SQL ASSESSMENT – FULL DETAILED SOLUTION

1. CREATING TABLES

We need to create three tables: Bank, Account_Holder, and Loan. Each table has its own fields, and we must also define primary keys and foreign keys to maintain data integrity.

SQL Code:

```
CREATE TABLE Bank (
   branch_id INT PRIMARY KEY,
   branch_name VARCHAR(50),
   branch_city VARCHAR(50)
);
```

Explanation:

- The Bank table stores branch details.
- branch_id is set as PRIMARY KEY, meaning each branch must have a unique ID.
- branch_name stores the name of the branch.
- branch_city stores the city where the branch is located.

Next, we create the Account_Holder table:

```
CREATE TABLE Account_Holder (
   account_holder_id INT PRIMARY KEY,
   account_no INT UNIQUE,
   account_holder_name VARCHAR(100),
   city VARCHAR(50),
   contact VARCHAR(15),
   date_of_account_created DATE,
   account_status VARCHAR(20),
   account_type VARCHAR(20),
   balance DECIMAL(10,2)
);
```

Explanation:

• account_holder_id uniquely identifies each account holder.
• account_no is UNIQUE because no two people can have the same account number.
• balance uses DECIMAL(10,2) because it stores money values with two decimal places.
• date_of_account_created keeps track of when the account was opened.
• account_status (e.g., active, inactive) and account_type (savings/current) describe the account.

Now, the Loan table:

```
CREATE TABLE Loan (
   loan_no INT PRIMARY KEY,
   branch_id INT,
   account_holder_id INT,
   loan_amount DECIMAL(10,2),
   loan_type VARCHAR(50),
   FOREIGN KEY (branch_id) REFERENCES Bank(branch_id),
   FOREIGN KEY (account_holder_id) REFERENCES Account_Holder(account_holder_id)
);
```

Explanation:

• loan_no uniquely identifies each loan.
• branch_id links to Bank(branch_id) → this is a FOREIGN KEY.
• account_holder_id links to Account_Holder(account_holder_id) → this is also a FOREIGN KEY.
• This ensures referential integrity between tables.

2. INSERTING SAMPLE DATA

Now we insert sample data into each table so we can run queries and test outputs.

```
INSERT INTO Bank VALUES
(1, 'Main Branch', 'Mumbai'),
(2, 'City Branch', 'Mumbai'),
(3, 'Town Branch', 'Pune');

INSERT INTO Account_Holder VALUES
(101, 5001, 'Amit Sharma', 'Mumbai', '9876543210', '2025-07-10', 'active', 'savings', 2000.00),
(102, 5002, 'Priya Mehta', 'Mumbai', '9876500000', '2025-07-20', 'active', 'current', 1500.00),
(103, 5003, 'Rahul Verma', 'Pune', '9876512345', '2025-07-18', 'active', 'savings', 5000.00);

INSERT INTO Loan VALUES
(201, 1, 101, 10000.00, 'Home Loan'),
```

(202, 2, 102, 5000.00, 'Car Loan');

Explanation:
• We added three branches, three account holders, and two loans.
• Sample data helps us see how queries work and what outputs they generate.

3. FUND TRANSFER TRANSACTION

We need to transfer ₹100 from Amit Sharma's account (5001) to Priya Mehta's account (5002).

```
START TRANSACTION;

UPDATE Account_Holder
SET balance = balance - 100
WHERE account_no = 5001;

UPDATE Account_Holder
SET balance = balance + 100
WHERE account_no = 5002;

COMMIT;
```

Step-by-step explanation:
1. START TRANSACTION → begins a safe block so that either all queries run or none.
2. First UPDATE deducts 100 from account 5001.
3. Second UPDATE adds 100 to account 5002.
4. COMMIT → saves changes permanently.
Resulting balances: Amit Sharma = 1900.00, Priya Mehta = 1600.00

4. FETCH ACCOUNT HOLDERS FROM SAME CITY

```
SELECT *
FROM Account_Holder
WHERE city = 'Mumbai';
```

Explanation:
• WHERE city='Mumbai' filters only Mumbai-based customers.
• This gives us a list of account holders from Mumbai.
Expected Output Table:

account_holder_id	account_no	account_holder_name	city	balance

| 101 | 5001 | Amit Sharma | Mumbai | 1900.00 |
| 102 | 5002 | Priya Mehta | Mumbai | 1600.00 |

5. ACCOUNTS CREATED AFTER THE 15TH OF ANY MONTH

```
SELECT account_no, account_holder_name
FROM Account_Holder
WHERE DAY(date_of_account_created) > 15;
```

Explanation:
• DAY() extracts the day part from the date.
• We select accounts created on days greater than 15.
Expected Output:

account_no	account_holder_name
5002	Priya Mehta
5003	Rahul Verma

6. COUNT BRANCHES PER CITY

```
SELECT branch_city, COUNT(branch_id) AS Count_Branch
FROM Bank
GROUP BY branch_city;
```

Explanation:
• GROUP BY groups rows by city.
• COUNT(branch_id) counts branches per city.
Expected Output:

branch_city	Count_Branch
Mumbai	2
Pune	1

7. JOIN QUERY FOR LOAN HOLDERS

```
SELECT
   A.account_holder_id,
   A.account_holder_name,
```

```
   L.branch_id,
   L.loan_amount
FROM Account_Holder A
JOIN Loan L
ON A.account_holder_id = L.account_holder_id;
```

Explanation:
• JOIN connects Account_Holder and Loan tables using account_holder_id.
• Only customers with loans will appear.
Expected Output:

account_holder_id	account_holder_name	branch_id	loan_amount
101	Amit Sharma	1	10000.00
102	Priya Mehta	2	5000.00