set with three columns: "user_id," "amount," and "row_number" (the assigned row number within each user's partition).

- The outer query selects the "user_id" column and the "amount" column as "second_highest_amount".
- It uses the result set obtained from the inner query as a derived table (subquery).
- The outer query filters the derived table by selecting only the rows where the "row_number" column equals 2, representing the second-highest amount for each user.

The result for the sample table is as follows.

	USER_ID	SECOND_HIGHEST_AMOUNT
1	1	100
2	2	500
3	3	1

---

2. Python

Write a Python program to reverse a string using recursion, but without using any built-in string reverse functions or slicing operations.

The code snippet with the output is provided below.

```
def reverse_string_using_recursion(input_string):
    if len(input_string) <= 1:  # Base case: If the length of the string is 0 or 1, return the same string
        return input_string
    else: # Recursive case: Reverse the substring and append the first character at the end
        return reverse_string_using_recursion (input_string[1:]) + input_string[0]

test = ".yduts esac s'diqiL rof detnemelpmi neeb sah gnirts a esrever ot noiitcnuf evisruceR"
transformed_string = reverse_string_using_recursion(test)  # Function call
print(transformed_string)
```

[9] ✓ 0.9s

```
Recursive functiion to reverse a string has been implemented for Liqid's case study.
```

---

3. Sleeping Beauty Problem

Sleeping Beauty agrees to participate in an experiment. On Sunday she is given a sleeping pill and falls asleep. One of the experimenters then tosses a coin. If “heads” comes up, the scientists awaken Sleeping Beauty on Monday. Afterward, they administer another sleeping pill. If “tails” comes up, they wake Sleeping Beauty up on Monday, put her back to sleep and wake her up again on Tuesday. Then they give her another sleeping pill. In both cases, they wake her up again on Wednesday, and the experiment ends.

The important thing here is that because of the sleeping drug, Sleeping Beauty has no memory of whether she was woken up before. So when she wakes up, she cannot distinguish whether it is Monday or Tuesday. The experimenters do not tell Sleeping Beauty either the outcome of the coin toss nor the day.

They ask her one question after each time she awakens, however: What is the probability that the coin shows heads?

What do you think would the correct answer be from the Sleeping Beauty’s perspective? Explain your reasoning.

Solution:
The sleeping beauty problem is a classic problem in the field of Probability. It is like “The Monty Hall problem” of a game show with 3 doors.
Before starting the experiment, the probability of heads with a fair coin would be ½. But, when the experiment is in progress and provided that the sleeping beauty does not know the outcome of the coin toss and she does not know whether she has been awakened before, the trick here is that there is a total of 3 awakening states; no matter the result of a coin toss, sleeping beauty is awakened on Monday. In the case of “heads”, she is not awakened on Tuesday, whereas if the coin toss results in a “tails”, she is also awakened on Tuesday. Therefore, she is not awakened on a single occasion but on 3 occasions.

In “heads”, 1 out of 3 awakenings occur; otherwise, in the case of “tails” 2 out of 3 awakenings occur.

Probability of awakening in case of Heads:
H: 1/3

Probability of awakening in the case of Tails:
T: 2/3