**SQL Analysis**

--create the database

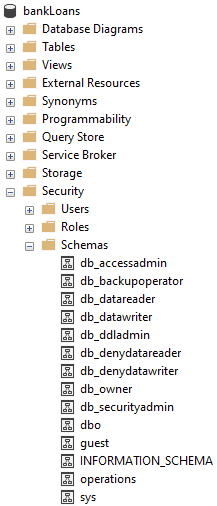
CREATE DATABASE bankLoans;

--use the database

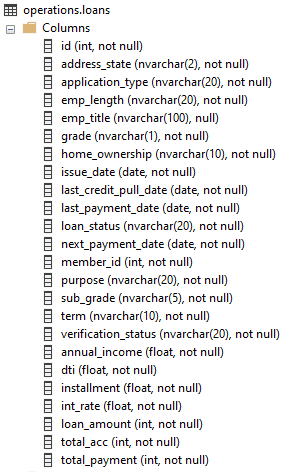
USE bankLoans;

--create an schema for the tables and more

create schema operations;

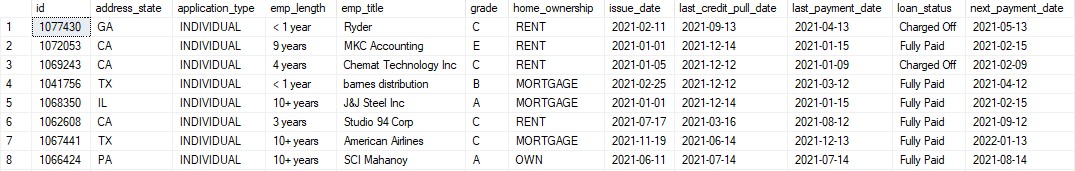


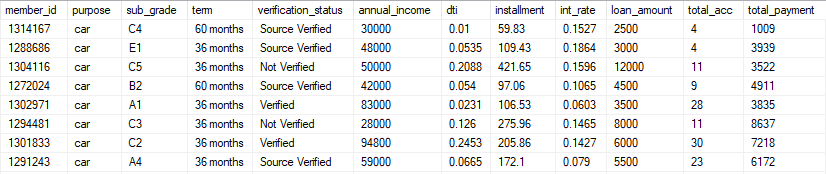
--we import the csv file into a new table in the database



--show the table

SELECT \* FROM operations.loans;





--measures

SELECT

COUNT(id) AS CountOfId,

SUM(loan\_amount) AS SumOfLoanAmount,

SUM(total\_payment) AS SumOfTotalPayment,

CAST((AVG(int\_rate) \* 100) AS DECIMAL(10,2)) AS AverageIntRate,

CAST((AVG(dti) \* 100) AS DECIMAL(10,2)) AS AverageDti

FROM

operations.loans



--month to date (MTD) measures

SELECT

COUNT(id) AS CountOfId,

SUM(loan\_amount) AS SumOfLoanAmount,

SUM(total\_payment) AS SumOfTotalPayment,

CAST((AVG(int\_rate) \* 100) AS DECIMAL(10,2)) AS AverageIntRate,

CAST((AVG(dti) \* 100) AS DECIMAL(10,2)) AS AverageDti

FROM

operations.loans

WHERE

MONTH(issue\_date) = 12



--previous month to date (PMTD) measures

SELECT

COUNT(id) AS CountOfId,

SUM(loan\_amount) AS SumOfLoanAmount,

SUM(total\_payment) AS SumOfTotalPayment,

CAST((AVG(int\_rate) \* 100) AS DECIMAL(10,2)) AS AverageIntRate,

CAST((AVG(dti) \* 100) AS DECIMAL(10,2)) AS AverageDti

FROM

operations.loans

WHERE

MONTH(issue\_date) = 11



--create a table to store measures

CREATE TABLE operations.#measures (

MonthToDateMeasures FLOAT NOT NULL,

PreviousMonthToDateMeasures FLOAT NOT NULL

)

SELECT \* FROM operations.#measures



--CountOfId MTD and CountOfId PMTD

DECLARE @CountOfIdMTD FLOAT = ( SELECT

COUNT(id) AS CountOfId

FROM

operations.loans

WHERE

MONTH(issue\_date) = 12 )

DECLARE @CountOfIdPMTD FLOAT = ( SELECT

COUNT(id) AS CountOfId

FROM

operations.loans

WHERE

MONTH(issue\_date) = 11 )

--SumOfLoanAmount MTD and SumOfLoanAmount PMTD

DECLARE @SumOfLoanAmountMTD FLOAT = ( SELECT

SUM(loan\_amount) AS SumOfLoanAmount

FROM

operations.loans

WHERE

MONTH(issue\_date) = 12 )

DECLARE @SumOfLoanAmountPMTD FLOAT = ( SELECT

SUM(loan\_amount) AS SumOfLoanAmount

FROM

operations.loans

WHERE

MONTH(issue\_date) = 11 )

--SumOfTotalPayment MTD and SumOfTotalPayment PMTD

DECLARE @SumOfTotalPaymentMTD FLOAT = ( SELECT

SUM(total\_payment) AS SumOfTotalPayment

FROM

operations.loans

WHERE

MONTH(issue\_date) = 12 )

DECLARE @SumOfTotalPaymentPMTD FLOAT = ( SELECT

SUM(total\_payment) AS SumOfTotalPayment

FROM

operations.loans

WHERE

MONTH(issue\_date) = 11 )

--AverageIntRate MTD and AverageIntRate PMTD

DECLARE @AverageIntRateMTD FLOAT = ( SELECT

CAST((AVG(int\_rate) \* 100) AS DECIMAL(10,2)) AS AverageIntRate

FROM

operations.loans

WHERE

MONTH(issue\_date) = 12 )

DECLARE @AverageIntRatePMTD FLOAT = ( SELECT

CAST((AVG(int\_rate) \* 100) AS DECIMAL(10,2)) AS AverageIntRate

FROM

operations.loans

WHERE

MONTH(issue\_date) = 11 )

--AverageDti MTD and AverageDti PMTD

DECLARE @AverageDtiMTD FLOAT = ( SELECT

CAST((AVG(dti) \* 100) AS DECIMAL(10,2)) AS AverageDti

FROM

operations.loans

WHERE

MONTH(issue\_date) = 12 )

DECLARE @AverageDtiPMTD FLOAT = ( SELECT

CAST((AVG(dti) \* 100) AS DECIMAL(10,2)) AS AverageDti

FROM

operations.loans

WHERE

MONTH(issue\_date) = 11 )

INSERT INTO operations.#measures

VALUES

(@CountOfIdMTD,@CountOfIdPMTD),

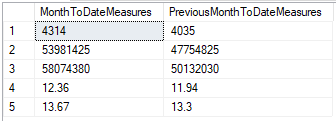
(@SumOfLoanAmountMTD,@SumOfLoanAmountPMTD),

(@SumOfTotalPaymentMTD,@SumOfTotalPaymentPMTD),

(@AverageIntRateMTD,@AverageIntRatePMTD),

(@AverageDtiMTD,@AverageDtiPMTD)

SELECT \* FROM operations.#measures



--getting MonthOverMonth measures

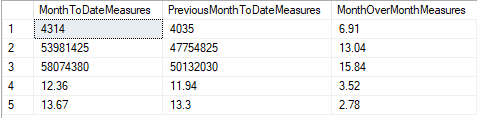
SELECT

\*,

CAST(((MonthToDateMeasures - PreviousMonthToDateMeasures) / PreviousMonthToDateMeasures \* 100) AS DECIMAL(10,2)) AS MonthOverMonthMeasures

FROM

operations.#measures



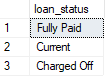
--there is three loan status

--Fully Paid is Good Loan

--Current is Good Loan

--Charged Off is Bad Loan

SELECT DISTINCT [loan\_status] FROM operations.loans



--create a new column depending in the loan status

BEGIN TRAN

ALTER TABLE

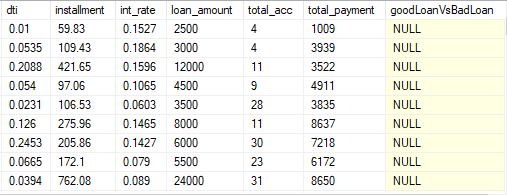
operations.loans

ADD

goodLoanVsBadLoan NVARCHAR(10) NULL;

SELECT \* FROM operations.loans;

COMMIT TRAN



--Filling the goodLoanVsBadLoan column

--Fully Paid is Good Loan

--Current is Good Loan

--Charged Off is Bad Loan

BEGIN TRAN

UPDATE

operations.loans

SET

goodLoanVsBadLoan =

CASE

WHEN loan\_status = 'Fully Paid' THEN 'Good Loan'

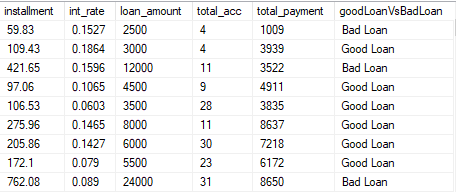
WHEN loan\_status = 'Current' THEN 'Good Loan'

WHEN loan\_status = 'Charged Off' THEN 'Bad Loan'

END

SELECT \* FROM operations.loans

COMMIT TRAN



--bad loan and good loan measures

DECLARE @totalCount FLOAT = ( SELECT

COUNT(\*)

FROM

operations.loans)

SELECT

goodLoanVsBadLoan,

CAST((COUNT(\*) / @totalCount \* 100) AS DECIMAL(10,2)) AS PercentageOfTotal,

COUNT(\*) AS CountOfLoans,

SUM(loan\_amount) AS SumOfLoanAmount,

SUM(total\_payment) AS SumOfTotalPayment

FROM

operations.loans

GROUP BY

goodLoanVsBadLoan



--loan status measures

DECLARE @totalCount FLOAT = ( SELECT

COUNT(\*)

FROM

operations.loans)

SELECT

loan\_status,

CAST((COUNT(\*) / @totalCount \* 100) AS DECIMAL(10,2)) AS PercentageOfTotal,

COUNT(\*) AS CountOfLoans,

SUM(loan\_amount) AS SumOfLoanAmount,

SUM(total\_payment) AS SumOfTotalPayment,

CAST((AVG(int\_rate) \* 100) AS DECIMAL(10,2)) AS AverageIntRate,

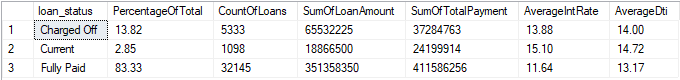
CAST((AVG(dti)\*100) AS DECIMAL(10,2)) AS AverageDti

FROM

operations.loans

GROUP BY

loan\_status



--Monthly Trends By Loan Issue Date

WITH monthAdded AS (

SELECT

\*,

FORMAT(

issue\_date,

'MMMM'

) AS loanIssueDateMonth

FROM

operations.loans

)

SELECT

loanIssueDateMonth,

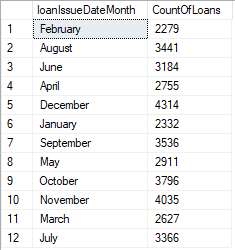
COUNT(\*) AS CountOfLoans

FROM

monthAdded

GROUP BY

loanIssueDateMonth



--Regional Analysis By State

SELECT

address\_state,

COUNT(\*) AS CountOfLoans

FROM

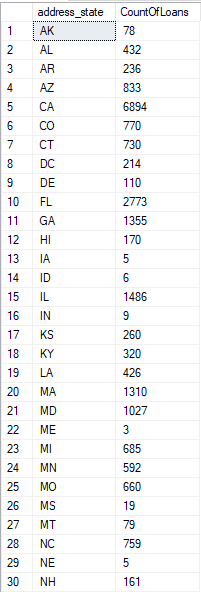
operations.loans

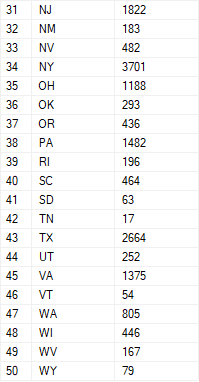
GROUP BY

address\_state

ORDER BY

address\_state ASC





--loan termn analysis

DECLARE @totalCount FLOAT = ( SELECT

COUNT(\*)

FROM

operations.loans)

SELECT

term,

COUNT(\*) AS CountOfLoans,

CAST((COUNT(\*) / @totalCount \* 100) AS DECIMAL(10,2)) AS PercentageOfTotal

FROM

operations.loans

GROUP BY

term



--employee lenght analysis

SELECT

emp\_length,

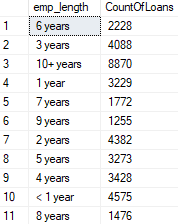
COUNT(\*) AS CountOfLoans

FROM

operations.loans

GROUP BY

emp\_length



--purpose analysis

SELECT

purpose,

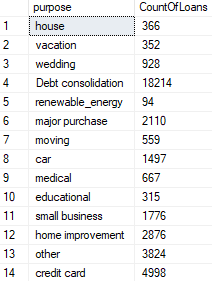
COUNT(\*) AS CountOfLoans

FROM

operations.loans

GROUP BY

purpose



--purpose analysis

SELECT

home\_ownership,

COUNT(\*) AS CountOfLoans

FROM

operations.loans

GROUP BY

home\_ownership

