# BANKING & TRANSACTION FRAUD ANALYSIS


Name :- Omkar Manohar Sambare

# INTRODUCTION

**Banking and digital payments have increased rapidly due to UPI, online banking, card payments, and ATM usage. Along with this growth, the number of fraudulent transactions has also increased. Fraud can happen due to phishing, card skimming, fake loan apps, unauthorized transfers, and suspicious merchant activity. Detecting fraud early is important for banks to reduce financial loss and protect customers.**

**This project focuses on analyzing banking transaction data to identify possible fraud patterns using SQL queries. The dataset contains details of customers, their bank accounts, transaction records, merchant information, and reported fraud cases. By using SQL concepts like joins, subqueries, aggregate functions, and window functions, we can generate insights such as:**

- **Customers with high risk scores**
- **High-risk merchant transactions**
- **Fraud flagged transactions and their categories**
- **Transaction trends based on account and customer level**
- **Suspicious transactions from different locations**

**The main goal of this project is to build an intermediate-level fraud analysis system using structured relational tables, helping to understand transaction behavior and supporting fraud monitoring and reporting.**

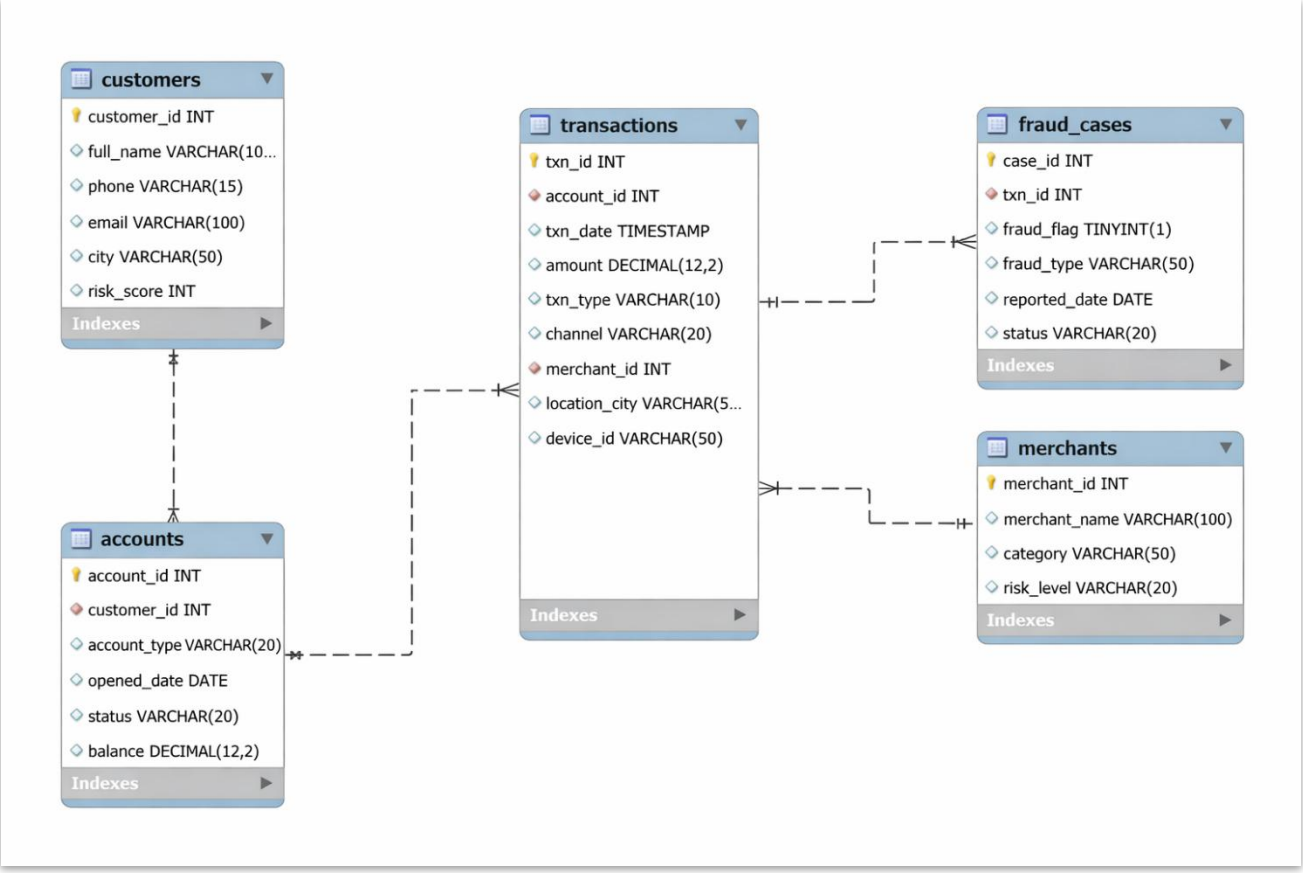
## Project Aim :

- **Customers Management** : To store and manage customer basic details and risk information, so we can identify high-risk customers and understand customer profiles involved in fraud transactions.
- **Accounts Management** : To maintain customer account details such as account type, status, and balance, which helps in analyzing account activity and checking unusual behavior like low balance accounts doing high-value transactions.
- **Merchants Management** : To store merchant information such as merchant category and risk level, which helps in tracking transactions made to high-risk merchants and identifying fraud-prone merchant categories
- **Transactions Management** : To record every transaction made by customers including amount, date, channel, location, and device, which helps detect suspicious transactions based on high amounts, unusual locations, and risky channels..
- **Fraud Cases Management :** To track fraud-reported transactions, fraud type, reported date, and case status, which helps in monitoring fraud investigations, fraud trends, and generating fraud reports.

## Project Objectives :

1. **To design and create a banking database structure** using tables like customers, accounts, transactions, merchants, and fraud cases.
2. **To analyze customer and account activity** such as account balance, transaction frequency, and transaction behavior.
3. **To identify suspicious and fraud-related transactions** using fraud flag details and transaction patterns.
4. **To monitor high-risk merchants** and evaluate fraud cases linked to merchant risk levels.
5. **To generate reports using SQL queries** such as total fraud amount, fraud cases by merchant category, and fraud status tracking.
6. **To practice intermediate SQL concepts** including joins, subqueries, aggregation, and window functions for real-world fraud analytics.

## ER Diagram :

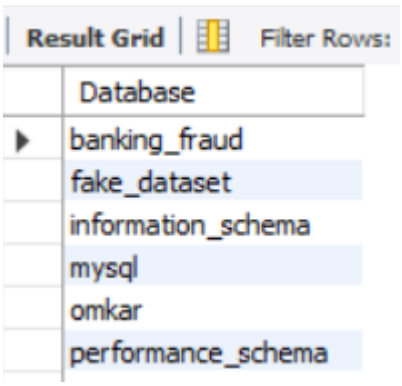


## Creating Database :

```
create database banking_fraud;

use banking_fraud;

Show databases;
```



Result Grid | Filter Rows:

| Database |
|---|
| banking_fraud |
| fake_dataset |
| information_schema |
| mysql |
| omkar |
| performance_schema |

## Tables in Bank_Management_System;

```
Show tables ;
```

| Tables_in_banking_fraud |
|---|
| accounts |
| customers |
| fraud_cases |
| merchants |
| transactions |

# DATA DEFINITION LANGUAGE (DDL) ;

## Creating Tables:

### A) Customers :

```
CREATE TABLE customers (
  customer_id INT PRIMARY KEY,
  full_name  VARCHAR(100) NOT NULL,
  phone      VARCHAR(15) UNIQUE,
  email      VARCHAR(100) UNIQUE,
  city       VARCHAR(50),
  risk_score  INT CHECK (risk_score BETWEEN 0 AND 100)
);

desc customers;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| customer_id | int | NO | PRI | NULL | |
| full_name | varchar(100) | NO | | NULL | |
| phone | varchar(15) | YES | UNI | NULL | |
| email | varchar(100) | YES | UNI | NULL | |
| city | varchar(50) | YES | | NULL | |
| risk_score | int | YES | | NULL | |

### B) Accounts :

```
CREATE TABLE accounts (
  account_id  INT PRIMARY KEY,
  customer_id  INT NOT NULL,
  account_type VARCHAR(20) CHECK (account_type IN ('Savings','Current')),
  opened_date  DATE,
  status      VARCHAR(20) CHECK (status IN ('Active','Closed')),
  balance     DECIMAL(12,2) DEFAULT 0,
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

desc accounts;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| account_id | int | NO | PRI | NULL | |
| customer_id | int | NO | MUL | NULL | |
| account_type | varchar(20) | YES | | NULL | |
| opened_date | date | YES | | NULL | |
| status | varchar(20) | YES | | NULL | |
| balance | decimal(12,2) | YES | | 0.00 | |

### C) Merchants :

```
CREATE TABLE merchants (
  merchant_id   INT PRIMARY KEY,
  merchant_name VARCHAR(100) NOT NULL,
  category      VARCHAR(50),
  risk_level    VARCHAR(20) CHECK (risk_level IN ('Low','Medium','High'))
);

desc merchants;
```

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| merchant_id | int | NO | PRI | NULL | |
| merchant_name | varchar(100) | NO | | NULL | |
| category | varchar(50) | YES | | NULL | |
| risk_level | varchar(20) | YES | | NULL | |

### D) Transactions :

```
CREATE TABLE transactions (
  txn_id      INT PRIMARY KEY,
  account_id  INT NOT NULL,
  txn_date    TIMESTAMP NOT NULL,
  amount      DECIMAL(12,2) NOT NULL,
  txn_type    VARCHAR(10) CHECK (txn_type IN ('Debit','Credit')),
  channel     VARCHAR(20) CHECK (channel IN ('ATM','UPI','CARD','NETBANK')),
  merchant_id  INT,
  location_city VARCHAR(50),
  device_id    VARCHAR(50),
        FOREIGN KEY (account_id) REFERENCES accounts(account_id),
  FOREIGN KEY (merchant_id) REFERENCES merchants(merchant_id)
```

```
);
desc transactions;
```

| | Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|---|
| ▶ | txn_id | int | NO | PRI | NULL | |
| | account_id | int | NO | MUL | NULL | |
| | txn_date | timestamp | NO | | NULL | |
| | amount | decimal(12,2) | NO | | NULL | |
| | txn_type | varchar(10) | YES | | NULL | |
| | channel | varchar(20) | YES | | NULL | |
| | merchant_id | int | YES | MUL | NULL | |
| | location_city | varchar(50) | YES | | NULL | |
| | device_id | varchar(50) | YES | | NULL | |

## E) Fraud_Cases :

```
CREATE TABLE fraud_cases (
  case_id     INT PRIMARY KEY,
  txn_id      INT NOT NULL,
  fraud_flag  BOOLEAN NOT NULL,
  fraud_type  VARCHAR(50),
  reported_date DATE,
  status      VARCHAR(20) CHECK (status IN ('Open','Closed','Investigating')),
        FOREIGN KEY (txn_id) REFERENCES transactions(txn_id)
);

desc fraud_cases;
```

# DATA MANUPULATION LANGUAGE (DML) :

## Insert into tables:

### Inserting values into tables(Customers):

Insert into **customers**

Value

**(1, 'Amit Sharma', '9876543210', 'amit.sharma@gmail.com', 'Mumbai', 25),**

**(2, 'Neha Verma', '9123456780', 'neha.verma@gmail.com', 'Delhi', 55),**

**(3, 'Rohit Mehta', '9988776655', 'rohit.mehta@gmail.com', 'Bangalore', 40),**

**(4, 'Priya Singh', '9090909090', 'priya.singh@gmail.com', 'Chennai', 70),**

**(5, 'Karan Patel', '9012345678', 'karan.patel@gmail.com', 'Ahmedabad', 35),**

**(6, 'Sneha Joshi', '9345678123', 'sneha.joshi@gmail.com', 'Pune', 60),**

**(7, 'Vikram Rao', '9765432109', 'vikram.rao@gmail.com', 'Hyderabad', 45),**

**(8, 'Anjali Gupta', '9898989898', 'anjali.gupta@gmail.com', 'Kolkata', 80),**

**(9, 'Suresh Kumar', '9555666777', 'suresh.kumar@gmail.com', 'Jaipur', 30),**

**(10,'Meera Nair', '9444555666', 'meera.nair@gmail.com', 'Kochi', 65);**

**Select * from customer;**

### Output:

| customer_id | full_name | phone | email | city | risk_score |
|---|---|---|---|---|---|
| 1 | Amit Sharma | 9876543210 | amit.sharma@gmail.com | Mumbai | 25 |
| 2 | Neha Verma | 9123456780 | neha.verma@gmail.com | Delhi | 55 |
| 3 | Rohit Mehta | 9988776655 | rohit.mehta@gmail.com | Bangalore | 40 |
| 4 | Priya Singh | 9090909090 | priya.singh@gmail.com | Chennai | 70 |
| 5 | Karan Patel | 9012345678 | karan.patel@gmail.com | Ahmedabad | 35 |
| 6 | Sneha Joshi | 9345678123 | sneha.joshi@gmail.com | Pune | 60 |
| 7 | Vikram Rao | 9765432109 | vikram.rao@gmail.com | Hyderabad | 45 |
| 8 | Anjali Gupta | 9898989898 | anjali.gupta@gmail.com | Kolkata | 80 |
| 9 | Suresh Kumar | 9555666777 | suresh.kumar@gmail.com | Jaipur | 30 |
| 10 | Meera Nair | 9444555666 | meera.nair@gmail.com | Kochi | 65 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Inserting values into tables(Accounts):

Insert into **accounts**

Value

**(101, 1, 'Savings', '2022-01-10', 'Active', 55000),**

**(102, 2, 'Current', '2021-08-05', 'Active', 120000),**

**(103, 3, 'Savings', '2020-11-20', 'Active', 75000),**

**(104, 4, 'Savings', '2019-03-15', 'Active', 20000),**

**(105, 5, 'Current', '2023-06-01', 'Active', 500000),**

**(106, 6, 'Savings', '2022-09-12', 'Active', 15000),**

**(107, 7, 'Savings', '2021-02-18', 'Active', 98000),**

**(108, 8, 'Current', '2020-05-30', 'Active', 670000),**

**(109, 9, 'Savings', '2023-01-25', 'Active', 34000),**

**(110, 10,'Savings', '2018-12-11', 'Closed', 0);**

Select * from **accounts;**

Output:

| account_id | customer_id | account_type | opened_date | status | balance |
|---|---|---|---|---|---|
| 101 | 1 | Savings | 2022-01-10 | Active | 55000.00 |
| 102 | 2 | Current | 2021-08-05 | Active | 120000.00 |
| 103 | 3 | Savings | 2020-11-20 | Active | 75000.00 |
| 104 | 4 | Savings | 2019-03-15 | Active | 20000.00 |
| 105 | 5 | Current | 2023-06-01 | Active | 500000.00 |
| 106 | 6 | Savings | 2022-09-12 | Active | 15000.00 |
| 107 | 7 | Savings | 2021-02-18 | Active | 98000.00 |
| 108 | 8 | Current | 2020-05-30 | Active | 670000.00 |
| 109 | 9 | Savings | 2023-01-25 | Active | 34000.00 |
| 110 | 10 | Savings | 2018-12-11 | Closed | 0.00 |
| NULL | NULL | NULL | NULL | NULL | NULL |

## Inserting values into tables(Merchants):

```
INSERT INTO merchants (merchant_id, merchant_name, category, risk_level)
VALUES
(201, 'Amazon', 'Shopping', 'Low'),
(202, 'Flipkart', 'Shopping', 'Low'),
(203, 'Swiggy', 'Food', 'Low'),
(204, 'Zomato', 'Food', 'Low'),
(205, 'IRCTC', 'Travel', 'Medium'),
(206, 'MakeMyTrip', 'Travel', 'Medium'),
(207, 'CryptoXChange', 'Crypto', 'High'),
(208, 'FastLoanApp', 'Loan', 'High'),
(209, 'PetrolPump-HP', 'Fuel', 'Medium'),
(210, 'LuxuryStore', 'Shopping', 'High');
```

```
Select * from merchants;
```

Output:

| merchant_id | merchant_name | category | risk_level |
|---|---|---|---|
| 201 | Amazon | Shopping | Low |
| 202 | Flipkart | Shopping | Low |
| 203 | Swiggy | Food | Low |
| 204 | Zomato | Food | Low |
| 205 | IRCTC | Travel | Medium |
| 206 | MakeMyTrip | Travel | Medium |
| 207 | CryptoXChange | Crypto | High |
| 208 | FastLoanApp | Loan | High |
| 209 | PetrolPump-HP | Fuel | Medium |
| 210 | LuxuryStore | Shopping | High |
| NULL | NULL | NULL | NULL |

## Inserting values into tables(Transactions):

```
INSERT INTO transactions (txn_id, account_id, txn_date, amount, txn_type, channel,
merchant_id, location_city, device_id)
VALUES
(301, 101, '2025-01-05 10:15:00', 2500,   'Debit', 'UPI',    203, 'Mumbai',   'DEV001'),
(302, 102, '2025-01-05 12:45:00', 75000,  'Debit', 'NETBANK', 205, 'Delhi',    'DEV002'),
(303, 103, '2025-01-06 09:10:00', 1500,   'Debit', 'CARD',   204, 'Bangalore', 'DEV003'),
(304, 104, '2025-01-06 20:20:00', 98000,  'Debit', 'CARD',   210, 'Chennai',  'DEV004'),
(305, 105, '2025-01-07 11:00:00', 250000, 'Debit', 'NETBANK', 207, 'Ahmedabad', 'DEV005'),
(306, 106, '2025-01-07 18:30:00', 5000,   'Debit', 'ATM',    209, 'Pune',     'DEV006'),
(307, 107, '2025-01-08 08:10:00', 12000,  'Debit', 'UPI',    201, 'Hyderabad', 'DEV007'),
(308, 108, '2025-01-08 22:05:00', 350000, 'Debit', 'NETBANK', 208, 'Kolkata',  'DEV008'),
(309, 109, '2025-01-09 14:55:00', 4000,   'Debit', 'UPI',    203, 'Jaipur',   'DEV009'),
(310, 101, '2025-01-09 23:59:00', 90000,  'Debit', 'CARD',   207, 'Goa',      'DEV010');
```

```
select * from transactions;
```

### Output:

| txn_id | account_id | txn_date | amount | txn_type | channel | merchant_id | location_city | device_id |
|---|---|---|---|---|---|---|---|---|
| 301 | 101 | 2025-01-05 10:15:00 | 2500.00 | Debit | UPI | 203 | Mumbai | DEV001 |
| 302 | 102 | 2025-01-05 12:45:00 | 75000.00 | Debit | NETBANK | 205 | Delhi | DEV002 |
| 303 | 103 | 2025-01-06 09:10:00 | 1500.00 | Debit | CARD | 204 | Bangalore | DEV003 |
| 304 | 104 | 2025-01-06 20:20:00 | 98000.00 | Debit | CARD | 210 | Chennai | DEV004 |
| 305 | 105 | 2025-01-07 11:00:00 | 250000.00 | Debit | NETBANK | 207 | Ahmedabad | DEV005 |
| 306 | 106 | 2025-01-07 18:30:00 | 5000.00 | Debit | ATM | 209 | Pune | DEV006 |
| 307 | 107 | 2025-01-08 08:10:00 | 12000.00 | Debit | UPI | 201 | Hyderabad | DEV007 |
| 308 | 108 | 2025-01-08 22:05:00 | 350000.00 | Debit | NETBANK | 208 | Kolkata | DEV008 |
| 309 | 109 | 2025-01-09 14:55:00 | 4000.00 | Debit | UPI | 203 | Jaipur | DEV009 |
| 310 | 101 | 2025-01-09 23:59:00 | 90000.00 | Debit | CARD | 207 | Goa | DEV010 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

## Inserting values into tables(Fraud_Cases):

```
INSERT INTO fraud_cases (case_id, txn_id, fraud_flag, fraud_type, reported_date, status)
VALUES
(401, 301, FALSE, NULL,          NULL,       'Closed'),
(402, 302, FALSE, NULL,          NULL,       'Closed'),
(403, 303, FALSE, NULL,          NULL,       'Closed'),
(404, 304, TRUE,  'Card Skimming', '2025-01-07','Investigating'),
(405, 305, TRUE,  'Crypto Scam',   '2025-01-08','Open'),
(406, 306, FALSE, NULL,          NULL,       'Closed'),
(407, 307, FALSE, NULL,          NULL,       'Closed'),
(408, 308, TRUE,  'Loan App Fraud','2025-01-09','Open'),
(409, 309, FALSE, NULL,          NULL,       'Closed'),
(410, 310, TRUE,  'Phishing',      '2025-01-10','Investigating');
```

```
select * from fraud_cases;
```

Output:

| case_id | txn_id | fraud_flag | fraud_type | reported_date | status |
|---|---|---|---|---|---|
| 401 | 301 | 0 | NULL | NULL | Closed |
| 402 | 302 | 0 | NULL | NULL | Closed |
| 403 | 303 | 0 | NULL | NULL | Closed |
| 404 | 304 | 1 | Card Skimming | 2025-01-07 | Investigating |
| 405 | 305 | 1 | Crypto Scam | 2025-01-08 | Open |
| 406 | 306 | 0 | NULL | NULL | Closed |
| 407 | 307 | 0 | NULL | NULL | Closed |
| 408 | 308 | 1 | Loan App Fraud | 2025-01-09 | Open |
| 409 | 309 | 0 | NULL | NULL | Closed |
| 410 | 310 | 1 | Phishing | 2025-01-10 | Investigating |
| NULL | NULL | NULL | NULL | NULL | NULL |

# Basic Questions

Q1) Show all customers from Mumbai ?

Answer:-

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

Output:-

| customer_id | full_name | phone | email | city | risk_score |
|---|---|---|---|---|---|
| 1 | Amit Sharma | 9876543210 | amit.sharma@gmail.com | Mumbai | 25 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Q2) Show all active accounts ?

Answer:-

```
SELECT * FROM accounts
WHERE status = 'Active' ;
```

Output:-

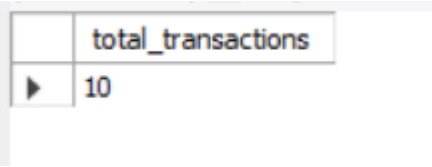
| account_id | customer_id | account_type | opened_date | status | balance |
|---|---|---|---|---|---|
| 101 | 1 | Savings | 2022-01-10 | Active | 55000.00 |
| 102 | 2 | Current | 2021-08-05 | Active | 120000.00 |
| 103 | 3 | Savings | 2020-11-20 | Active | 75000.00 |
| 104 | 4 | Savings | 2019-03-15 | Active | 20000.00 |
| 105 | 5 | Current | 2023-06-01 | Active | 500000.00 |
| 106 | 6 | Savings | 2022-09-12 | Active | 15000.00 |
| 107 | 7 | Savings | 2021-02-18 | Active | 98000.00 |
| 108 | 8 | Current | 2020-05-30 | Active | 670000.00 |
| 109 | 9 | Savings | 2023-01-25 | Active | 34000.00 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Q3) Find total number of transactions ?

Answers :-

```
SELECT COUNT(*) AS total_transactions
FROM transactions;
```

Output :-



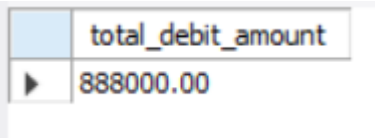
| total_transactions |
| --- |
| 10 |

Q4) Find total amount of debit transactions

Answer :-

```
SELECT SUM(amount) AS total_debit_amount
FROM transactions
WHERE txn_type = 'Debit';
```

Output :-



| total_debit_amount |
| --- |
| 888000.00 |

Q5) Find top 5 highest transactions ?

Answer :-

```
SELECT *FROM transactions
ORDER BY amount DESC LIMIT 5;
```

Output :-

| txn_id | account_id | txn_date | amount | txn_type | channel | merchant_id | location_city | device_id |
|---|---|---|---|---|---|---|---|---|
| 308 | 108 | 2025-01-08 22:05:00 | 350000.00 | Debit | NETBANK | 208 | Kolkata | DEV008 |
| 305 | 105 | 2025-01-07 11:00:00 | 250000.00 | Debit | NETBANK | 207 | Ahmedabad | DEV005 |
| 304 | 104 | 2025-01-06 20:20:00 | 98000.00 | Debit | CARD | 210 | Chennai | DEV004 |
| 310 | 101 | 2025-01-09 23:59:00 | 90000.00 | Debit | CARD | 207 | Goa | DEV010 |
| 302 | 102 | 2025-01-05 12:45:00 | 75000.00 | Debit | NETBANK | 205 | Delhi | DEV002 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Q6) Find merchants with risk_level = 'High' ?

Answer :-

```
SELECT * FROM merchants
WHERE risk_level = 'High';
```

Output :-

| merchant_id | merchant_name | category | risk_level |
|---|---|---|---|
| 207 | CryptoXChange | Crypto | High |
| 208 | FastLoanApp | Loan | High |
| 210 | LuxuryStore | Shopping | High |
| NULL | NULL | NULL | NULL |

Q7) Find total fraud amount (fraud_flag = 1) ?

Answer :-

```
SELECT SUM(t.amount) AS total_fraud_amount
FROM fraud_cases f
JOIN transactions t ON f.txn_id = t.txn_id
WHERE f.fraud_flag = TRUE;
```

Output :-

| total_fraud_amount |
|---|
| 788000.00 |

Q8) Insert a new merchant ?

Answer :-

```
INSERT INTO merchants (merchant_id, merchant_name, category, risk_level)
VALUES (211, 'Uber', 'Travel', 'Medium');
```

Output :-

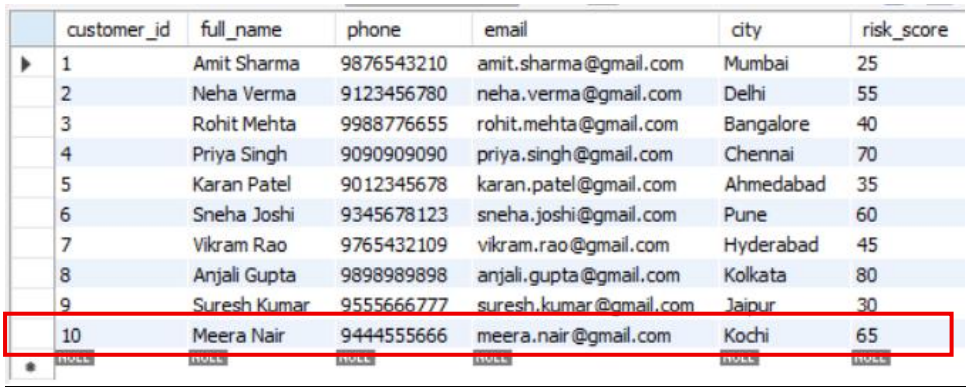
| merchant_id | merchant_name | category | risk_level |
|---|---|---|---|
| 202 | Flipkart | Shopping | Low |
| 203 | Swiggy | Food | Low |
| 204 | Zomato | Food | Low |
| 205 | IRCTC | Travel | Medium |
| 206 | MakeMyTrip | Travel | Medium |
| 207 | CryptoXChange | Crypto | High |
| 208 | FastLoanApp | Loan | High |
| 209 | PetrolPump-HP | Fuel | Medium |
| 210 | LuxuryStore | Shopping | High |
| 211 | Uber | Travel | Medium |
| NULL | NULL | NULL | NULL |

Q9) Update customer risk_score to 90 for customer_id = 11 ?

Answer :-

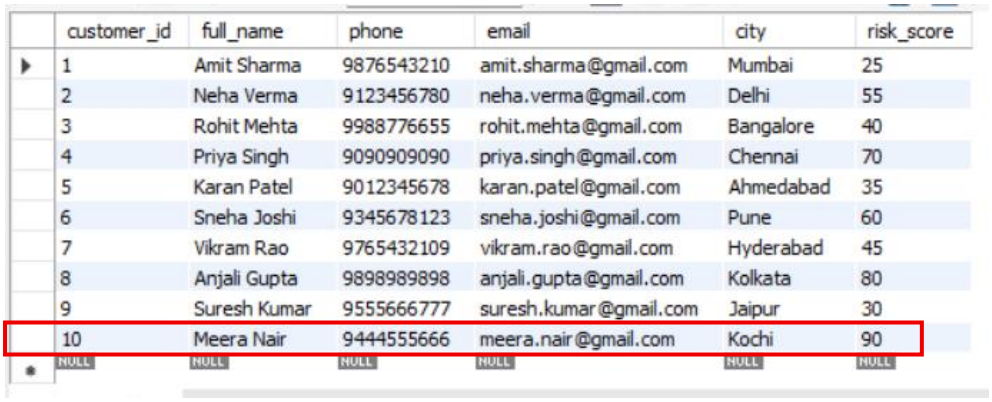
```
UPDATE customers
SET risk_score = 90
WHERE customer_id = 11;
```

Output :- OLD TABLE :-



| customer_id | full_name | phone | email | city | risk_score |
|---|---|---|---|---|---|
| 1 | Amit Sharma | 9876543210 | amit.sharma@gmail.com | Mumbai | 25 |
| 2 | Neha Verma | 9123456780 | neha.verma@gmail.com | Delhi | 55 |
| 3 | Rohit Mehta | 9988776655 | rohit.mehta@gmail.com | Bangalore | 40 |
| 4 | Priya Singh | 9090909090 | priya.singh@gmail.com | Chennai | 70 |
| 5 | Karan Patel | 9012345678 | karan.patel@gmail.com | Ahmedabad | 35 |
| 6 | Sneha Joshi | 9345678123 | sneha.joshi@gmail.com | Pune | 60 |
| 7 | Vikram Rao | 9765432109 | vikram.rao@gmail.com | Hyderabad | 45 |
| 8 | Anjali Gupta | 9898989898 | anjali.gupta@gmail.com | Kolkata | 80 |
| 9 | Suresh Kumar | 9555666777 | suresh.kumar@gmail.com | Jaipur | 30 |
| 10 | Meera Nair | 9444555666 | meera.nair@gmail.com | Kochi | 65 |

NEW TABLE :-



| customer_id | full_name | phone | email | city | risk_score |
|---|---|---|---|---|---|
| 1 | Amit Sharma | 9876543210 | amit.sharma@gmail.com | Mumbai | 25 |
| 2 | Neha Verma | 9123456780 | neha.verma@gmail.com | Delhi | 55 |
| 3 | Rohit Mehta | 9988776655 | rohit.mehta@gmail.com | Bangalore | 40 |
| 4 | Priya Singh | 9090909090 | priya.singh@gmail.com | Chennai | 70 |
| 5 | Karan Patel | 9012345678 | karan.patel@gmail.com | Ahmedabad | 35 |
| 6 | Sneha Joshi | 9345678123 | sneha.joshi@gmail.com | Pune | 60 |
| 7 | Vikram Rao | 9765432109 | vikram.rao@gmail.com | Hyderabad | 45 |
| 8 | Anjali Gupta | 9898989898 | anjali.gupta@gmail.com | Kolkata | 80 |
| 9 | Suresh Kumar | 9555666777 | suresh.kumar@gmail.com | Jaipur | 30 |
| 10 | Meera Nair | 9444555666 | meera.nair@gmail.com | Kochi | 90 |

Q10) Insert multiple transactions in one query ?

Answer :-

INSERT INTO transactions (txn_id, account_id, txn_date, amount, txn_type, channel, merchant_id, location_city, device_id)

VALUES

(312, 101, '2025-02-05 09:00:00', 1200, 'Debit', 'UPI', 203, 'Mumbai', 'DEV012'),

(313, 102, '2025-02-05 11:15:00', 50000, 'Debit', 'CARD', 210, 'Delhi', 'DEV013'),

(314, 103, '2025-02-05 14:45:00', 3000, 'Debit', 'ATM', 209, 'Bangalore', 'DEV014');

Output :-

| txn_id | account_id | txn_date | amount | txn_type | channel | merchant_id | location_city | device_id |
|---|---|---|---|---|---|---|---|---|
| 301 | 101 | 2025-01-05 10:15:00 | 2500.00 | Debit | UPI | 203 | Mumbai | DEV001 |
| 302 | 102 | 2025-01-05 12:45:00 | 75000.00 | Debit | NETBANK | 205 | Delhi | DEV002 |
| 303 | 103 | 2025-01-06 09:10:00 | 1500.00 | Debit | CARD | 204 | Bangalore | DEV003 |
| 304 | 104 | 2025-01-06 20:20:00 | 98000.00 | Debit | CARD | 210 | Chennai | DEV004 |
| 305 | 105 | 2025-01-07 11:00:00 | 250000.00 | Debit | NETBANK | 207 | Ahmedabad | DEV005 |
| 306 | 106 | 2025-01-07 18:30:00 | 5000.00 | Debit | ATM | 209 | Pune | DEV006 |
| 307 | 107 | 2025-01-08 08:10:00 | 12000.00 | Debit | UPI | 201 | Hyderabad | DEV007 |
| 308 | 108 | 2025-01-08 22:05:00 | 350000.00 | Debit | NETBANK | 208 | Kolkata | DEV008 |
| 309 | 109 | 2025-01-09 14:55:00 | 4000.00 | Debit | UPI | 203 | Jaipur | DEV009 |
| 310 | 101 | 2025-01-09 23:59:00 | 90000.00 | Debit | CARD | 207 | Goa | DEV010 |
| 312 | 101 | 2025-02-05 09:00:00 | 1200.00 | Debit | UPI | 203 | Mumbai | DEV012 |
| 313 | 102 | 2025-02-05 11:15:00 | 50000.00 | Debit | CARD | 210 | Delhi | DEV013 |
| 314 | 103 | 2025-02-05 14:45:00 | 3000.00 | Debit | ATM | 209 | Bangalore | DEV014 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

# SUB QUERIES

Q1) Find customers with risk_score higher than average ?

Answer:-

```
SELECT *FROM customers
WHERE risk_score > (SELECT AVG(risk_score) FROM customers);
```

Output :-

| customer_id | full_name | phone | email | city | risk_score |
|---|---|---|---|---|---|
| 2 | Neha Verma | 9123456780 | neha.verma@gmail.com | Delhi | 55 |
| 4 | Priya Singh | 9090909090 | priya.singh@gmail.com | Chennai | 70 |
| 6 | Sneha Joshi | 9345678123 | sneha.joshi@gmail.com | Pune | 60 |
| 8 | Anjali Gupta | 9898989898 | anjali.gupta@gmail.com | Kolkata | 80 |
| 10 | Meera Nair | 9444555666 | meera.nair@gmail.com | Kochi | 90 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Q2) Find accounts having balance greater than average balance ?

Answer :-

```
SELECT *FROM accounts
WHERE balance > (SELECT AVG(balance) FROM accounts);
```

Output :-

| account_id | customer_id | account_type | opened_date | status | balance |
|---|---|---|---|---|---|
| 105 | 5 | Current | 2023-06-01 | Active | 500000.00 |
| 108 | 8 | Current | 2020-05-30 | Active | 670000.00 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Q3) Find merchants who have at least 1 transaction ?

Answer :-

```
SELECT *FROM merchants
WHERE merchant_id IN (SELECT DISTINCT merchant_id FROM transactions);
```

Output :-

| merchant_id | merchant_name | category | risk_level |
|---|---|---|---|
| 201 | Amazon | Shopping | Low |
| 203 | Swiggy | Food | Low |
| 204 | Zomato | Food | Low |
| 205 | IRCTC | Travel | Medium |
| 207 | CryptoXChange | Crypto | High |
| 208 | FastLoanApp | Loan | High |
| 209 | PetrolPump-HP | Fuel | Medium |
| 210 | LuxuryStore | Shopping | High |
| NULL | NULL | NULL | NULL |

Q4) Find transactions greater than the highest transaction of account_id = 101 ?

Answer :-

```
SELECT * FROM transactions
WHERE amount > (SELECT MAX(amount)
        FROM transactions
        WHERE account_id = 101);
```

Output :-

| txn_id | account_id | txn_date | amount | txn_type | channel | merchant_id | location_city | device_id |
|---|---|---|---|---|---|---|---|---|
| 304 | 104 | 2025-01-06 20:20:00 | 98000.00 | Debit | CARD | 210 | Chennai | DEV004 |
| 305 | 105 | 2025-01-07 11:00:00 | 250000.00 | Debit | NETBANK | 207 | Ahmedabad | DEV005 |
| 308 | 108 | 2025-01-08 22:05:00 | 350000.00 | Debit | NETBANK | 208 | Kolkata | DEV008 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Q5) Find customers who have done at least 1 transaction ?

Answer :-

```
SELECT *FROM customers
WHERE customer_id IN (
  SELECT a.customer_id
  FROM accounts a
  JOIN transactions t ON a.account_id = t.account_id);
```

Output :-

| customer_id | full_name | phone | email | city | risk_score |
|---|---|---|---|---|---|
| 1 | Amit Sharma | 9876543210 | amit.sharma@gmail.com | Mumbai | 25 |
| 2 | Neha Verma | 9123456780 | neha.verma@gmail.com | Delhi | 55 |
| 3 | Rohit Mehta | 9988776655 | rohit.mehta@gmail.com | Bangalore | 40 |
| 4 | Priya Singh | 9090909090 | priya.singh@gmail.com | Chennai | 70 |
| 5 | Karan Patel | 9012345678 | karan.patel@gmail.com | Ahmedabad | 35 |
| 6 | Sneha Joshi | 9345678123 | sneha.joshi@gmail.com | Pune | 60 |
| 7 | Vikram Rao | 9765432109 | vikram.rao@gmail.com | Hyderabad | 45 |
| 8 | Anjali Gupta | 9898989898 | anjali.gupta@gmail.com | Kolkata | 80 |
| 9 | Suresh Kumar | 9555666777 | suresh.kumar@gmail.com | Jaipur | 30 |
| NULL | NULL | NULL | NULL | NULL | NULL |

# JOINS

Q1) Show customer name, account_id, account_type, and balance ?

Answer :-

```
SELECT c.full_name, a.account_id, a.account_type, a.balance
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id;
```

Output :-

| full_name | account_id | account_type | balance |
|---|---|---|---|
| Amit Sharma | 101 | Savings | 55000.00 |
| Neha Verma | 102 | Current | 120000.00 |
| Rohit Mehta | 103 | Savings | 75000.00 |
| Priya Singh | Rohit Mehta | Savings | 20000.00 |
| Karan Patel | 105 | Current | 500000.00 |
| Sneha Joshi | 106 | Savings | 15000.00 |
| Vikram Rao | 107 | Savings | 98000.00 |
| Anjali Gupta | 108 | Current | 670000.00 |
| Suresh Kumar | 109 | Savings | 34000.00 |
| Meera Nair | 110 | Savings | 0.00 |

Q2) List all transactions with merchant name and merchant category ?

Answer :-

```
SELECT t.txn_id, t.txn_date, t.amount, m.merchant_name, m.category
FROM transactions t
JOIN merchants m ON t.merchant_id = m.merchant_id;
```

Output :-

| txn_id | txn_date | amount | merchant_name | category |
|---|---|---|---|---|
| 301 | 2025-01-05 10:15:00 | 2500.00 | Swiggy | Food |
| 302 | 2025-01-05 12:45:00 | 75000.00 | IRCTC | Travel |
| 303 | 2025-01-06 09:10:00 | 1500.00 | Zomato | Food |
| 304 | 2025-01-06 20:20:00 | 98000.00 | LuxuryStore | Shopping |
| 305 | 2025-01-07 11:00:00 | 250000.00 | CryptoXChange | Crypto |
| 306 | 2025-01-07 18:30:00 | 5000.00 | PetrolPump-HP | Fuel |
| 307 | 2025-01-08 08:10:00 | 12000.00 | Amazon | Shopping |
| 308 | 2025-01-08 22:05:00 | 350000.00 | FastLoanApp | Loan |
| 309 | 2025-01-09 14:55:00 | 4000.00 | Swiggy | Food |
| 310 | 2025-01-09 23:59:00 | 90000.00 | CryptoXChange | Crypto |
| 312 | 2025-02-05 09:00:00 | 1200.00 | Swiggy | Food |
| 313 | 2025-02-05 11:15:00 | 50000.00 | LuxuryStore | Shopping |
| 314 | 2025-02-05 14:45:00 | 3000.00 | PetrolPump-HP | Fuel |

Q3) Show fraud cases with transaction amount, channel, and customer name ?

Answer :-

```
SELECT f.case_id, c.full_name, t.amount, t.channel, f.fraud_type
FROM fraud_cases f
JOIN transactions t ON f.txn_id = t.txn_id
JOIN accounts a ON t.account_id = a.account_id
JOIN customers c ON a.customer_id = c.customer_id
WHERE f.fraud_flag = TRUE;
```

Output :-

| case_id | full_name | amount | channel | fraud_type |
|---|---|---|---|---|
| 404 | Priya Singh | 98000.00 | CARD | Card Skimming |
| 405 | Karan Patel | 250000.00 | NETBANK | Crypto Scam |
| 408 | Anjali Gupta | 350000.00 | NETBANK | Loan App Fraud |
| 410 | Amit Sharma | 90000.00 | CARD | Phishing |

Q4) Show all transactions with account type and customer city ?

Answer :-

```
SELECT t.txn_id, t.amount, a.account_type, c.city
FROM transactions t
JOIN accounts a ON t.account_id = a.account_id
JOIN customers c ON a.customer_id = c.customer_id;
```

Output :-

| txn_id | amount | account_type | city |
|---|---|---|---|
| 301 | 2500.00 | Savings | Mumbai |
| 310 | 90000.00 | Savings | Mumbai |
| 312 | 1200.00 | Savings | Mumbai |
| 302 | 75000.00 | Current | Delhi |
| 313 | 50000.00 | Current | Delhi |
| 303 | 1500.00 | Savings | Bangalore |
| 314 | 3000.00 | Savings | Bangalore |
| 304 | 98000.00 | Savings | Chennai |
| 305 | 250000.00 | Current | Ahmedabad |
| 306 | 5000.00 | Savings | Pune |
| 307 | 12000.00 | Savings | Hyderabad |
| 308 | 350000.00 | Current | Kolkata |
| 309 | 4000.00 | Savings | Jaipur |

Q5) Show merchant risk level with total transaction count ?

Answer :-

```
SELECT m.risk_level, COUNT(t.txn_id) AS total_txns
FROM merchants m
JOIN transactions t ON m.merchant_id = t.merchant_id
GROUP BY m.risk_level;
```

Output :-

| risk_level | total_txns |
|---|---|
| Low | 5 |
| Medium | 3 |
| High | 5 |

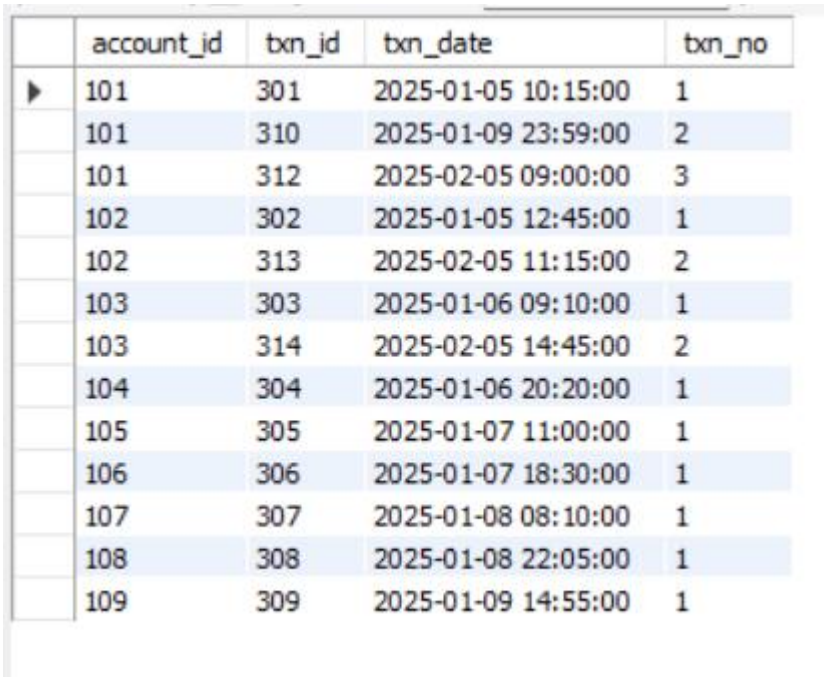
# Window Function

Q1) Show transaction number for each account ?

Answer :-

```
SELECT account_id, txn_id, txn_date,
ROW_NUMBER() OVER (PARTITION BY account_id ORDER BY txn_date) AS txn_no
FROM transactions;
```

Output :-



| account_id | txn_id | txn_date | txn_no |
|---|---|---|---|
| 101 | 301 | 2025-01-05 10:15:00 | 1 |
| 101 | 310 | 2025-01-09 23:59:00 | 2 |
| 101 | 312 | 2025-02-05 09:00:00 | 3 |
| 102 | 302 | 2025-01-05 12:45:00 | 1 |
| 102 | 313 | 2025-02-05 11:15:00 | 2 |
| 103 | 303 | 2025-01-06 09:10:00 | 1 |
| 103 | 314 | 2025-02-05 14:45:00 | 2 |
| 104 | 304 | 2025-01-06 20:20:00 | 1 |
| 105 | 305 | 2025-01-07 11:00:00 | 1 |
| 106 | 306 | 2025-01-07 18:30:00 | 1 |
| 107 | 307 | 2025-01-08 08:10:00 | 1 |
| 108 | 308 | 2025-01-08 22:05:00 | 1 |
| 109 | 309 | 2025-01-09 14:55:00 | 1 |

# CONCLUSION

In this **Banking & Transaction Fraud Analysis project**, we successfully created a structured relational database using five main tables: **customers, accounts, transactions, merchants, and fraud_cases.** This database helped us store and analyze banking transaction data in an organized way.

Using SQL queries, we performed different types of analysis such as **identifying customer transaction details, tracking high-risk merchant transactions, checking fraud flagged cases**, and generating summary reports like total fraud amount and fraud transactions by category. We also used important SQL concepts such as **joins, subqueries, aggregate functions, and window functions** to extract meaningful insights from the data.

Overall, this project improves understanding of how fraud monitoring works in banking systems and shows how SQL can be used to **detect suspicious activities, support fraud investigations, and reduce financial risk**. This project can be further enhanced by adding more data, applying real-time fraud rules, and building dashboards for better visualization and decision-making.