Sri Aurobindo Institute of Technology

CLASS WORK
SESSIONAL WORK

ASSIGNMENT NO. :- 01
EXPERIMENT NO. :- 01

SUBMITTED ON :- 08/11/2024

MARKS OR GRADE OBTAINED :- _ _ _

NAME :- Ganesh Sen

ROLL NO. :- 0873CS221041

CLASS :- 3rd Year/5th Sem

DEPATMENT :- Computer Science & Engineering

SUBJECT :- DBMS LAB

CODE NO. :-CS-502

Signature of Student

Signature of Professor

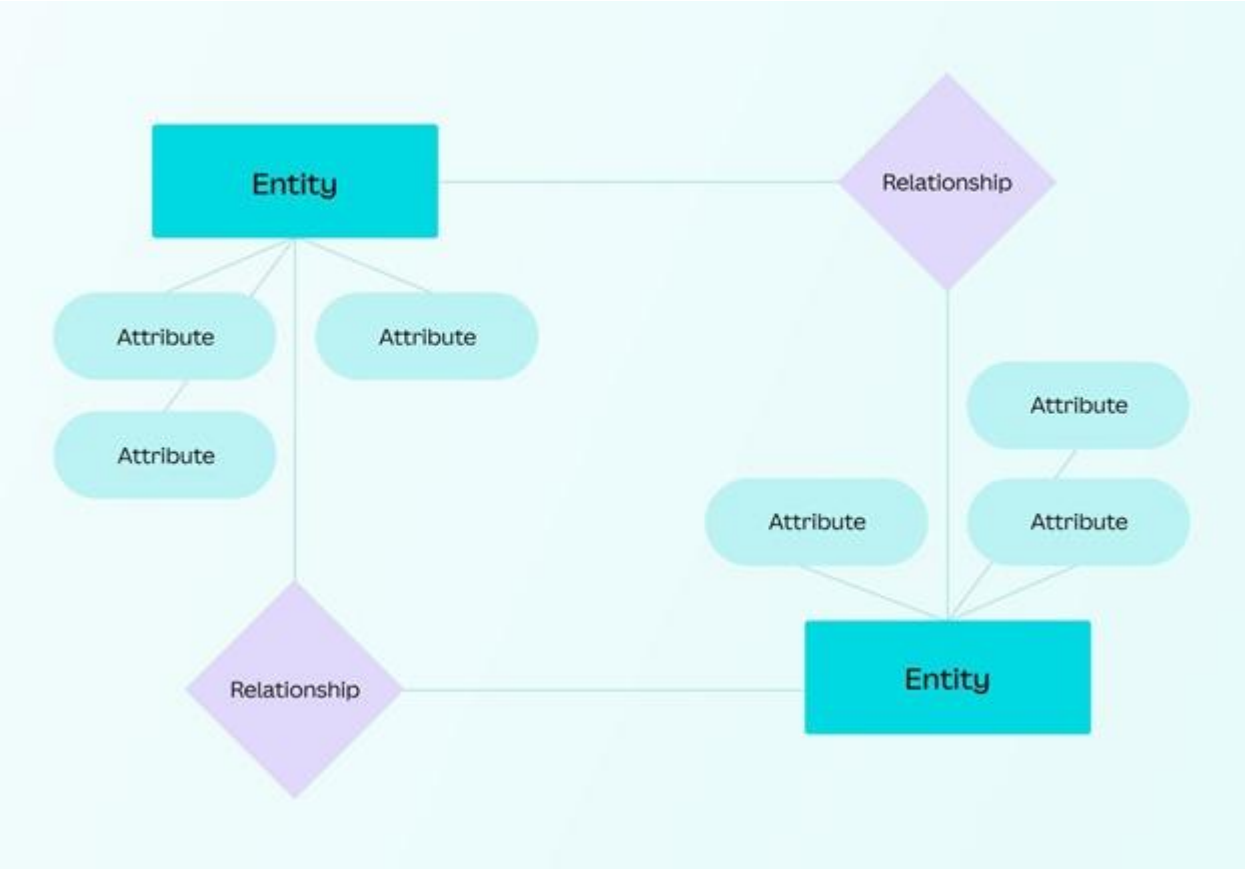
Q.1 Explain ER Diagram with diagram.

=> An *Entity-Relationship Diagram (ERD)* is a visual tool used to represent the structure of a database. It shows how different pieces of data (entities) are connected and related to each other.

Main Components:
1. *Entities: These are objects or things in the system, like **Customer, **Product, or **Order*.
2. *Attributes: These describe the properties of an entity, like a **Customer's* *Name* or *Address*.
3. *Relationships: These show how entities are related. For example, a **Customer* places an *Order*.
4. *Primary Key: A unique identifier for each entity, like **CustomerID*.

Types of Relationships:
1. *One-to-One (1:1): One entity is related to one other entity. Example: A **Person* has one *Passport*.
2. *One-to-Many (1:N): One entity is related to many others. Example: A **Department* has many *Employees*.
3. *Many-to-One (M:1): Many entities are related to one. Example: Many **Orders* can be placed by one *Customer*.
4. *Many-to-Many (M:N): Many entities are related to many others. Example: A **Student* can enroll in many *Courses*.

ERDs help to design databases by clearly showing how data elements are connected, making it easier to organize and manage data.



This image is a simple entity-relationship diagram (ERD). It illustrates:

- *Entities* (rectangles) that represent objects or concepts in the database (e.g., a "Person" or "Product").

- *Attributes* (oval shapes connected to entities) that define specific properties or characteristics of an entity (e.g., a person’s "Name" or "Age").

- *Relationships* (diamonds) that connect entities to show how they are related (e.g., "Works for" or "Buys").

This ERD visually organizes data by showing entities, their attributes, and the relationships between them.

Q.2 Study about RDBMS & Its Application, & installation of MySQL.

=> Relational Database Management System (RDBMS):-An **RDBMS** is a database management system that organizes data into tables (also known as relations) with rows and columns. This structure allows for efficient data retrieval, storage, and management using **Structured Query Language (SQL)**.

Key Features of RDBMS:-

1. **Tables and Schema**: Data is stored in tables with a defined schema. Tables are structured with rows (records) and columns (attributes).
2. **Relationships**: Data in different tables can be connected or related. These relationships are managed by using **foreign keys** and **primary keys**.
3. **Data Integrity**: Maintains data accuracy and consistency through constraints like primary keys, foreign keys, unique keys, etc.
4. **ACID Compliance**: RDBMS supports **Atomicity**, **Consistency**, **Isolation**, and **Durability** to ensure transaction reliability.
5. **SQL Support**: SQL is used to query, update, delete, and manage data in the RDBMS.

Applications of RDBMS:- RDBMS systems are commonly used in various applications due to their reliable data management capabilities:

1. **Banking Systems**: Store customer information, transaction records, and account details.
2. **E-commerce**: Manage product catalogs, customer orders, inventory, and user profiles
3. **Healthcare**: Store patient records, appointment schedules, and treatment details.
4. **Social Media**: Handle user data, friend connections, messages, and activity logs.
5. **Telecommunications**: Manage customer data, billing information, and call records.

Installing MySQL:-MySQL is a popular open-source RDBMS that is widely used for web applications and data management.

Step 1: Download MySQL

1. Go to the official MySQL website: MySQL Downloads.
2. Choose the MySQL Community Server version compatible with your operating system (Windows, macOS, Linux).
3. Download the installer file.

Step 2: Install MySQL

1. Open the downloaded installer file.
2. Follow the setup wizard:
    1. Choose the **Setup Type** (Developer Default, Server Only, Full, or Custom).
    2. Select the installation path.
3. **Configure MySQL Server**:
    1. Choose the **Server Configuration** options (Standalone, InnoDB cluster, or Replica).
    2. Set the **Root Password** and optionally add other users.
    3. Select the **Port** (default is 3306).
4. **Configure MySQL as a Windows Service** (on Windows) to allow MySQL to run in the background.
5. Finish the installation by clicking **Execute** to apply the configurations.

Step 3: Verify MySQL Installation

- Open a **command prompt** or **terminal**.
- Type the following command to start the MySQL command-line tool and log in

❖ **mysql -u root -p**

- Enter the root password you set during installation. If you see the MySQL prompt (mysql>), the installation was successful.

You're now ready to create databases, tables, and manage data using MySQL!

Q.3 Study about SQL Commands:-

> Create Table
> Insert Value in Table
> Drop Table
> Alter Table
> Truncate Table

=>

> **Create Table :-**The CREATE TABLE command is used to define a new table and its columns.

```
>       CREATE TABLE Employees (
        EmployeeID INT PRIMARY KEY,
        FirstName VARCHAR(50),
        LastName VARCHAR(50),
        Age INT,
        Salary DECIMAL(10, 2)
      );
```

```
mysql> use office;
Database changed
mysql> CREATE TABLE Employees (
    ->     EmployeeID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Age INT,
    ->     Salary DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

> Insert Value in Table:-The INSERT INTO command adds new rows (records) into a table.

```
>         INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Salary)
          VALUES (1, 'John', 'Doe', 30, 50000.00);
```

```
ysql> INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Salary)
   -> VALUES (1, 'John', 'Doe', 30, 50000.00);
uery OK, 1 row affected (0.00 sec)

ysql> INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Salary)
   -> values (2, '
   '> values (2, 'Jay', 'Shah', '25','60000.00);
RROR 1064 (42000): You have an error in your SQL syntax; check the manual
for the right syntax to use near 'Jay', 'Shah', '25','60000.00)' at line 2
ysql> values (2, 'Jay', 'Shah', '25','60000.00');
RROR 1064 (42000): You have an error in your SQL syntax; check the manual
for the right syntax to use near 'values (2, 'Jay', 'Shah', '25','60000.00
ysql> values (2, 'Jay', 'Shah', 25, 60000.00);
RROR 1064 (42000): You have an error in your SQL syntax; check the manual
for the right syntax to use near 'values (2, 'Jay', 'Shah', 25, 60000.00)'
ysql> INSERT INTO Employees (EmployeeID, FirstName, LastName, Age, Salary)
   ->  values (2, 'Jay', 'Shah', 25, 60000.00);
uery OK, 1 row affected (0.00 sec)

ysql> Select * from Employees;
+------------+-----------+----------+------+----------+
| EmployeeID | FirstName | LastName | Age  | Salary   |
+------------+-----------+----------+------+----------+
|          1 | John      | Doe      |   30 | 50000.00 |
|          2 | Jay       | Shah     |   25 | 60000.00 |
+------------+-----------+----------+------+----------+
```

> Drop Table :- The DROP TABLE command permanently deletes a table and all its data from the database.
>DROP TABLE Employees;

```
mysql> SHOW TABLES LIKE '%Employees'%';
    '>
```

> Alter Table:- Add a new column:

```
ALTER TABLE Employees
ADD Address Varchar(255);
```

```
mysql> ALTER TABLE Employees
    -> ADD Address Varchar(255);
Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> Select * from Emplyees;
ERROR 1146 (42S02): Table 'office.emplyees' doesn't exist
mysql> Select * from Employees;
+------------+-----------+----------+------+----------+---------+
| EmployeeID | FirstName | LastName | Age  | Salary   | Address |
+------------+-----------+----------+------+----------+---------+
|          1 | John      | Doe      |   30 | 50000.00 | NULL    |
+------------+-----------+----------+------+----------+---------+
1 row in set (0.00 sec)
```

Modify an existing column:-

```
ALTER TABLE Employees
MODIFY COLUMN Salary numeric;
```

```
mysql> ALTER TABLE table_name
    -> MODIFY COLUMN column_name new_datatype;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
 for the right syntax to use near 'new_datatype' at line 2
mysql> ALTER TABLE Employee
    -> ALTER TABLE Employees
    -> MODIFY COLUMN Salary int;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
 for the right syntax to use near 'TABLE Employees
MODIFY COLUMN Salary int' at line 2
mysql> ALTER TABLE Employees
    -> MODIFY COLUMN Salary numeric;
Query OK, 1 row affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> Select * from Employees;
+------------+-----------+----------+------+--------+---------+
| EmployeeID | FirstName | LastName | Age  | Salary | Address |
+------------+-----------+----------+------+--------+---------+
|          1 | John      | Doe      |   30 |  50000 | NULL    |
+------------+-----------+----------+------+--------+---------+
```

Delete a column:-

```
ALTER TABLE Employees
DROP COLUMN Address;
```

```
mysql> ALTER TABLE table_name
    -> DROP COLUMN column_name;
ERROR 1146 (42S02): Table 'office.table_name' doesn't exist
mysql> ALTER TABLE Employees
    -> DROP COLUMN Address;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> Select * from Employees;
+------------+-----------+----------+------+--------+
| EmployeeID | FirstName | LastName | Age  | Salary |
+------------+-----------+----------+------+--------+
|          1 | John      | Doe      |   30 |  50000 |
+------------+-----------+----------+------+--------+
```

> Truncate Table:-

```
TRUNCATE TABLE Employees;
```

```
mysql> TRUNCATE TABLE Employees;
Query OK, 0 rows affected (0.00 sec)

mysql> Select * from Employees;
Empty set (0.00 sec)
```

Q.4 – Use select command for following structure :-

1. **Basic select command:-**The SELECT statement in SQL is used to retrieve data from one or more tables in a database. It allows you to specify which columns to retrieve and can be used with various clauses to filter, sort, and organize the result set.

Here's the basic command syntax :-

```
mysql> select * from employee;
+------------+-----------+----------+------+------------+
| EmployeeID | FirstName | LastName | Age  | HireDate   |
+------------+-----------+----------+------+------------+
|          1 | John      | Doe      |   30 | 2024-01-15 |
|          2 | Jane      | Smith    |   28 | 2023-05-10 |
|          3 | Alice     | Johnson  |   35 | 2022-09-30 |
|          4 | Bob       | Lee      |   40 | 2021-03-22 |
+------------+-----------+----------+------+------------+
4 rows in set (0.00 sec)
```

2. Using Where clause:-To retrieve data from the Employee table using the WHERE clause, you can filter the rows based on specific conditions. The WHERE clause allows you to specify criteria for the data you want to retrieve.

Here's the basic command syntax :-

```
mysql> SELECT * FROM Employee WHERE Age > 30;
+------------+-----------+----------+------+------------+
| EmployeeID | FirstName | LastName | Age  | HireDate   |
+------------+-----------+----------+------+------------+
|          3 | Alice     | Johnson  |   35 | 2022-09-30 |
|          4 | Bob       | Lee      |   40 | 2021-03-22 |
+------------+-----------+----------+------+------------+
2 rows in set (0.03 sec)
```

Q.5 Study about order by , group by, having by:-

=> ORDER BY Clause in SQL :- The ORDER BY clause is used to sort the result set of a SELECT query based on one or more columns. You can sort the results in ascending (default) or descending order.

Here's the basic command syntax :-

```
mysql> SELECT * FROM Employee ORDER BY Age;
+------------+-----------+----------+------+------------+
| EmployeeID | FirstName | LastName | Age  | HireDate   |
+------------+-----------+----------+------+------------+
|          2 | Jane      | Smith    |   28 | 2023-05-10 |
|          1 | John      | Doe      |   30 | 2024-01-15 |
|          3 | Alice     | Johnson  |   35 | 2022-09-30 |
|          4 | Bob       | Lee      |   40 | 2021-03-22 |
+------------+-----------+----------+------+------------+
4 rows in set (0.00 sec)
```

GROUP BY Clause in SQL :- The GROUP BY clause is used to **group rows** that have the same values in specified columns into **summary rows**, like calculating totals, averages, or counts for each group. This is often used with **aggregate functions** like COUNT(), SUM(), AVG(), MAX(), and MIN() to perform calculations on each group of data.

Here's the basic command syntax :-

```
mysql> SELECT Department, COUNT(*) AS NumEmployees
    -> FROM Employee
    -> GROUP BY Department;
+------------+--------------+
| Department | NumEmployees |
+------------+--------------+
| HR         |            3 |
| IT         |            3 |
+------------+--------------+
2 rows in set (0.00 sec)
```

HAVING Clause in SQL :- The HAVING clause is used in SQL to filter records after grouping the data with the GROUP BY clause, typically used with aggregate functions like COUNT(), SUM(), AVG(), etc.

Here's the basic command syntax :-

```
mysql> SELECT Department, COUNT(*) AS NumEmployees
    -> FROM Employee
    -> GROUP BY Department
    -> HAVING COUNT(*) > 2;
+------------+--------------+
| Department | NumEmployees |
+------------+--------------+
| HR         |            3 |
| IT         |            3 |
+------------+--------------+
2 rows in set (0.00 sec)
```

Q.6 Difference between different DCL commands:-

=> Here’s a comparison of the DCL (Data Control Language) commands GRANT and REVOKE:

Command	Purpose	Used To	Example	Effect
GRANT	Grants specific privileges to a user or role on a database object.	- Assign privileges like SELECT, INSERT, UPDATE, DELETE, etc., to users or roles. - Provide access to specific tables, views, or other objects.	GRANT SELECT, INSERT ON Employee TO John; (This grants John the ability to SELECT and INSERT data into the Employee table.)	- The specified user/role gains the granted privileges on the given object. - Can be granted with the WITH GRANT OPTION to allow the user to grant those privileges to others.
REVOKE	Removes specific privileges that were previously granted.	- Revoke privileges from users or roles. - Prevents users from performing certain actions (like selecting, inserting, etc.) on database objects.	REVOKE INSERT ON Employee FROM John; (This revokes John's ability to insert data into the Employee table.)	- The specified user/role loses the revoked privileges. - If the user/role no longer has any privileges, they cannot perform any operations on the object.

Here’s the basic GRANT command syntax :-

```
mysql> GRANT SELECT, INSERT ON Employee TO John;
Query OK, 0 rows affected (0.01 sec)
```

Here’s the basic REVOKE command syntax :-

```
mysql> REVOKE INSERT ON Employee FROM John;
Query OK, 0 rows affected (0.00 sec)
```

Q.7 Study and use join command:-

=> In SQL, the JOIN command is used to combine rows from two or more tables based on a related column between them. There are several types of joins: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, and SELF JOIN.

Types of JOINs:-

1. INNER JOIN: Returns rows when there is a match in both tables.
2. LEFT JOIN (or LEFT OUTER JOIN): Returns all rows from the left table and matched rows from the right table. If there's no match, NULL values will be returned for columns from the right table.
3. RIGHT JOIN (or RIGHT OUTER JOIN): Returns all rows from the right table and matched rows from the left table. If no match exists, NULL values will be returned for columns from the left table.
4. FULL JOIN (or FULL OUTER JOIN): Returns rows when there is a match in one of the tables. If no match exists, NULL values will be returned for non-matching rows in either table.
5. SELF JOIN: Joins a table with itself, used when a table has hierarchical data or when you need to compare rows within the same table.

Here's the basic INNER JOIN command syntax :-

```
mysql> SELECT e.EmployeeID, e.FirstName AS EmployeeFirstName, e.LastName AS EmployeeLastName,
    ->        e1.FirstName AS Employee1FirstName, e1.LastName AS Employee1LastName
    -> FROM Employee e
    -> INNER JOIN Employee1 e1 ON e.EmployeeID = e1.EmployeeID;
+------------+-------------------+------------------+--------------------+-------------------+
| EmployeeID | EmployeeFirstName | EmployeeLastName | Employee1FirstName | Employee1LastName |
+------------+-------------------+------------------+--------------------+-------------------+
|          1 | Alice             | Williams         | Grace              | King              |
|          2 | Bob               | Miller           | Hugh               | Stone             |
+------------+-------------------+------------------+--------------------+-------------------+
2 rows in set (0.00 sec)
```

Here's the basic LEFT JOIN command syntax :-

```
mysql> SELECT e.EmployeeID, e.FirstName AS EmployeeFirstName, e.LastName AS EmployeeLastName,
    ->        e1.FirstName AS Employee1FirstName, e1.LastName AS Employee1LastName
    -> FROM Employee e
    -> LEFT JOIN Employee1 e1 ON e.EmployeeID = e1.EmployeeID;
+------------+-------------------+------------------+--------------------+-------------------+
| EmployeeID | EmployeeFirstName | EmployeeLastName | Employee1FirstName | Employee1LastName |
+------------+-------------------+------------------+--------------------+-------------------+
|          1 | Alice             | Williams         | Grace              | King              |
|          2 | Bob               | Miller           | Hugh               | Stone             |
|          3 | Charlie           | Taylor           | NULL               | NULL              |
|          4 | David             | Anderson         | NULL               | NULL              |
|          5 | Eve               | Moore            | NULL               | NULL              |
|          6 | Frank             | Davis            | NULL               | NULL              |
+------------+-------------------+------------------+--------------------+-------------------+
```

Here's the basic RIGHT JOIN command syntax :-

```
mysql> SELECT e.EmployeeID, e.FirstName AS EmployeeFirstName, e.LastName AS EmployeeLastName,
    ->        e1.FirstName AS Employee1FirstName, e1.LastName AS Employee1LastName
    -> FROM Employee e
    -> RIGHT JOIN Employee1 e1 ON e.EmployeeID = e1.EmployeeID;
+------------+-------------------+------------------+--------------------+-------------------+
| EmployeeID | EmployeeFirstName | EmployeeLastName | Employee1FirstName | Employee1LastName |
+------------+-------------------+------------------+--------------------+-------------------+
|          1 | Alice             | Williams         | Grace              | King              |
|          2 | Bob               | Miller           | Hugh               | Stone             |
|       NULL | NULL              | NULL             | Irene              | Black             |
|       NULL | NULL              | NULL             | Jack               | White             |
|       NULL | NULL              | NULL             | Karen              | Green             |
+------------+-------------------+------------------+--------------------+-------------------+
5 rows in set (0.00 sec)
```

Here’s the basic FULL JOIN command syntax :-

```
mysql> SELECT e.EmployeeID AS Employee1_ID, e.FirstName AS Employee1_FirstName, e.LastName AS Employee1_LastName,
    ->        e1.EmployeeID AS Employee2_ID, e1.FirstName AS Employee2_FirstName, e1.LastName AS Employee2_LastName,
    ->        e.DepartmentID
    -> FROM Employee e
    -> LEFT JOIN Employee1 e1 ON e.DepartmentID = e1.DepartmentID
    ->
    -> UNION
    ->
    -> -- RIGHT JOIN: Get all records from Employee1 and matching records from Employee
    -> SELECT e.EmployeeID AS Employee1_ID, e.FirstName AS Employee1_FirstName, e.LastName AS Employee1_LastName,
    ->        e1.EmployeeID AS Employee2_ID, e1.FirstName AS Employee2_FirstName, e1.LastName AS Employee2_LastName,
    ->        e.DepartmentID
    -> FROM Employee e
    -> RIGHT JOIN Employee1 e1 ON e.DepartmentID = e1.DepartmentID;
+--------------+--------------------+--------------------+--------------+--------------------+--------------------+--------------+
| Employee1_ID | Employee1_FirstName | Employee1_LastName | Employee2_ID | Employee2_FirstName | Employee2_LastName | DepartmentID |
+--------------+--------------------+--------------------+--------------+--------------------+--------------------+--------------+
|            1 | Alice              | Williams           |            1 | Grace              | King               |            1 |
|            1 | Alice              | Williams           |            9 | Karen              | Green              |            1 |
|            2 | Bob                | Miller             |            2 | Hugh               | Stone              |            2 |
|            3 | Charlie            | Taylor             |            1 | Grace              | King               |            1 |
|            3 | Charlie            | Taylor             |            9 | Karen              | Green              |            1 |
|            4 | David              | Anderson           |            7 | Irene              | Black              |            3 |
|            5 | Eve                | Moore              |            2 | Hugh               | Stone              |            2 |
|            6 | Frank              | Davis              |            7 | Irene              | Black              |            3 |
|         NULL | NULL               | NULL               |            8 | Jack               | White              |         NULL |
+--------------+--------------------+--------------------+--------------+--------------------+--------------------+--------------+
9 rows in set (0.00 sec)
```

Here’s the basic SELF JOIN command syntax :-

```
mysql> SELECT e.EmployeeID AS Employee1_ID, e.FirstName AS Employee1_FirstName, e.LastName AS Employee1_LastName,
    ->        e1.EmployeeID AS Employee2_ID, e1.FirstName AS Employee2_FirstName, e1.LastName AS Employee2_LastName,
    ->        e.DepartmentID
    -> FROM Employee e
    -> JOIN Employee1 e1 ON e.DepartmentID = e1.DepartmentID
    -> WHERE e.EmployeeID < e1.EmployeeID;
+--------------+--------------------+--------------------+--------------+--------------------+--------------------+--------------+
| Employee1_ID | Employee1_FirstName | Employee1_LastName | Employee2_ID | Employee2_FirstName | Employee2_LastName | DepartmentID |
+--------------+--------------------+--------------------+--------------+--------------------+--------------------+--------------+
|            4 | David              | Anderson           |            7 | Irene              | Black              |            3 |
|            6 | Frank              | Davis              |            7 | Irene              | Black              |            3 |
|            1 | Alice              | Williams           |            9 | Karen              | Green              |            1 |
|            3 | Charlie            | Taylor             |            9 | Karen              | Green              |            1 |
+--------------+--------------------+--------------------+--------------+--------------------+--------------------+--------------+
4 rows in set (0.00 sec)
```