**SQL**

**SQL SERVER (preferably)**

Graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated

Take a look at the table structures and write down the following queries:

1. ALL the employees and their salaries, even if there is no data for the salary for the employee

Select ED.FullName, ES.Salary

From EmployeeDetails as ED

FULL OUTER JOIN EmployeeSalary as ES

ON ES.EmpId= ED.EmpId;

1. All employees that are also the managers

Select ED.FullName

From EmployeeDetails as ED

Where ED.ManagerId IS NOT NULL;

1. Managers and the projects they are involved

Select ED.FullName, ES.Project

From EmployeeDetails as ED

INNER JOIN EmployeeSalary as ES

ON ES.EmpId= ED.EmpId

Where ED.ManagerId IS NOT NULL;

1. Second highest paid employee by the each city

With Temptable

As

{

Select

Row\_Number() Over (partition by ED.City order by ES.Salary desc) as A, ED.FullName as Name, ES.Salary)

From EmployeeDetails as ED

INNER JOIN EmployeeSalary as ES

ON ES.EmpId= ED.EmpId

}

Select Name, Salary

From Temptable

Where A = 2

Order by Name

2.2

Table TABLE has duplicate rows in the production environment. It does not contain Primary Key , but from the business side it is expected to be an unique combination of all columns for each row. Describe how would you execute this in order to remove duplicates.

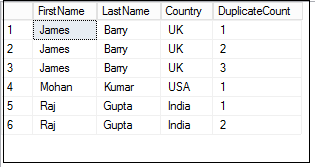
The duplicate rows can be removed by using **ROW\_NUMBER** function that’s adds a unique sequential row number for the row. It partitions the data using the **PARTITION BY** clause. In our case, it can be done by using **PARTITION BY** clause over column(s) that are expected to have unique values in it and create new column of DuplicateCount. **ROW\_NUMBER** function will do numbering for every duplicate row value. Now we can simply filter our distinct rows by using WHERE DuplicateCount = 1.  
  
 **CODE:**

ROW\_NUMBER() OVER

(  
PARTITION BY  
Unique Column(s)  
)

AS DuplicateCount

**SAMPLE OUTPUT:**



3.3

The following table is defined:

Transaction Item TITM, with columns:

* Datatype (beginning of the row for TITM, string)
* TransactionID, string
* ProdcutID, string
* ProductName, string
* SoldPrice, numerical column

Write CREATE TABLE statement for the following table.

Create table TITM

(

TransactionID VARCHAR(40),

ProdcutID VARCHAR(40),

ProductName VARCHAR(40),

SoldPrice INT);

How would you additionally optimize queries knowing that join is a common used function or where condition and ProductID column?

In a query where common used function is Join & Where can be used, it is recommended to use **INNER JOIN** instead of OUTER JOIN & WHERE clause because of its limited option to the optimize database which typically results in slower SQL execution.