# DELHI TECHNOLOGICAL UNIVERSITY

DATABASE MANAGEMENT SYSTEM

MC 209

THIRD SEMESTER (2024-2025)

Submitted to:

Prof. Brij Kishore Tyagi

&

Ms. Shweta Kalson

Submitted by:

Himanshu

23/MC/65

INDEX

| S. No. | Experiment | Date | Signature |
|---|---|---|---|
| 1 | Synopsis of the Project, along with its E-R Diagram. | 12-08-2024 | |
| 2 | Implementation of the **DDL statements:**<br>1. Create<br>2. Create table with constraints<br>3. Alter Table<br>a. Add column<br>b. Drop column<br>c. Add/Drop constraint<br>d. Rename column<br>4. Drop Table | 02-09-2024 | |
| 3 | Implementation of the **DML statements:**<br>1. Insert<br>2. Update<br>3. Delete<br>4. Truncate | 09-09-2024 | |
| 4 | Implementation of the **SELECT statements:**<br>1. Simple SELECT statement<br>2. Where clause + IN/NOT IN<br>3. Aggregate functions<br>4. Group by + Having<br>5. Order by<br>6. Views<br>7. Inbuilt function | 16-09-2024 | |
| 5 | Implement and perform **Nested Queries** along with **Joins.** | 30-09-2024 | |
| 6 | Introduction to **PL/SQL**. Create a PL/SQL block and implement the following:<br>1. Variables<br>2. Packages<br>3. Procedures<br>4. Functions | 07-10-2024 | |
| 7 | Perform **Exception handling** in a PL/SQL block. | 07-10-2024 | |
| 8 | Project Report with Code and Screenshots. | 14-09-2024 | |
| 9 | Implement **Triggers** in SQL. | 14-09-2024 | |
| 10 | Implement the following transaction statements:<br>1. Commit<br>2. Rollback<br>3. Savepoint | 21-10-2024 | |

# DBMS PROJECT

ON

# RAILWAY MANAGEMENT SYSTEM

## Experiment 1

**Aim**: Synopsis of the Project, along with its E-R Diagram.

**Project Title:** Railway System Database Management

### Project Synopsis:

The Railway System Database Management project is dedicated to developing a sophisticated database for managing India's extensive railway network. The primary goal is to create a structured, relational database that streamlines user management, passenger reservations, train schedules, ticketing, and more.

### Objective:

To design and implement a relational database that supports efficient data storage, retrieval, and management for various aspects of the railway system. This project ensures data integrity, consistency, and security, providing a solid foundation for railway operations.

### Scope:

The database encompasses tables for Users, Passengers, Trains, Stations, Tickets, and Train Status. Relationships between these entities are established to reflect real-world interactions, such as booking tickets, train schedules, and station stops.

### Entities and Attributes:

1. **Users:** UserID, Password, FirstName, LastName, Gender, Age, Email, AadharNumber, MobileNumber, City, State, PinCode, SecurityQuestion, SecurityAnswer.
2. **Passengers**: PassengerID, Name, Gender, Age, PnrNumber, SeatNumber, BookedBy, ReservationStatus.

3. **Trains**: TrainNumber, TrainName, Source, Destination, ArrivalTime, DepartureTime, AvailabilityOfSeats, Date.
4. **Stations**: StationNumber, Name, TrainNumber, ArrivalTime, Hault.
5. **Tickets**: TicketID, TrainNumber, BookedUser, Status, NumberOfPassengers.
6. **Train Status:** TrainNumber, A_Seats1, A_Seats2, A_Seats3, B_Seats1, B_Seats2, B_Seats3, W_Seats1, W_Seats2, W_Seats3, Fare1, Fare2, Fare3.

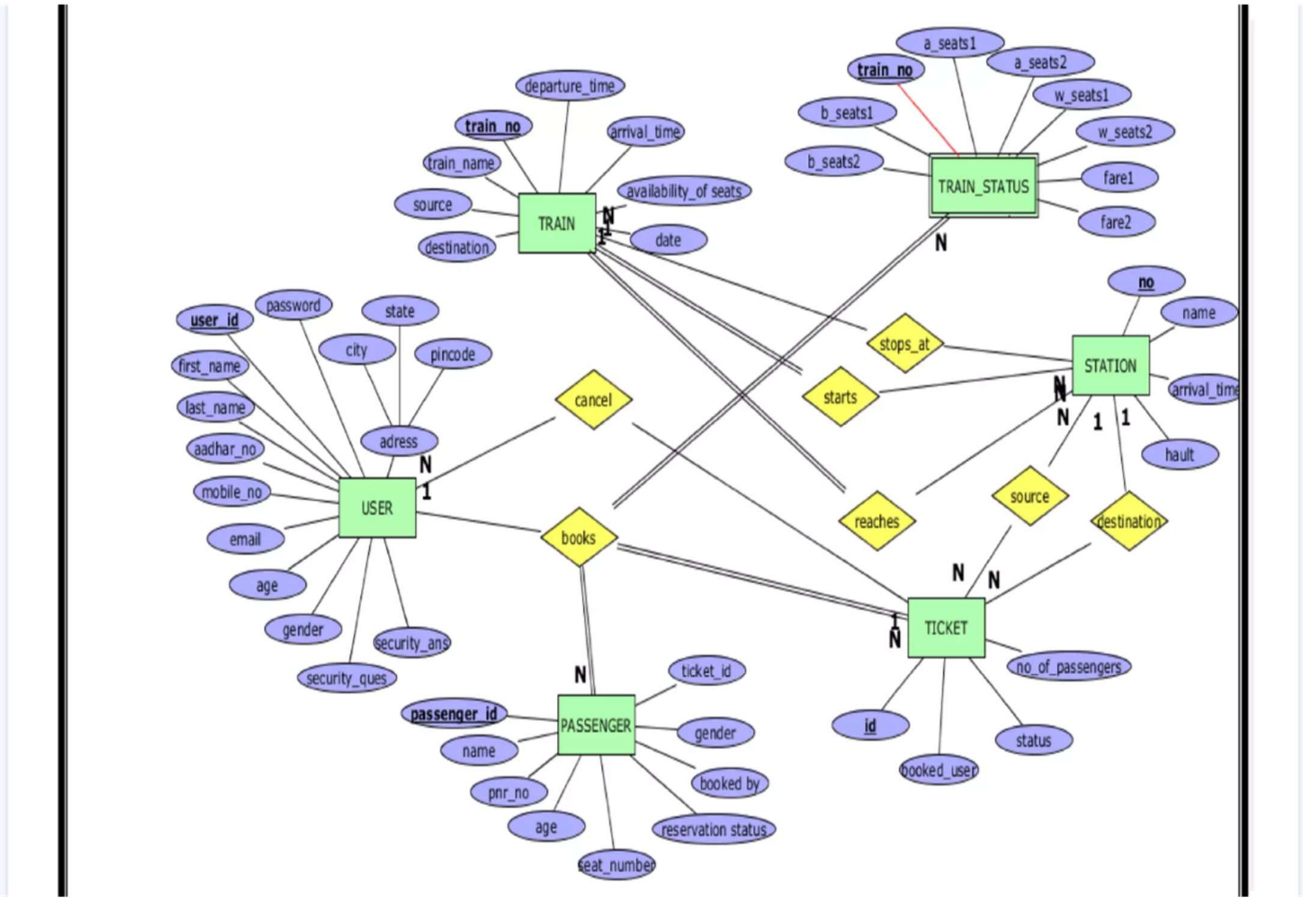
**Relationships:**

1. **Cancel**: Between User (one) and Ticket (one).
2. **Books**: Between User (one), Passenger (many), and Ticket (many).
3. **Starts**: Between Train and Station.
4. **Stops At**: Between Train and Station.
5. **Reaches**: Between Train and Station.
6. **Source**: Between Ticket and Station.
7. **Destination**: Between Ticket and Station.

## Conclusion:

This project delivers a robust and scalable database management system that enhances the efficiency of railway operations in India. By providing accurate data management and seamless integration of railway functions, it significantly contributes to the modernization and optimization of the Indian railway system.

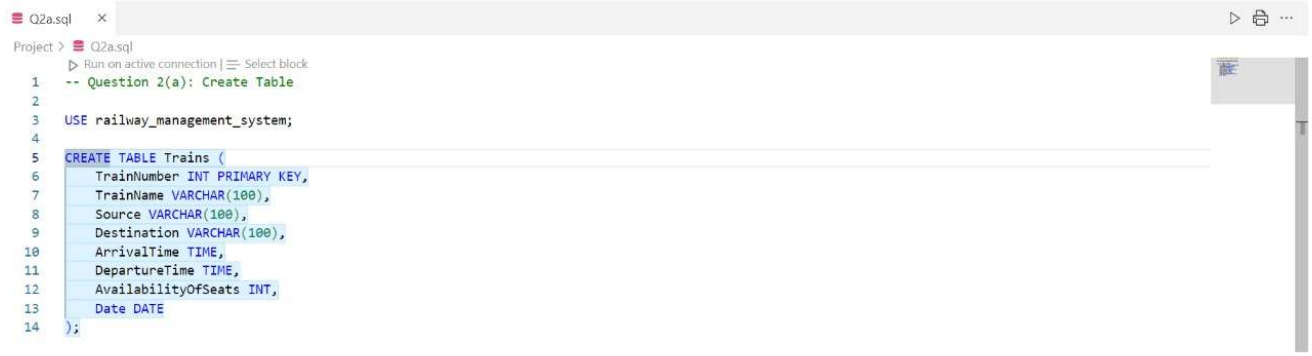
## E-R Diagram:


a_seats1
train_no
a_seats2
w_seats1
b_seats1
w_seats2
b_seats2
TRAIN_STATUS
fare1
fare2
N
departure_time
train_no
arrival_time
train_name
availability_of seats
source
TRAIN
N
1
date
destination
no
name
stops_at
STATION
starts
arrival_time
N
N
N
1
1
hault
user_id
password
state
city
pincode
first_name
last_name
cancel
aadhar_no
adress
N
1
mobile_no
USER
reaches
source
destination
books
email
age
N
N
gender
1
N
TICKET
security_ans
no_of_passengers
security_ques
N
ticket_id
passenger_id
PASSENGER
gender
name
id
status
booked_user
booked by
pnr_no
age
reservation status
seat_number

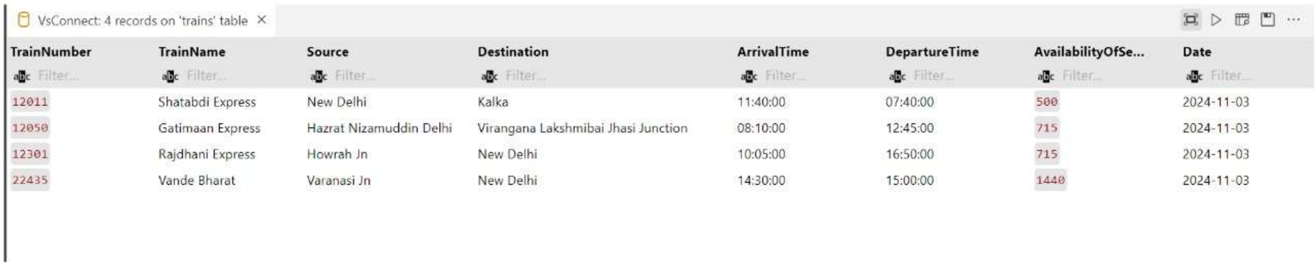
# Experiment 2

**Aim**: Implementation of the DDL statements:

a) Create

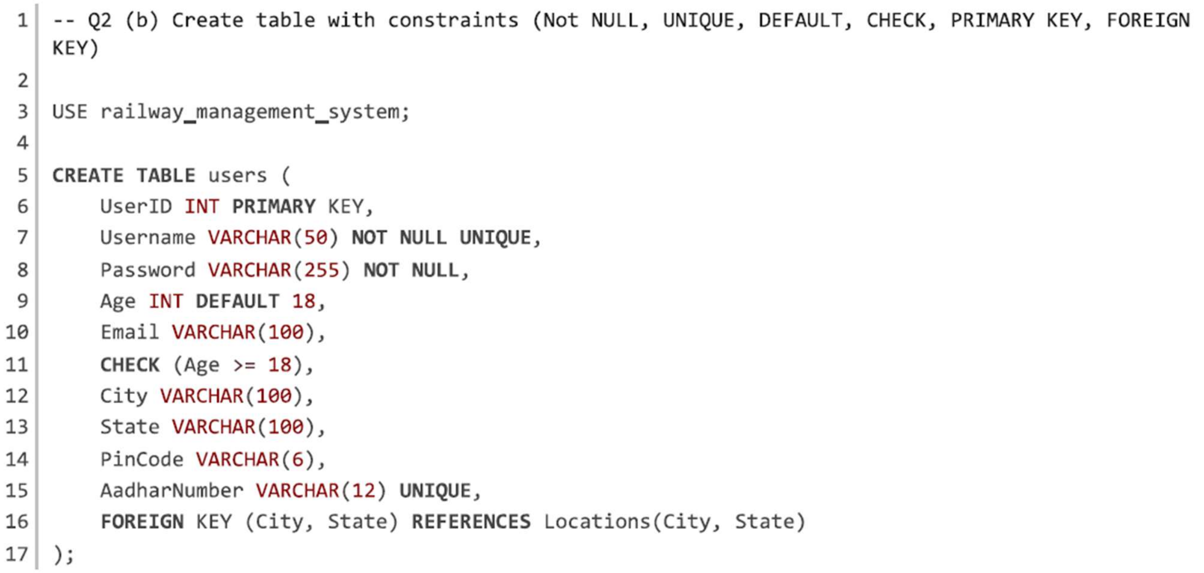


```sql
-- Question 2(a): Create Table

USE railway_management_system;

CREATE TABLE Trains (
    TrainNumber INT PRIMARY KEY,
    TrainName VARCHAR(100),
    Source VARCHAR(100),
    Destination VARCHAR(100),
    ArrivalTime TIME,
    DepartureTime TIME,
    AvailabilityOfSeats INT,
    Date DATE
);
```

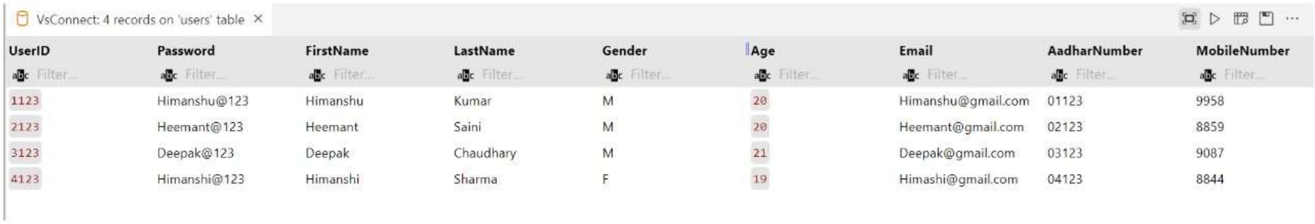


| TrainNumber | TrainName | Source | Destination | ArrivalTime | DepartureTime | AvailabilityOfSe... | Date |
|---|---|---|---|---|---|---|---|
| 12011 | Shatabdi Express | New Delhi | Kalka | 11:40:00 | 07:40:00 | 500 | 2024-11-03 |
| 12050 | Gatimaan Express | Hazrat Nizamuddin Delhi | Virangana Lakshmibai Jhasi Junction | 08:10:00 | 12:45:00 | 715 | 2024-11-03 |
| 12301 | Rajdhani Express | Howrah Jn | New Delhi | 10:05:00 | 16:50:00 | 715 | 2024-11-03 |
| 22435 | Vande Bharat | Varanasi Jn | New Delhi | 14:30:00 | 15:00:00 | 1440 | 2024-11-03 |

b) Create table with constraints



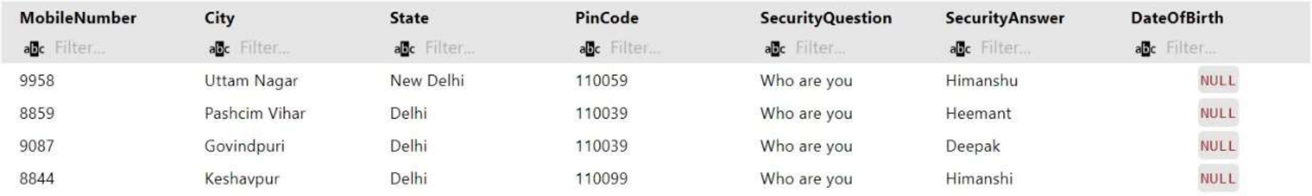
```sql
-- Q2 (b) Create table with constraints (Not NULL, UNIQUE, DEFAULT, CHECK, PRIMARY KEY, FOREIGN KEY)

USE railway_management_system;

CREATE TABLE users (
    UserID INT PRIMARY KEY,
    Username VARCHAR(50) NOT NULL UNIQUE,
    Password VARCHAR(255) NOT NULL,
    Age INT DEFAULT 18,
    Email VARCHAR(100),
    CHECK (Age >= 18),
    City VARCHAR(100),
    State VARCHAR(100),
    PinCode VARCHAR(6),
    AadharNumber VARCHAR(12) UNIQUE,
    FOREIGN KEY (City, State) REFERENCES Locations(City, State)
);
```



| UserID | Password | FirstName | LastName | Gender | Age | Email | AadharNumber | MobileNumber |
|---|---|---|---|---|---|---|---|---|
| 1123 | Himanshu@123 | Himanshu | Kumar | M | 20 | Himanshu@gmail.com | 01123 | 9958 |
| 2123 | Heemant@123 | Heemant | Saini | M | 20 | Heemant@gmail.com | 02123 | 8859 |
| 3123 | Deepak@123 | Deepak | Chaudhary | M | 21 | Deepak@gmail.com | 03123 | 9087 |
| 4123 | Himanshi@123 | Himanshi | Sharma | F | 19 | Himashi@gmail.com | 04123 | 8844 |

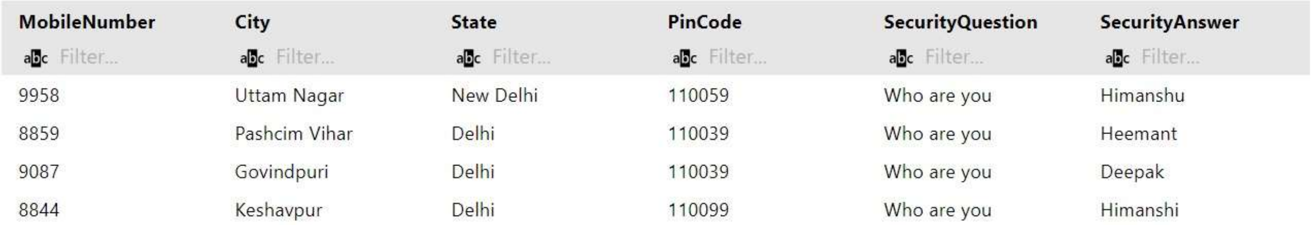
c) Alter table

1. Add column



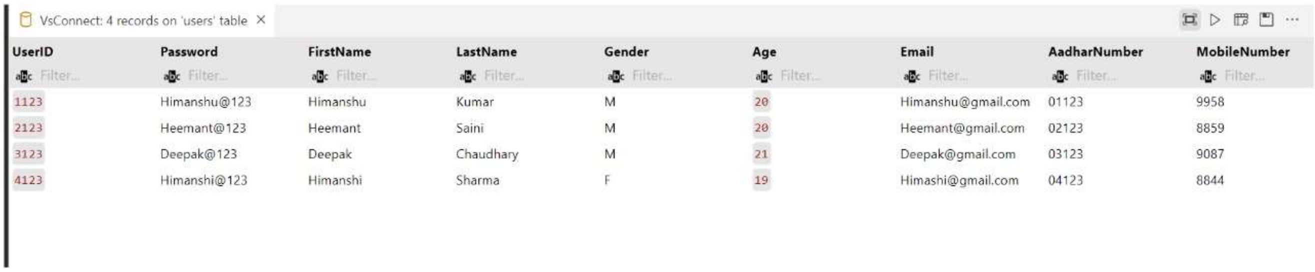
| MobileNumber | City | State | PinCode | SecurityQuestion | SecurityAnswer | DateOfBirth |
|---|---|---|---|---|---|---|
| 9958 | Uttam Nagar | New Delhi | 110059 | Who are you | Himanshu | NULL |
| 8859 | Pashcim Vihar | Delhi | 110039 | Who are you | Heemant | NULL |
| 9087 | Govindpuri | Delhi | 110039 | Who are you | Deepak | NULL |
| 8844 | Keshavpur | Delhi | 110099 | Who are you | Himanshi | NULL |

2. Drop column



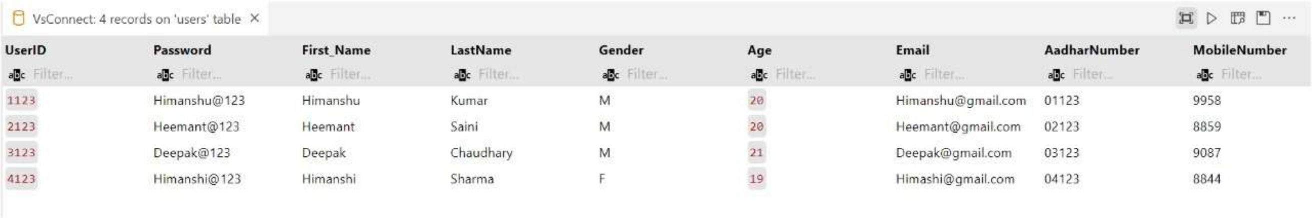
| MobileNumber | City | State | PinCode | SecurityQuestion | SecurityAnswer |
|---|---|---|---|---|---|
| 9958 | Uttam Nagar | New Delhi | 110059 | Who are you | Himanshu |
| 8859 | Pashcim Vihar | Delhi | 110039 | Who are you | Heemant |
| 9087 | Govindpuri | Delhi | 110039 | Who are you | Deepak |
| 8844 | Keshavpur | Delhi | 110099 | Who are you | Himanshi |

3. Add/Drop constraint



| UserID | Password | FirstName | LastName | Gender | Age | Email | AadharNumber | MobileNumber |
|---|---|---|---|---|---|---|---|---|
| 1123 | Himanshu@123 | Himanshu | Kumar | M | 20 | Himanshu@gmail.com | 01123 | 9958 |
| 2123 | Heemant@123 | Heemant | Saini | M | 20 | Heemant@gmail.com | 02123 | 8859 |
| 3123 | Deepak@123 | Deepak | Chaudhary | M | 21 | Deepak@gmail.com | 03123 | 9087 |
| 4123 | Himanshi@123 | Himanshi | Sharma | F | 19 | Himashi@gmail.com | 04123 | 8844 |

4. Rename column

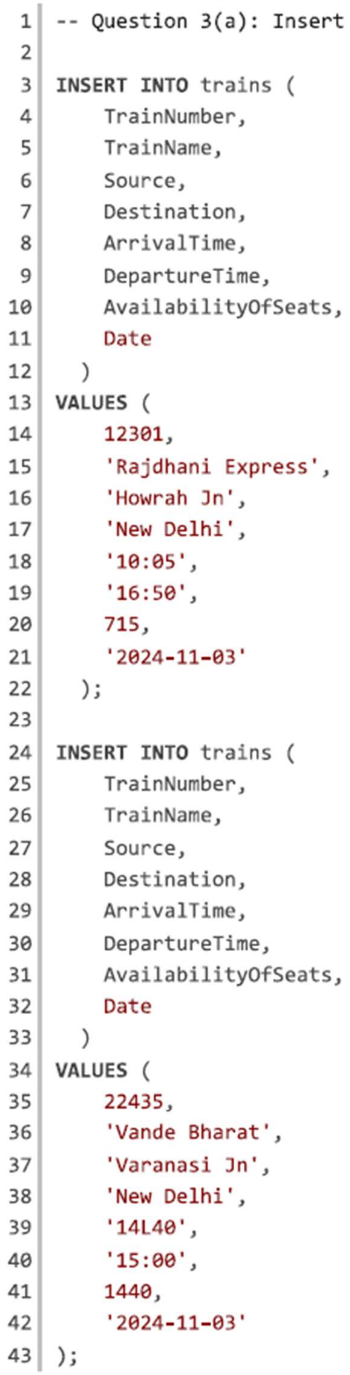


| UserID | Password | First_Name | LastName | Gender | Age | Email | AadharNumber | MobileNumber |
|---|---|---|---|---|---|---|---|---|
| 1123 | Himanshu@123 | Himanshu | Kumar | M | 20 | Himanshu@gmail.com | 01123 | 9958 |
| 2123 | Heemant@123 | Heemant | Saini | M | 20 | Heemant@gmail.com | 02123 | 8859 |
| 3123 | Deepak@123 | Deepak | Chaudhary | M | 21 | Deepak@gmail.com | 03123 | 9087 |
| 4123 | Himanshi@123 | Himanshi | Sharma | F | 19 | Himashi@gmail.com | 04123 | 8844 |

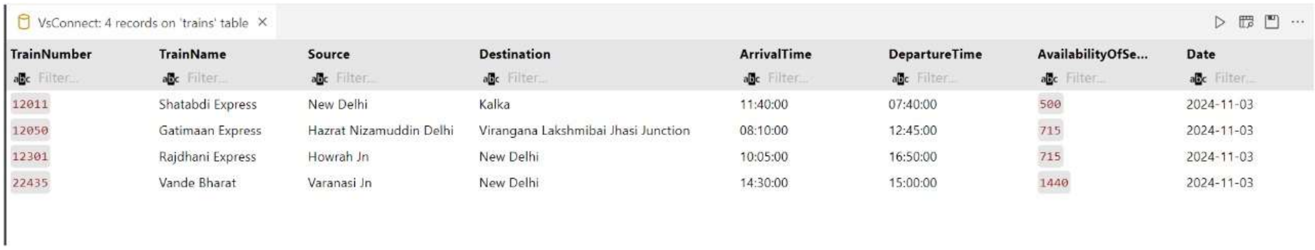
# Experiment 3

**Aim**: Implementation of the DML statements:

a) Insert



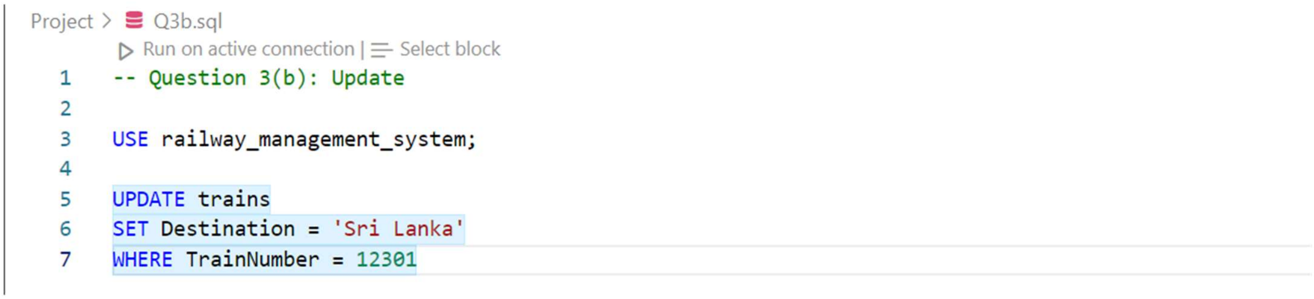
```
-- Question 3(a): Insert

INSERT INTO trains (
    TrainNumber,
    TrainName,
    Source,
    Destination,
    ArrivalTime,
    DepartureTime,
    AvailabilityOfSeats,
    Date
  )
VALUES (
    12301,
    'Rajdhani Express',
    'Howrah Jn',
    'New Delhi',
    '10:05',
    '16:50',
    715,
    '2024-11-03'
  );

INSERT INTO trains (
    TrainNumber,
    TrainName,
    Source,
    Destination,
    ArrivalTime,
    DepartureTime,
    AvailabilityOfSeats,
    Date
  )
VALUES (
    22435,
    'Vande Bharat',
    'Varanasi Jn',
    'New Delhi',
    '14L40',
    '15:00',
    1440,
    '2024-11-03'
);
```



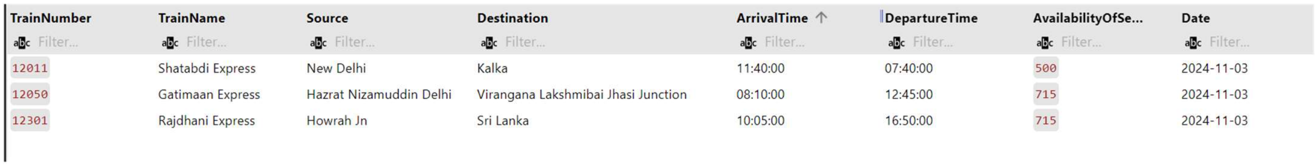
VsConnect: 4 records on 'trains' table

| TrainNumber | TrainName | Source | Destination | ArrivalTime | DepartureTime | AvailabilityOfSe... | Date |
|---|---|---|---|---|---|---|---|
| 12011 | Shatabdi Express | New Delhi | Kalka | 11:40:00 | 07:40:00 | 500 | 2024-11-03 |
| 12050 | Gatimaan Express | Hazrat Nizamuddin Delhi | Virangana Lakshmibai Jhasi Junction | 08:10:00 | 12:45:00 | 715 | 2024-11-03 |
| 12301 | Rajdhani Express | Howrah Jn | New Delhi | 10:05:00 | 16:50:00 | 715 | 2024-11-03 |
| 22435 | Vande Bharat | Varanasi Jn | New Delhi | 14:30:00 | 15:00:00 | 1440 | 2024-11-03 |

## b) Update

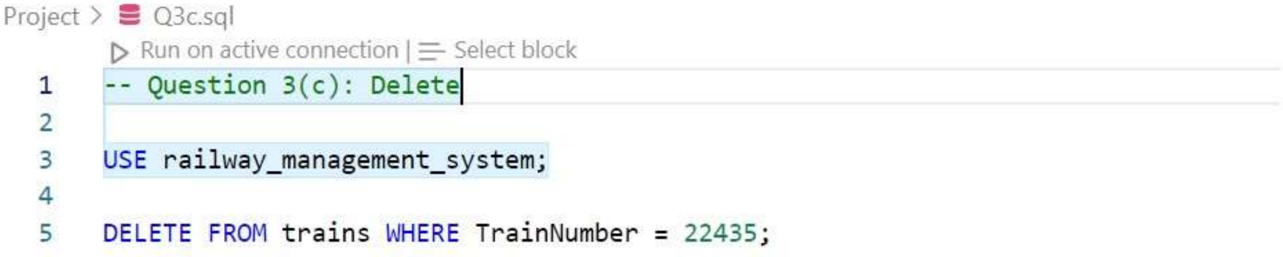


```sql
-- Question 3(b): Update

USE railway_management_system;

UPDATE trains
SET Destination = 'Sri Lanka'
WHERE TrainNumber = 12301
```

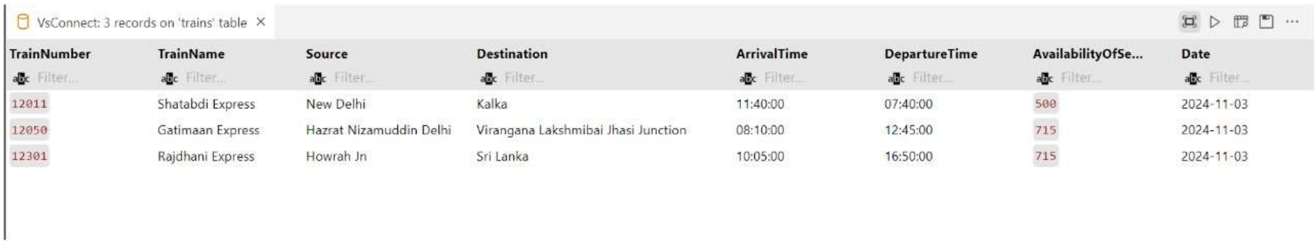


| TrainNumber | TrainName | Source | Destination | ArrivalTime | DepartureTime | AvailabilityOfSe... | Date |
|---|---|---|---|---|---|---|---|
| 12011 | Shatabdi Express | New Delhi | Kalka | 11:40:00 | 07:40:00 | 500 | 2024-11-03 |
| 12050 | Gatimaan Express | Hazrat Nizamuddin Delhi | Virangana Lakshmibai Jhasi Junction | 08:10:00 | 12:45:00 | 715 | 2024-11-03 |
| 12301 | Rajdhani Express | Howrah Jn | Sri Lanka | 10:05:00 | 16:50:00 | 715 | 2024-11-03 |

## c) Delete



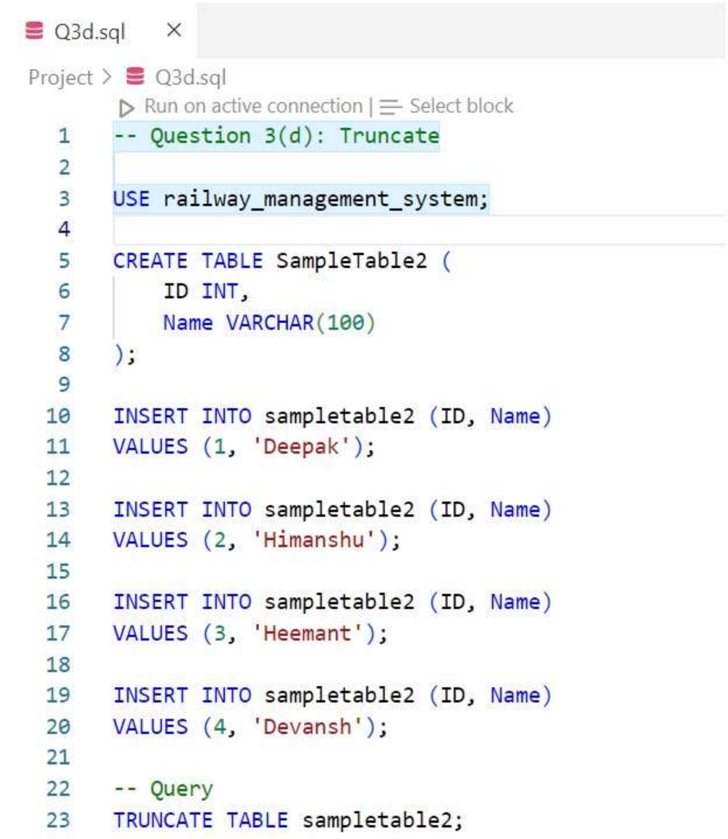
```sql
-- Question 3(c): Delete

USE railway_management_system;

DELETE FROM trains WHERE TrainNumber = 22435;
```



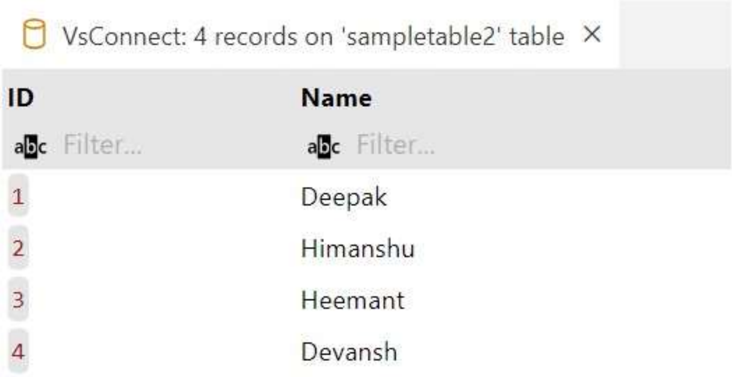
VsConnect: 3 records on 'trains' table

| TrainNumber | TrainName | Source | Destination | ArrivalTime | DepartureTime | AvailabilityOfSe... | Date |
|---|---|---|---|---|---|---|---|
| 12011 | Shatabdi Express | New Delhi | Kalka | 11:40:00 | 07:40:00 | 500 | 2024-11-03 |
| 12050 | Gatimaan Express | Hazrat Nizamuddin Delhi | Virangana Lakshmibai Jhasi Junction | 08:10:00 | 12:45:00 | 715 | 2024-11-03 |
| 12301 | Rajdhani Express | Howrah Jn | Sri Lanka | 10:05:00 | 16:50:00 | 715 | 2024-11-03 |

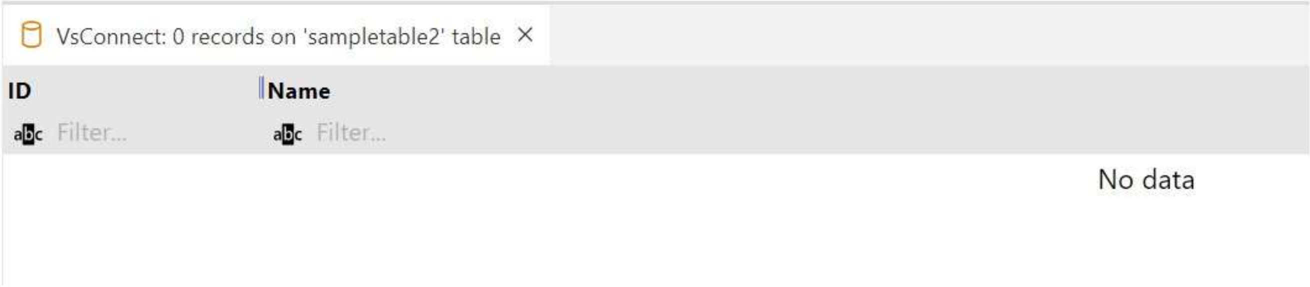
## d) Truncate



```sql
-- Question 3(d): Truncate

USE railway_management_system;

CREATE TABLE SampleTable2 (
    ID INT,
    Name VARCHAR(100)
);

INSERT INTO sampletable2 (ID, Name)
VALUES (1, 'Deepak');

INSERT INTO sampletable2 (ID, Name)
VALUES (2, 'Himanshu');

INSERT INTO sampletable2 (ID, Name)
VALUES (3, 'Heemant');

INSERT INTO sampletable2 (ID, Name)
VALUES (4, 'Devansh');

-- Query
TRUNCATE TABLE sampletable2;
```


VsConnect: 4 records on 'sampletable2' table ×

| ID | Name |
|---|---|
| 1 | Deepak |
| 2 | Himanshu |
| 3 | Heemant |
| 4 | Devansh |


VsConnect: 0 records on 'sampletable2' table ×

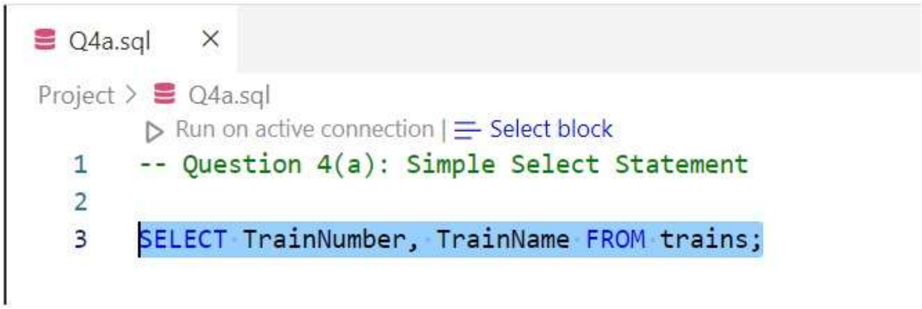
| ID | Name |
|---|---|

No data

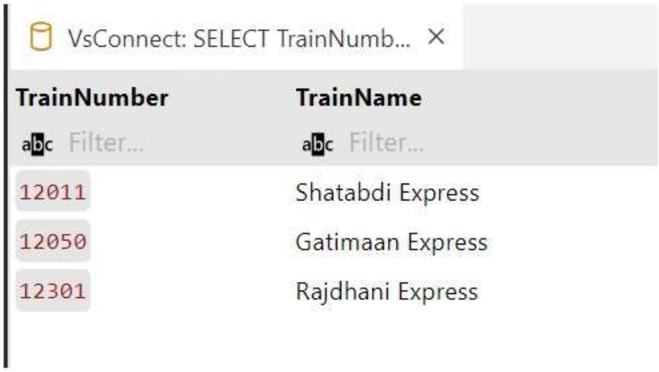
# Experiment 4

**Aim**: Implementation of the SELECT statements:

a) Simple SELECT statement

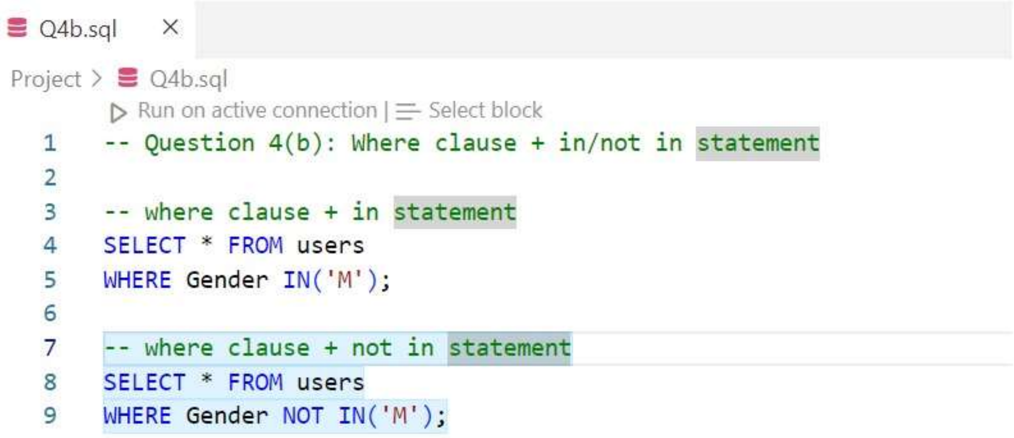


```sql
-- Question 4(a): Simple Select Statement

SELECT TrainNumber, TrainName FROM trains;
```

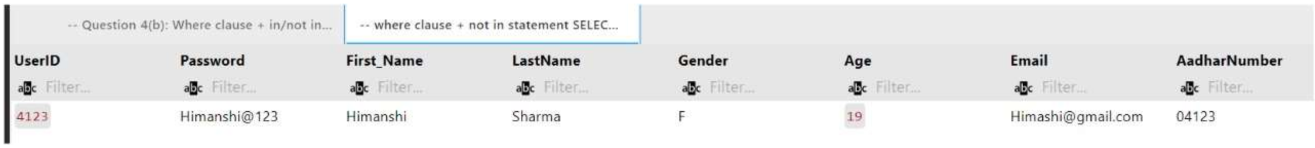


| TrainNumber | TrainName |
|---|---|
| 12011 | Shatabdi Express |
| 12050 | Gatimaan Express |
| 12301 | Rajdhani Express |

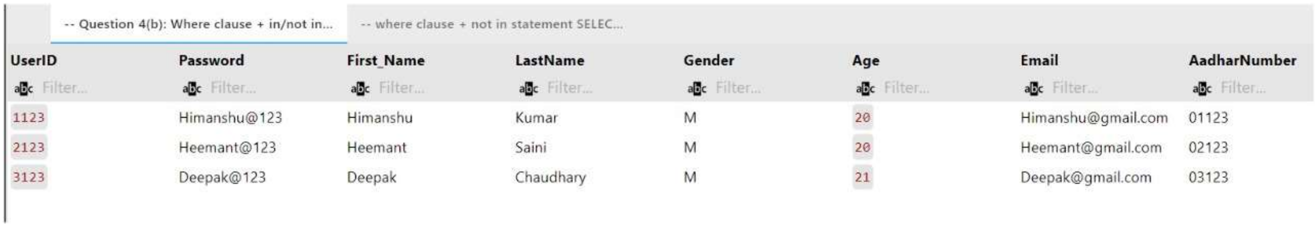
b) Where clause + IN/NOT IN



```sql
-- Question 4(b): Where clause + in/not in statement

-- where clause + in statement
SELECT * FROM users
WHERE Gender IN('M');

-- where clause + not in statement
SELECT * FROM users
WHERE Gender NOT IN('M');
```

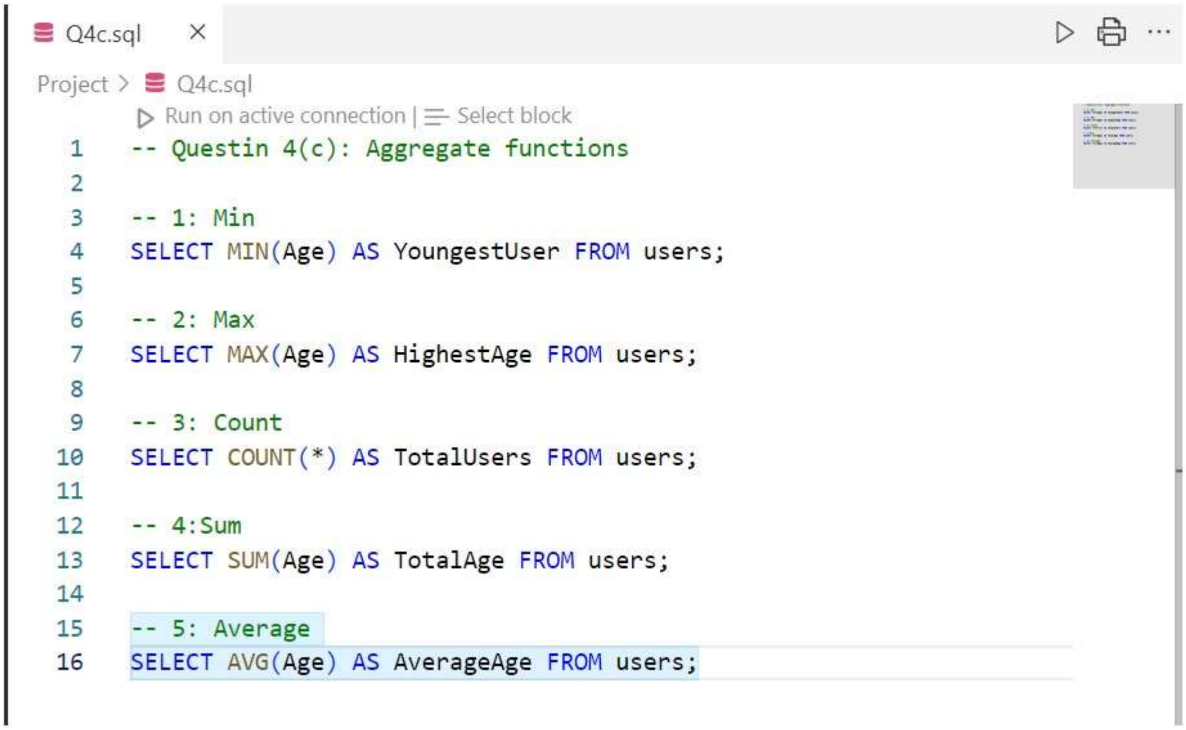

-- Question 4(b): Where clause + in/not in...
-- where clause + not in statement SELEC...
Filter...

| UserID | Password | First_Name | LastName | Gender | Age | Email | AadharNumber |
|---|---|---|---|---|---|---|---|
| 4123 | Himanshi@123 | Himanshi | Sharma | F | 19 | Himashi@gmail.com | 04123 |

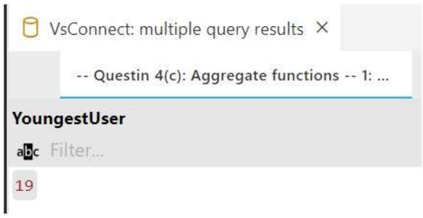

-- Question 4(b): Where clause + in/not in...
-- where clause + not in statement SELEC...

| UserID | Password | First_Name | LastName | Gender | Age | Email | AadharNumber |
|---|---|---|---|---|---|---|---|
| 1123 | Himanshu@123 | Himanshu | Kumar | M | 20 | Himanshu@gmail.com | 01123 |
| 2123 | Heemant@123 | Heemant | Saini | M | 20 | Heemant@gmail.com | 02123 |
| 3123 | Deepak@123 | Deepak | Chaudhary | M | 21 | Deepak@gmail.com | 03123 |

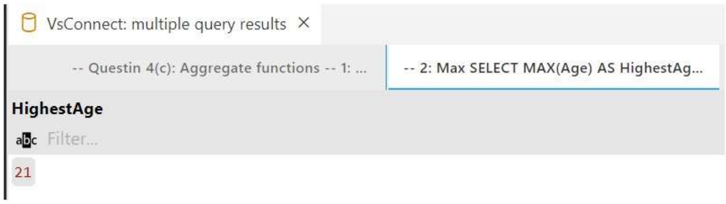
c) Aggregate functions



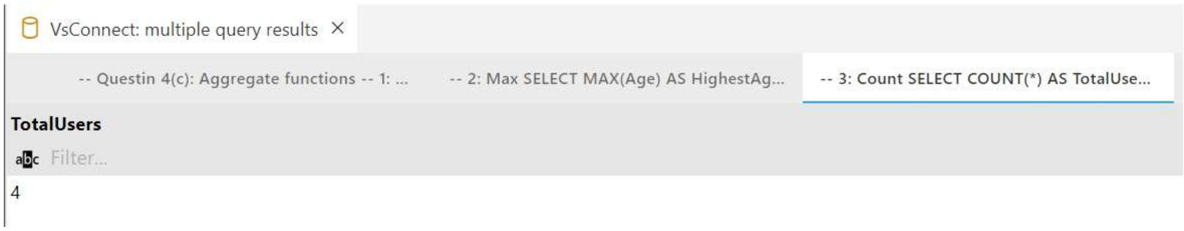
```
-- Questin 4(c): Aggregate functions

-- 1: Min
SELECT MIN(Age) AS YoungestUser FROM users;

-- 2: Max
SELECT MAX(Age) AS HighestAge FROM users;

-- 3: Count
SELECT COUNT(*) AS TotalUsers FROM users;

-- 4:Sum
SELECT SUM(Age) AS TotalAge FROM users;

-- 5: Average
SELECT AVG(Age) AS AverageAge FROM users;
```


VsConnect: multiple query results
-- Questin 4(c): Aggregate functions -- 1: ...

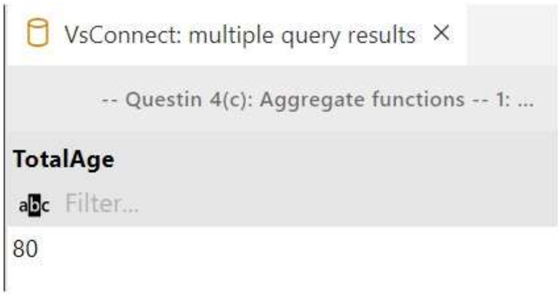
| YoungestUser |
|---|
| 19 |


VsConnect: multiple query results
-- Questin 4(c): Aggregate functions -- 1: ...
-- 2: Max SELECT MAX(Age) AS HighestAg...

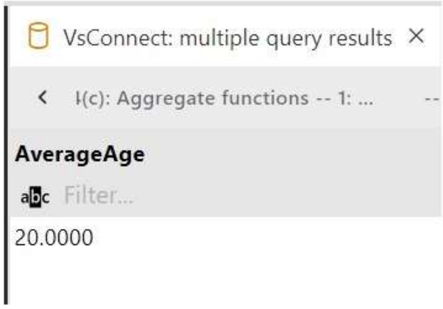
| HighestAge |
|---|
| 21 |


VsConnect: multiple query results
-- Questin 4(c): Aggregate functions -- 1: ...
-- 2: Max SELECT MAX(Age) AS HighestAg...
-- 3: Count SELECT COUNT(*) AS TotalUse...

| TotalUsers |
|---|
| 4 |

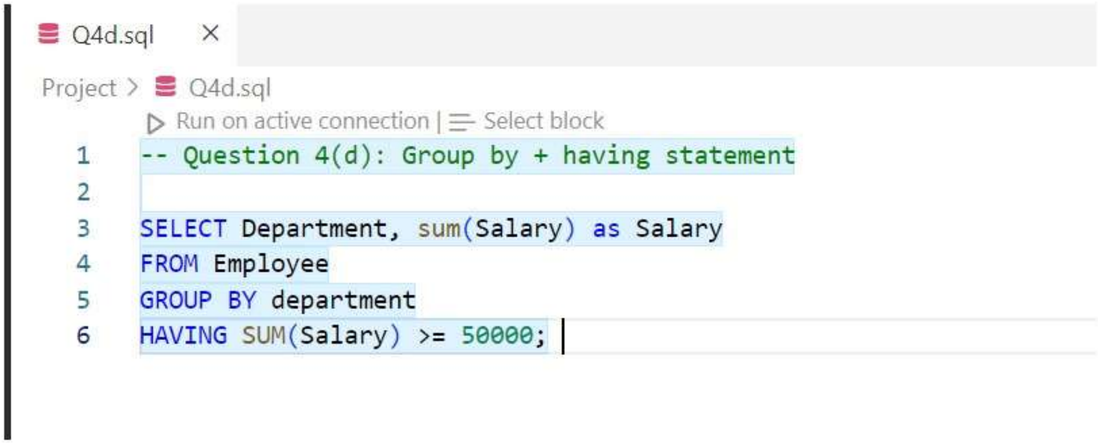

VsConnect: multiple query results
-- Questin 4(c): Aggregate functions -- 1: ...
Filter...

| TotalAge |
|---|
| 80 |

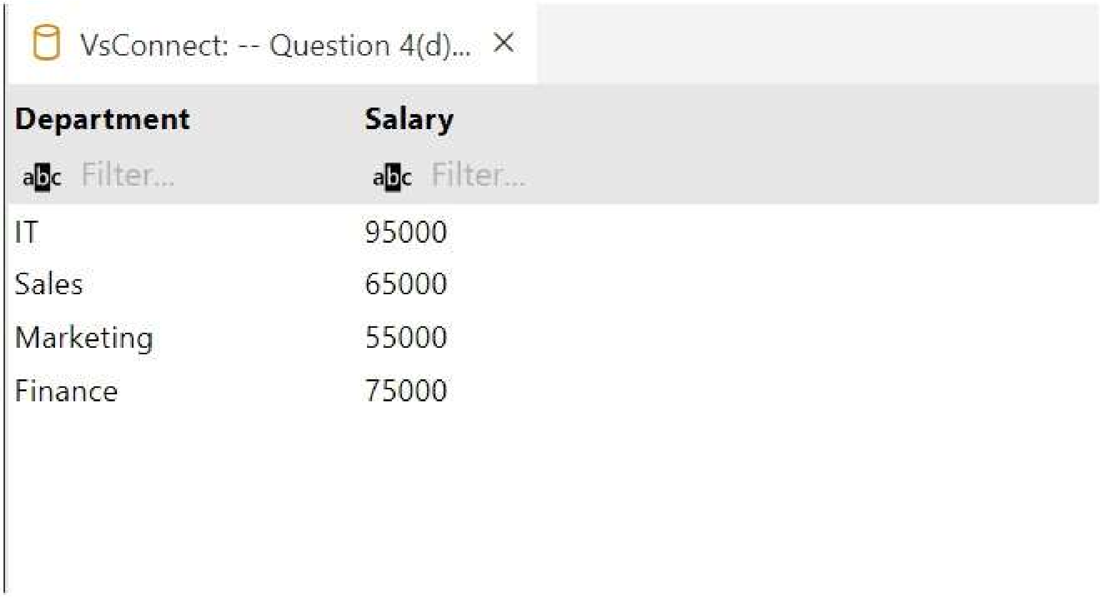

VsConnect: multiple query results
ı(c): Aggregate functions -- 1: ...
Filter...

| AverageAge |
|---|
| 20.0000 |

d) Group by + Having

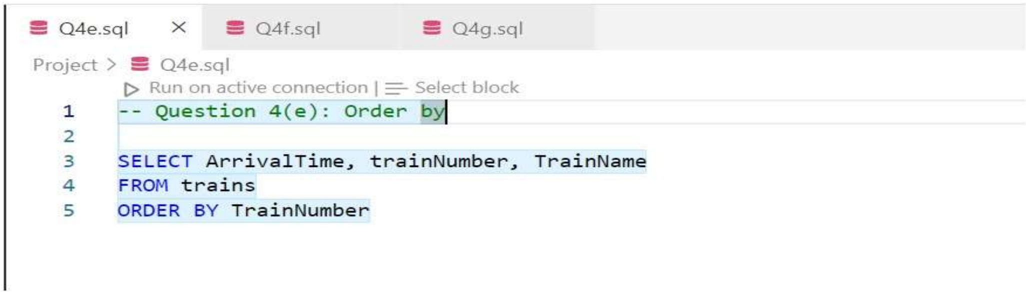


```sql
-- Question 4(d): Group by + having statement

SELECT Department, sum(Salary) as Salary
FROM Employee
GROUP BY department
HAVING SUM(Salary) >= 50000;
```

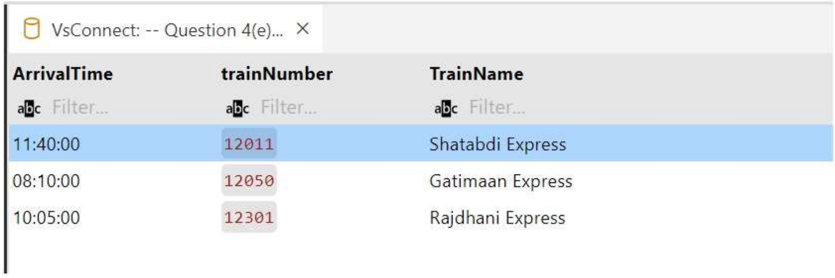


| Department | Salary |
|---|---|
| IT | 95000 |
| Sales | 65000 |
| Marketing | 55000 |
| Finance | 75000 |

e) Order by

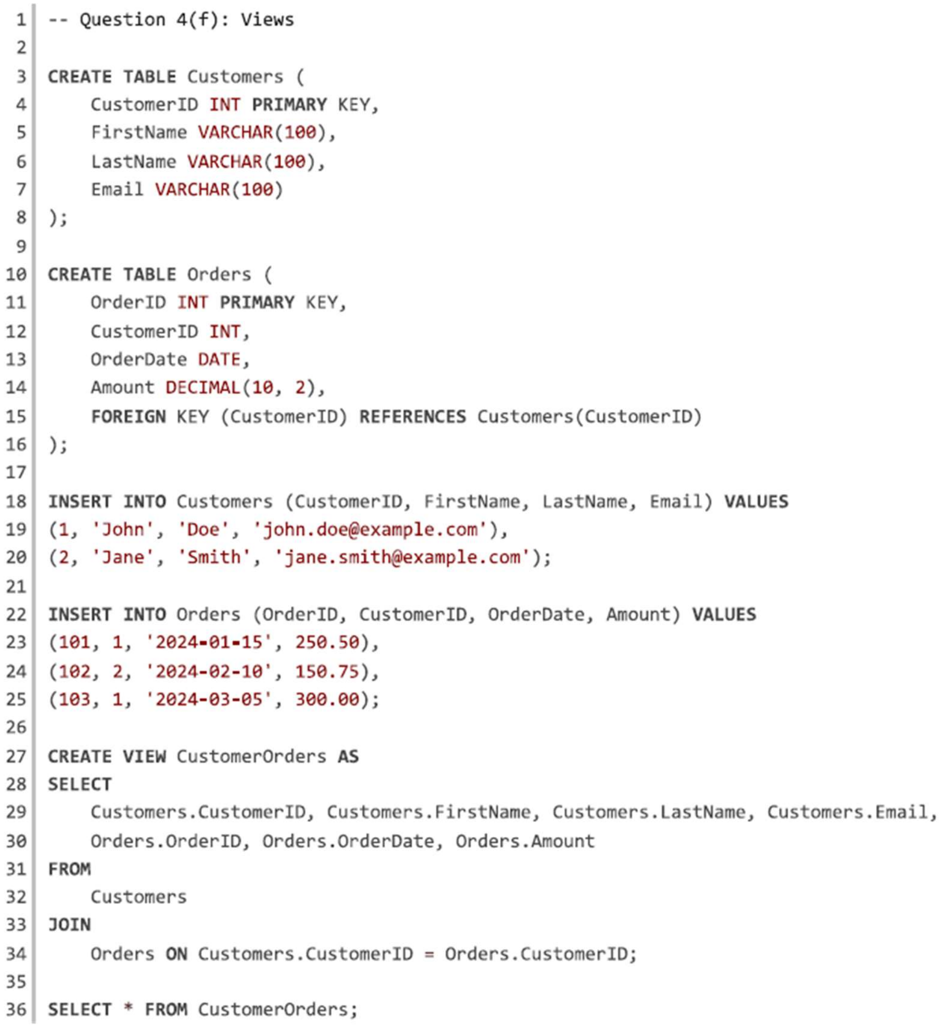


```sql
-- Question 4(e): Order by

SELECT ArrivalTime, trainNumber, TrainName
FROM trains
ORDER BY TrainNumber
```

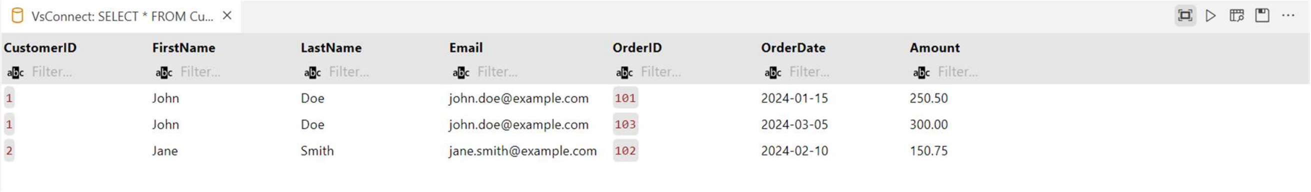


| ArrivalTime | trainNumber | TrainName |
|---|---|---|
| 11:40:00 | 12011 | Shatabdi Express |
| 08:10:00 | 12050 | Gatimaan Express |
| 10:05:00 | 12301 | Rajdhani Express |

## f) Views



```
-- Question 4(f): Views

CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Email VARCHAR(100)
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    Amount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO Customers (CustomerID, FirstName, LastName, Email) VALUES
(1, 'John', 'Doe', 'john.doe@example.com'),
(2, 'Jane', 'Smith', 'jane.smith@example.com');

INSERT INTO Orders (OrderID, CustomerID, OrderDate, Amount) VALUES
(101, 1, '2024-01-15', 250.50),
(102, 2, '2024-02-10', 150.75),
(103, 1, '2024-03-05', 300.00);

CREATE VIEW CustomerOrders AS
SELECT
    Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email,
    Orders.OrderID, Orders.OrderDate, Orders.Amount
FROM
    Customers
JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID;

SELECT * FROM CustomerOrders;
```

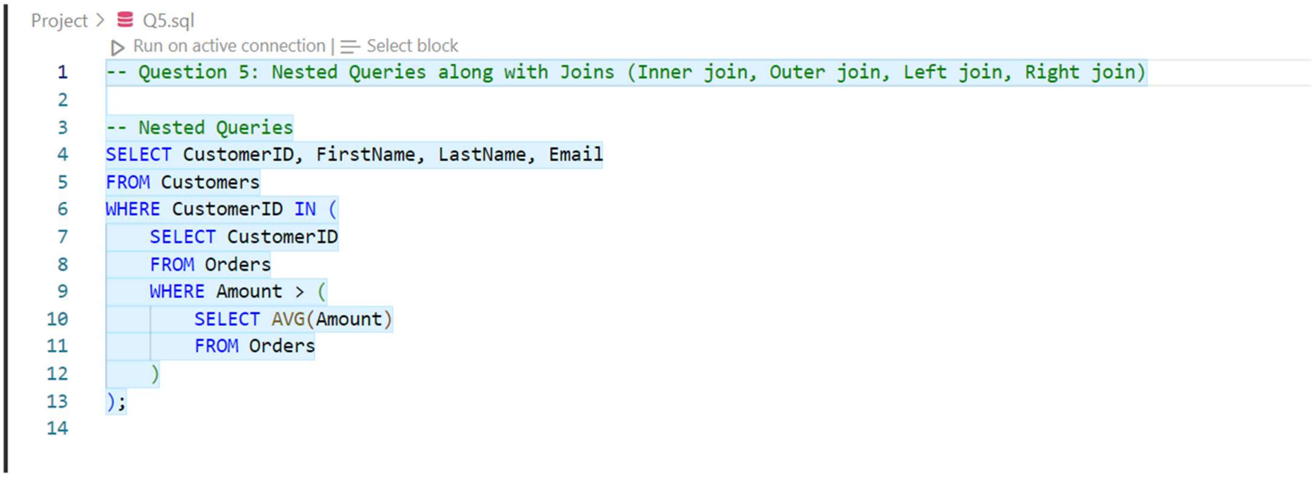


| CustomerID | FirstName | LastName | Email | OrderID | OrderDate | Amount |
|---|---|---|---|---|---|---|
| 1 | John | Doe | john.doe@example.com | 101 | 2024-01-15 | 250.50 |
| 1 | John | Doe | john.doe@example.com | 103 | 2024-03-05 | 300.00 |
| 2 | Jane | Smith | jane.smith@example.com | 102 | 2024-02-10 | 150.75 |

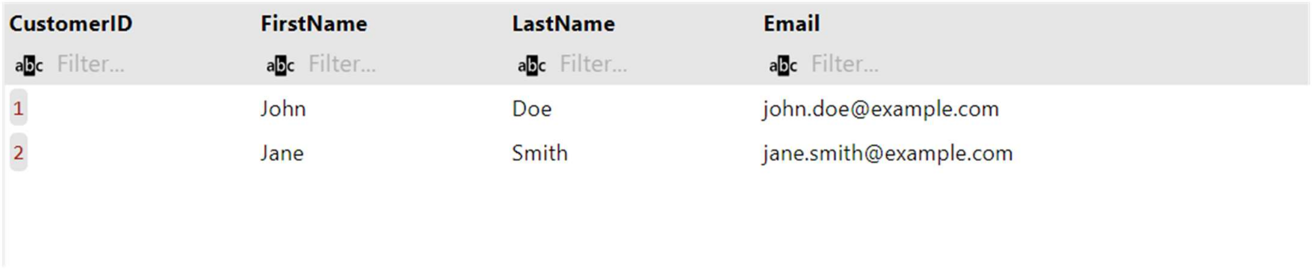
# Experiment 5

**Aim**: Implement and perform Nested Queries along with Joins (Inner join, Outer join, Left join, Right join).

## Nested Queries

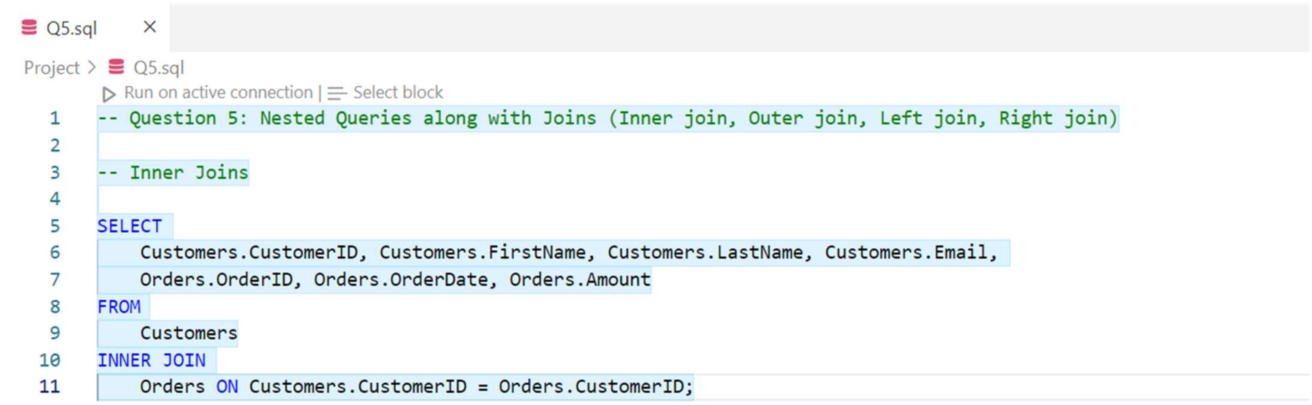


```sql
-- Question 5: Nested Queries along with Joins (Inner join, Outer join, Left join, Right join)

-- Nested Queries
SELECT CustomerID, FirstName, LastName, Email
FROM Customers
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE Amount > (
        SELECT AVG(Amount)
        FROM Orders
    )
);
```

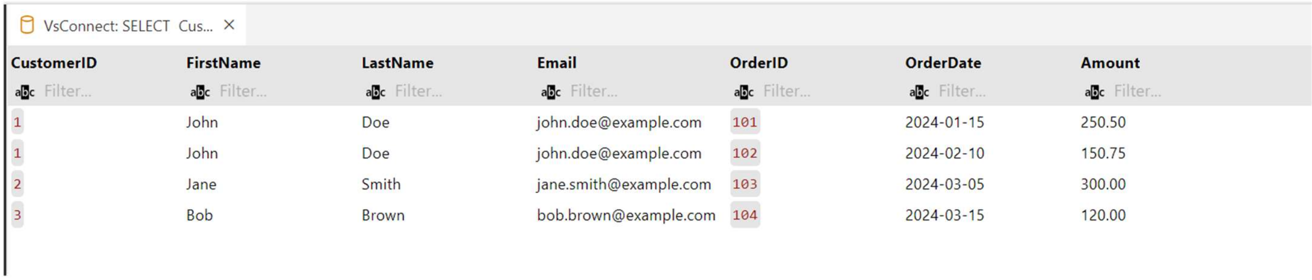


| | CustomerID | FirstName | LastName | Email |
|---|---|---|---|---|
| 1 | | John | Doe | john.doe@example.com |
| 2 | | Jane | Smith | jane.smith@example.com |

## Inner Join

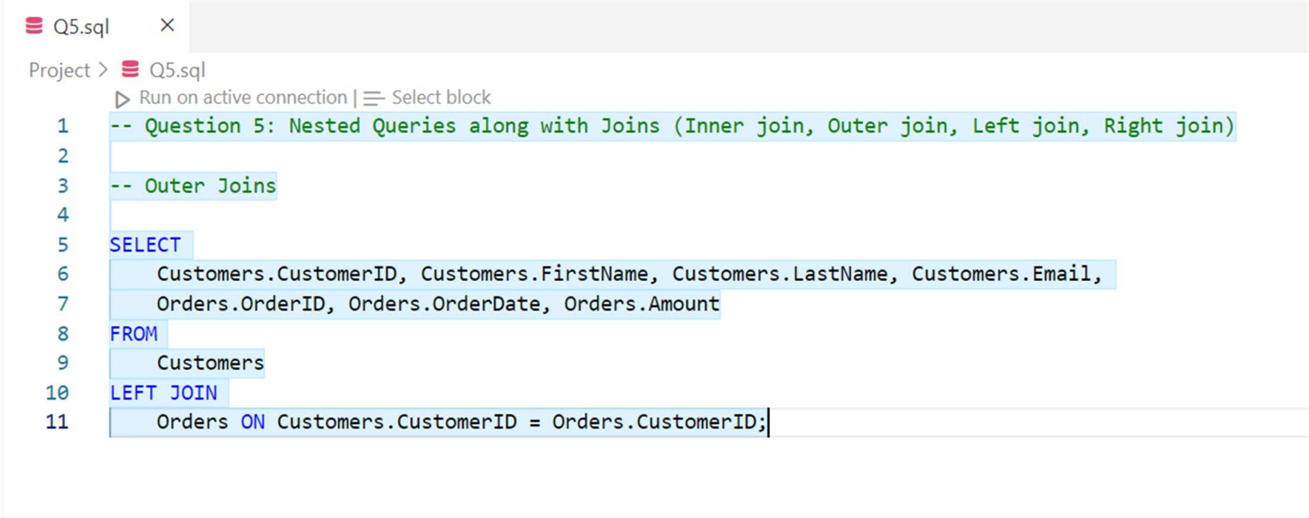


```sql
-- Question 5: Nested Queries along with Joins (Inner join, Outer join, Left join, Right join)

-- Inner Joins

SELECT
    Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email,
    Orders.OrderID, Orders.OrderDate, Orders.Amount
FROM
    Customers
INNER JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID;
```

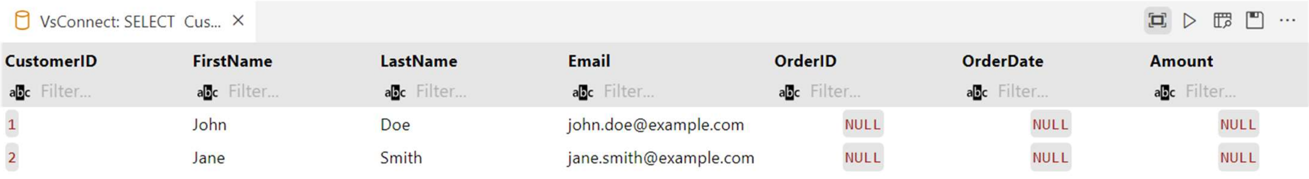


| CustomerID | FirstName | LastName | Email | OrderID | OrderDate | Amount |
|---|---|---|---|---|---|---|
| 1 | John | Doe | john.doe@example.com | 101 | 2024-01-15 | 250.50 |
| 1 | John | Doe | john.doe@example.com | 102 | 2024-02-10 | 150.75 |
| 2 | Jane | Smith | jane.smith@example.com | 103 | 2024-03-05 | 300.00 |
| 3 | Bob | Brown | bob.brown@example.com | 104 | 2024-03-15 | 120.00 |

## Outer Join

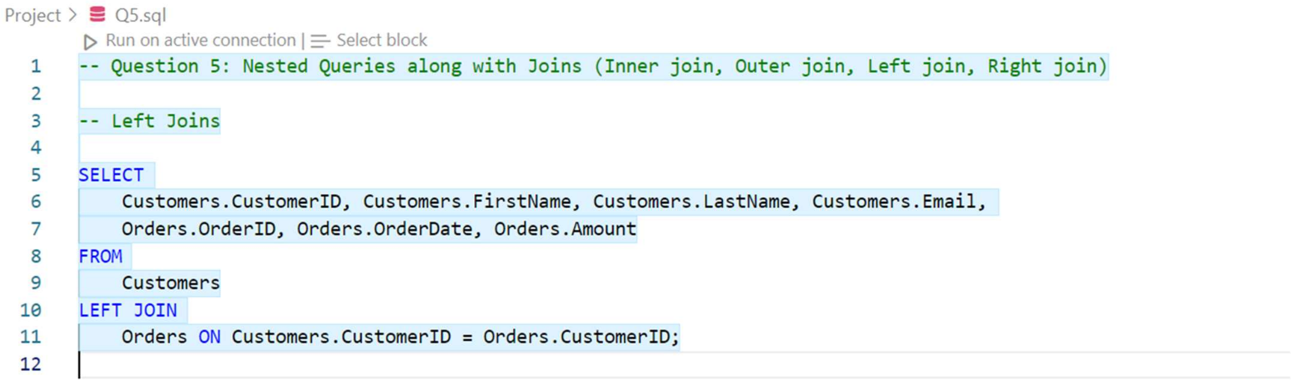


```sql
-- Question 5: Nested Queries along with Joins (Inner join, Outer join, Left join, Right join)

-- Outer Joins

SELECT
    Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email,
    Orders.OrderID, Orders.OrderDate, Orders.Amount
FROM
    Customers
LEFT JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID;
```

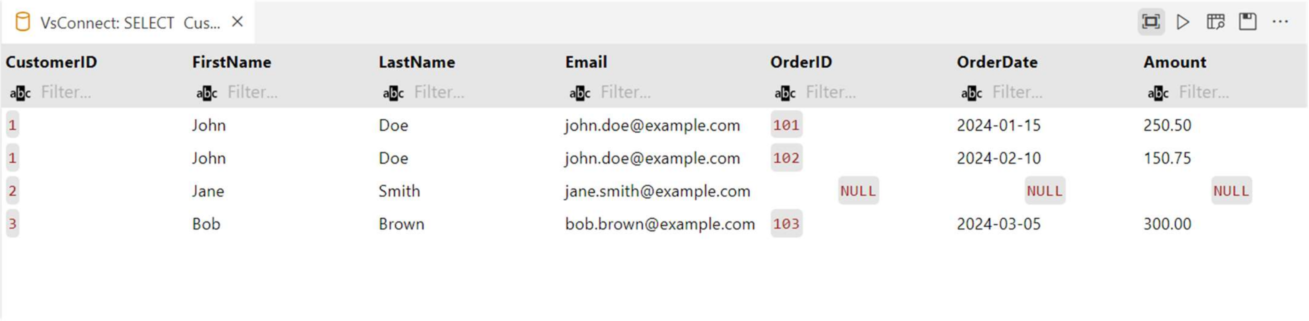


| CustomerID | FirstName | LastName | Email | OrderID | OrderDate | Amount |
|---|---|---|---|---|---|---|
| 1 | John | Doe | john.doe@example.com | NULL | NULL | NULL |
| 2 | Jane | Smith | jane.smith@example.com | NULL | NULL | NULL |

## Left Join

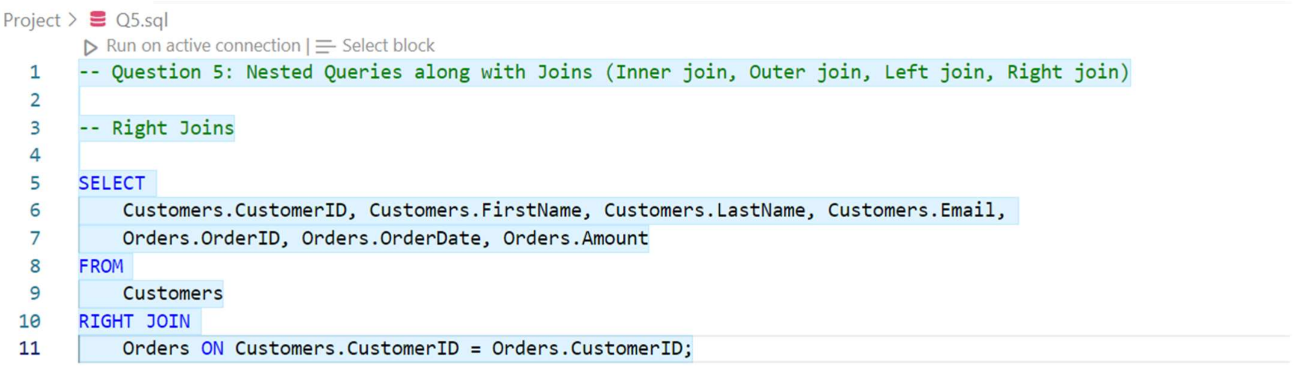


```sql
-- Question 5: Nested Queries along with Joins (Inner join, Outer join, Left join, Right join)

-- Left Joins

SELECT
    Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email,
    Orders.OrderID, Orders.OrderDate, Orders.Amount
FROM
    Customers
LEFT JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID;

```

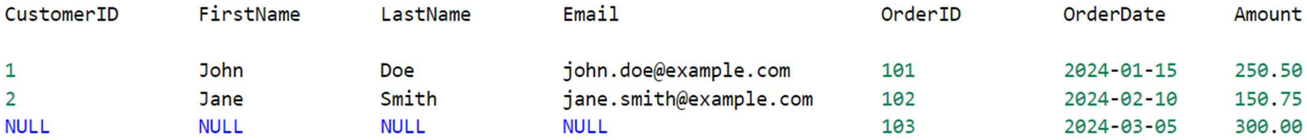


| CustomerID | FirstName | LastName | Email | OrderID | OrderDate | Amount |
|---|---|---|---|---|---|---|
| 1 | John | Doe | john.doe@example.com | 101 | 2024-01-15 | 250.50 |
| 1 | John | Doe | john.doe@example.com | 102 | 2024-02-10 | 150.75 |
| 2 | Jane | Smith | jane.smith@example.com | NULL | NULL | NULL |
| 3 | Bob | Brown | bob.brown@example.com | 103 | 2024-03-05 | 300.00 |

## Right Join



```sql
-- Question 5: Nested Queries along with Joins (Inner join, Outer join, Left join, Right join)

-- Right Joins

SELECT
    Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email,
    Orders.OrderID, Orders.OrderDate, Orders.Amount
FROM
    Customers
RIGHT JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID;
```

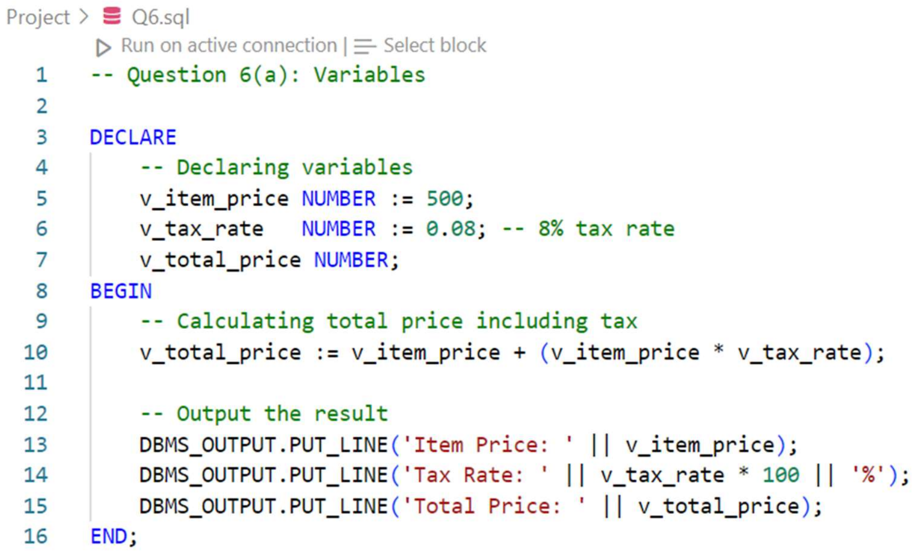


```
CustomerID    FirstName    LastName    Email                     OrderID    OrderDate     Amount

1             John         Doe         john.doe@example.com      101        2024-01-15    250.50
2             Jane         Smith       jane.smith@example.com    102        2024-02-10    150.75
NULL          NULL         NULL        NULL                      103        2024-03-05    300.00
```

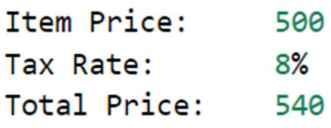
# Experiment 6

**Aim**: Introduction to PL/SQL. Create a PL/SQL block and implement the following:

a) Variables

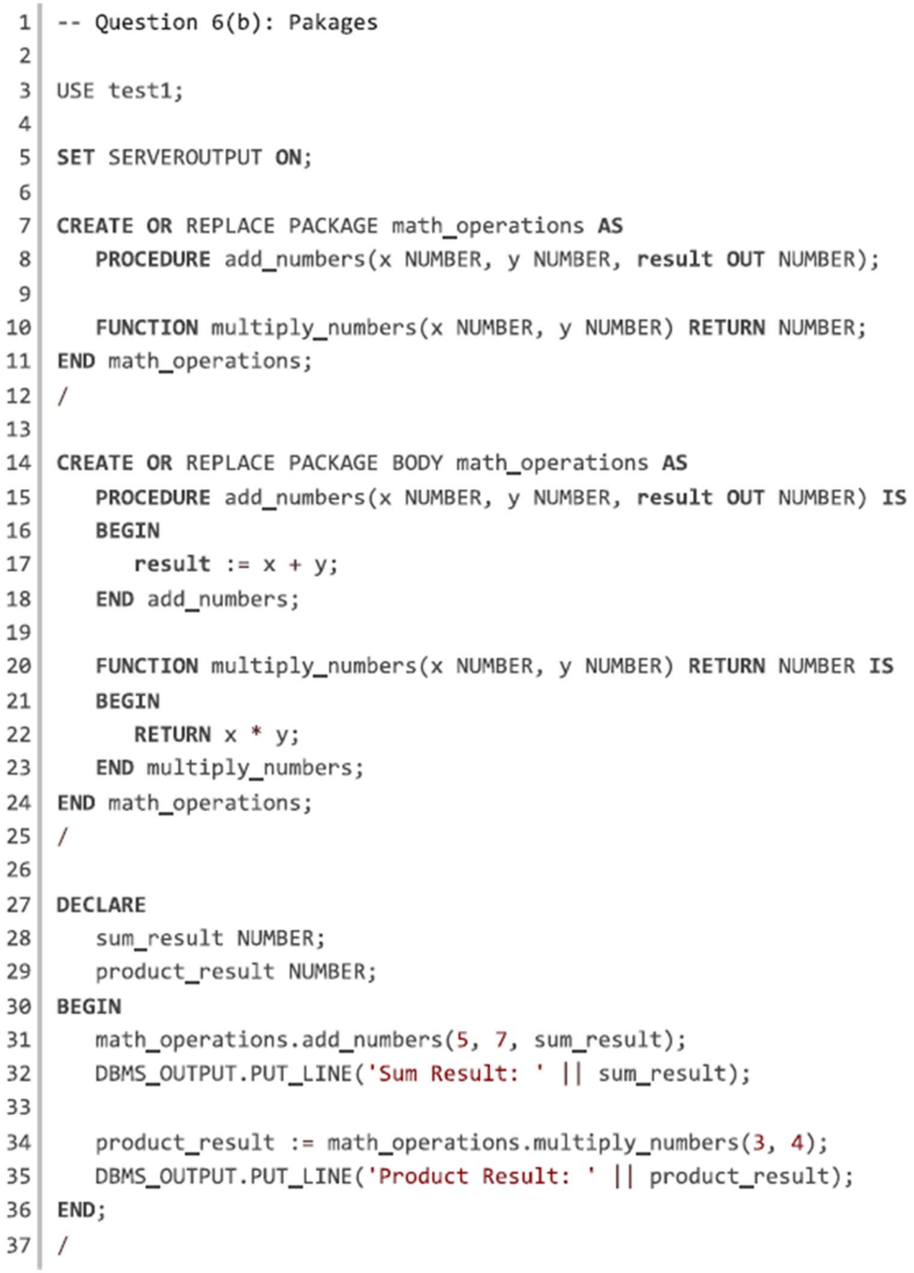


```sql
-- Question 6(a): Variables

DECLARE
    -- Declaring variables
    v_item_price NUMBER := 500;
    v_tax_rate   NUMBER := 0.08; -- 8% tax rate
    v_total_price NUMBER;
BEGIN
    -- Calculating total price including tax
    v_total_price := v_item_price + (v_item_price * v_tax_rate);

    -- Output the result
    DBMS_OUTPUT.PUT_LINE('Item Price: ' || v_item_price);
    DBMS_OUTPUT.PUT_LINE('Tax Rate: ' || v_tax_rate * 100 || '%');
    DBMS_OUTPUT.PUT_LINE('Total Price: ' || v_total_price);
END;
```

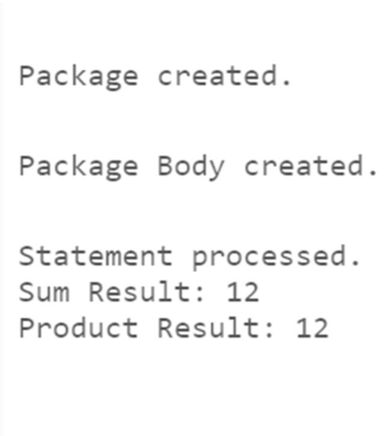


```
Item Price:     500
Tax Rate:       8%
Total Price:    540
```

## b) Packages

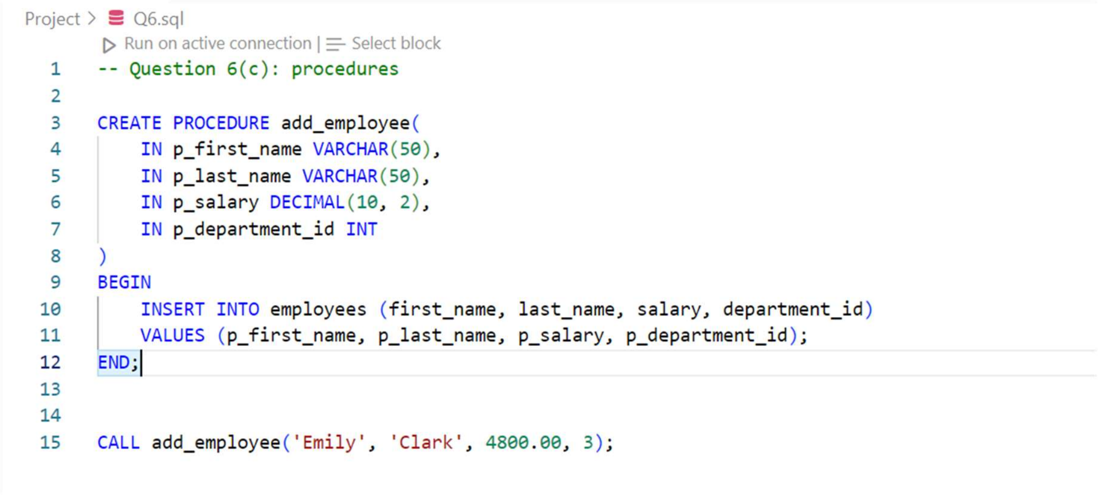


```
-- Question 6(b): Pakages

USE test1;

SET SERVEROUTPUT ON;

CREATE OR REPLACE PACKAGE math_operations AS
   PROCEDURE add_numbers(x NUMBER, y NUMBER, result OUT NUMBER);

   FUNCTION multiply_numbers(x NUMBER, y NUMBER) RETURN NUMBER;
END math_operations;
/

CREATE OR REPLACE PACKAGE BODY math_operations AS
   PROCEDURE add_numbers(x NUMBER, y NUMBER, result OUT NUMBER) IS
   BEGIN
      result := x + y;
   END add_numbers;

   FUNCTION multiply_numbers(x NUMBER, y NUMBER) RETURN NUMBER IS
   BEGIN
      RETURN x * y;
   END multiply_numbers;
END math_operations;
/

DECLARE
   sum_result NUMBER;
   product_result NUMBER;
BEGIN
   math_operations.add_numbers(5, 7, sum_result);
   DBMS_OUTPUT.PUT_LINE('Sum Result: ' || sum_result);

   product_result := math_operations.multiply_numbers(3, 4);
   DBMS_OUTPUT.PUT_LINE('Product Result: ' || product_result);
END;
/
```

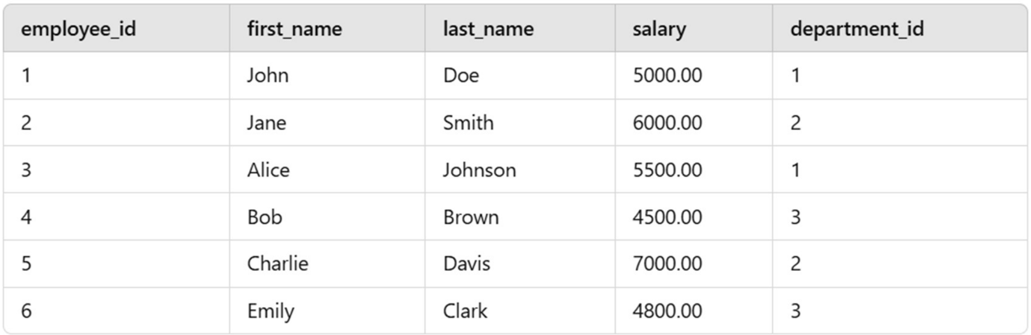


```
Package created.


Package Body created.


Statement processed.
Sum Result: 12
Product Result: 12
```

## c) Procedures

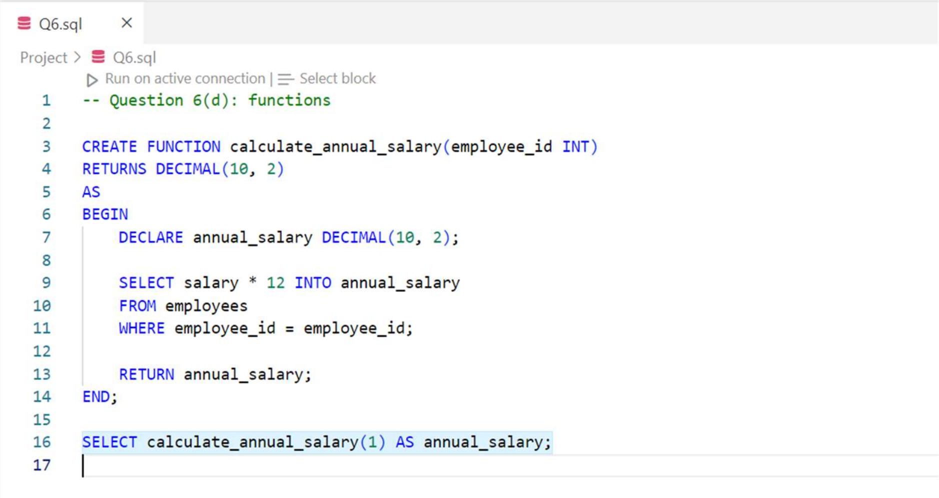


```sql
-- Question 6(c): procedures

CREATE PROCEDURE add_employee(
    IN p_first_name VARCHAR(50),
    IN p_last_name VARCHAR(50),
    IN p_salary DECIMAL(10, 2),
    IN p_department_id INT
)
BEGIN
    INSERT INTO employees (first_name, last_name, salary, department_id)
    VALUES (p_first_name, p_last_name, p_salary, p_department_id);
END;


CALL add_employee('Emily', 'Clark', 4800.00, 3);
```

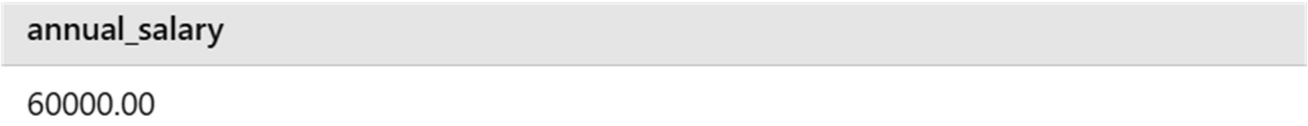


| employee_id | first_name | last_name | salary | department_id |
|---|---|---|---|---|
| 1 | John | Doe | 5000.00 | 1 |
| 2 | Jane | Smith | 6000.00 | 2 |
| 3 | Alice | Johnson | 5500.00 | 1 |
| 4 | Bob | Brown | 4500.00 | 3 |
| 5 | Charlie | Davis | 7000.00 | 2 |
| 6 | Emily | Clark | 4800.00 | 3 |

## d) Functions



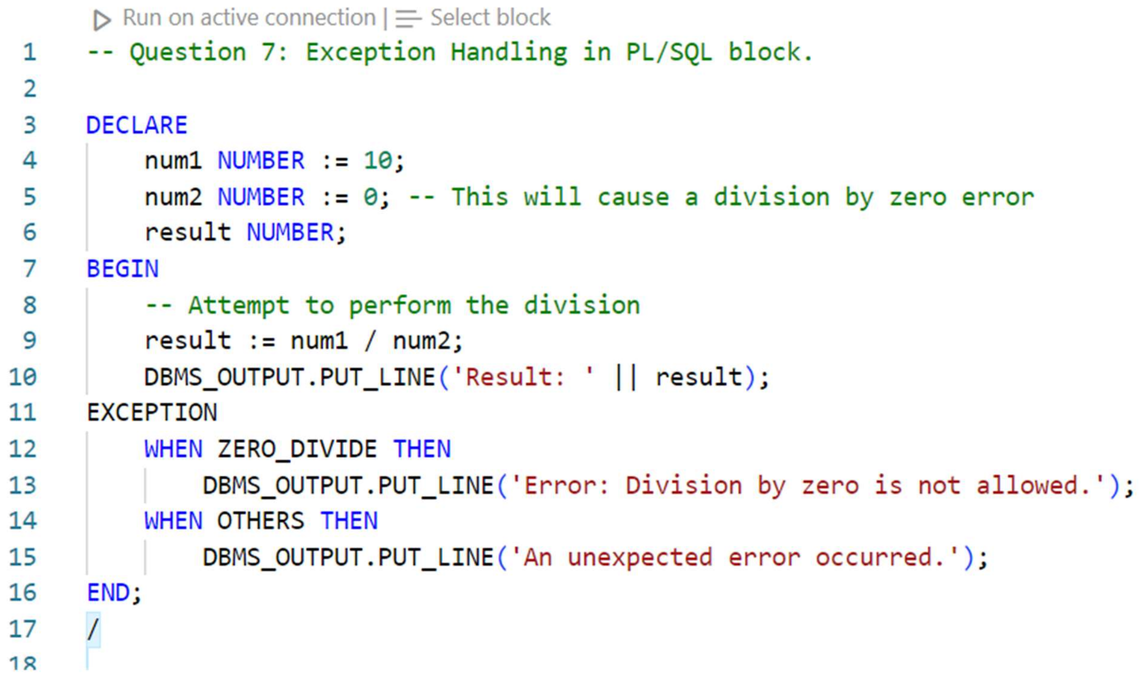
```sql
-- Question 6(d): functions

CREATE FUNCTION calculate_annual_salary(employee_id INT)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE annual_salary DECIMAL(10, 2);

    SELECT salary * 12 INTO annual_salary
    FROM employees
    WHERE employee_id = employee_id;

    RETURN annual_salary;
END;

SELECT calculate_annual_salary(1) AS annual_salary;
```



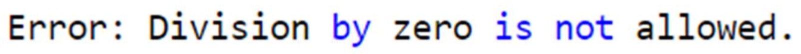
| annual_salary |
|---|
| 60000.00 |

# Experiment 7

**Aim**: Perform Exception handling in a PL/SQL block.



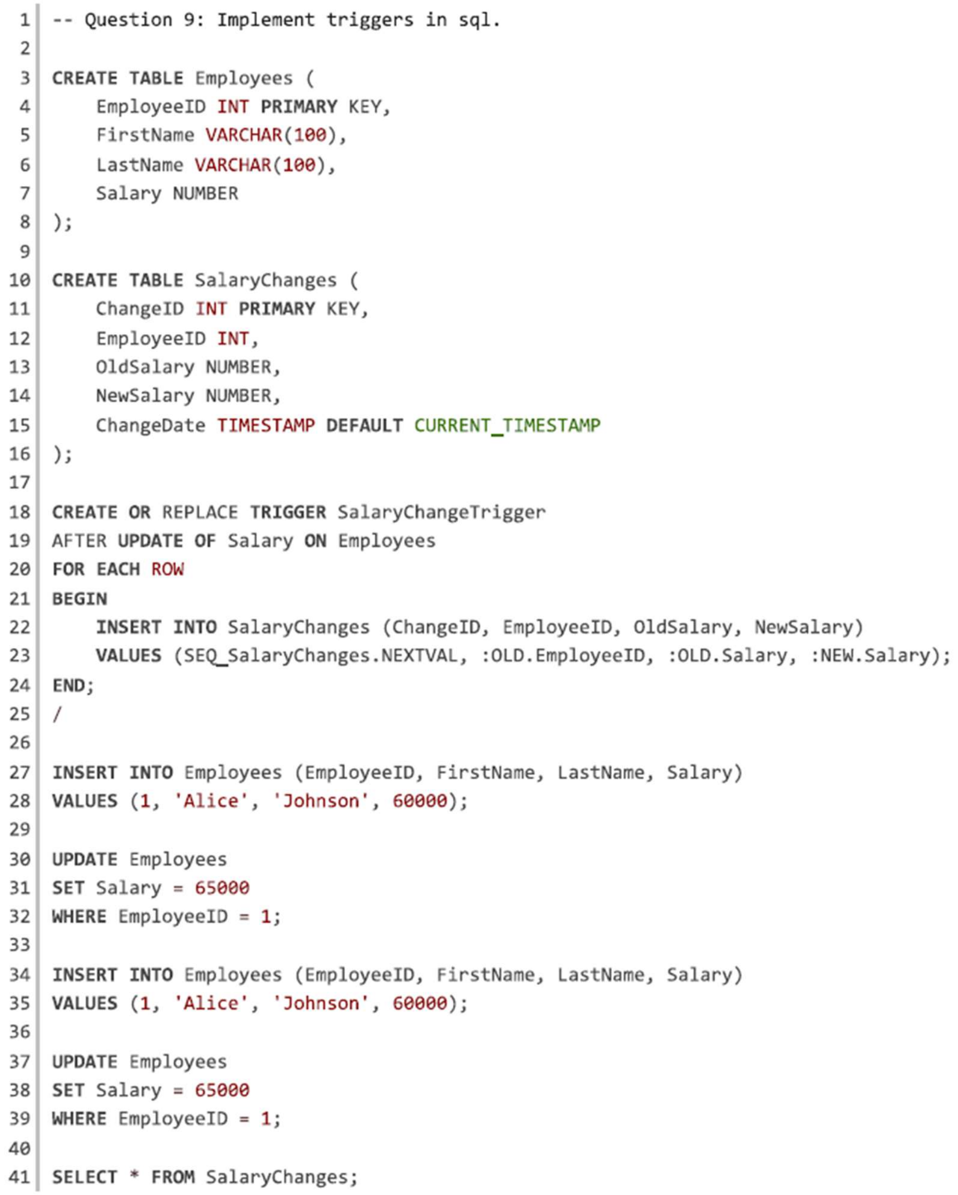
```
-- Question 7: Exception Handling in PL/SQL block.

DECLARE
    num1 NUMBER := 10;
    num2 NUMBER := 0; -- This will cause a division by zero error
    result NUMBER;
BEGIN
    -- Attempt to perform the division
    result := num1 / num2;
    DBMS_OUTPUT.PUT_LINE('Result: ' || result);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred.');
END;
/
```



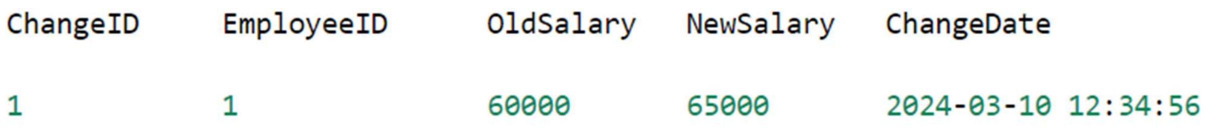
```
Error: Division by zero is not allowed.
```

# Experiment 9

**Aim**: Implement Triggers in SQL.



```
-- Question 9: Implement triggers in sql.

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Salary NUMBER
);

CREATE TABLE SalaryChanges (
    ChangeID INT PRIMARY KEY,
    EmployeeID INT,
    OldSalary NUMBER,
    NewSalary NUMBER,
    ChangeDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE TRIGGER SalaryChangeTrigger
AFTER UPDATE OF Salary ON Employees
FOR EACH ROW
BEGIN
    INSERT INTO SalaryChanges (ChangeID, EmployeeID, OldSalary, NewSalary)
    VALUES (SEQ_SalaryChanges.NEXTVAL, :OLD.EmployeeID, :OLD.Salary, :NEW.Salary);
END;
/

INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary)
VALUES (1, 'Alice', 'Johnson', 60000);

UPDATE Employees
SET Salary = 65000
WHERE EmployeeID = 1;

INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary)
VALUES (1, 'Alice', 'Johnson', 60000);

UPDATE Employees
SET Salary = 65000
WHERE EmployeeID = 1;

SELECT * FROM SalaryChanges;
```

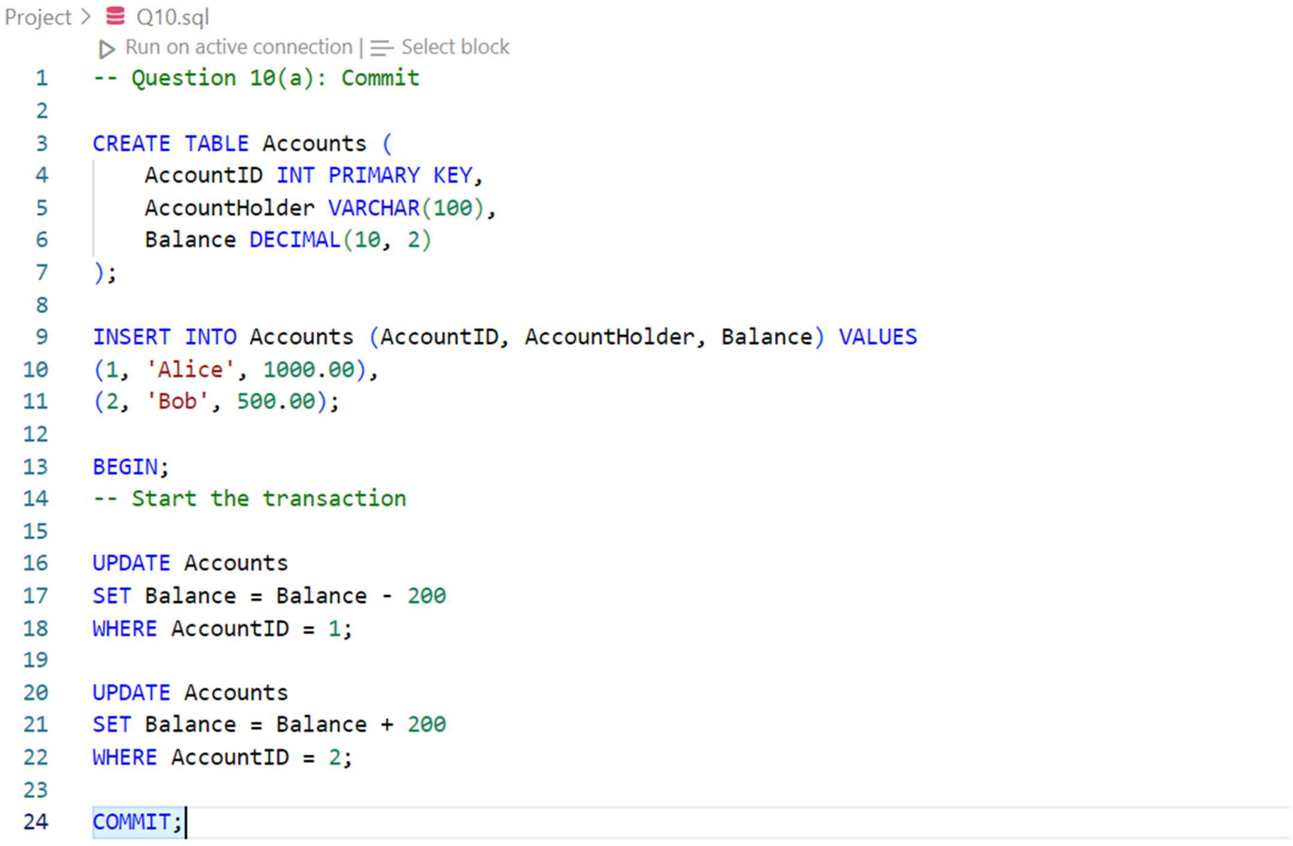


```
ChangeID      EmployeeID      OldSalary    NewSalary    ChangeDate

1             1               60000        65000        2024-03-10 12:34:56
```

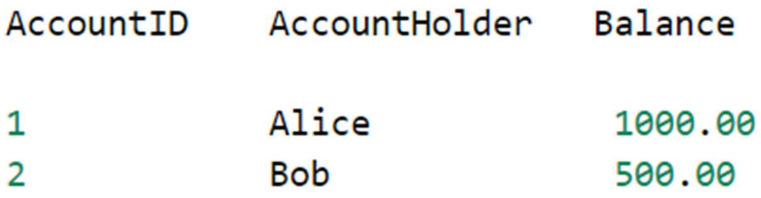
# Experiment 10

**Aim**: Implement the following transaction statements:

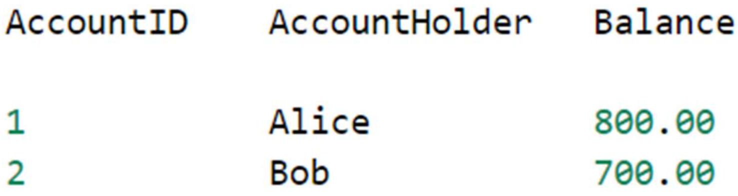
a) Commit



```sql
-- Question 10(a): Commit

CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    AccountHolder VARCHAR(100),
    Balance DECIMAL(10, 2)
);

INSERT INTO Accounts (AccountID, AccountHolder, Balance) VALUES
(1, 'Alice', 1000.00),
(2, 'Bob', 500.00);

BEGIN;
-- Start the transaction

UPDATE Accounts
SET Balance = Balance - 200
WHERE AccountID = 1;

UPDATE Accounts
SET Balance = Balance + 200
WHERE AccountID = 2;

COMMIT;
```

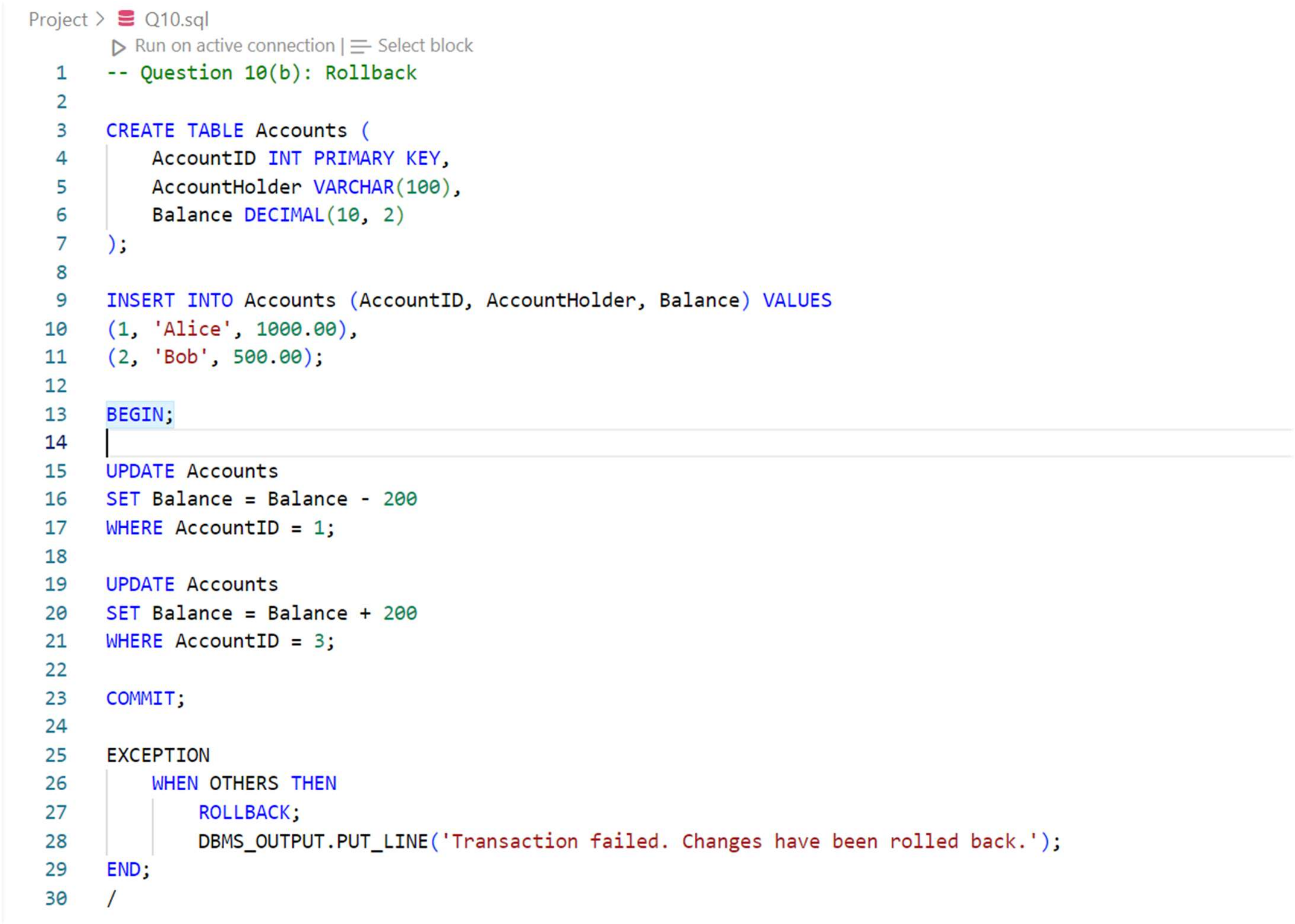


```
AccountID    AccountHolder   Balance

1            Alice            1000.00
2            Bob              500.00
```

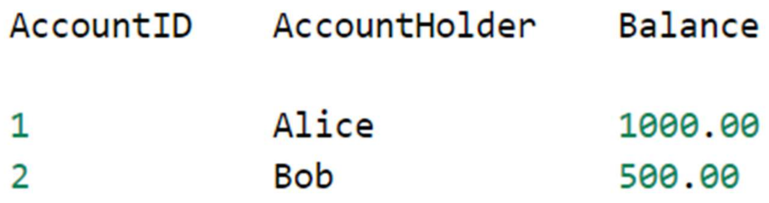


```
AccountID    AccountHolder   Balance

1            Alice           800.00
2            Bob             700.00
```

## b) Rollback

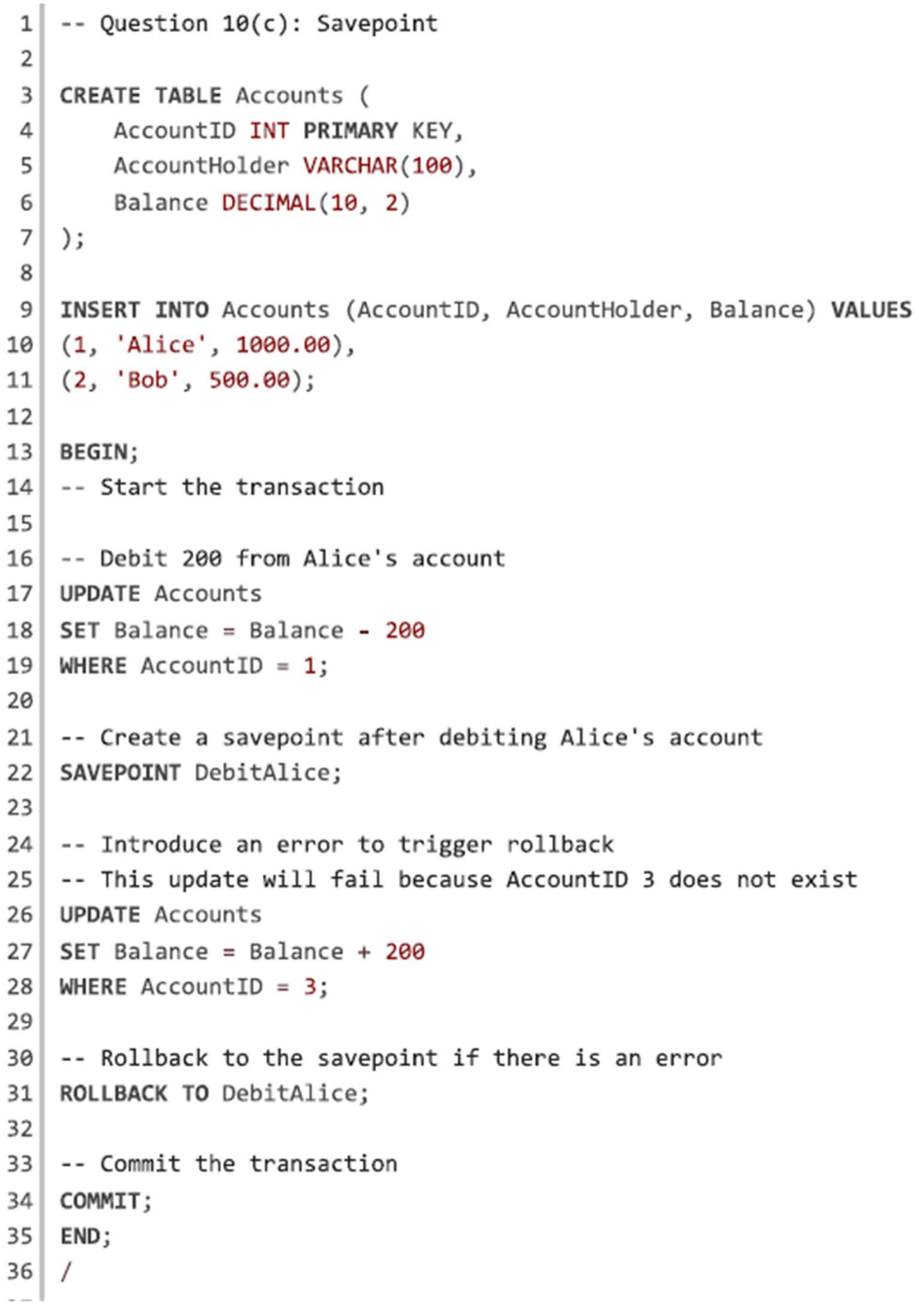


```sql
-- Question 10(b): Rollback

CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    AccountHolder VARCHAR(100),
    Balance DECIMAL(10, 2)
);

INSERT INTO Accounts (AccountID, AccountHolder, Balance) VALUES
(1, 'Alice', 1000.00),
(2, 'Bob', 500.00);

BEGIN;

UPDATE Accounts
SET Balance = Balance - 200
WHERE AccountID = 1;

UPDATE Accounts
SET Balance = Balance + 200
WHERE AccountID = 3;

COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Transaction failed. Changes have been rolled back.');
END;
/
```

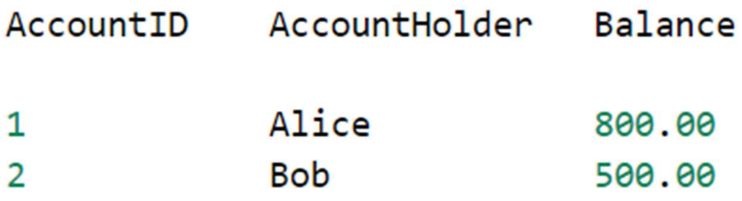


| AccountID | AccountHolder | Balance |
|---|---|---|
| 1 | Alice | 1000.00 |
| 2 | Bob | 500.00 |

c) Savepoint



```sql
-- Question 10(c): Savepoint

CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    AccountHolder VARCHAR(100),
    Balance DECIMAL(10, 2)
);

INSERT INTO Accounts (AccountID, AccountHolder, Balance) VALUES
(1, 'Alice', 1000.00),
(2, 'Bob', 500.00);

BEGIN;
-- Start the transaction

-- Debit 200 from Alice's account
UPDATE Accounts
SET Balance = Balance - 200
WHERE AccountID = 1;

-- Create a savepoint after debiting Alice's account
SAVEPOINT DebitAlice;

-- Introduce an error to trigger rollback
-- This update will fail because AccountID 3 does not exist
UPDATE Accounts
SET Balance = Balance + 200
WHERE AccountID = 3;

-- Rollback to the savepoint if there is an error
ROLLBACK TO DebitAlice;

-- Commit the transaction
COMMIT;
END;
/
```



```
AccountID    AccountHolder    Balance

1            Alice            800.00
2            Bob              500.00
```