Q1) Create a table to represent account of a bank consisting of account-no,
Customer-name, balance-amount.
Write a Query to Update the table name account to sb-account.
Write a Query to Fetch sb-account with balance amount >10000.
Write a Query to convert all customer-name into Upper case.
Write a Query to Collect all customers balance-amount and customer-name into Upper case.

ANSWER---->

1st code:
```
CREATE TABLE account (
    account_no INT PRIMARY KEY,
    customer_name VARCHAR(50),
    balance_amount DECIMAL(10, 2)
);
INSERT INTO account (account_no, customer_name, balance_amount)
VALUES
    (1, 'John Doe', 1500.00),
    (2, 'Jane Smith', 1200.50),
    (3, 'Alice Johnson', 18000.75),
    (4, 'Bob Williams', 1050.25),
    (5, 'Eve Brown', 20000.00);
```

2nd code:
```
ALTER TABLE account
RENAME TO sb_account;
```

3rd code:
```
SELECT * FROM sb_account
WHERE balance_amount > 10000;
```

4th code:
```
UPDATE sb_account
SET customer_name = UPPER(customer_name);
```

5th code:
```
SELECT UPPER(customer_name) AS upper_case_name, balance_amount
FROM sb_account;
```

Q2)
Create The following two tables :
College-info
Faculty-info
College-info consists of fields : college-code, college-name, address
Faculty-info consists of fields : college-code, faculty-code, faculty-name,
qualification, experience-in-no-of-years, address.
The field college-code is foreign key.

Generate queries to do the following :
List all those faculty members whose experience is greater than or equal to 10 years and have M. Tech degree.
List all those faculty members, who have at least 10 years of experience but do not have M. Tech degree

ANSWER---->

1st code:
```
-- College-info table
CREATE TABLE College_info (
    college_code INT PRIMARY KEY,
    college_name VARCHAR(50),
    address VARCHAR(100)
);

-- Faculty-info table
CREATE TABLE Faculty_info (
    college_code INT,
    faculty_code INT PRIMARY KEY,
    faculty_name VARCHAR(50),
    qualification VARCHAR(50),
    experience_in_years INT,
    address VARCHAR(100),
    FOREIGN KEY (college_code) REFERENCES College_info(college_code)
);
-- Inserting values into College_info table
INSERT INTO College_info (college_code, college_name, address)
VALUES
    (1, 'ABC College', '123 Main Street, Cityville'),
    (2, 'XYZ College', '456 Oak Avenue, Townsville'),
    (3, 'PQR College', '789 Pine Road, Villagetown');

-- Inserting values into Faculty_info table
INSERT INTO Faculty_info (college_code, faculty_code, faculty_name, 
qualification, experience_in_years, address)
VALUES
    (1, 101, 'John Doe', 'M. Tech', 12, '321 Elm Lane, Suburbia'),
    (1, 102, 'Jane Smith', 'Ph.D.', 8, '654 Birch Street, Outskirts'),
    (2, 201, 'Bob Williams', 'M. Tech', 15, '987 Maple Drive, 
Countryside'),
    (2, 202, 'Alice Johnson', 'B. Tech', 10, '741 Cedar Court, 
Riverside'),
    (3, 301, 'Eve Brown', 'Ph.D.', 11, '852 Pine Lane, Hillside');
```

2nd code:
```
SELECT *
FROM Faculty_info
WHERE experience_in_years >= 10 AND qualification = 'M. Tech';
```

3rd code:
```
SELECT *
FROM Faculty_info
WHERE experience_in_years >= 10 AND qualification != 'M. Tech';
```

```
Q3)
Create the following tables :
      Student(roll-no, name, age, course-id)
      Course  (Course-id, name, fee, duration)
(a) Insert the appropriate data.
(b) Generate queries to do the following :
      (i) List all those students who are greater than 18 years of age
and have opted
              for MCA course.
      (ii) List all those courses whose fee is greater than that of MCA
course.
```

ANSWER---->

1st code:

```
-- Create Student table
CREATE TABLE Student (
    roll_no INT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    course_id INT,
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

-- Create Course table
CREATE TABLE Course (
    course_id INT PRIMARY KEY,
    name VARCHAR(50),
    fee DECIMAL(10, 2),
    duration INT -- Assuming duration is in months
);
```

2nd code:

```
-- Inserting data into Course table
INSERT INTO Course (course_id, name, fee, duration)
VALUES
    (1, 'MCA', 15000.00, 24),
    (2, 'B.Tech', 20000.00, 48),
    (3, 'MBA', 18000.00, 36),
    (4, 'BBA', 16000.00, 36);

-- Inserting data into Student table
INSERT INTO Student (roll_no, name, age, course_id)
VALUES
    (101, 'John Doe', 20, 1),
    (102, 'Jane Smith', 22, 2),
    (103, 'Bob Williams', 19, 1),
    (104, 'Alice Johnson', 21, 3),
    (105, 'Eve Brown', 18, 2);
```

3rd code:

```
SELECT *
FROM Student
WHERE age > 18 AND course_id = (SELECT course_id FROM Course WHERE name =
'MCA');
```

4th code:

```
SELECT *
FROM Course
WHERE fee > (SELECT fee FROM Course WHERE name = 'MCA');
```

Q4)4. Create the following tables :
salesperson(salesman_id,name,city,commission)
customer(customer_id,cust_name,city,grade,salesman_id)

Status could be issued, present in the library, sent for binding, and account be
issued.
(a) Insert the appropriate data.
(b) From the following tables write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust_name and city.

ANSWER---->

1st code:

```
-- Create salesperson table
CREATE TABLE salesperson (
    salesman_id INT PRIMARY KEY,
    name VARCHAR(50),
    city VARCHAR(50),
    commission DECIMAL(10, 2)
);

-- Create customer table
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    cust_name VARCHAR(50),
    city VARCHAR(50),
    grade VARCHAR(10),
    salesman_id INT,
    FOREIGN KEY (salesman_id) REFERENCES salesperson(salesman_id)
);

-- Inserting data into salesperson table
INSERT INTO salesperson (salesman_id, name, city, commission)
VALUES
    (1, 'John Doe', 'New York', 0.1),
    (2, 'Jane Smith', 'Los Angeles', 0.15),
    (3, 'Bob Williams', 'Chicago', 0.12),
    (4, 'Alice Johnson', 'New York', 0.08);

-- Inserting data into customer table
INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id)
VALUES
    (101, 'Eve Brown', 'Los Angeles', 'A', 2),
    (102, 'Charlie Wilson', 'New York', 'B', 1),
    (103, 'David Clark', 'Chicago', 'C', 3),
    (104, 'Grace Taylor', 'New York', 'A', 1);
```

code 2:

```
SELECT s.name AS Salesman, c.cust_name, c.city
FROM salesperson s
JOIN customer c ON s.city = c.city;
```

Q5)Create the following table :

      Student (roll-no, name, subject-opted)
      Subject (subject-code, subject-name, faculty-code)
      Faculty (faculty-code, faculty-name, specialization)
      (a) Insert the appropriate data.
      (b) Generate queries to do the following:
            (i) Find the number of students who have enrolled for the subject "DBMS".
      (ii) Find all those faculty members who have not offered any subject

ANSWER---->

1st code:

```
-- Create Student table
```

```
CREATE TABLE Student (
    roll_no INT PRIMARY KEY,
    name VARCHAR(50),
    subject_opted VARCHAR(50)
);

-- Create Subject table
CREATE TABLE Subject (
    subject_code INT PRIMARY KEY,
    subject_name VARCHAR(50),
    faculty_code INT,
    FOREIGN KEY (faculty_code) REFERENCES Faculty(faculty_code)
);

-- Create Faculty table
CREATE TABLE Faculty (
    faculty_code INT PRIMARY KEY,
    faculty_name VARCHAR(50),
    specialization VARCHAR(50)
);
```

2nd code:

```
-- Inserting data into Faculty table
INSERT INTO Faculty (faculty_code, faculty_name, specialization)
VALUES
    (1, 'Dr. Smith', 'Database Management'),
    (2, 'Prof. Johnson', 'Artificial Intelligence'),
    (3, 'Dr. Brown', 'Software Engineering');

-- Inserting data into Subject table
INSERT INTO Subject (subject_code, subject_name, faculty_code)
VALUES
    (101, 'DBMS', 1),
    (102, 'AI Algorithms', 2),
    (103, 'Software Design', 3);

-- Inserting data into Student table
INSERT INTO Student (roll_no, name, subject_opted)
VALUES
    (1, 'John Doe', 'DBMS'),
    (2, 'Jane Smith', 'AI Algorithms'),
    (3, 'Bob Williams', 'DBMS'),
    (4, 'Alice Johnson', 'Software Design');
```

3rd code:

```
SELECT COUNT(*) AS num_students
FROM Student
WHERE subject_opted = 'DBMS';
```

4th code:

```
SELECT f.faculty_name
FROM Faculty f
LEFT JOIN Subject s ON f.faculty_code = s.faculty_code
WHERE s.subject_code IS NULL;
```

Q6)

CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

(a) insert values into this table using the INSERT statement

(b) create the stored procedure with the name GetCustomerInfo. Provide it a single input parameter   called @CutomerAge. The stored procedure then selects all records from the CUSTOMERS table where the value of the CutomerAge matches the input parameter.

(c) stored procedure that displays the student's list in the increasing order of a salary from the CUSTOMERS table in the selected database:

ANSWER---->

1st code:

```
-- Assuming the CUSTOMERS table is created
INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES
    (1, 'John Doe', 25, '123 Main Street', 50000),
    (2, 'Jane Smith', 30, '456 Oak Avenue', 60000),
    (3, 'Bob Williams', 28, '789 Pine Road', 55000),
    (4, 'Alice Johnson', 22, '101 Elm Lane', 48000);
```

2nd code:

```
CREATE PROCEDURE GetCustomerInfo
    @CustomerAge INT
AS
BEGIN
    SELECT *
    FROM CUSTOMERS
    WHERE AGE = @CustomerAge;
END;
```

3rd code:

```
CREATE PROCEDURE GetCustomerListBySalary
AS
BEGIN
    SELECT *
    FROM CUSTOMERS
    ORDER BY SALARY ASC;
END;
```

Q7)
Create Orders Table as (order_no ,purchase_amtount ,ordere_date ,customer_id,salesman_id)
Calculate total purchase amount of all orders. Return total purchase amount.
Calculate average purchase amount of all orders. Return average purchase amount.
 Count the number of unique salespeople. Return number of salespeople.
Find the maximum purchase amount.
Find the minimum purchase amount.
Find the highest purchase amount ordered by each customer. Return customer ID, maximum purchase amount.

ANSWER---->

1st code:

```
-- Create Orders table
CREATE TABLE Orders (
    order_no INT PRIMARY KEY,
    purchase_amount DECIMAL(10, 2),
    order_date DATE,
    customer_id INT,
    salesman_id INT
);

-- Inserting data into Orders table (example values)
INSERT INTO Orders (order_no, purchase_amount, order_date, customer_id,
salesman_id)
VALUES
    (1, 500.00, '2023-01-01', 101, 201),
    (2, 800.50, '2023-01-02', 102, 202),
    (3, 600.25, '2023-01-03', 103, 203),
    (4, 700.75, '2023-01-04', 104, 204),
    (5, 900.00, '2023-01-05', 105, 205);

-- Calculate total purchase amount of all orders
SELECT SUM(purchase_amount) AS total_purchase_amount
FROM Orders;

-- Calculate average purchase amount of all orders
SELECT AVG(purchase_amount) AS average_purchase_amount
FROM Orders;

-- Count the number of unique salespeople
SELECT COUNT(DISTINCT salesman_id) AS number_of_salespeople
FROM Orders;

-- Find the maximum purchase amount
SELECT MAX(purchase_amount) AS max_purchase_amount
FROM Orders;

-- Find the minimum purchase amount
SELECT MIN(purchase_amount) AS min_purchase_amount
FROM Orders;

-- Find the highest purchase amount ordered by each customer
SELECT customer_id, MAX(purchase_amount) AS max_purchase_amount
FROM Orders
GROUP BY customer_id;
```

Q8)Create T1_EMP and T2_EMP table as follows And perform Set operations on the tables

ID
Name
Department
Salary
Year_of_Experience
1
Aakash Singh
Development
72000
2
2
Abhishek Pawar
Production
45000

```
1
3
Pranav Deshmukh
HR
59900
3
4
Shubham Mahale
Accounts
57000
2
5
Sunil Kulkarni
Development
87000
3
6
Bhushan Wagh
R&D
75000
2
7
Paras Jaiswal
Marketing
32000
1
```

```
ID
Name
Department
Salary
Year_of_Experience
1
Prashant Wagh
R&D
49000
1
2
Abhishek Pawar
Production
45000
1
3
Gautam Jain
Development
56000
4
4
Shubham Mahale
Accounts
57000
2
```

5
Rahul Thakur
Production
76000
4
6
Bhushan Wagh
R&D
75000
2
7
Anand Singh
Marketing
28000
1

1.Write a query to perform union between the table T1_EMP and the table T2_EMP .
2.Write a query to perform union all operation between the table T1_EMP and the table T2_EMP .
3. Write a query to perform intersect operation between the table T1_EMP and the table T2_EMP .
4. Write a query to perform a minus operation between the table T1_EMP and the table T2_EMP .

ANSWER---->

1st code:

```
-- Create T1_EMP table
CREATE TABLE T1_EMP (
    ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Department VARCHAR(50),
    Salary INT,
    Year_of_Experience INT
);

-- Inserting data into T1_EMP table
INSERT INTO T1_EMP (ID, Name, Department, Salary, Year_of_Experience)
VALUES
    (1, 'Aakash Singh', 'Development', 72000, 2),
    (2, 'Abhishek Pawar', 'Production', 45000, 1),
    (3, 'Pranav Deshmukh', 'HR', 59900, 3),
    (4, 'Shubham Mahale', 'Accounts', 57000, 2),
    (5, 'Sunil Kulkarni', 'Development', 87000, 3),
    (6, 'Bhushan Wagh', 'R&D', 75000, 2),
    (7, 'Paras Jaiswal', 'Marketing', 32000, 1);

-- Create T2_EMP table
CREATE TABLE T2_EMP (
    ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Department VARCHAR(50),
    Salary INT,
    Year_of_Experience INT
);

-- Inserting data into T2_EMP table
INSERT INTO T2_EMP (ID, Name, Department, Salary, Year_of_Experience)
VALUES
    (1, 'Prashant Wagh', 'R&D', 49000, 1),
    (2, 'Abhishek Pawar', 'Production', 45000, 1),
    (3, 'Gautam Jain', 'Development', 56000, 4),
    (4, 'Shubham Mahale', 'Accounts', 57000, 2),
    (5, 'Rahul Thakur', 'Production', 76000, 4),
    (6, 'Bhushan Wagh', 'R&D', 75000, 2),
    (7, 'Anand Singh', 'Marketing', 28000, 1);
```

2nd code:

```
-- Union between T1_EMP and T2_EMP
SELECT * FROM T1_EMP
UNION
SELECT * FROM T2_EMP;
```

3rd code:

```
-- Union All between T1_EMP and T2_EMP
SELECT * FROM T1_EMP
UNION ALL
SELECT * FROM T2_EMP;
```

4th code:

```
-- Intersect between T1_EMP and T2_EMP
SELECT * FROM T1_EMP
INTERSECT
SELECT * FROM T2_EMP;
```

5th code:

```
-- Minus (or Except) between T1_EMP and T2_EMP
SELECT * FROM T1_EMP
EXCEPT
SELECT * FROM T2_EMP;
```


EXTRAS

Q1)
Create the following tables for Library Information System :
Book : (accession-no, title, publisher, author, status)
Status could be issued, present in the library, sent for binding, and cannot be
issued.
Write a trigger which sets the status of a book to "cannot be issued", if it is
published 20 years back.

ANSWER---->

1st code:

```
-- Create Book table
CREATE TABLE Book (
    accession_no INT PRIMARY KEY,
    title VARCHAR(100),
    publisher VARCHAR(50),
    author VARCHAR(50),
    status VARCHAR(50)
);

-- Inserting sample data into Book table
INSERT INTO Book (accession_no, title, publisher, author, status)
VALUES
    (1, 'Book1', 'Publisher1', 'Author1', 'present in the library'),
    (2, 'Book2', 'Publisher2', 'Author2', 'issued'),
    (3, 'Book3', 'Publisher3', 'Author3', 'sent for binding'),
    (4, 'Book4', 'Publisher4', 'Author4', 'present in the library'),
    (5, 'Book5', 'Publisher5', 'Author5', 'issued');

-- Create a trigger to set the status of a book to "cannot be issued" if
it is published 20 years back
CREATE TRIGGER set_status_cannot_be_issued
BEFORE INSERT ON Book
FOR EACH ROW
BEGIN
    IF (YEAR(CURDATE()) - YEAR(NEW.publisher) >= 20) THEN
        SET NEW.status = 'cannot be issued';
    END IF;
END;
```

Q2)
Create the following tables for Library Information System :

Book(accession-no, title, publisher, author, status, date-of-purchase)
Status could be issued, present in the library, sent for binding, and account be
issued.
(a) Create a form to accept the data from the user Create a form to accept the
data from the user with appropriate validation checks.
(b) Generate queries to do the following :
(i)
List all those books which are new arrivals. The books which are acquired
during the last 6 months are categorized as new arrivals.
(ii)
List all those books that cannot be issued and purchased 20 years ago.

ANSWER---->

1st code:

```
-- Assuming you have a Book table already created
CREATE TABLE Book (
    accession_no INT PRIMARY KEY,
    title VARCHAR(100),
    publisher VARCHAR(50),
    author VARCHAR(50),
    status VARCHAR(50),
    date_of_purchase DATE
);

-- Example SQL statement to insert data into the Book table
INSERT INTO Book (accession_no, title, publisher, author, status,
date_of_purchase)
VALUES
    (1, 'Book1', 'Publisher1', 'Author1', 'present in the library',
'2023-01-01'),
    (2, 'Book2', 'Publisher2', 'Author2', 'issued', '2023-01-15'),
    (3, 'Book3', 'Publisher3', 'Author3', 'sent for binding', '2023-02-
05');
```

2nd code:

```
SELECT *
FROM Book
WHERE date_of_purchase >= CURDATE() - INTERVAL 6 MONTH;
```

3rd code

```
SELECT *
FROM Book
```

```
WHERE status = 'cannot be issued' AND YEAR(date_of_purchase) =
YEAR(CURDATE()) - 20;
```

Q3)
Create the following tables :
Student(roll-no, name, date-of-birth, course-id)
Course (Course-id, name, fee, duration)
(a) Create a form to accept the data from the user with appropriate validation
   checks.
(b) Generate queries to do the following :
(i) List all those students who are greater than 18 years of age and have opted for
    MCA course.
(ii) List all those courses whose fee is greater than that of MCA course.

ANSWER---->

1st code:

```
-- Assuming you have a Student and Course table already created
CREATE TABLE Student (
    roll_no INT PRIMARY KEY,
    name VARCHAR(50),
    date_of_birth DATE,
    course_id INT
);

CREATE TABLE Course (
    course_id INT PRIMARY KEY,
    name VARCHAR(50),
    fee DECIMAL(10, 2),
    duration INT -- Assuming duration is in months
);

-- Example SQL statements to insert data into the Student and Course
tables
-- You can customize these statements based on your actual requirements
INSERT INTO Student (roll_no, name, date_of_birth, course_id)
VALUES
    (1, 'John Doe', '2000-01-01', 1),
    (2, 'Jane Smith', '1999-05-15', 2),
    (3, 'Bob Williams', '2001-02-10', 1);

INSERT INTO Course (course_id, name, fee, duration)
VALUES
```

```
    (1, 'MCA', 15000.00, 24),
    (2, 'B.Tech', 20000.00, 48),
    (3, 'MBA', 18000.00, 36);
```

2nd code:

```
SELECT *
FROM Student s
JOIN Course c ON s.course_id = c.course_id
WHERE TIMESTAMPDIFF(YEAR, s.date_of_birth, CURDATE()) > 18
  AND c.name = 'MCA';
```

3rd code:

```
SELECT *
FROM Course
WHERE fee > (SELECT fee FROM Course WHERE name = 'MCA');
```

Q4)Create the following table :
Student (roll-no, name, subject-name, subject-opted)
Subject(faculty-code, faculty-name, specialization)
(a) Create a form to accept the data from the user with appropriate validation
     checks.
(b) Generate queries to do the following :
(i) Find the number of students who have enrolled for the subject "DBMS".
(ii) Find all those faculty members who have not offered any subject.

ANSWER---->

1st code:

```
-- Creating Student table
CREATE TABLE Student (
    roll_no INT PRIMARY KEY,
    name VARCHAR(50),
    subject_name VARCHAR(50),
    subject_opted VARCHAR(50)
);

-- Creating Subject table
```

```
CREATE TABLE Subject (
    faculty_code INT PRIMARY KEY,
    faculty_name VARCHAR(50),
    specialization VARCHAR(50)
);
```

2nd code:

```
-- Example SQL statements to insert data into the Student table
INSERT INTO Student (roll_no, name, subject_name, subject_opted)
VALUES
    (1, 'John Doe', 'DBMS', 'Database Management'),
    (2, 'Jane Smith', 'AI Algorithms', 'Artificial Intelligence'),
    (3, 'Bob Williams', 'DBMS', 'Database Management');

-- Example SQL statements to insert data into the Subject table
INSERT INTO Subject (faculty_code, faculty_name, specialization)
VALUES
    (101, 'Dr. Smith', 'Database Management'),
    (102, 'Prof. Johnson', 'Artificial Intelligence'),
    (103, 'Dr. Brown', 'Software Engineering');
```

3rd code:

```
SELECT COUNT(*) AS num_students
FROM Student
WHERE subject_name = 'DBMS';
```

4th code:

```
SELECT *
FROM Subject
WHERE faculty_code NOT IN (SELECT DISTINCT faculty_code FROM Student);
```

Q5)
Create the following table :
Item (item-code, item-name, qty-in-stock, reorder-level)
Supplier (supplier-code, supplier-name, address)
Can-supply(supplier-code, item-code)
(a)
Create a form to accept the data from the user with appropriate validation
checks.
(b)
Generate queries to do the following :
(i)
List all those suppliers who can supply the given item.
(ii)

List all those items which cannot be supplied by given company.

ANSWER---->

1st code:

```
-- Creating Item table
CREATE TABLE Item (
    item_code INT PRIMARY KEY,
    item_name VARCHAR(50),
    qty_in_stock INT,
    reorder_level INT
);

-- Creating Supplier table
CREATE TABLE Supplier (
    supplier_code INT PRIMARY KEY,
    supplier_name VARCHAR(50),
    address VARCHAR(100)
);

-- Creating Can-Supply table
CREATE TABLE Can_Supply (
    supplier_code INT,
    item_code INT,
    PRIMARY KEY (supplier_code, item_code),
    FOREIGN KEY (supplier_code) REFERENCES Supplier(supplier_code),
    FOREIGN KEY (item_code) REFERENCES Item(item_code)
);
```

2nd code:

```
-- Example SQL statements to insert data into the Item table
INSERT INTO Item (item_code, item_name, qty_in_stock, reorder_level)
VALUES
    (1, 'Widget', 100, 20),
    (2, 'Gadget', 50, 10),
    (3, 'Doodad', 75, 15);

-- Example SQL statements to insert data into the Supplier table
INSERT INTO Supplier (supplier_code, supplier_name, address)
VALUES
    (101, 'ABC Suppliers', '123 Main Street'),
    (102, 'XYZ Distributors', '456 Oak Avenue'),
    (103, 'LMN Inc.', '789 Pine Road');

-- Example SQL statements to insert data into the Can_Supply table
INSERT INTO Can_Supply (supplier_code, item_code)
VALUES
    (101, 1),
    (102, 2),
    (103, 3);
```

3rd code:

```
SELECT s.supplier_name
FROM Supplier s
JOIN Can_Supply cs ON s.supplier_code = cs.supplier_code
WHERE cs.item_code = 1; -- Change 1 to the desired item_code
```

4th code:

```
SELECT i.item_name
FROM Item i
WHERE i.item_code NOT IN (SELECT item_code FROM Can_Supply WHERE
supplier_code = 101); -- Change 101 to the desired supplier_code
```

Q6)
Create the following tables:
Student (roll-no, marks, category, district, state)
Student-rank(roll-no, marks, rank)
(a) Create a form to accept the data from the user with appropriate validation
checks.
(b) Generate queries to do the following :
(i) List all those students who have come from Tamilnadu state and secured a rank
above 100.
(ii) List all those students who come from Andhra Pradesh state and belong to
given category who have secured a rank above 100.

ANSWER---->

1st code:

```
-- Creating Student table
CREATE TABLE Student (
    roll_no INT PRIMARY KEY,
    marks INT,
    category VARCHAR(50),
    district VARCHAR(50),
    state VARCHAR(50)
);

-- Creating Student_Rank table
CREATE TABLE Student_Rank (
    roll_no INT PRIMARY KEY,
    marks INT,
    rank INT
);
```

2nd code:

```
-- Example SQL statements to insert data into the Student table
INSERT INTO Student (roll_no, marks, category, district, state)
VALUES
    (1, 85, 'General', 'Chennai', 'Tamilnadu'),
    (2, 92, 'SC', 'Hyderabad', 'Andhra Pradesh'),
    (3, 78, 'OBC', 'Coimbatore', 'Tamilnadu');

-- Example SQL statements to insert data into the Student_Rank table
INSERT INTO Student_Rank (roll_no, marks, rank)
VALUES
    (1, 85, 120),
    (2, 92, 85),
    (3, 78, 150);
```

3rd code:

```
SELECT s.*
FROM Student s
JOIN Student_Rank sr ON s.roll_no = sr.roll_no
WHERE s.state = 'Tamilnadu' AND sr.rank > 100;
```

4th code:

```
SELECT s.*
FROM Student s
JOIN Student_Rank sr ON s.roll_no = sr.roll_no
WHERE s.state = 'Andhra Pradesh' AND s.category = 'SC' AND sr.rank > 100;
```