Zain Wajid

2023775

Section D (DS)

CREATE TABLE Customers

(

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR,

email VARCHAR,

phone\_number INT

);

INSERT INTO Customers(customer\_id , customer\_name , email , phone\_number)

VALUES

(1 , 'Alice' , 'alice@example.com' , 123456789),

(2 , 'Bob' , 'bob@example.com' , 987654321),

(3 , 'Charlie' , 'charlie@example' , NULL);

CREATE TABLE Products

(

product\_id INT PRIMARY KEY,

product\_name VARCHAR,

price INT,

stock\_quantity INT

);

INSERT INTO Products(product\_id , product\_name , price , stock\_quantity)

VALUES

(10 , 'Laptop' , 1000 , 10),

(11 , 'Phone' , 600 , 20),

(12 , 'Tablet' , 300 , 15);

CREATE TABLE Orders

(

order\_id INT PRIMARY KEY,

customer\_id INT,

FOREIGN KEY(customer\_id) REFERENCES Customers(customer\_id),

product\_id INT,

FOREIGN KEY(product\_id) REFERENCES Products(product\_id),

order\_date DATE,

quantity INT,

total\_price INT

);

INSERT INTO Orders(order\_id , customer\_id , product\_id , order\_date , quantity , total\_price)

VALUES

(20 , 1 , 10 , '2025-03-01' , 10 , 10000),

(21 , 2 , 11 , '2025-03-02' , 2 , 1200),

(22 , 2 , 12 , '2025-03-03' , 5 , 1500);

Tasks:

1. Inner Join: Write a query to retrieve the details of all orders along with the customer

names and product names. The output should include order\_id, customer\_name, product\_name, order\_date, quantity, and total\_price.

*-- View which customer ordered whihc product*

SELECT order\_id , order\_date , quantity , total\_price , Customers.customer\_name , Products.product\_name

FROM Orders

INNER JOIN Customers

ON Orders.customer\_id = Customers.customer\_id

INNER JOIN Products

ON Products.product\_id = Orders.product\_id;

1. Left Join: Write a query to find all customers who have not placed any orders. The output should include customer\_id, customer\_name, and email.

*-- View customers without orders*

SELECT Customers.customer\_id , Customers.customer\_name , Customers.email

FROM Customers

LEFT JOIN Orders

ON Customers.customer\_id = Orders.customer\_id

WHERE order\_id IS NULL;

1. Full Outer Join: Write a query to retrieve all customers and all orders, including those customers who have not placed any orders and those orders that do not have a corresponding customer. The output should include customer\_id, customer\_name, order\_id, and order\_date.

*-- View all orders and customers in database*

SELECT Customers.customer\_id , Customers.customer\_name , Orders.order\_id , Orders.order\_date

FROM Customers

FULL OUTER JOIN Orders

ON Customers.customer\_id = Orders.customer\_id;

1. Cross Join: Write a query to generate a report that shows all possible combinations of customers and products. The output should include customer\_name and product\_name.

*--Cross join to view all order customer combinations*

SELECT Customers.customer\_name , Products.product\_name

FROM Customers

CROSS JOIN Products;

CREATE TABLE employees

(

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR,

department\_id INT,

hire\_date DATE

);

INSERT INTO employees (employee\_id , employee\_name , department\_id , hire\_date)

VALUES

(1 , 'John' , 101 , '2020-01-15'),

(2 , 'Sarah' , 101 , '2021-03-20'),

(3 , 'Mike' , 102 , '2019-07-10'),

(4 , 'Emma' , NULL , '2022-05-01');

CREATE TABLE projects

(

project\_id INT PRIMARY KEY,

project\_name VARCHAR,

employee\_id INT,

start\_date DATE,

end\_date DATE,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

INSERT INTO projects(project\_id , project\_name , employee\_id , start\_date , end\_date)

VALUES

(1 , 'project\_alpha' , 1 , '2023-01-01' , '2023-06-30'),

(2 , 'project\_beta' , 2 , '2023-02-15' , '2023-08-31'),

(3 , 'project\_gamma' , 1 , '2023-03-01' , '2023-09-30');

Tasks:

1. Anti-Join: Write a query to find all employees who have not been assigned to any project. The output should include employee\_id, employee\_name, and department\_id.

*--Employees not assigned to any projects*

SELECT e.employee\_id , e.employee\_name , e.department\_id

FROM employees e

LEFT JOIN projects

ON e.employee\_id = projects.employee\_id

WHERE e.department\_id IS NULL;

1. Semi-Join: Write a query to retrieve the details of all employees who have been assigned to at least one project. The output should include employee\_id, employee\_name, and department\_id.

*-- Show employees assigned to at least one projects*

SELECT e.employee\_id , e.employee\_name , e.department\_id

FROM employees e

WHERE EXISTS

(

SELECT 1

FROM projects p

WHERE e.employee\_id = p.employee\_id

)

1. Self-Join: Write a query to find pairs of employees who work in the same department. The output should include employee1\_name, employee2\_name, and department\_id.

*-- Employees who work in the same department*

SELECT a.employee\_name AS employee1\_name , b.employee\_name AS employee2\_name , a.department\_id

FROM employees a

JOIN employees b

ON a.department\_id = b.department\_id

WHERE a.employee\_id < b.employee\_id;

1. Natural Join: Write a query to retrieve the details of all projects along with the employee names using a natural join. The output should include project\_id, project\_name, employee\_name, start\_date, and end\_date.

*-- Detail of all projects along with employee name*

SELECT p.project\_id , p.project\_name , e.employee\_name , p.start\_date , p.end\_date

FROM projects p

NATURAL JOIN employees e