

EXP1:

PROGRAM :

```
CREATE DATABASE SampleDB;
```

```
USE SampleDB;
```

```
CREATE TABLE Employee (
```

```
    EmployeeID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
    FirstName VARCHAR(50) NOT NULL,
```

```
    LastName VARCHAR(50) NOT NULL,
```

```
    Email VARCHAR(100) UNIQUE,
```

```
    Age INT CHECK (Age >= 18),
```

```
    Salary DECIMAL(10, 2) NOT NULL
```

```
);
```

```
INSERT INTO Employee (FirstName, LastName, Email, Age, Salary)
```

```
VALUES
```

```
('John', 'Doe', 'john.doe@example.com', 30, 50000.00),
```

```
('Jane', 'Smith', 'jane.smith@example.com', 25, 60000.00),
```

```
('Alice', 'Johnson', 'alice.johnson@example.com', 40, 70000.00);
```

```
UPDATE Employee
```

```
SET Salary = 55000.00
```

```
WHERE EmployeeID = 1;
```

```
DELETE FROM Employee
```

```
WHERE EmployeeID = 2;
```

```
SELECT * FROM Employee;
```

OUTPUT:

Insert:

Result Grid

Filter Rows:

Edit:

Export/Import:

	EmployeeID	FirstName	LastName	Email	Age	Salary
▶	1	John	Doe	john.doe@example.com	30	50000.00
	2	Jane	Smith	jane.smith@example.com	25	60000.00
	3	Alice	Johnson	alice.johnson@example.com	40	70000.00
✱	NULL	NULL	NULL	NULL	NULL	NULL

Update:

Result Grid

Filter Rows:

Edit:

Export/Import:

	EmployeeID	FirstName	LastName	Email	Age	Salary
▶	1	John	Doe	john.doe@example.com	30	55000.00
	2	Jane	Smith	jane.smith@example.com	25	60000.00
	3	Alice	Johnson	alice.johnson@example.com	40	70000.00
★	NULL	NULL	NULL	NULL	NULL	NULL

Delete:

Result Grid

Filter Rows:

Edit:

Export/Import:

	EmployeeID	FirstName	LastName	Email	Age	Salary
	1	John	Doe	john.doe@example.com	30	55000.00
	3	Alice	Johnson	alice.johnson@example.com	40	70000.00
	NULL	NULL	NULL	NULL	NULL	NULL

EXP 2:

PROGRAM :

```
CREATE DATABASE IF NOT EXISTS ReferentialIntegrityDB;
```

```
USE ReferentialIntegrityDB;
```

```
CREATE TABLE Department (
```

```
    DepartmentID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
    DepartmentName VARCHAR(100) NOT NULL
```

```
);
```

```
CREATE TABLE Employee (
```

```
    EmployeeID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
    FirstName VARCHAR(50) NOT NULL,
```

```
    LastName VARCHAR(50) NOT NULL,
```

```
    DepartmentID INT, -- Foreign Key
```

```
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
```

```
    ON DELETE CASCADE -- Enforce referential integrity
```

```
);
```

```
INSERT INTO Department (DepartmentName)
```

```
VALUES
```

```
('HR'),
```

```
('Finance'),
```

```
('IT');
```

```
INSERT INTO Employee (FirstName, LastName, DepartmentID)
```

```
VALUES
```

```
('John', 'Doe', 1), -- Belongs to HR
```

('Jane', 'Smith', 2), -- Belongs to Finance

('Alice', 'Johnson', 3); -- Belongs to IT

SELECT * FROM Department;

SELECT * FROM Employee;

DELETE FROM Department WHERE DepartmentID = 2;

SELECT * FROM Department;

SELECT * FROM Employee;

OUTPUT:

Employee:



The screenshot shows the 'Result Grid' for the Employee table. The table has four columns: EmployeeID, FirstName, LastName, and DepartmentID. The first row contains NULL values for all columns. The interface includes a 'Filter Rows' field, an 'Edit' button, and an 'Export/Import' button.

	EmployeeID	FirstName	LastName	DepartmentID
*	NULL	NULL	NULL	NULL


Department:



The screenshot shows the 'Result Grid' for the Department table. The table has two columns: DepartmentID and DepartmentName. The first row contains NULL values for both columns. The interface includes a 'Filter Rows' field, an 'Edit' button, and an 'Export/Import' button.

	DepartmentID	DepartmentName
*	NULL	NULL

Insert to department:



The screenshot shows the 'Result Grid' for the Department table after an insert operation. The table has two columns: DepartmentID and DepartmentName. The first three rows contain data: (1, HR), (2, Finance), and (3, IT). The fourth row contains NULL values. The interface includes a 'Filter Rows' field, an 'Edit' button, and an 'Export/Import' button.

	DepartmentID	DepartmentName
▶	1	HR
	2	Finance
	3	IT
*	NULL	NULL

Insert to employee:

Result Grid	Filter Rows:	Edit:	Export/Impo
EmployeeID	FirstName	LastName	DepartmentID
1	John	Doe	1
2	Jane	Smith	2
3	Alice	Johnson	3
NULL	NULL	NULL	NULL

Delete:

Result Grid	Filter Rows:	Edit:	Export/Impo
EmployeeID	FirstName	LastName	DepartmentID
1	John	Doe	1
3	Alice	Johnson	3
4	John	Doe	1
6	Alice	Johnson	3
NULL	NULL	NULL	NULL

EXP 3:

PROGRAM :

USE UniversityDB;

SELECT Name, Age, DeptID FROM

Students

WHERE DeptID = 1;

SELECT DeptID, COUNT(*) AS TotalStudents FROM

Students

GROUP BY DeptID;

SELECT DeptID, AVG(Age) AS AverageAge FROM

Students

WHERE DeptID = 1

GROUP BY DeptID;

SELECT MAX(Age) AS MaxAge, MIN(Age) AS MinAge FROM

Students;

SELECT DeptID, COUNT(*) AS TotalStudents FROM

Students

GROUP BY DeptID HAVING

COUNT(*) > 1;

SELECT SUM(Age) AS TotalAge FROM

Students

WHERE DeptID = (

```
SELECT DeptID  
FROM Departments  
WHERE DeptName = 'Computer Science'  
);
```

OUTPUT:

CREATE:



The screenshot shows a database application interface with a 'Result Grid' tab. The grid has five columns: SaleID, ProductName, Quantity, PricePerUnit, and SaleDate. A single row is displayed with all values as NULL. The interface includes a 'Filter Rows' search bar, an 'Edit' button with a pencil icon, and an 'Export/Import' button.

	SaleID	ProductName	Quantity	PricePerUnit	SaleDate
*	NULL	NULL	NULL	NULL	NULL

Insert:



The screenshot shows the same database application interface, but now with six rows of data inserted. The columns are SaleID, ProductName, Quantity, PricePerUnit, and SaleDate. The data is as follows:

	SaleID	ProductName	Quantity	PricePerUnit	SaleDate
▶	1	Laptop	5	50000.00	2025-01-01
	2	Mouse	10	500.00	2025-01-02
	3	Keyboard	7	1500.00	2025-01-03
	4	Monitor	3	12000.00	2025-01-04
	5	Laptop	2	48000.00	2025-01-05
	6	Mouse	8	550.00	2025-01-06
*	NULL	NULL	NULL	NULL	NULL

Exp 4:

PROGRAM :

```
CREATE DATABASE IF NOT EXISTS SubqueriesAndJoinsDB;
```

```
USE SubqueriesAndJoinsDB;
```

```
CREATE TABLE IF NOT EXISTS Customers (
```

```
    CustomerID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
    CustomerName VARCHAR(100) NOT NULL,
```

```
    City VARCHAR(50),
```

```
    Country VARCHAR(50)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS Orders (
```

```
    OrderID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
    CustomerID INT NOT NULL,
```

```
    OrderDate DATE NOT NULL,
```

```
    TotalAmount DECIMAL(10, 2) NOT NULL,
```

```
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE  
CASCADE
```

```
);
```

```
INSERT IGNORE INTO Customers (CustomerName, City, Country)
```

```
VALUES
```

```
('John Doe', 'New York', 'USA'),
```

```
('Jane Smith', 'Los Angeles', 'USA'),
```

```
('Alice Johnson', 'London', 'UK'),
```



```
('Bob Brown', 'Sydney', 'Australia');  
INSERT IGNORE INTO Orders (CustomerID, OrderDate, TotalAmount)  
VALUES  
(1, '2025-01-01', 500.00),  
(2, '2025-01-02', 1200.00),  
(3, '2025-01-03', 700.00),  
(1, '2025-01-04', 300.00),  
(4, '2025-01-05', 450.00);  
SELECT CustomerName  
FROM Customers  
WHERE CustomerID IN  
    ( SELECT CustomerID  
      FROM Orders  
      WHERE TotalAmount > 1000  
    );  
SELECT SUM(TotalAmount) AS TotalSpent  
FROM Orders  
WHERE CustomerID =  
    ( SELECT CustomerID  
      FROM Customers  
      WHERE CustomerName = 'John Doe'  
    );
```

```
SELECT OrderID, CustomerID, TotalAmount
```

```
FROM Orders
```

```
WHERE CustomerID IN
```

```
    ( SELECT CustomerID
```

```
      FROM Customers
```

```
      WHERE Country = 'USA'
```

```
);
```

```
SELECT Customers.CustomerName, Orders.OrderID, Orders.OrderDate,
```

```
Orders.TotalAmount
```

```
FROM Customers
```

```
INNER JOIN Orders
```

```
ON Customers.CustomerID = Orders.CustomerID;
```

```
SELECT Customers.CustomerName, Orders.OrderID, Orders.TotalAmount
```

```
FROM Customers
```

```
LEFT OUTER JOIN Orders
```

```
ON Customers.CustomerID = Orders.CustomerID;
```

```
SELECT Customers.CustomerName, Orders.OrderID, Orders.TotalAmount
```

```
FROM Customers
```

```
RIGHT OUTER JOIN Orders
```

```
ON Customers.CustomerID = Orders.CustomerID;
```

```
SELECT *
```

```
FROM Customers
```

```
NATURAL JOIN Orders;
```

```
SELECT Customers.CustomerName, SUM(Orders.TotalAmount) AS TotalSpent
```

```
FROM Customers
```

INNER JOIN Orders

ON Customers.CustomerID = Orders.CustomerID

GROUP BY Customers.CustomerName;

SELECT * FROM Customers;

SELECT * FROM Orders;

OUTPUT:

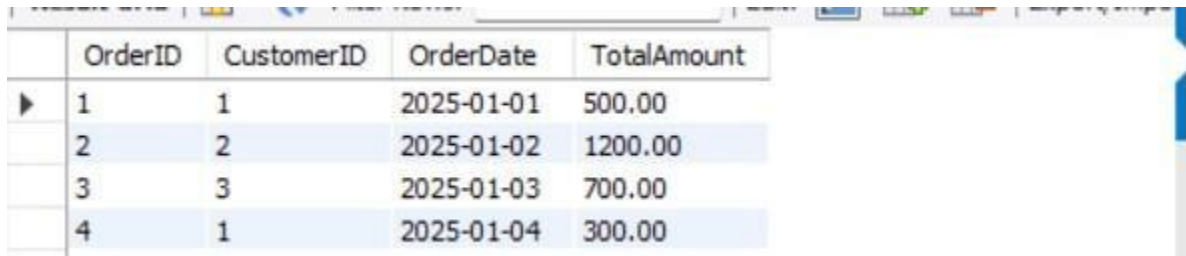
Customer :



A screenshot of a database application's 'Result Grid' window. The window has a toolbar with icons for 'Filter Rows', 'Edit', and 'Export/Import'. The table displayed has five columns: 'CustomerID', 'CustomerName', 'City', and 'Country'. There are four rows of data.

	CustomerID	CustomerName	City	Country
▶	1	John Doe	New York	USA
	2	Jane Smith	Los Angeles	USA
	3	Alice Johnson	London	UK
	4	Bob Brown	Sydney	Australia

Order:



A screenshot of a database application's 'Result Grid' window showing order data. The table has five columns: 'OrderID', 'CustomerID', 'OrderDate', and 'TotalAmount'. There are four rows of data.

	OrderID	CustomerID	OrderDate	TotalAmount
▶	1	1	2025-01-01	500.00
	2	2	2025-01-02	1200.00
	3	3	2025-01-03	700.00
	4	1	2025-01-04	300.00

Exp 5:

PROGRAM :

```
CREATE TABLE Employees (  
    EmployeeID INT AUTO_INCREMENT PRIMARY KEY,  
    EmployeeName VARCHAR(100),  
    Salary DECIMAL(10,2),  
    Department VARCHAR(50)  
);  
  
INSERT INTO Employees (EmployeeName, Salary, Department)  
VALUES  
('John Doe', 50000, 'HR'),  
('Jane Smith', 60000, 'IT'),  
('Alice Johnson', 70000, 'HR'),  
('Bob Brown', 55000, 'Finance');  
  
DELIMITER //  
  
CREATE PROCEDURE GetDepartmentSalaryTotal(IN department_name  
VARCHAR(100), OUT total_salary DECIMAL(10,2))  
BEGIN  
    SET total_salary = 0;  
    SELECT SUM(Salary) INTO total_salary  
    FROM Employees  
    WHERE Department = department_name;  
END //
```

DELIMITER ;

CALL GetDepartmentSalaryTotal('HR', @total_salary);

SELECT @total_salary;

OUTPUT:



The screenshot shows a SQL query result grid. The grid has a header row with the column name '@total_salary' and a data row with the value '480000.00'. The grid is displayed in a window with a toolbar at the top containing icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'.

	@total_salary
▶	480000.00