Chakradhar's Approach:

He has used a procedure for each question and called the procedure to get the result.

Q1.

```
DELIMITER &&
CREATE PROCEDURE get_num_of_male_and_female ()
BEGIN
SELECT
 IFNULL(Department, 'Not Assigned') as Department,
 COUNT(
   CASE WHEN UPPER(Gender)= 'MALE' THEN 1 END
 ) AS 'Num of Male',
 COUNT(
   CASE WHEN UPPER(Gender)= 'FEMALE' THEN 1 END
 ) AS 'Num of Female'
FROM
 employees
GROUP BY
 Department
order by
 Department;
END &&
DELIMITER ;
```

The query is a SQL stored procedure. The procedure is named "get_num_of_male_and_female". The procedure performs the following actions upon execution :

- Selects data from the "employees" table.
- Groups the data by the "Department" column.
- Counts the number of occurrences of "Male" and "Female" in the "Gender" column and returns the results as two separate columns named "Num of Male" and "Num of Female".
- Returns the department name along with the count of male and female employees in each department, with a default value of "Not Assigned" for departments that have no assigned employees.
- The results are ordered by the "Department" column in ascending order.

Q2.

```
DELIMITER &&
CREATE PROCEDURE max_amount_from_rows_with_month_name()
BEGIN
```

```
select
  emp_name as Name,
  value,
  case when idx = 1 then 'Jan' when idx = 2 then 'Feb' when idx = 3 then 'Mar' end
as Month
from
  (
    select
      emp_name,
      greatest(Jan, Feb, March) as value,
      field(
        greatest(Jan, Feb, March),
        Jan,
        Feb,
        March
      ) as idx
    from
      employeesalaries
  ) emps;
END &&
DELIMITER ;
```

Procedure holding Query to find the max amount from the rows with month name -

The query is a SQL stored procedure. The procedure is named "max_amount_from_rows_with_month_name". The procedure performs the following actions upon execution:

- Selects data from a table named "employeesalaries" and Finds the maximum value from the columns "Jan", "Feb", and "March" and stores it in the "value" column.
- Determines the month corresponding to the maximum value using the "field" function, which returns the index of a value in a list of values, and stores the result in the "idx" column.
- Uses a "CASE WHEN" statement to convert the "idx" column into the month name (Jan, Feb, or Mar) and returns the result in the "Month" column.

Q3.

```
DELIMITER &&
CREATE PROCEDURE rank_in_order ()
BEGIN
SELECT
```

```
  Marks,
  dense_rank() OVER (
    order by
      marks desc
  ) as 'Rank',
  GROUP_CONCAT(Candidate_id) as Candidate_id
FROM
  test
GROUP BY
  marks;
END &&
DELIMITER ;
```

Procedure holding Query to rank marks in proper order -
The query is a SQL procedure named "rank_in_order". The procedure holds a query that calculates the rank of each candidate based on their marks. The query performs the following action upon execution :

1. The query selects three columns: "Marks", "Rank", and "Candidate_id".
2. The dense_rank() calculates the rank of each candidate based on their marks.
3. The query groups the data by marks, so that the same marks are grouped together.
4. The "GROUP_CONCAT()" function concatenates the candidate IDs for each group.

Q4.

```
DELIMITER &&
CREATE PROCEDURE Delete_rows_with_same_value_except_smallest_id ()
BEGIN
DELETE FROM
  mailids
WHERE
  candidate_id in (
    Select
      tempcandidate_id
    from
      (
        select
          Distinct a.candidate_id as tempcandidate_id
        from
          mailids a
          inner join mailids b
        where
```

```
        a.mail = b.mail
        and a.candidate_id > b.candidate_id
    ) as c
);
```

Procedure holding Query to keep the value that has smallest id and delete all the other rows having same value -
The query below is a stored procedure named "Delete_rows_with_same_value_except_smallest_id". The query performs the below actions upon execution

1. It deletes all rows from the "mailids" table where the candidate IDs have duplicate email addresses, and the candidate IDs are greater than the smallest candidate IDs with the same email addresses.
2. The inner join operation in the subquery is used to find all the duplicates in the "mail" column of the "mailids" table. The subquery returns the candidate IDs with duplicate email addresses, and the DELETE statement deletes all these rows except the ones with the smallest candidate IDs.
3. It displays the remaining rows in the "mailids" table, ordered by the candidate IDs in descending order.

```
-- Calling procedures
-- Question - 1
CALL get_num_of_male_and_female ();
-- Question - 2
CALL max_amount_from_rows_with_month_name();
-- Question - 3
CALL rank_in_order();
-- Question - 4
CALL Delete_rows_with_same_value_except_smallest_id ();
```

Rithish Punna's Approach:

Rithish has also written SQL code inside procedure for each question and called the procedure to answer the query

Q1.

```
CREATE PROCEDURE get_num_of_male_and_female ()
    -> BEGIN
    -> SELECT
    ->   IFNULL(Department, 'Not Assigned') as Department,
    ->   COUNT(
    ->     CASE WHEN UPPER(Gender)= 'MALE' THEN 1 END
    ->   ) AS 'Num of Male',
    ->   COUNT(
    ->     CASE WHEN UPPER(Gender)= 'FEMALE' THEN 1 END
    ->   ) AS 'Num of Female'
    -> FROM
    ->   employees
    -> GROUP BY
    ->   Department
    -> order by
    ->   Department;
    -> END &&
```

The query is a SQL stored procedure. The procedure is named "get_num_of_male_and_female". The procedure performs the following actions upon execution :

- Selects data from the "employees" table.
- Groups the data by the "Department" column.
- Counts the number of occurrences of "Male" and "Female" in the "Gender" column and returns the results as two separate columns named "Num of Male" and "Num of Female".
- Returns the department name along with the count of male and female employees in each department, with a default value of "Not Assigned" for departments that have no assigned employees.
- The results are ordered by the "Department" column in ascending order.

Q2.

```
CREATE PROCEDURE max_amount_from_rows_with_month_name()
    -> BEGIN
    -> select
    ->   emp_name as Name,
    ->   value,
    ->   case when idx = 1 then 'Jan' when idx = 2 then 'Feb' when idx = 3 then
'Mar' end as Month
    -> from
    ->   (
    ->     select
```

```
    ->       emp_name,
    ->       greatest(Jan, Feb, March) as value,
    ->       field(
    ->         greatest(Jan, Feb, March),
    ->         Jan,
    ->         Feb,
    ->         March
    ->       ) as idx
    ->     from
    ->       employeesalaries
    ->   ) emps;
    -> END &&
```

The above procedure selects three columns, name of the employee, value to store the largest salary , and month. The greatest function is used to get the highest salary per employee in the columns in Jan, Feb and March. the feild function is used to get the month where the highest salary exists

Q3.

```
DELIMITER &&
mysql> CREATE PROCEDURE rank_in_order ()
    -> BEGIN
    -> SELECT
    ->   Marks,
    ->   dense_rank() OVER (
    ->     order by
    ->       marks desc
    ->   ) as 'Rank',
    ->   GROUP_CONCAT(Candidate_id) as Candidate_id
    -> FROM
    ->   test
    -> GROUP BY
    ->   marks;
    -> END &&
DELIMITER ;
```

The above procedure selects the three columns Marks, Rank and candidate id the dense rank function calculates the rank of the students on the marks achieved the ranks are grouped together so that two students who score the same marks are given the same rank instead of different ranks.

Q4.

```
DELIMITER &&
CREATE PROCEDURE Delete_rows_with_same_value_except_smallest_id ()
BEGIN
DELETE FROM
  mailids
WHERE
  candidate_id in (
    Select
      tempcandidate_id
    from
      (
        select
          Distinct a.candidate_id as tempcandidate_id
        from
          mailids a
          inner join mailids b
        where
          a.mail = b.mail
          and a.candidate_id > b.candidate_id
      ) as c
  );
```

The inner Join query is a subquery that returns all the duplicate elements in the table in the mailids and the delete statement deletes these records but the records with the minimum candidate_id.