```
CREATE DATABASE project_bank;
USE project_bank;

CREATE TABLE Raw_Data ( -- Create the inital raw data table
    cust_id VARCHAR(255) PRIMARY KEY,
    age INT,
    occupation VARCHAR(100),
    risk_tolerance VARCHAR(10),
    investment_goals VARCHAR(255),
    education VARCHAR(100),
    marital_status VARCHAR(20),
    dependents INT,
    region VARCHAR(100),
    financial_history VARCHAR(50),
    sector VARCHAR(100),
    income_level DECIMAL(65,30),
    account_balance DECIMAL(65,30),
    account_deposits DECIMAL(65,30),
    account_withdrawals DECIMAL(65,30),
    account_transfers DECIMAL(65,30),
    international_transfers DECIMAL(65,30),
    account_investments DECIMAL(65,30),
    account_type VARCHAR(50),
    loan_amount DECIMAL(65,30),
    loan_purpose VARCHAR(255),
    employment_status VARCHAR(50),
    loan_term INT,
    interest_rate DECIMAL(65,15),
    loan_status VARCHAR(50)
);

CREATE TABLE Customer_Demographics ( -- Creating and Insterting into
the Customer Demographics table
    cust_id VARCHAR(255) PRIMARY KEY,
    age INT,
    occupation VARCHAR(100),
    risk_tolerance VARCHAR(10),
    investment_goals VARCHAR(255),
    education VARCHAR(100),
    marital_status VARCHAR(20),
    dependents INT,
    region VARCHAR(100),
    financial_history VARCHAR(50),
    sector VARCHAR(100),
    income_level DECIMAL(65,30)
);
INSERT INTO Customer_Demographics (cust_id, age,
occupation,risk_tolerance,investment_goals,education,marital_status,
 dependents,region,financial_history,sector, income_level)
```

```
SELECT cust_id, age,
occupation,risk_tolerance,investment_goals,education,marital_status,
 dependents,region,financial_history,sector, income_level
FROM Raw_Data;

CREATE TABLE Account_Activity ( -- Account Activity Table creation
    cust_id VARCHAR(255),
    account_balance DECIMAL(65,30),
    account_deposits DECIMAL(65,30),
    account_withdrawals DECIMAL(65,30),
    account_transfers DECIMAL(65,30),
    international_transfers DECIMAL(65,30),
    account_investments DECIMAL(65,30),
    account_type VARCHAR(50),
    FOREIGN KEY (cust_id) REFERENCES Customer_Demographics(cust_id)
);
INSERT INTO Account_Activity
(cust_id,account_balance,account_deposits,account_withdrawals,accoun
t_transfers,international_transfers,
account_investments,account_type)
SELECT
cust_id,account_balance,account_deposits,account_withdrawals,account
_transfers,international_transfers,
account_investments,account_type
FROM Raw_Data;

CREATE TABLE Loan_Details ( -- Loan details table
    cust_id VARCHAR(255),
    loan_amount DECIMAL(65,30),
    loan_purpose VARCHAR(255),
    employment_status VARCHAR(50),
    loan_term INT,
    interest_rate DECIMAL(65,15),
    loan_status VARCHAR(50),
    FOREIGN KEY (cust_id) REFERENCES Customer_Demographics(cust_id)
);
INSERT INTO Loan_Details
(cust_id,loan_amount,loan_purpose,employment_status,loan_term,intere
st_rate,loan_status)
SELECT
cust_id,loan_amount,loan_purpose,employment_status,loan_term,interes
t_rate,loan_status
FROM Raw_Data;




SELECT -- Customer Segmentation Analysis based on Income Groups
    CASE
        WHEN cd.income_level < 30000 THEN 'Low Income'
        WHEN cd.income_level BETWEEN 30000 AND 70000 THEN 'Middle
Income'
        ELSE 'High Income'
```

```
    END AS income_group,-- Categorize customers into income groups
    COUNT(cd.cust_id) AS customer_count, -- Count the number of
customers in each income group
    ROUND(COUNT(cd.cust_id) * 100.0 / (SELECT COUNT(*) FROM
Customer_Demographics), 2) AS customer_percentage, -- Calculate the
percentage of customers in each income group
    ROUND(AVG(aa.account_balance), 2) AS avg_balance, -- Calculate
the average account balance for each income group
    ROUND(SUM(aa.account_balance), 2) AS total_balance,-- Calculate
the total account balance for each income group
    ROUND(SUM(aa.account_balance) * 100.0 / (SELECT
SUM(account_balance) FROM Account_Activity), 2) AS
balance_contribution_percentage, -- Calculate the percentage
contribution of each income group's total balance to the overall
account balance
    ROUND(AVG(aa.account_investments), 2) AS avg_investments, --
Calculate the average investment amount for each income group
    ROUND(SUM(aa.account_investments), 2) AS total_investments, --
Calculate the total investment amount for each income group
    ROUND(SUM(aa.account_investments) * 100.0 / (SELECT
SUM(account_investments) FROM Account_Activity), 2) AS
investment_contribution_percentage -- Calculate the percentage
contribution of each income group's total investments to the overall
investments
    -- Join the Customer_Demographics and Account_Activity tables to
link customer details with their account data
    FROM Customer_Demographics AS cd
    JOIN Account_Activity AS aa ON cd.cust_id = aa.cust_id
    GROUP BY income_group;

SELECT
    loan_status,
    COUNT(*) AS total_loans,
    COUNT(CASE WHEN employment_status = 'Unemployed' THEN 1 END) AS
unemployed_count,
    ROUND(AVG(loan_amount), 2) AS avg_loan_amount,
    ROUND(AVG(interest_rate), 3) AS avg_interest_rate
FROM Loan_Details
JOIN Customer_Demographics ON Loan_Details.cust_id =
Customer_Demographics.cust_id
GROUP BY loan_status
ORDER BY total_loans DESC;

SELECT -- Rejection/Approval Rates per region (4)
    region,
    COUNT(CASE WHEN loan_status = 'rejected' THEN 1 END) AS
rejected_count,
    ROUND((COUNT(CASE WHEN loan_status = 'rejected' THEN 1 END) *
100.0 / COUNT(Customer_Demographics.cust_id)), 2) AS rejected_rate,
    COUNT(CASE WHEN loan_status = 'approved' THEN 1 END) AS
approved_count,
    ROUND((COUNT(CASE WHEN loan_status = 'approved' THEN 1 END) *
100.0 / COUNT(Customer_Demographics.cust_id)), 2) AS approved_rate,
    COUNT(Customer_Demographics.cust_id) AS total_customers
```

```
FROM Customer_Demographics
JOIN Loan_Details ON Customer_Demographics.cust_id =
Loan_Details.cust_id
GROUP BY region
ORDER BY rejected_rate DESC;

SELECT -- Account Activity Analysis for High Income earners
    cust_id,
    ROUND(account_balance, 2) AS account_balance,
    ROUND(account_deposits, 2) AS account_deposits,
    ROUND(account_withdrawals, 2) AS account_withdrawals,
    ROUND(account_deposits - account_withdrawals, 2) AS net_flow
FROM Account_Activity
WHERE account_balance >= 70000
ORDER BY net_flow DESC;

SELECT -- Account Activity for all customers and Risk
    Account_Activity.cust_id,
    ROUND(account_balance, 2) AS account_balance,
    ROUND(account_deposits, 2) AS account_deposits,
    ROUND(account_withdrawals, 2) AS account_withdrawals,
    ROUND(account_deposits - account_withdrawals, 2) AS net_flow,
    Customer_Demographics.risk_tolerance
FROM Account_Activity
JOIN Customer_Demographics ON Account_Activity.cust_id =
Customer_Demographics.cust_id
ORDER BY account_balance;


SELECT -- Average Investment per risk category
    Customer_Demographics.risk_tolerance,
    ROUND(AVG(Account_Activity.account_investments),2) AS
avg_investments,
    COUNT(Customer_Demographics.cust_id) AS total_customers
FROM Customer_Demographics
JOIN Account_Activity ON Customer_Demographics.cust_id =
Account_Activity.cust_id
GROUP BY Customer_Demographics.risk_tolerance
ORDER BY avg_investments DESC;
```