-- Create a new database

CREATE DATABASE customer\_churn\_analysis;

-- Use the created database

USE customer\_churn\_analysis;

-- Create a table to match the dataset structure

CREATE TABLE customer\_data (

CustomerID VARCHAR(255),

Gender VARCHAR(10),

Age INT,

Married VARCHAR(5),

NumberOfDependents INT,

City VARCHAR(100),

ZipCode INT,

Latitude FLOAT,

Longitude FLOAT,

NumberOfReferrals INT,

TenureInMonths INT,

Offer VARCHAR(50),

PhoneService VARCHAR(10),

AvgMonthlyLongDistanceCharges FLOAT,

MultipleLines VARCHAR(20),

InternetService VARCHAR(20),

InternetType VARCHAR(20),

AvgMonthlyGBDownload FLOAT,

OnlineSecurity VARCHAR(20),

OnlineBackup VARCHAR(20),

DeviceProtectionPlan VARCHAR(20),

PremiumTechSupport VARCHAR(20),

StreamingTV VARCHAR(20),

StreamingMovies VARCHAR(20),

StreamingMusic VARCHAR(20),

UnlimitedData VARCHAR(10),

Contract VARCHAR(20),

PaperlessBilling VARCHAR(5),

PaymentMethod VARCHAR(50),

MonthlyCharge FLOAT,

TotalCharges FLOAT,

TotalRefunds FLOAT,

TotalExtraDataCharges INT,

TotalLongDistanceCharges FLOAT,

TotalRevenue FLOAT,

CustomerStatus VARCHAR(20),

ChurnCategory VARCHAR(50),

ChurnReason VARCHAR(255)

);

SELECT \* FROM customer\_data LIMIT 10;

# Replace null value as No service in internet type

UPDATE customer\_data

SET InternetType = 'No Service'

WHERE InternetType IS NULL;

# Repace the missing numeric values

-- Calculate the average value

SET @AvgCharges = (SELECT AVG(AvgMonthlyLongDistanceCharges)

FROM customer\_data

WHERE AvgMonthlyLongDistanceCharges IS NOT NULL);

-- Replace NULL with average

UPDATE customer\_data

SET AvgMonthlyLongDistanceCharges = @AvgCharges

WHERE AvgMonthlyLongDistanceCharges IS NULL;

# To standardize the categorical values

UPDATE customer\_data

SET Gender = 'Male'

WHERE Gender IN ('M', 'male', 'MALE');

UPDATE customer\_data

SET Married = 'Yes'

WHERE Married IN ('yes', 'YES', 'y');

# Outlier detection and correction

-- Check for outliers in monthly charges

SELECT CustomerId, MonthlyCharge

FROM customer\_data

WHERE MonthlyCharge < 0 OR MonthlyCharge > 1000;

# To nullify the outliers

-- Calculate the median monthly charge

-- Step 1: Create a temporary table with ordered row numbers

CREATE TEMPORARY TABLE temp\_median1 AS

SELECT

MonthlyCharge,

ROW\_NUMBER() OVER (ORDER BY MonthlyCharge) AS row\_num,

(SELECT COUNT(\*) FROM customer\_data) AS total\_rows

FROM customer\_data;

-- Step 2: Calculate the median

-- If total number of rows is odd

SET @median = (

SELECT MonthlyCharge

FROM temp\_median1

WHERE row\_num = (total\_rows DIV 2) + 1

);

-- If total number of rows is even, calculate the average of the two middle values

SET @even\_median = (

SELECT AVG(MonthlyCharge)

FROM temp\_median1

WHERE row\_num IN ((total\_rows DIV 2), (total\_rows DIV 2) + 1)

);

-- Use the even median if the total number of rows is even

SET @median = IF((SELECT total\_rows FROM temp\_median1 LIMIT 1) % 2 = 0, @even\_median, @median);

-- Step 3: Update outliers with the median value

UPDATE customer\_data

SET MonthlyCharge = @median

WHERE MonthlyCharge < 0 OR MonthlyCharge > 1000;

-- Step 4: Drop the temporary table

DROP TEMPORARY TABLE temp\_median;

drop temporary Table temp\_median1;

# to handle missing churn information

UPDATE customer\_data

SET ChurnCategory = 'Not Applicable',

ChurnReason = 'Not Applicable'

WHERE CustomerStatus != 'Churned';

# To dervie new column

ALTER TABLE customer\_data ADD COLUMN LifetimeRevenue FLOAT;

UPDATE customer\_data

SET LifetimeRevenue = TenureInMonths \* MonthlyCharge;

# Add Age Group column for segmentation

ALTER TABLE customer\_data ADD COLUMN AgeGroup VARCHAR(20);

UPDATE customer\_data

SET AgeGroup = CASE

WHEN age < 25 THEN 'Under 25'

WHEN age BETWEEN 25 AND 40 THEN '25-40'

WHEN age BETWEEN 41 AND 60 THEN '41-60'

ELSE 'Above 60'

END;

# To remove duplicates

DELETE FROM customer\_data

WHERE CustomerId IN (

SELECT CustomerId FROM (

SELECT CustomerId, COUNT(\*) AS cnt FROM customer\_data

GROUP BY CustomerId

HAVING cnt > 1

) AS duplicate\_ids

);

#validate data realtionships

SELECT \*

FROM customer\_data

WHERE CustomerStatus != 'Churned' AND ChurnReason IS NOT NULL;

UPDATE customer\_data

SET ChurnReason = NULL

WHERE CustomerStatus != 'Churned';

CREATE INDEX Idx\_Customer\_status ON customer\_data(CustomerStatus);

CREATE INDEX Idx\_Internet\_service ON customer\_data(InternetService);

# Overall Churn Rate (1.Identify the total number of customers and the churn rate)

SELECT

COUNT(\*) AS TotalCustomers,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) AS TotalChurnedCustomers,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRate

FROM customer\_data;

#2.Find the average age of churned customers

SELECT

ROUND(AVG(age), 2) AS AverageAgeOfChurned

FROM customer\_data

WHERE CustomerStatus = 'Churned';

#3.Discover the most common contract types among churned customers

SELECT

Contract,

COUNT(\*) AS ChurnedCount

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY Contract

ORDER BY ChurnedCount DESC

LIMIT 1;

#4. Analyze the distribution of monthly charges among churned customers

SELECT

MonthlyCharge,

COUNT(\*) AS Frequency

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY MonthlyCharge

ORDER BY MonthlyCharge;

#5.Identify the contract types that are most prone to churn

SELECT

Contract,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRatePercentage

FROM customer\_data

GROUP BY Contract

ORDER BY ChurnRatePercentage DESC;

#6. Identify customers with high total charges who have churned

SELECT

CustomerID,

TotalCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

ORDER BY TotalCharges DESC

LIMIT 10;

# 7 Calculate the total charges distribution for churned and non-churned customers

SELECT

CustomerStatus,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges,

MIN(TotalCharges) AS MinTotalCharges,

MAX(TotalCharges) AS MaxTotalCharges

FROM customer\_data

GROUP BY CustomerStatus;

#8. Calculate the average monthly charges for different contract types among churned customers

SELECT

Contract,

ROUND(AVG(MonthlyCharge), 2) AS AverageMonthlyCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY Contract;

#9. Identify customers who have both online security and online backup services and have not churned

SELECT

CustomerID

FROM customer\_data

WHERE OnlineSecurity = 'Yes' AND OnlineBackup = 'Yes' AND CustomerStatus != 'Churned';

#10. Determine the most common combinations of services among churned customers

SELECT

CONCAT\_WS(', ',

CASE WHEN OnlineSecurity = 'Yes' THEN 'Online Security' ELSE NULL END,

CASE WHEN OnlineBackup = 'Yes' THEN 'Online Backup' ELSE NULL END,

CASE WHEN StreamingTV = 'Yes' THEN 'Streaming TV' ELSE NULL END,

CASE WHEN StreamingMovies = 'Yes' THEN 'Streaming Movies' ELSE NULL END

) AS ServiceCombination,

COUNT(\*) AS ChurnedCount

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY ServiceCombination

ORDER BY ChurnedCount DESC;

#11. Identify the average total charges for customers grouped by gender and marital status

ALTER TABLE customer\_data

RENAME COLUMN Married TO MaritalStatus;

SELECT

Gender,

MaritalStatus,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges

FROM customer\_data

GROUP BY Gender, MaritalStatus;

#12. Calculate the average monthly charges for different age groups among churned customers

SELECT

CASE

WHEN AgeGroup < 25 THEN 'Under 25'

WHEN AgeGroup BETWEEN 25 AND 40 THEN '25-40'

WHEN AgeGroup BETWEEN 41 AND 60 THEN '41-60'

ELSE 'Above 60'

END AS AgeGroup,

ROUND(AVG(MonthlyCharge), 2) AS AverageMonthlyCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY AgeGroup

ORDER BY FIELD(AgeGroup, 'Under 25', '25-40', '41-60', 'Above 60');

# 13. Determine the average age and total charges for customers with multiple lines and online backup

SELECT

ROUND(AVG(age), 2) AS AverageAge,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges

FROM customer\_data

WHERE MultipleLines = 'Yes' AND OnlineBackup = 'Yes';

# 14. Identify the contract types with the highest churn rate among senior citizens (age 65 and over)

SELECT

Contract,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRatePercentage

FROM customer\_data

WHERE age >= 65

GROUP BY Contract

ORDER BY ChurnRatePercentage DESC;

# 15. Calculate the average monthly charges for customers who have multiple lines and streaming TV

SELECT

ROUND(AVG(MonthlyCharge), 2) AS AverageMonthlyCharges

FROM customer\_data

WHERE MultipleLines = 'Yes' AND StreamingTV = 'Yes';

#16.Identify the customers who have churned and used the most online services

SELECT

CustomerID,

(CASE WHEN OnlineSecurity = 'Yes' THEN 1 ELSE 0 END +

CASE WHEN OnlineBackup = 'Yes' THEN 1 ELSE 0 END +

CASE WHEN StreamingTV = 'Yes' THEN 1 ELSE 0 END +

CASE WHEN StreamingMovies = 'Yes' THEN 1 ELSE 0 END) AS TotalOnlineServices

FROM customer\_data

WHERE CustomerStatus = 'Churned'

ORDER BY TotalOnlineServices DESC;

#17. Calculate the average age and total charges for customers with different combinations of streaming services

SELECT

CONCAT\_WS(', ',

CASE WHEN StreamingTV = 'Yes' THEN 'Streaming TV' ELSE NULL END,

CASE WHEN StreamingMovies = 'Yes' THEN 'Streaming Movies' ELSE NULL END

) AS StreamingServices,

ROUND(AVG(age), 2) AS AverageAge,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges

FROM customer\_data

GROUP BY StreamingServices;

# 18. Identify the gender distribution among customers who have churned and are on yearly contracts

SELECT

Gender,

COUNT(\*) AS ChurnedCount

FROM customer\_data

WHERE CustomerStatus = 'Churned' AND Contract = 'Yearly'

GROUP BY Gender;

#19. Calculate the average monthly charges and total charges for customers who have churned, grouped by contract type and internet service type

SELECT

Contract,

InternetService,

ROUND(AVG(MonthlyCharge), 2) AS AverageMonthlyCharges,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY Contract, InternetService;

#20. Find the customers who have churned and are not using online services, and their average total charges

SELECT

CustomerID,

TotalCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

AND OnlineSecurity = 'No'

AND OnlineBackup = 'No'

AND StreamingTV = 'No'

AND StreamingMovies = 'No';

#21. Calculate the average monthly charges and total charges for customers who have churned, grouped by the number of dependents

SELECT

NumberOfDependents,

ROUND(AVG(MonthlyCharge), 2) AS AverageMonthlyCharges,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY NumberOfDependents;

#22. Identify the customers who have churned, and their contract duration in months (for monthly contracts)

SELECT

CustomerID,

TenureInMonths AS ContractDurationInMonths

FROM customer\_data

WHERE CustomerStatus = 'Churned' AND Contract = 'Monthly';

#23. Determine the average age and total charges for customers who have churned, grouped by internet service and phone service

SELECT

InternetService,

PhoneService,

ROUND(AVG(AgeGroup), 2) AS AverageAge,

ROUND(AVG(TotalCharges), 2) AS AverageTotalCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY InternetService, PhoneService;

#24. Create a view to find the customers with the highest monthly charges in each contract type

CREATE VIEW HighestMonthlyCharge AS

SELECT

Contract,

CustomerID,

MonthlyCharge

FROM customer\_data

WHERE MonthlyCharge = (

SELECT MAX(MonthlyCharge)

FROM customer\_data AS sub

WHERE sub.Contract = customer\_data.Contract

);

SELECT \* FROM HighestMonthlyCharge;

#25. Create a view to identify customers who have churned and the average monthly charges compared to the overall average

CREATE VIEW ChurnedVsOverallAvg AS

SELECT

CustomerID,

MonthlyCharge,

(SELECT AVG(MonthlyCharge) FROM customer\_data) AS OverallAvgMonthlyCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned';

SELECT \* FROM ChurnedVsOverallAvg;

#26. Create a view to find the customers who have churned and their cumulative total charges over time

CREATE VIEW CumulativeTotalCharges AS

SELECT

CustomerID,

TotalCharges,

SUM(TotalCharges) OVER (ORDER BY CustomerID) AS CumulativeTotalCharges

FROM customer\_data

WHERE CustomerStatus = 'Churned';

SELECT \* FROM CumulativeTotalCharges;

#27. Stored Procedure to Calculate Churn Rate

DELIMITER //

CREATE PROCEDURE CalculateChurnRate()

BEGIN

SELECT

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRatePercentage

FROM customer\_data;

END //

DELIMITER ;

CALL CalculateChurnRate();

#28. Stored Procedure to Identify High-Value Customers at Risk of Churning

DELIMITER //

CREATE PROCEDURE HighValueCustomersAtRisk()

BEGIN

SELECT

CustomerID,

TotalCharges,

MonthlyCharge

FROM customer\_data

WHERE CustomerStatus != 'Churned'

AND TotalCharges > (SELECT AVG(TotalCharges) FROM customer\_data)

AND MonthlyCharge > (SELECT AVG(MonthlyCharge) FROM customer\_data);

END //

DELIMITER ;

CALL HighValueCustomersAtRisk ();

#Churn Rate by Gender

SELECT

Gender,

COUNT(\*) AS TotalCustomers,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) AS ChurnedCustomers,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRate

FROM customer\_data

GROUP BY Gender;

# To check age distribution

SELECT

age,

COUNT(\*) AS TotalCustomers

FROM customer\_data

GROUP BY age

ORDER BY age;

#Churn by Age group

SELECT

AgeGroup,

TotalCustomers,

ChurnedCustomers,

ChurnRate

FROM (

SELECT

CASE

WHEN age < 25 THEN 'Under 25'

WHEN age BETWEEN 25 AND 40 THEN '25-40'

WHEN age BETWEEN 41 AND 60 THEN '41-60'

ELSE 'Above 60'

END AS AgeGroup,

COUNT(\*) AS TotalCustomers,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) AS ChurnedCustomers,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRate

FROM customer\_data

GROUP BY

CASE

WHEN age < 25 THEN 'Under 25'

WHEN age BETWEEN 25 AND 40 THEN '25-40'

WHEN age BETWEEN 41 AND 60 THEN '41-60'

ELSE 'Above 60'

END

) AS grouped\_data

ORDER BY FIELD(AgeGroup, 'Under 25', '25-40', '41-60', 'Above 60');

#revenue loss due to churn

SELECT

SUM(MonthlyCharge) AS TotalMonthlyRevenue,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN MonthlyCharge ELSE 0 END) AS LostMonthlyRevenue,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN MonthlyCharge ELSE 0 END) / SUM(MonthlyCharge) \* 100, 2) AS RevenueLossPercentage

FROM customer\_data;

#churn by internet service type

SELECT

InternetService,

COUNT(\*) AS TotalCustomers,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) AS ChurnedCustomers,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRate

FROM customer\_data

GROUP BY InternetService;

#Churn by Contract Type

SELECT

Contract,

COUNT(\*) AS TotalCustomers,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) AS ChurnedCustomers,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRate

FROM customer\_data

GROUP BY Contract;

#High value customers

SELECT

CustomerID,

TotalRevenue,

CustomerStatus

FROM customer\_data

WHERE TotalRevenue > (

SELECT AVG(TotalRevenue) FROM customer\_data

)

ORDER BY TotalRevenue DESC;

# Top Churn Reasons

SELECT

ChurnReason,

COUNT(\*) AS Occurrences,

ROUND(COUNT(\*) / (SELECT COUNT(\*) FROM customer\_data WHERE CustomerStatus = 'Churned') \* 100, 2) AS Percentage

FROM customer\_data

WHERE CustomerStatus = 'Churned'

GROUP BY ChurnReason

ORDER BY Occurrences DESC;

# Churn BY Tenure

SELECT

CASE

WHEN TenureInMonths < 12 THEN 'Less than 1 year'

WHEN TenureInMonths BETWEEN 12 AND 24 THEN '1-2 years'

WHEN TenureInMonths BETWEEN 25 AND 36 THEN '2-3 years'

ELSE 'More than 3 years'

END AS TenureGroup,

COUNT(\*) AS TotalCustomers,

SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) AS ChurnedCustomers,

ROUND(SUM(CASE WHEN CustomerStatus = 'Churned' THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS ChurnRate

FROM customer\_data

GROUP BY TenureGroup;

# To identify at risk customers

ALTER TABLE customer\_data ADD COLUMN ChurnProbability FLOAT;

UPDATE customer\_data

SET ChurnProbability =

CASE

WHEN TenureInMonths < 12 AND MonthlyCharge > 70 THEN 0.8

WHEN TenureInMonths < 12 AND MonthlyCharge <= 70 THEN 0.6

WHEN TenureInMonths BETWEEN 12 AND 36 THEN 0.5

ELSE 0.3

END;

SELECT \*

FROM customer\_data

WHERE ChurnProbability > 0.7;

#segemnt customer by ChurnProbability

SELECT ChurnProbability, COUNT(\*) AS customer\_count

FROM customer\_data

GROUP BY ChurnProbability

ORDER BY ChurnProbability DESC;

#segment customers by Monthy Charge

SELECT

CASE

WHEN MonthlyCharge < 50 THEN 'Low Spend'

WHEN MonthlyCharge BETWEEN 50 AND 100 THEN 'Medium Spend'

ELSE 'High Spend'

END AS SpendCategory,

COUNT(\*) AS CustomerCount

FROM customer\_data

GROUP BY SpendCategory;