1.) Write a SQL query to create the tables in your database and insert the data into these tables.

```
create database Final_Assessment;
use Final_Assessment;
```

a.) Add the primary key “Emp_ID” to the Employees Table:

```
CREATE table EmployeesTable (
        EMP_ID int NOT NULL,
        FIRST_NAME varchar(200),
        LAST_NAME varchar(200),
        SALARY float,
        JOINING_DATE timestamp,
        DEPARTMENT varchar(200),
        PRIMARY KEY (EMP_ID)
);
```

b.) Add foreign key “EMP_REF_ID” in Variables Details and Designation Table that references “Emp_ID” in the Employees Table.

```
CREATE table VariablesDetails(
        EMP_REF_ID int,
        VARIABLES_DATE timestamp,
        VARIABLES_AMOUNT float,
        FOREIGN KEY (EMP_REF_ID) REFERENCES EmployeesTable (EMP_ID)
);

CREATE table DesignationTable(
        EMP_REF_ID int,
        EMP_TITLE varchar(200),
        AFFECTED_FROM timestamp,
        FOREIGN KEY (EMP_REF_ID) REFERENCES EmployeesTable (EMP_ID)
);

#INSERTING VALUES IN TABLES
INSERT INTO EmployeesTable (
        EMP_ID,
        FIRST_NAME,
        LAST_NAME,
        SALARY,
        JOINING_DATE,
        DEPARTMENT
)
VALUES
        (001, 'Manish', 'Agrawal', 700000, '2019-04-20 09:00:00', 'HR'),
        (002, 'Niranjan', 'Bose', 20000, '2019-02-11 09:00:00', 'DA'),
        (003, 'Vivek', 'Singh', 100000, '2019-01-20 09:00:00', 'DA'),
        (004, 'Asutosh', ‘Kapoor’, 700000, '2019-03-20 09:00:00', 'HR'),
        (005, 'Vihaan', 'Banerjee', 300000, '2019-06-11 09:00:00', 'DA'),
        (006, 'Atul', 'Diwedi', 400000, '2019-05-11 09:00:00', 'Account'),
        (007, 'Satyendra', ‘Tripathi’, 95000, '2019-03-20 09:00:00', 'Account'),
        (008, 'Pritika', 'Bhatt', 80000, '2019-02-11 09:00:00', 'DA');
```

```
INSERT INTO VariablesDetails (
        EMP_REF_ID,
        VARIABLES_DATE,
        VARIABLES_AMOUNT
)
VALUES
        (1, '2019-02-20 00:00:00', 15000),
        (2, '2019-06-11 00:00:00', 30000),
        (3, '2019-02-20 00:00:00', 42000),
        (4, '2019-02-20 00:00:00', 14500),
        (5, '2019-06-11 00:00:00', 23500);


INSERT INTO DesignationTable (
        EMP_REF_ID,
        EMP_TITLE,
        AFFECTED_FROM
)
VALUES
        (1, 'Asst. Manager', '2019-02-20 00:00:00'),
        (2, 'Senior Analyst', '2019-01-11 00:00:00'),
        (8, 'Senior Analyst' , '2019-04-06 00:00:00'),
        (5, 'Manager', '2019-10-06 00:00:00'),
        (4, 'Asst. Manager' , '2019-12-06 00:00:00'),
        (7, 'Team Lead' , '2019-06-06 00:00:00'),
        (6, 'Team Lead' , '2019-09-06 00:00:00'),
        (3, 'Senior Analyst', '2019-08-06 00:00:00');
```

2.) Name the four different types of joins? Give examples of each by performing all the joins on the Employees table and Designation Table

i) INNER JOIN: INNER JOINs are used to retrieve **only common matching records**. This join is like an intersection in mathematics.
Example:

```
SELECT e.EMP_ID, d.EMP_TITLE
   FROM EmployeesTable e
      INNER JOIN DesignationTable d ON e.EMP_ID = d.EMP_REF_ID;
```

ii) LEFT JOIN: LEFT JOIN retrieves all records from Table A, along with those records from Table B for which the join condition is met. For the records from Table A that do not match the condition, the NULL values are displayed.
Example:

```
SELECT e.EMP_ID, d.EMP_TITLE
   FROM EmployeesTable e
      LEFT JOIN DesignationTable d ON e.EMP_ID = d.EMP_REF_ID
   ORDER BY e.EMP_ID;
```

iii) RIGHT JOIN: Accordingly, RIGHT JOINs allow retrieving all records from Table B, along with those records from Table A for which the join condition is met. For the records from Table B that do not match the condition, the NULL values are displayed.
Example:

```
SELECT  e.FIRST_NAME, d.EMP_REF_ID
   FROM EmployeesTable e
      RIGHT JOIN DesignationTable d ON d.EMP_REF_ID = e.EMP_ID
   ORDER BY d.EMP_REF_ID;
```

iv) CROSS JOIN: The CROSS JOIN keyword returns all records from both tables (table1 and table2).This is like a union in mathematics.
Example:

```
SELECT * FROM EmployeesTable e
        CROSS JOIN DesignationTable d;
```

a.) Write a query to get the employee details(columns - full name and department) of those who received the highest and the least variables:

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT
   FROM EmployeesTable
   JOIN VariablesDetails ON EmployeesTable.EMP_ID = VariablesDetails.EMP_REF_ID
   WHERE VariablesDetails.VARIABLES_AMOUNT = (
      SELECT MAX(VARIABLES_AMOUNT) FROM VariablesDetails) OR VariablesDetails.VARIABLES_AMOUNT = (
      SELECT MIN(VARIABLES_AMOUNT) FROM VariablesDetails);
```

b.) Write a query to get the designation which has got the highest and second lowest amount (salary + variables) for the whole year of 2019. Get the corresponding amount values.

```
with T as (SELECT d.EMP_TITLE, ifnull(v.VARIABLES_AMOUNT,0)+e.SALARY as total_salary, d.emp_ref_ID
        FROM DesignationTable d
                LEFT JOIN EmployeesTable e
                        ON d.EMP_REF_ID = e.EMP_ID
                LEFT JOIN VariablesDetails v
                        ON v.EMP_REF_ID = e.EMP_ID)
SELECT T.EMP_TITLE, T.total_salary FROM T
        WHERE total_salary = (
                SELECT max(total_salary) FROM T) OR total_salary = (
      SELECT total_salary from T ORDER BY total_salary limit 1,1);
```

NOTE: In the above question, left join is performed because there exist some employees whose names are not mentioned in the VariablesDetails table, considering their variables as 0, I have fetched the employees having maximum (salary+varibales) and second minimum (salary+variables).

c.) What is cross join? Write a query to give an example of the same by performing it on the Employees table and Designation table.

The CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN.This kind of result is called a Cartesian Product. It returns all matching records from both tables whether the other table matches or not. So, if there are rows in "EmployeesTable" that do not have matches in "DesignationTable", or if there are rows in "DesignationTable" that do not have matches in "EmployeesTable", those rows will be listed as well.

Example:

```
SELECT * FROM EmployeesTable e
        CROSS JOIN DesignationTable d;
```

d.) What are the clauses used with Select statements and what are the preference orders of it?

- SELECT - Specifies which values are to be returned
- FROM - Specifies the source tables and views from which data is to be read
- WHERE - Specifies criteria that restrict the contents of the results table.
- GROUP BY - Groups the selected rows based on identical values in a column or expression.
- HAVING - Hilters the results of the GROUP BY clause by using an aggregate function.
- ORDER BY - Allows you to specify the columns on which the results table is to be sorted.
- OFFSET - Used to return a subset of rows from a result set.
- UNION - Combines the results of SELECT statements into a single result table.
- INTERSECT - Takes the results of two SELECT statements and returns only rows that appear in both result tables.
- EXCEPT - Takes the results of two SELECT statements and returns the rows of the first result table that do not appear in the second result table.

3. What is the stored procedure?

The stored procedure is similar to a function in any programming language, The stored procedure is SQL statements wrapped within the “CREATE PROCEDURE” statement. The stored procedure may contain a conditional statement like IF or CASE or the Loops. The stored procedure can also execute another stored procedure or a function that modularizes the code.

Benefits of Stored Procedure:

1. **Reduce the Network Traffic**: Multiple SQL Statements are encapsulated in a stored procedure. When you execute it, instead of sending multiple queries, we are sending only the name and the parameters of the stored procedure
2. **Easy to maintain**: The stored procedures are reusable. We can implement the business logic within an Stored Procedure, and it can be used by applications multiple times, or different modules of an application can use the same procedure
3. **Secure**: The stored procedures are more secure than the AdHoc queries. The permission can be granted to the user to execute the stored procedure without giving permission to the tables used in the stored procedure. The stored procedure helps to prevent the database from SQL Injection

a.) Write a query to get the employee details who got their designations updated in the second half of the year 2019(July to December), sorted by the “variables_amount” (highest to lowest).

```
SELECT * FROM EmployeesTable e
        LEFT JOIN DesignationTable d
                ON e.EMP_ID = d.EMP_REF_ID
        LEFT JOIN VariablesDetails v
                ON e.EMP_ID = v.EMP_REF_ID
        WHERE d.AFFECTED_FROM BETWEEN '2019-07-01 00:00:00' AND '2019-12-31 00:00:00'
   ORDER BY v.VARIABLES_AMOUNT DESC;
```

b.) Write a stored procedure to call the query that you have written for Q2.a

```
DELIMITER //

CREATE PROCEDURE highest_least_variables()
BEGIN
   SELECT FIRST_NAME, LAST_NAME, DEPARTMENT
   FROM EmployeesTable
   JOIN VariablesDetails ON EmployeesTable.EMP_ID = VariablesDetails.EMP_REF_ID
   WHERE VariablesDetails.VARIABLES_AMOUNT = (
      SELECT MAX(VARIABLES_AMOUNT) FROM VariablesDetails) OR VariablesDetails.VARIABLES_AMOUNT = (
      SELECT MIN(VARIABLES_AMOUNT) FROM VariablesDetails);
END //

DELIMITER ;

#Running the created stored procedure
call highest_least_variables
```