SQL Problems

```
ROW_NUMBER()        RANK()          DENSE_RANK()

A   1               A   1           A   1
A   2               A   1           A   1
B   1               B   3           B   2
C   1               C   4           C   3
C   2               C   4           C   3
D   1               D   6           D   4
```

1. Find Second Highest Salary

Column Name	Type
id	int
salary	int

id is the primary key (column with unique values) for this table.
Each row of this table contains information about the salary of an employee.

Write a solution to find the second highest salary from the Employee table. If there is no second highest salary, return null

```sql
select
(select distinct Salary
from Employee order by salary desc
limit 1 offset 1)
as SecondHighestSalary;
```

PySpark

```python
second_highest_salary = employee_df \
                        .select("Salary") \
                        .distinct() \
                        .orderBy(col("Salary").desc()) \
                        .limit(1) \
                        .offset(1) \
                        .first()[0]

print("Second Highest Salary:", second_highest_salary)
```

2. Delete Duplicates

Column Name	Type

id	int
email	varchar

Write a solution to **delete** all duplicate emails, keeping only one unique email with the smallest id.

```
DELETE p1 FROM Person p1, Person p2
WHERE p1.email = p2.email AND p1.id > p2.id
```

Simple delete duplicate

```
DELETE FROM
(SELECT *,
ROW_NUMBER() OVER(PARTITION BY email ORDER BY id) as rn
FROM table) t
WHERE rn > 1
```

PySpark

```
# Step 1: Identify minimum id for each email
min_ids_df =
df.groupBy('email').agg(min('id').alias('min_id'))

# Step 2: Join to filter out rows with duplicate emails and
keep only the one with the minimum id
unique_emails_df = df.join(min_ids_df, (df['email'] ==
min_ids_df['email']) & (df['id'] == min_ids_df['min_id']),
'inner') \
.select(df['id'], df['email'])

# Step 3: Show or save the unique emails DataFrame as per
your requirement unique_emails_df.show()
```

3. GROUP CONCAT

Input:
Activities table:

sell_date	product
2020-05-30	Headphone
2020-06-01	Pencil
2020-06-02	Mask
2020-05-30	Basketball
2020-06-01	Bible
2020-06-02	Mask
2020-05-30	T-Shirt

```
+------------+------------+
```
Output:
```
+------------+----------+------------------------------+
| sell_date  | num_sold | products                     |
+------------+----------+------------------------------+
| 2020-05-30 | 3        | Basketball,Headphone,T-shirt |
| 2020-06-01 | 2        | Bible,Pencil                 |
| 2020-06-02 | 1        | Mask                         |
+------------+----------+------------------------------+
```

```
SELECT sell_date, COUNT(DISTINCT product) as 'num_sold',
GROUP_CONCAT(DISTINCT product ORDER BY product) AS
'products'
FROM Activities
GROUP BY sell_date
ORDER BY sell_date
```

4. There is a table given as

   You need to find the gap between sequential numbers
   e.g. between 3, 50 the gap is 4, 49
   so output format will be

   ```
   4   49
   56  99
   .....
   ```

   Test1

col
1
2
3
50
51
52
53
54
55
100
101
102
500

950
951
952
954

```
SELECT
    col + 1 AS start_gap,
    next_col - 1 AS end_gap
FROM (
  SELECT col, LEAD(col)OVER(ORDER BY col) as next_col
  FROM Test1
) t
WHERE
    next_col - col > 1
```