Bank Fraud Analysis Project - Portfolio Ready

Folder Structure

```
Bank_Fraud_Analysis/
│
├── README.md
├── schema.sql
├── data.sql
├── queries.sql
└── screenshots/
```

README.md Content

```
# Bank Fraud Analysis Project

## Project Overview
This project analyzes bank transactions to detect fraud patterns using SQL
queries.

## Database Schema
1. customers(customer_id, name, age, gender, branch_id)
2. branches(branch_id, branch_name, city)
3. transactions(transaction_id, customer_id, branch_id, amount,
transaction_date, is_fraud)

## Sample Queries & Insights
1. Month-wise Fraud Trend
2. High-Risk Customers
3. Branch-wise Fraud Analysis
4. Fraud Amount Patterns

## How to Run
1. Run schema.sql to create tables.
2. Run data.sql to insert sample data.
3. Run queries.sql to see insights.
```

---

schema.sql

```sql
CREATE TABLE customers(
    customer_id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    gender VARCHAR(10),
    branch_id INT
);

CREATE TABLE branches(
    branch_id INT PRIMARY KEY,
    branch_name VARCHAR(50),
    city VARCHAR(50)
);

CREATE TABLE transactions(
    transaction_id INT PRIMARY KEY,
    customer_id INT,
    branch_id INT,
    amount DECIMAL(10,2),
    transaction_date DATE,
    is_fraud TINYINT(1),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (branch_id) REFERENCES branches(branch_id)
);
```

---

data.sql

```sql
INSERT INTO branches VALUES
(1, 'Mumbai Main', 'Mumbai'),
(2, 'Delhi Central', 'Delhi');

INSERT INTO customers VALUES
(101, 'Sanjana', 25, 'Female', 1),
(102, 'Rahul', 30, 'Male', 2);

INSERT INTO transactions VALUES
(1001, 101, 1, 5000, '2025-01-10', 0),
(1002, 101, 1, 20000, '2025-02-15', 1),
(1003, 102, 2, 15000, '2025-01-20', 0),
(1004, 102, 2, 80000, '2025-03-05', 1);
```

---

queries.sql

```sql
-- 1. Month-wise Fraud Trend
SELECT
    DATE_FORMAT(transaction_date, '%Y-%m') AS month,
    COUNT(*) AS total_transactions,
    SUM(is_fraud) AS fraud_transactions,
    ROUND((SUM(is_fraud)/COUNT(*))*100,2) AS fraud_percentage
FROM transactions
GROUP BY month
ORDER BY month;

-- 2. High-Risk Customers
SELECT
    c.customer_id,
    c.name,
    COUNT(t.transaction_id) AS total_transactions,
    SUM(t.is_fraud) AS fraud_transactions
FROM customers c
JOIN transactions t ON c.customer_id = t.customer_id
GROUP BY c.customer_id, c.name
HAVING SUM(t.is_fraud) > 1
ORDER BY fraud_transactions DESC;

-- 3. Branch-wise Fraud Analysis
SELECT
    b.branch_name,
    COUNT(t.transaction_id) AS total_transactions,
    SUM(t.is_fraud) AS fraud_transactions,
    ROUND((SUM(t.is_fraud)/COUNT(*))*100,2) AS fraud_percentage
FROM branches b
JOIN transactions t ON b.branch_id = t.branch_id
GROUP BY b.branch_name
ORDER BY fraud_percentage DESC;

-- 4. Fraud Amount Patterns
SELECT
    is_fraud,
    AVG(amount) AS avg_transaction_amount,
    MAX(amount) AS max_transaction_amount,
    MIN(amount) AS min_transaction_amount
FROM transactions
GROUP BY is_fraud;

-- 5. View for all fraud transactions
CREATE VIEW fraud_transactions AS
```

```sql
SELECT t.transaction_id, t.customer_id, c.name, t.branch_id, b.branch_name,
t.amount, t.transaction_date
FROM transactions t
JOIN customers c ON t.customer_id = c.customer_id
JOIN branches b ON t.branch_id = b.branch_id
WHERE t.is_fraud = 1;

-- 6. View for branch fraud summary
CREATE VIEW branch_fraud_summary AS
SELECT
    b.branch_name,
    COUNT(t.transaction_id) AS total_transactions,
    SUM(t.is_fraud) AS fraud_transactions,
    ROUND((SUM(t.is_fraud)/COUNT(*))*100,2) AS fraud_percentage
FROM branches b
JOIN transactions t ON b.branch_id = t.branch_id
GROUP BY b.branch_name;
```