SQL QUERIES

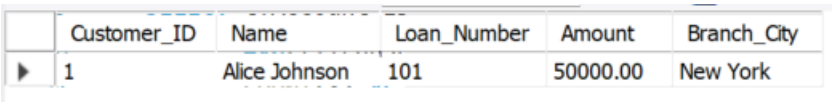
🔷 Query 1: Customers with Loans from a Specific Branch City

Query:

```
SELECT c.Customer_ID,
    c.Name,
    l.Loan_Number,
    l.Amount,
    br.Branch_City
FROM Customer c
JOIN Borrow b ON c.Customer_ID = b.Customer_ID
JOIN Loan l ON b.Loan_Number = l.Loan_Number
JOIN Branch br ON l.Branch_Name = br.Branch_Name
WHERE br.Branch_City = 'New York';
```

Explanation:

- This query retrieves all customers **who have taken loans from branches in New York**.
- It joins the **Customer, Borrow, Loan, and Branch** tables to filter loans **by branch city**.



Customer_ID	Name	Loan_Number	Amount	Branch_City
1	Alice Johnson	101	50000.00	New York

🔷 Query 2: Average Transaction Amount on Active Accounts

Query:

```
SELECT t.Account_ID,
    AVG(t.Transaction_Amount) AS avg_transaction,
    COUNT(*) AS transaction_count
FROM Transaction t
GROUP BY t.Account_ID
HAVING AVG(t.Transaction_Amount) > 100;
```

Explanation:

- This query finds accounts where the **average transaction amount is greater than $100**.
- It groups transactions by `Account_ID`, computes `AVG(Transaction_Amount)`, and applies a `HAVING` clause to filter out accounts that don't meet the criteria.

Account_ID	avg_transaction	transaction_count
1001	200.000000	1
1002	300.000000	1
1003	150.000000	1
1004	500.000000	1
1005	250.000000	1

🔷 Query 3: Employee Service Load (Number of Customers Served)

Query:

```
SELECT e.Employee_ID,
    e.Name,
    COUNT(b.Customer_ID) AS num_customers
FROM Employee e
JOIN Banker b ON e.Employee_ID = b.Employee_ID
GROUP BY e.Employee_ID, e.Name
ORDER BY num_customers DESC;
```

Explanation:

- This query identifies **employees who serve as bankers** and counts the number of customers they manage.
- Results are **sorted in descending order**, showing employees handling the highest number of customers first.

Employee_ID	Name	num_customers
2001	David White	1
2002	Emma Stone	1
2003	Frank Martin	1
2004	Grace Hall	1
2005	Hank Wilson	1
2006	Ivy Adams	1

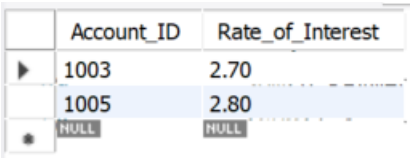
🔷 Query 4: Savings Accounts with Above-Average Interest Rates

Query:

```
SELECT sa.Account_ID,
    sa.Rate_of_Interest
FROM Savings_Acc sa
WHERE sa.Rate_of_Interest > (SELECT AVG(Rate_of_Interest) FROM Savings_Acc);
```

Explanation:

- This query finds **savings accounts** offering an **interest rate higher than the average interest rate** among all savings accounts.
- A **subquery** calculates the **average interest rate**, and the main query filters results based on that value.



Account_ID	Rate_of_Interest
1003	2.70
1005	2.80
NULL	NULL

🔷 Query 5: Branch Loan and Payment Summary

Query:

```
SELECT o.Branch_Name,
    SUM(l.Amount) AS total_loan,
    SUM(DISTINCT p.Payment_Amount) AS total_payment,
    (SUM(l.Amount) - SUM(p.Payment_Amount)) AS outstanding_amount
FROM Originated_By o
JOIN Loan l ON o.Loan_Number = l.Loan_Number
JOIN Payment p ON l.Loan_Number = p.Loan_Number
GROUP BY o.Branch_Name;
```

Explanation:

- This query provides a **financial summary for each bank branch**, showing:
  - **Total loans issued**
  - **Total payments received**
  - **Outstanding loan amount (loan amount - payments made)**

Branch_Name	total_loan	total_payment	outstanding_amount
Central	90000.00	900.00	89100.00
Downtown	50000.00	1000.00	49000.00
Eastside	55000.00	1100.00	53900.00
Midtown	60000.00	1300.00	58700.00
Uptown	75000.00	1200.00	73800.00
Westside	80000.00	1400.00	78600.00

🔷 Query 6: Top 3 Customers by Total Account Balance

Query:

```
SELECT d.Customer_ID,
    SUM(a.Balance) AS total_balance
FROM Deposit d
```

```
JOIN Account a ON d.Account_ID = a.Account_ID
GROUP BY d.Customer_ID
ORDER BY total_balance DESC
LIMIT 3;
```

Explanation:

- This query finds the **top 3 customers** who hold the highest total balances across all their accounts.
- It **aggregates balances per customer**, sorts them in descending order, and limits the results to **top 3 customers**.

Customer_ID	total_balance
6	4000.00
4	3000.00
3	2500.00

🔷 Query 7: Latest Transaction per Account

Query:

```
SELECT t.Account_ID,
       t.Transaction_Amount,
       t.Transaction_Date
FROM Transaction t
WHERE t.Transaction_Date = (
    SELECT MAX(t2.Transaction_Date)
    FROM Transaction t2
    WHERE t2.Account_ID = t.Account_ID
);
```

Explanation:

- This query retrieves **the most recent transaction for each account**.
- A **correlated subquery** finds the **maximum transaction date** for each account.

Account_ID	Transaction_Amount	Transaction_Date
1001	200.00	2024-02-10
1002	300.00	2024-02-11
1003	150.00	2024-02-12
1004	500.00	2024-02-13
1005	250.00	2024-02-14
1006	100.00	2024-02-15

🔷 Query 8: Counting Subordinates under Each Manager

Query:

```
SELECT m.Employee_ID,
    m.Name,
    COUNT(s.Employee_ID) AS subordinate_count
FROM Employee m
JOIN Employee s ON m.Employee_ID = s.Manager_ID
GROUP BY m.Employee_ID, m.Name;
```

Explanation:

- This query **identifies managers** and counts how many **subordinates** report to them.
- The Employee table **self-joins** on `Manager_ID` to track hierarchical relationships.

Employee_ID	Name	subordinate_count
2001	David White	2
2002	Emma Stone	1
2003	Frank Martin	1
2004	Grace Hall	1

🔷 Query 9: Cross-State Money Transfers

Query:

```
SELECT tm.From_Account_ID,
    tm.To_Account_ID
FROM Transfer_Money tm
JOIN Deposit d1 ON tm.From_Account_ID = d1.Account_ID
JOIN Customer c1 ON d1.Customer_ID = c1.Customer_ID
JOIN Deposit d2 ON tm.To_Account_ID = d2.Account_ID
JOIN Customer c2 ON d2.Customer_ID = c2.Customer_ID
WHERE c1.City = 'New York'
  AND c2.City = 'Los Angeles';
```

Explanation:

- This query finds **money transfers where the sender is from New York and the receiver is from Los Angeles**.
- It joins the **Transfer_Money, Deposit, and Customer tables** to filter accounts based on city.

From_Account_ID	To_Account_ID
1001	1002

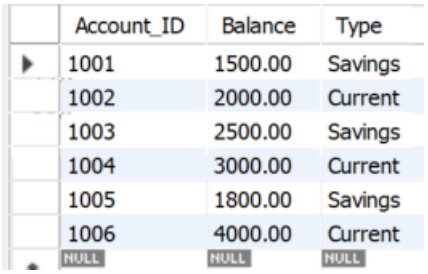
🔷 Query 10: Accounts with Transactions in Only a Specific Month

Query:

```
SELECT a.Account_ID,
       a.Balance,
       a.Type
FROM Account a
WHERE a.Account_ID IN (
    SELECT t.Account_ID
    FROM Transaction t
    GROUP BY t.Account_ID
    HAVING MAX(EXTRACT(MONTH FROM t.Transaction_Date)) = 2
);
```

Explanation:

- This query lists **accounts that had transactions exclusively in February** (`Month = 2`).
- It groups transactions by `Account_ID` and filters them using the **HAVING** clause to include only accounts where the most recent transaction is in February.



Account_ID	Balance	Type
1001	1500.00	Savings
1002	2000.00	Current
1003	2500.00	Savings
1004	3000.00	Current
1005	1800.00	Savings
1006	4000.00	Current
NULL	NULL	NULL

CONTRIBUTORS

1. Prateek Dhar
2. Priyanshu Sharma
3. Vansh tyagi
4. Umang Aggarwal