**ASSIGNMENT 1**

**CSA0593- DBMS**

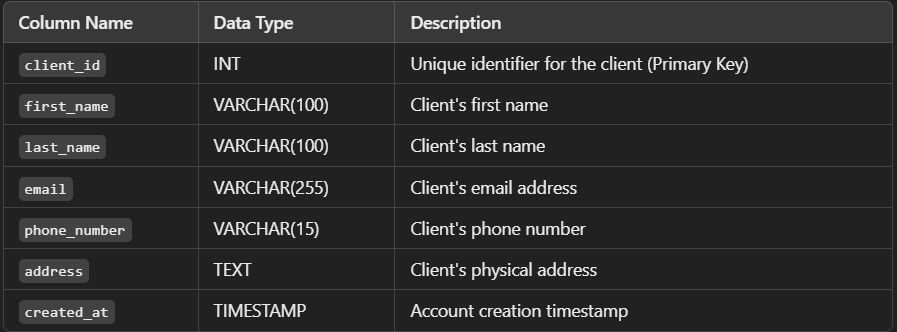
**S. Deva priya**

**192311419**

**QUESTION: Financial Risk Management System for Stock Portfolios  
Develop a database to manage stock portfolios, trading histories, and risk analytics for financial advisors.  
Requirements:  
Create tables for clients, portfolios, stocks, transactions, and risk assessments.  
Write procedures to calculate portfolio risk scores based on historical performance and stock volatility.  
Write SQL queries to identify high-risk portfolios and suggest risk mitigation strategies.  
Implement partitioning for performance optimization with high-volume transaction data.**

**1.Clients Table**

Stores details about each client.



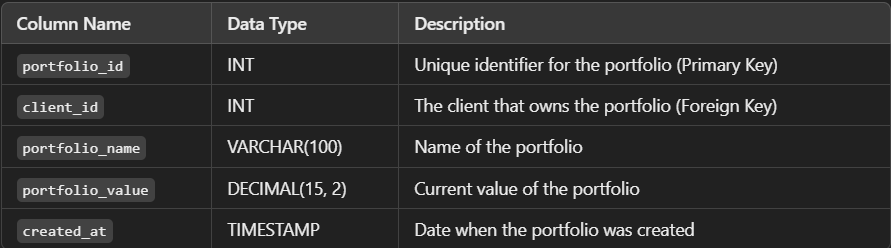
2.Stocks Table

Stores details about individual stocks.



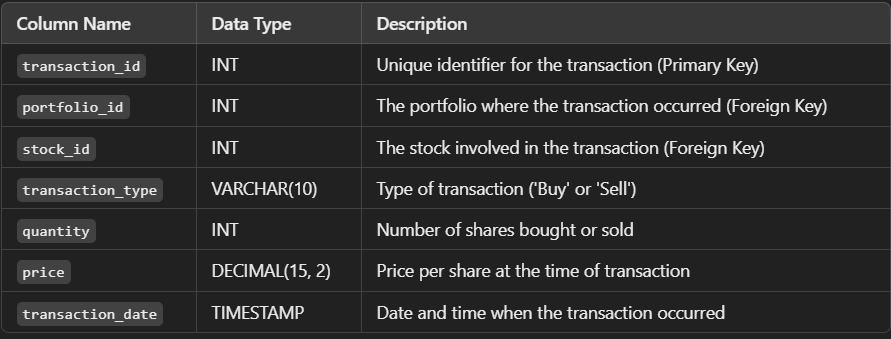
3.Portfolios Table

Stores information about portfolios associated with each client



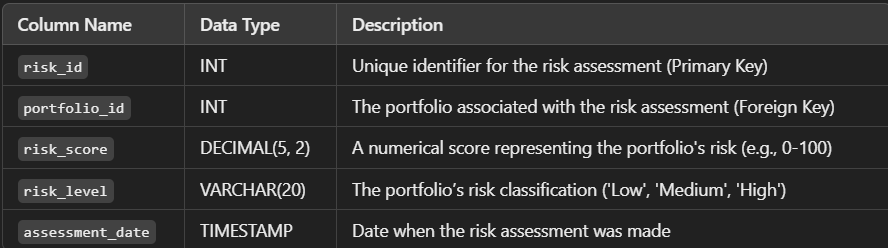
4.Transaction Table

Tracks all stock transactions (buy/sell) within a portfolio.



5.Risk assessements Table

Stores calculated risk scores for each portfolio, based on historical stock performance and volatility.



SQL CODE:

1.CREATE DATABASE TABLE:

-- Creating the Clients table

CREATE TABLE Clients (

client\_id INT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100),

phone VARCHAR(20)

);

-- Creating the Portfolios table

CREATE TABLE Portfolios (

portfolio\_id INT PRIMARY KEY,

client\_id INT,

name VARCHAR(100),

created\_at DATE,

FOREIGN KEY (client\_id) REFERENCES Clients(client\_id)

);

-- Creating the Stocks table

CREATE TABLE Stocks (

stock\_id INT PRIMARY KEY,

ticker VARCHAR(10) UNIQUE,

sector VARCHAR(50),

volatility FLOAT -- Volatility as a measure of risk (e.g., standard deviation of returns)

);

-- Creating the Transactions table

CREATE TABLE Transactions (

transaction\_id INT PRIMARY KEY,

portfolio\_id INT,

stock\_id INT,

transaction\_date DATE,

transaction\_type VARCHAR(4), -- 'BUY' or 'SELL'

quantity INT,

price FLOAT,

FOREIGN KEY (portfolio\_id) REFERENCES Portfolios(portfolio\_id),

FOREIGN KEY (stock\_id) REFERENCES Stocks(stock\_id)

);

-- Creating the Risk\_Assessments table

CREATE TABLE Risk\_Assessments (

risk\_assessment\_id INT PRIMARY KEY,

portfolio\_id INT,

risk\_score FLOAT,

assessed\_at DATE,

FOREIGN KEY (portfolio\_id) REFERENCES Portfolios(portfolio\_id)

);

2.CREATE SCORED PROCEDURE FOR RISK SCORE FACTOR:

CREATE PROCEDURE CalculateRiskScore()

BEGIN

DECLARE portfolioVolatility FLOAT;

-- Clear previous risk scores

DELETE FROM Risk\_Assessments;

-- Calculate and insert new risk scores for each portfolio

INSERT INTO Risk\_Assessments (portfolio\_id, risk\_score, assessed\_at)

SELECT

p.portfolio\_id,

SUM(s.volatility \* t.quantity) / SUM(t.quantity) AS risk\_score, -- Weighted average volatility

CURDATE()

FROM

Portfolios p

JOIN

Transactions t ON p.portfolio\_id = t.portfolio\_id

JOIN

Stocks s ON t.stock\_id = s.stock\_id

GROUP BY

p.portfolio\_id;

END //

DELIMITER ;

3.IDENTIFY HIGH RISK PORTFOLIOS:

-- Define a high-risk threshold, e.g., 0.7 (adjust based on risk tolerance)

SET @high\_risk\_threshold = 0.7;

-- Query to find high-risk portfolios

SELECT

ra.portfolio\_id,

c.name AS client\_name,

ra.risk\_score,

p.name AS portfolio\_name

FROM

Risk\_Assessments ra

JOIN

Portfolios p ON ra.portfolio\_id = p.portfolio\_id

JOIN

Clients c ON p.client\_id = c.client\_id

WHERE

ra.risk\_score > @high\_risk\_threshold;

4.SUGGEST RISK MITIGATION STRATEGIES:

SELECT

p.portfolio\_id,

c.name AS client\_name,

s.ticker AS high\_volatility\_stock,

s.volatility,

t.quantity

FROM

Portfolios p

JOIN

Clients c ON p.client\_id = c.client\_id

JOIN

Transactions t ON p.portfolio\_id = t.portfolio\_id

JOIN

Stocks s ON t.stock\_id = s.stock\_id

JOIN

Risk\_Assessments ra ON p.portfolio\_id = ra.portfolio\_id

WHERE

ra.risk\_score > @high\_risk\_threshold

AND s.volatility > (SELECT AVG(volatility) FROM Stocks);

5.IMPLEMENT PARTITIONING FOR PERFORMANCE OPTIMIZATION:

-- Partition Transactions table by year for efficient querying

ALTER TABLE Transactions

PARTITION BY RANGE (YEAR(transaction\_date)) (

PARTITION p\_before\_2020 VALUES LESS THAN (2020),

PARTITION p\_2020 VALUES LESS THAN (2021),

PARTITION p\_2021 VALUES LESS THAN (2022),

PARTITION p\_2022 VALUES LESS THAN (2023),

PARTITION p\_future VALUES LESS THAN MAXVALUE

);