# EXPERIMENT-1

**Aim:**

Perform DDL commands.

**Input:**

**(**Commands used are **CREATE, ALTER, DROP, TRUNCATE and RENAME)**

CREATE TABLE Employee ( id INT PRIMARY KEY, name VARCHAR(50),

age INT

);

# OUTPUT:

# 

ALTER TABLE Employee ADD email VARCHAR(100); **OUTPUT:**



DROP TABLE Employee; TRUNCATE TABLE Employee;

RENAME TABLE Employees TO Employee29;

# EXPERIMENT-2

**Aim:**

Perform DML commands.

**Input:**

**(**Commands used are **CREATE TABLE , INSERT INTO, SELECT, WHERE, UPDATE, DELETE, COUNT, SUM** and **AVG)**

CREATE TABLE Students ( StudentID int,

FirstName varchar (100), LastName varchar(100), Marks int,

Course varchar(100)

);

INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course) VALUES ('1529', 'Barbie', 'Kumari', '100', 'BTech(CSE)');

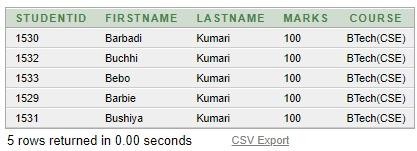
INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course) VALUES ('1530', 'Barbadi', 'Kumari', '100', 'BTech(CSE)');

INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course) VALUES ('1531', 'Bushiya', 'Kumari', '100', 'BTech(CSE)');

INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course) VALUES ('1532', 'Buchhi', 'Kumari', '100', 'BTech(CSE)');

INSERT INTO Students (StudentID, Firstname, Lastname, Marks, Course) VALUES ('1533', 'Bebo', 'Kumari', '100', 'BTech(CSE)');

# OUTPUT:



SELECT StudentID, Firstname, Course FROM Students WHERE StudentID = '1529';

### OUTPUT:

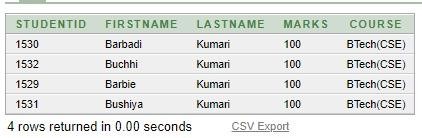


UPDATE Students SET Lastname= 'Jha' ,Marks = '20' WHERE StudentID= '1529'; SELECT \* FROM Students;

### A screenshot of a table Description automatically generatedOUTPUT:

DELETE FROM Students WHERE Firstname = 'Bebo';

### OUTPUT:



SELECT COUNT(Firstname) FROM Students

WHERE Lastname = 'Kumari'; SELECT AVG(Marks)

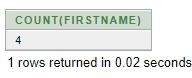
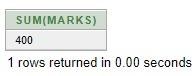
FROM Students

WHERE Course = 'BTech(CSE)';

SELECT SUM(Marks)

FROM Students

WHERE Lastname = 'Kumari';

**OUTPUT:**

# EXPERIMENT-3

**Aim:**

Perform SQL constraints commands.

#### Input:

#### (Commands used are CREATE TABLE, CONSTRAINTS, NOT NULL, PRIMARY KEY, FOREIGN KEY, ALTER, DROP and INSERT INTO)

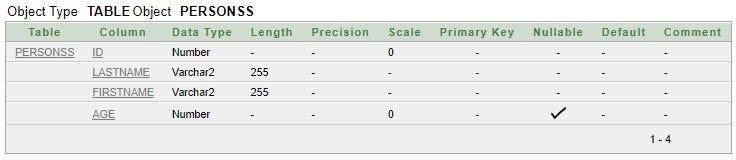
* CREATE TABLE Persons ( ID int NOT NULL,

LastName varchar(255) NOT NULL, FirstName varchar(255),

Age int, UNIQUE (ID)

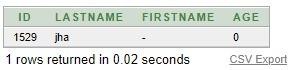
);

### OUTPUT:



* INSERT INTO Person1 VALUES('1529','Jha','',0);

### OUTPUT:



* SELECT \* FROM Person1;
* ALTER TABLE Person1

ADD CONSTRAINT UC\_Person1 UNIQUE (FirstName,LastName);

* ALTER TABLE Person1

DROP CONSTRAINT UC\_Person1;

* ALTER TABLE Persons ADD PRIMARY KEY (ID);
* CREATE TABLE Order1(

OrderID int NOT NULL, OrderNumber int NOT NULL, ID int,

PRIMARY KEY (OrderID), FOREIGN KEY (ID) REFERENCES Person1(ID)

);

* INSERT INTO Order1 VALUES('112', '23', '1529');

### OUTPUT:

#### EXPERIMENT-4

**Aim:**

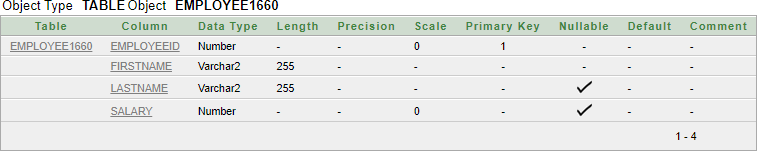
Perform different types of joins.

#### Input:

#### (Left Join. Right Join, Full Join, Cross Join, Inner Join, Left Outer Join, Right Outer Join, Full Outer Join)

* CREATE TABLE Employee1660( EmployeeID int NOT NULL PRIMARY KEY, FirstName varchar(255) NOT NULL, LastName varchar(255),

Salary int

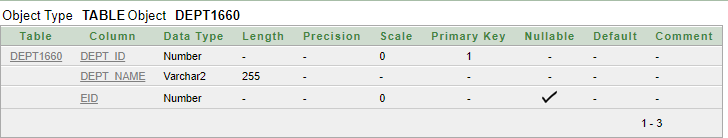
);

* CREATE TABLE Dept1660(

Dept\_ID int NOT NULL PRIMARY KEY,

Dept\_Name varchar(255) NOT NULL, EID int

);



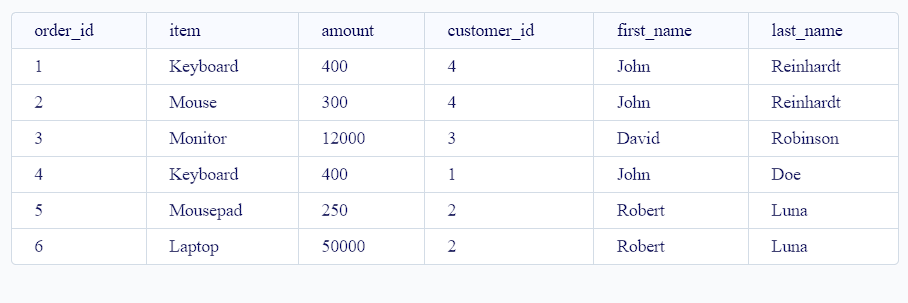
* INSERT INTO Orders(order\_id, item, amount, customer\_id) VALUES(6, "Laptop","50000", "2");

A table with numbers and text

Description automatically generated

* INSERT INTO Dept1660(Dept\_Id, Dept\_Name, EID) VALUES(32, 'Finance', 1);

.....

* **Left Outer Join**
* SELECT e.order\_id, e.item, e.amount, e.customer\_id, d.first\_name,d.last\_name FROM Orders e LEFT OUTER JOIN Customers d ON e.customer\_id = d.customer\_id;
* **Right Outer Join**

SELECT e.order\_id, e.item, e.amount, e.customer\_id, d.first\_name,d.last\_name FROM Orders e RIGHT OUTER JOIN Customers d ON e.customer\_id = d.customer\_id;

A screenshot of a computer

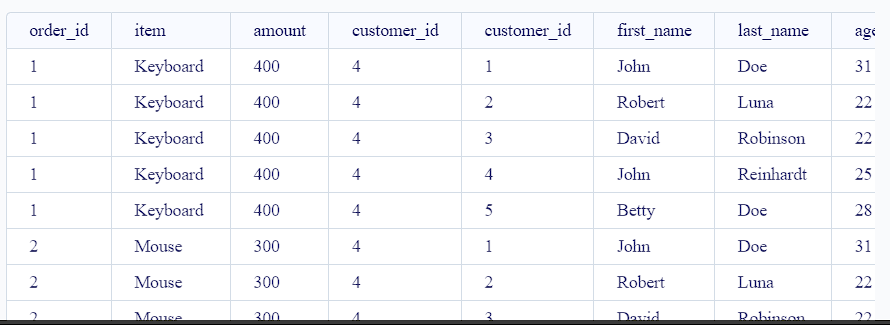
Description automatically generated

* **Full Outer Join**
* A screenshot of a computer

  Description automatically generatedSELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept\_ID, d.Dept\_Name FROM Dept1660 d LEFT OUTER JOIN Employee1660 e ON e.EmployeeID = d.EID UNION SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept\_ID, d.Dept\_Name FROM Dept1660 d RIGHT OUTER JOIN Employee1660 e ON e.EmployeeID = d.EID;
* **Inner Join**
* SELECT EmployeeID, FirstName, LastName, Salary FROM Employee1660 e, Dept1660 d

WHERE e.EmployeeID =d.EID; A table with numbers and text

Description automatically generated

* **Cross Join**
* SELECT \*FROM Orders CROSS JOIN Customers;
* 
* **Left Join**
* SELECT e.order\_id, e.item, e.amount, e.customer\_id, d.customer\_id, d.country FROM Orders e LEFT JOIN Customers d ON e.customer\_id = d.customer\_id;

A screenshot of a computer

Description automatically generated

* **Right Join**
* SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary, d.Dept\_ID, d.Dept\_Name FROM Employee1660 e RIGHT JOIN Dept1660 d ON e.EmployeeID = d.EID;

A screenshot of a computer

Description automatically generated

* **Full Join**
* SELECT \*

FROM Employee1660 e

FULL OUTER JOIN Dept1660 d ON e.EmployeeID = d.EID;

A screenshot of a number of salary

Description automatically generated

# 

# EXPERIMENT -5

**Aim:**

To complete **LAB ASSIGNMENT 1.**

**Input:**

To Create the following five tables **(Course, Course\_fee, Student, Installments, Course\_Taken)**

#### --Create Course table--

CREATE TABLE Course (

course\_no CHAR(4) PRIMARY KEY, course\_name VARCHAR(20)

);

**--Insert Five Values into the Course Table—** INSERT INTO Course (course\_no, course\_name) VALUES

('C001', 'Introduction to Programming'), ('C002', 'Database Management'), ('C003', 'Web Development'),

('C004', 'Data Structures'), ('C005', 'Machine Learning');

#### A screenshot of a computer Description automatically generatedOutput:

**--Create Course\_Fee table--** CREATE TABLE Course\_fee ( course\_no CHAR(4) PRIMARY KEY,

full\_part CHAR(1) CHECK (full\_part IN ('F', 'P')), fees NUMBER(10),

CONSTRAINT fk\_course\_fee\_course FOREIGN KEY (course\_no) REFERENCES Course (course\_no)

);

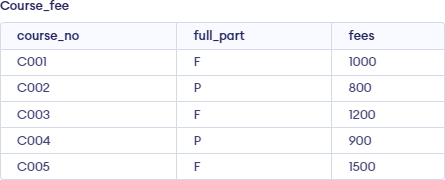
**--Insert Five Values into the Course\_Fee Table--** INSERT INTO Course\_fee (course\_no, full\_part, fees) VALUES

('C001', 'F', 1000),

('C002', 'P', 800),

('C003', 'F', 1200),

('C004', 'P', 900),

('C005', 'F', 1500);

#### Output:

**-- Create Student table--**

CREATE TABLE Student (

prospectus\_no NUMBER(10) PRIMARY KEY, name VARCHAR(20),

address VARCHAR(30), phone\_no NUMBER(11), D\_O\_B DATE,

total\_amt NUMBER(10, 2),

amt\_paid NUMBER(10, 2),

installment CHAR(1) CHECK (installment IN ('I', 'F'))

);

#### -- Insert into Student table--

INSERT INTO Student (prospectus\_no, name, address, phone\_no, D\_O\_B, total\_amt, amt\_paid, installment) VALUES

(1001, 'JIggi Fegusa', '125 Amity University', 1234567890, '2000-05-15', 1500.00, 500.00, 'I'),

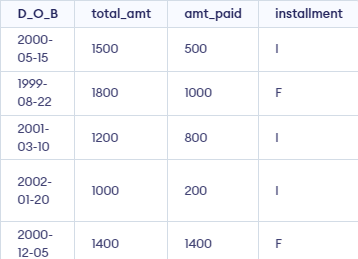
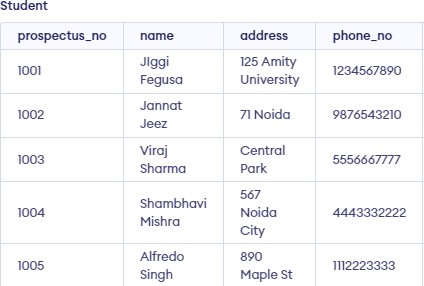
(1002, 'Jannat Jeez', '71 Noida', 9876543210, '1999-08-22', 1800.00, 1000.00, 'F'),

(1003, 'Viraj Sharma', 'Central Park', 5556667777, '2001-03-10', 1200.00, 800.00, 'I'),

(1004, 'Shambhavi Mishra', '567 Noida City', 4443332222, '2002-01-20', 1000.00, 200.00, 'I'),

(1005, 'David Wiley', '890 Maple St', 1112223333, '2000-12-05', 1400.00, 600.00, 'F');

#### Output:



**-- Create Installment table--** CREATE TABLE Installment ( prospectus\_no NUMBER(10), installment\_amt NUMBER(10, 2), due\_dt DATE,

paid CHAR(1) CHECK (paid IN ('P', 'U')),

PRIMARY KEY (prospectus\_no, due\_dt),

CONSTRAINT fk\_installment\_student FOREIGN KEY (prospectus\_no) REFERENCES Student (prospectus\_no) ON DELETE CASCADE

);

#### -- Insert into Installment table--

INSERT INTO Installment (prospectus\_no, installment\_amt, due\_dt, paid) VALUES

(1001, 200.00, '2023-08-20', 'U'),

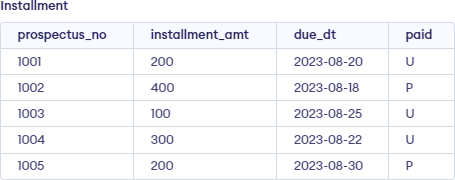
(1002, 400.00, '2023-08-18', 'P'),

(1003, 100.00, '2023-08-25', 'U'),

(1004, 300.00, '2023-08-22', 'U'),

(1005, 200.00, '2023-08-30', 'P');

#### Output:



**-- Create Course\_taken table--** CREATE TABLE Course\_taken ( prospectus\_no NUMBER(10), course\_no CHAR(4),

start\_dt DATE,

full\_part CHAR(1) CHECK (full\_part IN ('F', 'P')), time\_slot CHAR(2),

performance VARCHAR(20),

CONSTRAINT fk\_course\_taken\_student FOREIGN KEY (prospectus\_no) REFERENCES Student (prospectus\_no),

CONSTRAINT fk\_course\_taken\_course FOREIGN KEY (course\_no) REFERENCES Course (course\_no)

);

#### -- Insert into Course\_taken table--

INSERT INTO Course\_taken (prospectus\_no, course\_no, start\_dt, full\_part, time\_slot, performance) VALUES

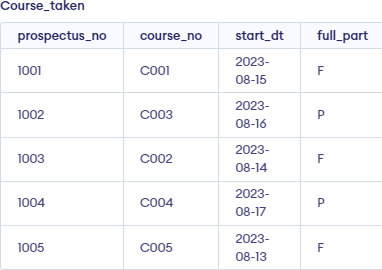
(1001, 'C001', '2023-08-15', 'F', 'M1', 'Good'),

(1002, 'C003', '2023-08-16', 'P', 'E1', 'Excellent'),

(1003, 'C002', '2023-08-14', 'F', 'A2', 'Average'),

(1004, 'C004', '2023-08-17', 'P', 'M2', 'Good'),

(1005, 'C005', '2023-08-13', 'F', 'E2', 'Very Good');



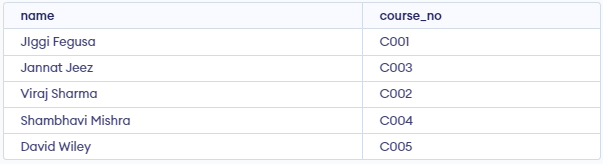
**Output:**

**Write the following SQL Queries:**

## Retrieve name and course no of all the students.

* + SELECT s.name, ct.course\_no FROM Student s

JOIN Course\_taken ct ON s.prospectus\_no = ct.prospectus\_no;



## List the names of students who have paid the full amount at the time of admission.

* + SELECT name FROM Student

WHERE total\_amt = amt\_paid;

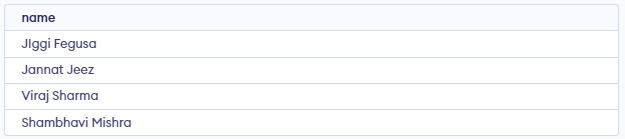


1. Find the names of students starting with A.
   * SELECT name FROM Student

WHERE name LIKE 'A%';

1. Print the names of students whose total amount is not equal to the amount due.
   * SELECT name FROM Student

WHERE total\_amt <> amt\_paid;



1. Count the number of students who have joined in the current year, current month.
   * SELECT COUNT(\*)

FROM Student

WHERE EXTRACT(YEAR FROM D\_O\_B) = EXTRACT(YEAR FROM CURRENT\_DATE) AND EXTRACT(MONTH FROM D\_O\_B) = EXTRACT(MONTH FROM CURRENT\_DATE);

1. Determine the maximum and minimum course fees.
   * SELECT MAX(fees) AS max\_fee, MIN(fees) AS min\_fee FROM Course\_fee;



1. Increase the fee of oracle by 50%.
   * UPDATE Course\_fee SET fees = fees \* 1.5

WHERE course\_no = 'C001';

## Print the details of courses whose fees are between 5000 and 10000.

* + SELECT \*

FROM Course\_fee

WHERE fees BETWEEN 5000 AND 10000;

## Display the admission date in Date, Month, Year format.

* + SELECT TO\_CHAR(D\_O\_B, 'DD MONTH YYYY') AS admission\_date FROM Student;

1. Find out in which course the maximum number of students have taken admission.

SELECT course\_no FROM (

SELECT course\_no, RANK() OVER (ORDER BY COUNT(\*) DESC) AS rank

FROM Course\_taken GROUP BY course\_no

) ranked

WHERE rank = 1;

# EXPERIMENT-6

**Aim:** To complete **LAB ASSIGNMENT 2.**

**Input:**

To Create the following five tables **(SUPPLIER, PART, PROJECTS and SPJ)**

**-- Create SUPPLIER table --** CREATE TABLE SUPPLIER ( SNO CHAR(4) PRIMARY KEY, SNAME VARCHAR(50), STATUS INT,

CITY VARCHAR(50)

);

#### --Insert Five Values into the SUPPLIER Table --

INSERT INTO SUPPLIER (SNO, SNAME, STATUS, CITY) VALUES

('S1', 'Supplier A', 1, 'New York'),

('S2', 'Supplier B', 2, 'Los Angeles'),

('S3', 'Supplier C', 3, 'Chicago'),

('S4', 'Supplier D', 1, 'Houston'),

('S5', 'Supplier E', 2, 'San Francisco');

### A table with text on it Description automatically generatedOUTPUT:

#### --Create PARTS table --

CREATE TABLE PARTS (

PNO CHAR(4) PRIMARY KEY, PNAME VARCHAR(50), COLOR VARCHAR(20), WEIGHT DECIMAL(10, 2), CITY VARCHAR(50)

);

#### --Insert data into PARTS table --

INSERT INTO PARTS (PNO, PNAME, COLOR, WEIGHT, CITY) VALUES

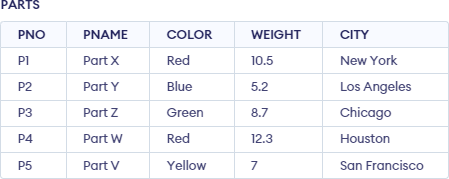
('P1', 'Part X', 'Red', 10.5, 'New York'),

('P2', 'Part Y', 'Blue', 5.2, 'Los Angeles'),

('P3', 'Part Z', 'Green', 8.7, 'Chicago'),

('P4', 'Part W', 'Red', 12.3, 'Houston'),

('P5', 'Part V', 'Yellow', 7.0, 'San Francisco');

OUTPUT:

--Create PROJECT table-- CREATE TABLE PROJECT ( JNO CHAR(4) PRIMARY KEY, JNAME VARCHAR(50),

CITY VARCHAR(50)

);

#### -- Insert data into PROJECT table --

INSERT INTO PROJECT (JNO, JNAME, CITY) VALUES

('J1', 'Project Alpha', 'New York'),

('J2', 'Project Beta', 'Los Angeles'),

('J3', 'Project Gamma', 'Chicago'),

('J4', 'Project Delta', 'Houston'),

('J5', 'Project Epsilon', 'San Francisco');

### A screenshot of a computer Description automatically generatedOUTPUT:

**-- Create SPJ table --** CREATE TABLE SPJ ( SNO CHAR(4),

PNO CHAR(4), JNO CHAR(4), QTY INT,

PRIMARY KEY (SNO, PNO, JNO),

FOREIGN KEY (SNO) REFERENCES SUPPLIER(SNO), FOREIGN KEY (PNO) REFERENCES PARTS(PNO), FOREIGN KEY (JNO) REFERENCES PROJECT(JNO)

);

#### -- Insert data into SPJ table --

INSERT INTO SPJ (SNO, PNO, JNO, QTY) VALUES

('S1', 'P1', 'J1', 100),

('S1', 'P2', 'J2', 200),

('S2', 'P3', 'J2', 150),

('S2', 'P4', 'J3', 300),

('S3', 'P1', 'J3', 75),

('S3', 'P3', 'J3', 50),

('S4', 'P5', 'J4', 250),

('S5', 'P2', 'J5', 100),

('S5', 'P4', 'J5', 50);

### OUTPUT:

**Write the following SQL Queries:**

## Get sno values for suppliers who supply project j1.

* + SELECT SNO FROM SPJ WHERE JNO = 'J1';

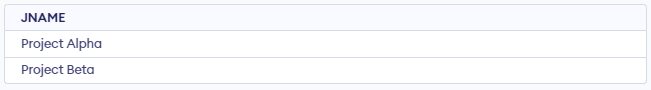


1. Get sno values for suppliers who supply project j1 with part p1.
   * SELECT SNO FROM SPJ

WHERE JNO = 'J1' AND PNO = 'P1';

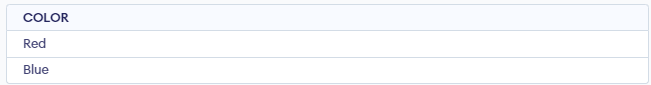
1. Get JNAME values for projects supplied by supplier S1.
   * SELECT DISTINCT PROJECT.JNAME FROM PROJECT

JOIN SPJ ON PROJECT.JNO = SPJ.JNO WHERE SPJ.SNO = 'S1';



1. Get COLOR values for parts supplied by supplier S1.
   * SELECT DISTINCT PARTS.COLOR FROM PARTS

JOIN SPJ ON PARTS.PNO = SPJ.PNO WHERE SPJ.SNO = 'S1';



1. Get PNO values for parts supplied to any project in New York.
   * SELECT DISTINCT SPJ.PNO FROM SPJ

JOIN PROJECT ON SPJ.JNO = PROJECT.JNO WHERE PROJECT.CITY = 'New York';



1. Get SNO values for suppliers who supply project J1 with a red part.
   * SELECT DISTINCT SPJ.SNO FROM SPJ

JOIN PARTS ON SPJ.PNO = PARTS.PNO

WHERE SPJ.JNO = 'J1' AND PARTS.COLOR = 'Red';

1. Get SNO values for suppliers who supply a New York or Los Angeles project with a red part.
   * SELECT DISTINCT SPJ.SNO FROM SPJ

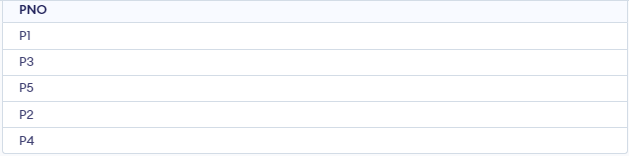
JOIN PARTS ON SPJ.PNO = PARTS.PNO JOIN PROJECT ON SPJ.JNO = PROJECT.JNO

WHERE PARTS.COLOR = 'Red' AND (PROJECT.CITY = 'New York' OR PROJECT.CITY = 'Los

Angeles');

1. Get PNO values for parts supplied to any project by a supplier in the same city.
   * SELECT DISTINCT SPJ.PNO FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO JOIN PROJECT ON SPJ.JNO = PROJECT.JNO WHERE SUPPLIER.CITY = PROJECT.CITY;



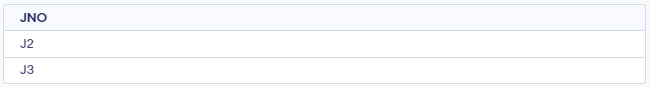
1. Get PNO values for parts supplied to any project in New York by a supplier in New York.
   * SELECT DISTINCT SPJ.PNO FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO JOIN PROJECT ON SPJ.JNO = PROJECT.JNO

WHERE SUPPLIER.CITY = 'New York' AND PROJECT.CITY = 'New York';

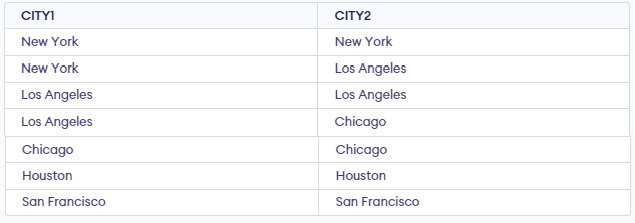
1. Get JNO values for projects supplied by at least one supplier not in the same city.
   * SELECT DISTINCT SPJ.JNO FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO JOIN PROJECT ON SPJ.JNO = PROJECT.JNO WHERE SUPPLIER.CITY <> PROJECT.CITY;



1. Get all pairs of CITY values such that a supplier in the first CITY supplies a project in the second CITY.
   * SELECT DISTINCT SUPPLIER.CITY AS CITY1, PROJECT.CITY AS CITY2 FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO JOIN PROJECT ON SPJ.JNO = PROJECT.JNO;



1. Get SNO values for suppliers who supply the same part to all projects.
   * SELECT DISTINCT SPJ.SNO FROM SPJ

WHERE SPJ.PNO IN ( SELECT PNO

FROM SPJ GROUP BY PNO

HAVING COUNT(DISTINCT JNO) = (SELECT COUNT(\*) FROM PROJECT)

);

1. Get PNO values for parts supplied to all projects in New York.
   * SELECT DISTINCT SPJ.PNO FROM SPJ

JOIN PROJECT ON SPJ.JNO = PROJECT.JNO

WHERE PROJECT.CITY = 'New York' AND SPJ.PNO NOT IN (

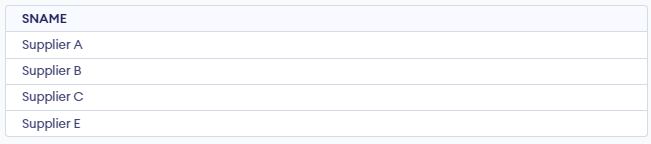
SELECT PNO FROM SPJ

JOIN PROJECT ON SPJ.JNO = PROJECT.JNO WHERE PROJECT.CITY != 'New York'

);

1. Get SNAME values for suppliers who supply at least one red part to any project.
   * SELECT DISTINCT SUPPLIER.SNAME FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO JOIN PARTS ON SPJ.PNO = PARTS.PNO WHERE PARTS.COLOR = 'Red';



1. Get total quantity of part P1 supplied by supplier S1.
   * SELECT SUM(QTY) AS total\_quantity FROM SPJ

WHERE SNO = 'S1' AND PNO = 'P1';

1. Get the total number of projects supplied by supplier S3.
   * SELECT COUNT(DISTINCT JNO) AS total\_projects FROM SPJ

WHERE SNO = 'S3';

1. Change COLOR of all red parts to orange.
   * UPDATE PARTS

SET COLOR = 'Orange' WHERE COLOR = 'Red';

1. Get SNAME values for suppliers who supply to both projects J1 and J2.
   * SELECT DISTINCT SUPPLIER.SNAME FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO WHERE JNO = 'J1'

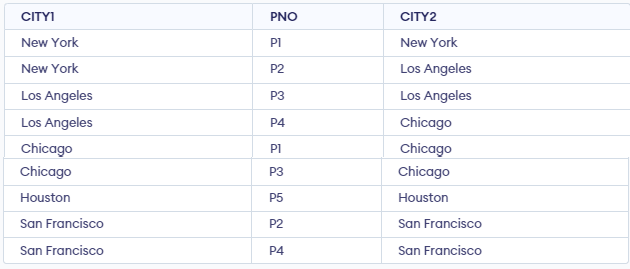
AND SUPPLIER.SNO IN ( SELECT SNO

FROM SPJ WHERE JNO = 'J2'

);

1. Get all CITY, PNO, CITY triples such that a supplier in the first CITY supplies the specified part to a project in the second CITY.
   * SELECT DISTINCT SUPPLIER.CITY AS CITY1, SPJ.PNO, PROJECT.CITY AS CITY2 FROM SPJ

JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO JOIN PROJECT ON SPJ.JNO = PROJECT.JNO;



1. Get JNAMEs for projects which are supplied by supplier XYZ.
   * SELECT DISTINCT PROJECT.JNAME FROM SPJ

JOIN PROJECT ON SPJ.JNO = PROJECT.JNO JOIN SUPPLIER ON SPJ.SNO = SUPPLIER.SNO WHERE SUPPLIER.SNAME = 'XYZ';

EXPERIMENT -7

**Aim:** To complete **LAB ASSIGNMENT 3.**

**Input:**

To Create the following two tables **(Employee and department)**

**Create DEPARTMENT table** CREATE TABLE Departments ( DepartmentID INT PRIMARY KEY, DepartmentName VARCHAR(255)

);

# Insert Values into the DEPARTMENT Table

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'Systems'),

(2, 'Marketing'),

(3, 'Sales');

# OUTPUT:

**Create EMPLOYEE table** CREATE TABLE Employees ( EmployeeID INT PRIMARY KEY,

EmployeeName VARCHAR(255), DepartmentID INT,

HireDate DATE,

Salary DECIMAL(10, 2), JobType VARCHAR(50),

ManagerID INT

);

# Insert data into EMPLOYEE table

INSERT INTO Employees (EmployeeID, EmployeeName, DepartmentID, HireDate, Salary, JobType, ManagerID) VALUES

(1, 'Apple singh', 1, '2020-01-15', 55000, 'Manager', NULL),

(2, 'Jamun jha', 1, '2021-03-10', 48000, 'Engineer', 1),

(3, 'Pineapple mishra', 1, '2022-05-20', 52000, 'Engineer', 1),

(4, 'Banana wilson', 2, '2019-11-02', 60000, 'Manager', NULL),

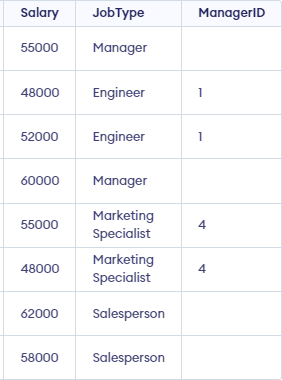
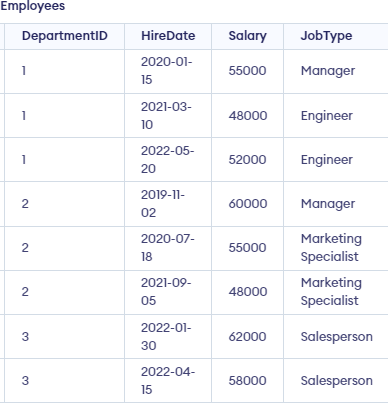
(5, 'Mango kaur', 2, '2020-07-18', 55000, 'Marketing Specialist', 4),

(6, 'Peach dhawan', 2, '2021-09-05', 48000, 'Marketing Specialist', 4),

(7, 'orange roy', 3, '2022-01-30', 62000, 'Salesperson', NULL),

(8, 'raspberry pie', 3, '2022-04-15', 58000, 'Salesperson', NULL);

# OUTPUT:



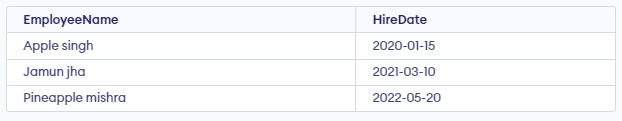
**Write the following SQL Queries:**

1.) Display each employee name and hiredate of the Systems department.

* SELECT EmployeeName, HireDate FROM Employees

WHERE DepartmentID = 1;SELECT SNO FROM SPJ

WHERE JNO = 'J1';



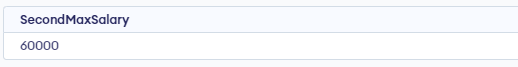
2.) Calculate length of service of each employee

* SELECT EmployeeName, DATEDIFF(CURDATE(), HireDate) AS LengthOfService FROM Employees;

3.) Find the second maximum salary of all employee

* SELECT MAX(Salary) AS SecondMaxSalary FROM Employees

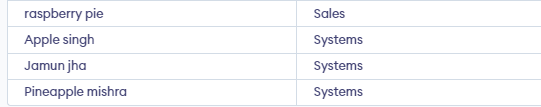
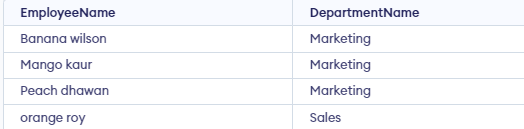
WHERE Salary < (SELECT MAX(Salary) FROM Employees);



4.) Display all employee names and department names in department name order.

* SELECT E.EmployeeName, D.DepartmentName FROM Employees E

JOIN Departments D ON E.DepartmentID = D.DepartmentID ORDER BY D.DepartmentName;



5.) Find the name of the lowest paid employee for each manager.

* SELECT M.EmployeeName AS ManagerName, E.EmployeeName AS LowestPaidEmployee FROM Employees E

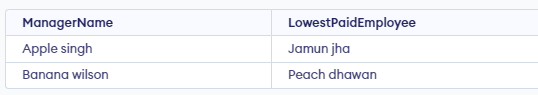
LEFT JOIN Employees M ON E.ManagerID = M.EmployeeID WHERE E.Salary = (

SELECT MIN(Salary)

FROM Employees

WHERE ManagerID = E.ManagerID

);



6.) Display the department that has no employee.

* SELECT DepartmentName FROM Departments

WHERE DepartmentID NOT IN (SELECT DISTINCT DepartmentID FROM Employees);

7.) Find the employees who earn the maximum salary in each job type. Sort in descending order of salary.

* SELECT JobType, EmployeeName, Salary FROM Employees E1

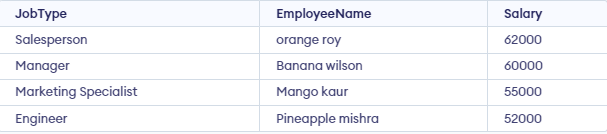
WHERE Salary = ( SELECT MAX(Salary)

FROM Employees E2

WHERE E1.JobType = E2.JobType

)

ORDER BY Salary DESC, JobType;



8.) In which year did most people join the company? Display the year and number of employees.

* SELECT YEAR(HireDate) AS JoinYear, COUNT(\*) AS NumberOfEmployees

FROM Employees GROUP BY JoinYear

ORDER BY NumberOfEmployees DESC LIMIT 1;

9.) Display the details of those employees who earn greater than average of their department.

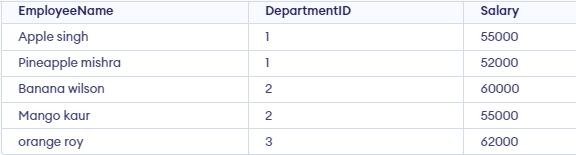
* SELECT E.EmployeeName, E.DepartmentID, E.Salary FROM Employees E

JOIN (

SELECT DepartmentID, AVG(Salary) AS AvgSalary FROM Employees

GROUP BY DepartmentID

) AS AvgSalaries ON E.DepartmentID = AvgSalaries.DepartmentID WHERE E.Salary > AvgSalaries.AvgSalary;



10.) List the employees having salary between 10000 and 20000.

* SELECT EmployeeName, Salary FROM Employees

WHERE Salary BETWEEN 10000 AND 20000;

11.)Display all employees hired during 1983 and who earn greater than average of their department

* SELECT E.EmployeeName, E.HireDate, E.Salary FROM Employees E

WHERE YEAR(HireDate) = 1983

AND E.Salary > ( SELECT AVG(Salary)

FROM Employees

WHERE DepartmentID = E.DepartmentID

);

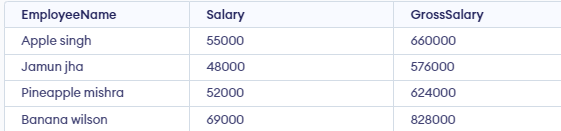
12.) Update the salaries of all employees in marketing department & hike it by 15%.

* UPDATE Employees

SET Salary = Salary \* 1.15 WHERE DepartmentID = 2;

13.) Get the gross salaries of all the employees.

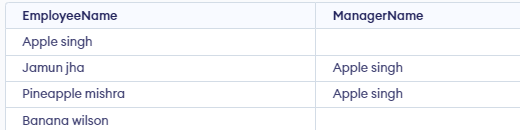
* SELECT EmployeeName, Salary, Salary \* 12 AS GrossSalary FROM Employees;



14.) Get the names of employees and their managers' names.

* SELECT E.EmployeeName, M.EmployeeName AS ManagerName FROM Employees E

LEFT JOIN Employees M ON E.ManagerID = M.EmployeeID;



15.) Display the name, location, and department name of all the employees earning more than 1500.

* SELECT E.EmployeeName, D.DepartmentName, E.Salary FROM Employees E

JOIN Departments D ON E.DepartmentID = D.DepartmentID WHERE E.Salary > 1500;



16.) Show all the employees in Dallas.

* SELECT EmployeeName FROM Employees E

JOIN Departments D ON E.DepartmentID = D.DepartmentID WHERE D.DepartmentName = 'Dallas';

17.) List the employees' name, job, salary, grade, and department for employees in the company except clerks. Sort on employee names.

* SELECT EmployeeName, JobType, Salary, Grade, DepartmentName FROM Employees E

JOIN Departments D ON E.DepartmentID = D.DepartmentID WHERE E.JobType != 'Clerk'

ORDER BY EmployeeName;

18.) Find the employees who earn the minimum salary for their job. Sort in descending order of salary.

* SELECT JobType, EmployeeName, Salary FROM Employees E1

WHERE Salary = ( SELECT MIN(Salary)

FROM Employees E2

WHERE E1.JobType = E2.JobType

)

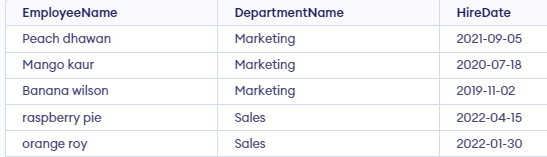
ORDER BY Salary DESC, JobType;



19.) Find the most recently hired employees in department order by hire date.

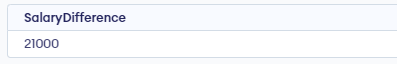
* SELECT EmployeeName, DepartmentName, HireDate FROM Employees E

JOIN Departments D ON E.DepartmentID = D.DepartmentID ORDER BY DepartmentName, HireDate DESC;



20.) Find out the difference between the highest and lowest salaries.

* SELECT MAX(Salary) - MIN(Salary) AS SalaryDifference FROM Employees;



**AMITY SCHOOL OF ENGINEERING & TECHNOLOGY**

AMITY UNIVERSITY CAMPUS, SECTOR-125, NOIDA-201303

A logo of a university

Description automatically generated

**DATABASE MANAGEMENT SYSTEMS**

PRACTICAL FILE

COURSE CODE: CSE201

Submitted to: Submitted by:

Dr. Sumit kumar Boddu Asmitha Bhavya

A2305221386

5CSE-6X

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Experiment name | Date of assigned | Date of submission | Signature |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |