Churn Prediction and Customer Behaviour Analysis for E-commerce Platform

Churn analysis prediction project on SQL .

INTRODUCTION

In Today's competitive e-commerce landscape, customer retention is crucial for business success.
This project aims to analyse user behaviour, marketing engagement, and customer support to understand and predict churn- when customers stop using our platform.
Created a e-commerce dataset consisting of six interconnected tables (users, orders, activities, marketing engagement, support tickets, churn flags), we have derived key insights and recommendations to minimize churn and improve retention.

Overview Of the Schema

This table has six interconnected table that provide insights into the e-commerce churn prediction.

Tables used:

➡ **user_info:** customer personal and engagement details

➡ **orders:** order history including purchase amounts and status.

➡ **user_activity:** records of user logins and actions**.**

➡ **marketing_engagement:** tracks campaign interactions.

➡ **customer_support:** Support ticket resolutions and delays.

➡ **churn_flag:** Identifies users who have churned**.**

Key metrics:

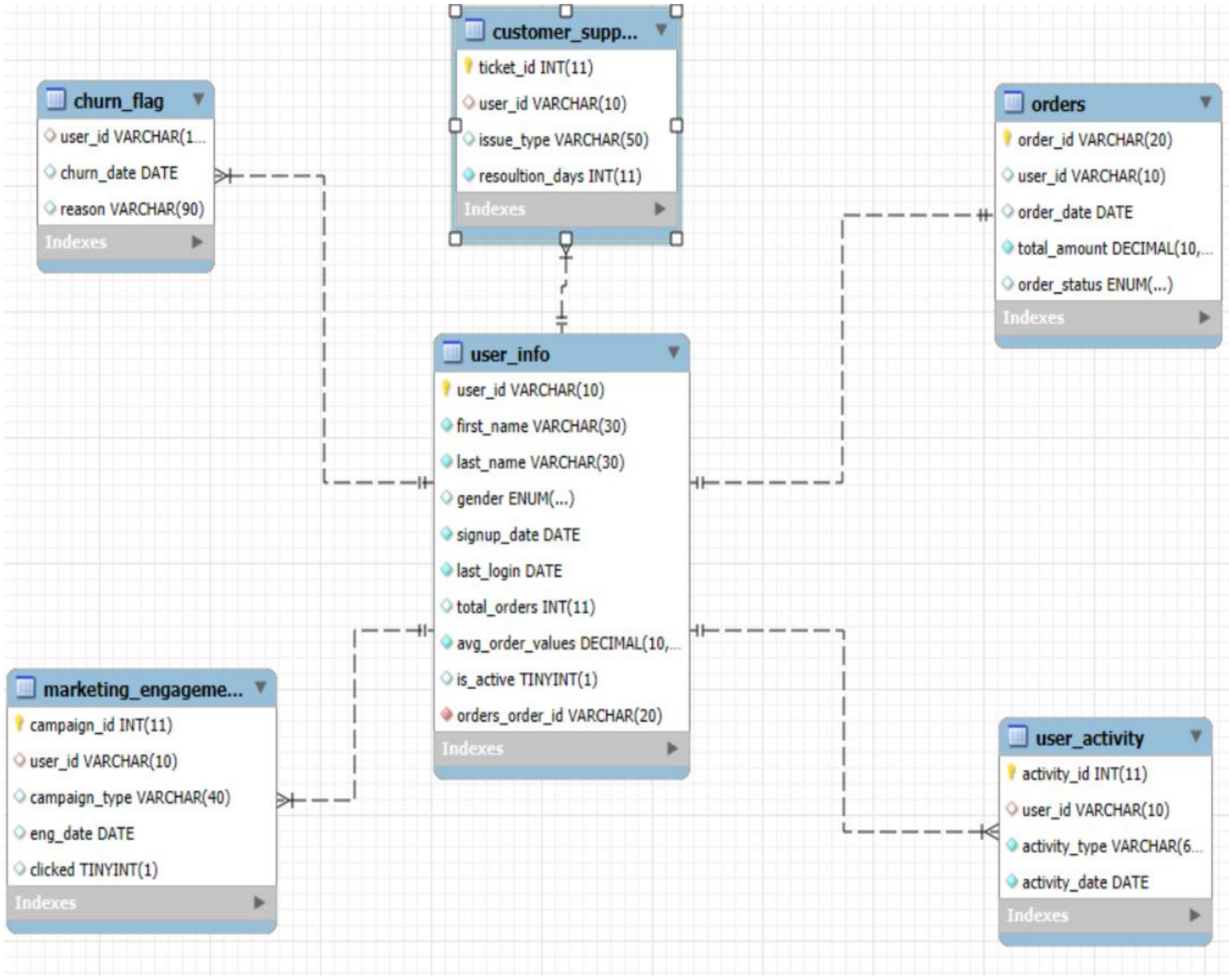
- Total users: 20
- Churned users: 10
- Inactive users: 8

```
-- CREATE DATABASE --> E-COMMERCE CUSTOMER RETENTION CHURN
PREDICTION

CREATE DATABASE IF NOT EXISTS CHURN_PREDICTION.


-- USE DATABASE--

USE churn_prediction;


-- CREATE TABLE --> USER_INFO

-- core customer data--


CREATE TABLE user_info (

user_id int primary key,

first_name varchar (30),

last_name varchar (30),

gender ENUM('male','female','others'),

signup_date DATE NOT NULL,

last_login DATE NOT NULL,

total_orders INT DEFAULT 0,

avg_order_values DECIMAL (10,2) NOT NULL,

is_active boolean
```

```
);

SELECT * FROM user_info;


-- create table orders

CREATE TABLE orders (

order_id varchar (20) PRIMARY KEY,

user_id VARCHAR (10),

order_date DATE,

total_amount DECIMAL (10,2) NOT NULL,

order_status ENUM ('delivered', 'canceled', 'returned'),

FOREIGN KEY (user_id) REFERENCES user_info(user_id)

);

SELECT * FROM orders;


-- CREATE TABLE USER_ACTIVITY

CREATE TABLE user_activity (

activity_id INT PRIMARY KEY ,

user_id varchar(20) REFERENCES user_info(user_id),

activity_type VARCHAR(60) NOT NULL,

activity_date DATE NOT NULL,
FOREIGN KEY (user_id) REFERENCES user_info(user_id)

);

SELECT * FROM user_activity;



-- CREATE TABLE CUSTOMER_SUPPORT

CREATE TABLE customer_support(

ticket_id Int PRIMARY KEY,

user_id varchar (20) REFERENCES user_info(user_id),

issue_type VARCHAR (50),

resoultion_days INT,

FOREIGN KEY (user_id) REFERENCES user_info(user_id)

);

SELECT * FROM customer_support;

-- CREATE TABLE MARKETING_ENGAGEMENT

CREATE TABLE marketing_engagement(

campaign_id INT PRIMARY KEY,

user_id varchar (20) ,

campaign_type VARCHAR (40),

eng_date DATE,

clicked boolean,
FOREIGN KEY (user_id) REFERENCES user_info(user_id)

);

SELECT * FROM marketing_engagement;


-- CREATE TABLE CHURN_FLAG

CREATE TABLE churn_flag(

user_id varchar(20) references user_info (user_id),

churn_date DATE,

reason VARCHAR (90),

FOREIGN KEY (user_id) REFERENCES user_info(user_id)

);

SELECT * FROM churn_flag;
```

- **Entity Relationship Diagram [ERD]**

- ## Entity Relationship Diagram [ERD]



One-to-Many Relationship in the ER Diagram

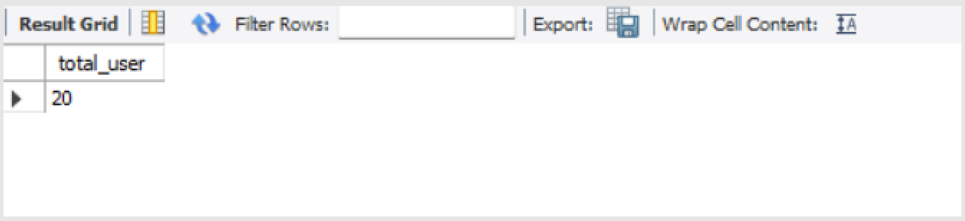
In this project, I've used a **one-to-many relationship** to capture how different parts of an e-commerce platform are connected, specifically focusing on user behavior and churn analysis.

For instance, let's take the relationship between **user_info** and **orders**. Each user can place multiple orders, but each order is linked to only one user. This relationship helps us understand how user activity influences order patterns, which is crucial for analyzing churn.
to improve retention. Similarly, **customer_support** and **user_activity** are related to users, where one user might interact with customer support multiple times or have various activities recorded. These relationships provide valuable insights into why users might churn and help identify patterns that we can address

• SQL Queries and Key Insights

Q1. How many total users are there?

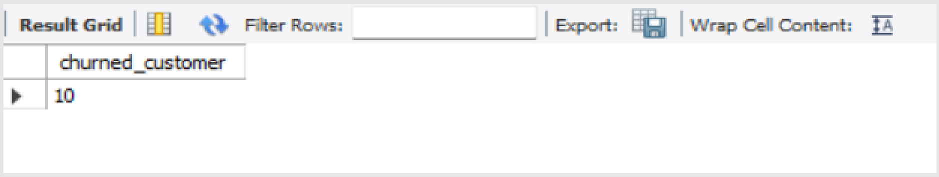
```
Select count (*) AS total_user from user_info
From user_info;
```



total_user
20

Q2. How many customers have churned?

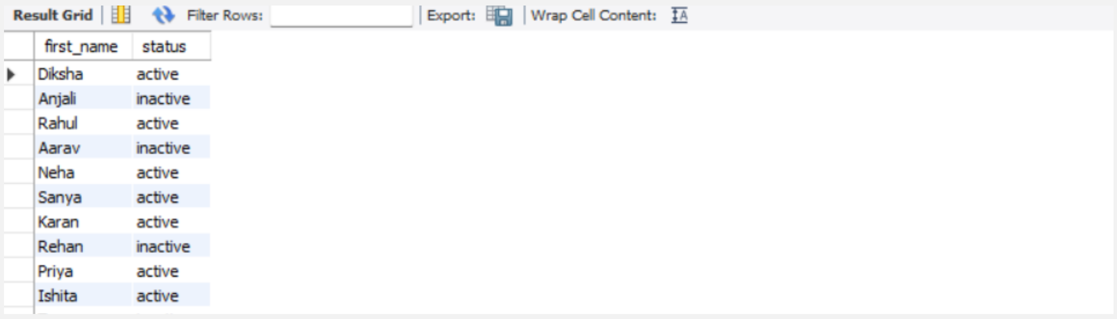
```
Select count (*) AS churned_customer
From churn_flag;
```



churned_customer
10

Q3. how many users are active vs inactive?

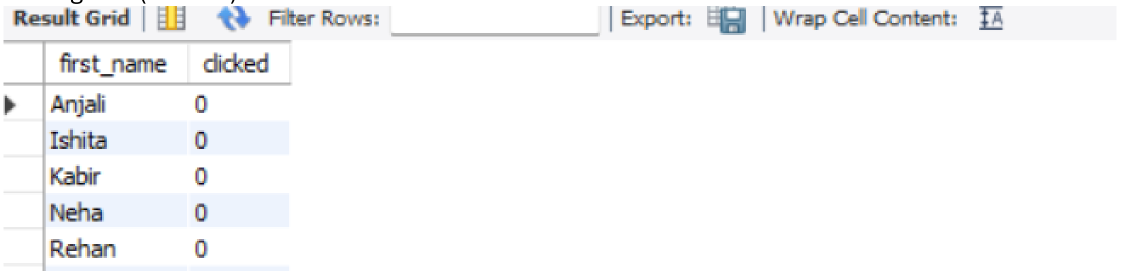
```
Select first_name
Case
When is_active = 1 then 'active'
Else 'inactive'
End as status
From user_info;
```



first_name	status
Diksha	active
Anjali	inactive
Rahul	active
Aarav	inactive
Neha	active
Sanya	active
Karan	active
Rehan	inactive
Priya	active
Ishita	active

Q4. users who never clicked on any campaign ?

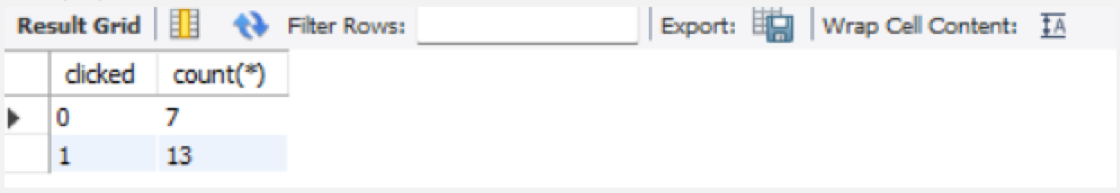
```
Select u.first_name ,m. clicked
From marketing_campaign m
Join user_info
On u.user_id = m.user_id
Group by first_name;
Having sum(clicked) > 0
```



first_name	clicked
Anjali	0
Ishita	0
Kabir	0
Neha	0
Rehan	0

Q5. Are our campaigns effectively engaging users?

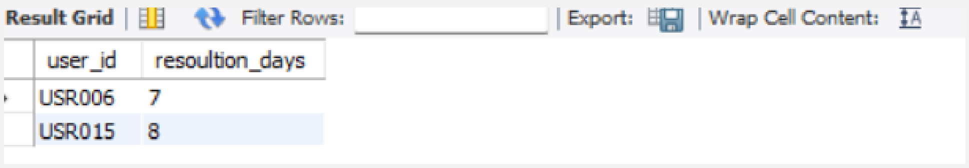
```
Select clicked, count (*)
From marketing_engagment
Group by clicked;
```



clicked	count(*)
0	7
1	13

Q5. Are delayed ticket resolutions causing churn?

```
Select user_id, resolution_days

From customer_support

Where resolution_days > 6;
```



user_id	resoultion_days
USR006	7
USR015	8