

SQL PROJECT

Customer Churn Analysis For A Subscription-Based Service

Dataset Link: <https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data>

Content:

Each row represents a customer, each column contains customer's attributes described on the column Metadata.

Objective:

Identify patterns that lead to customer churn and provide actionable insights to reduce it.

The data set includes information about:

- **Customers who left within the last month** – the column is called Churn.
- **Services that each customer has signed up for** – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
- **Customer account information** – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges.
- **Demographic info about customers** – gender, age range, and if they have partners and dependents.

About this file:

Telcom Customer Churn

- Each row represents a customer, each column contains customer's attributes described on the column Metadata.
- The raw data contains 7043 rows (customers) and 21 columns (features).
- The "Churn" column is our target.

Steps for Analysing Customer Churn and Designing the Database Schema

➤ Database Analysis:

- Conducted a thorough review of raw database columns and values to understand data structure and relationships.

➤ Data Segmentation:

- Created separate CSV files to organize data into logical tables:
 - Customer Details
 - Subscriptions
 - Services
 - Billing

➤ Key Identification:

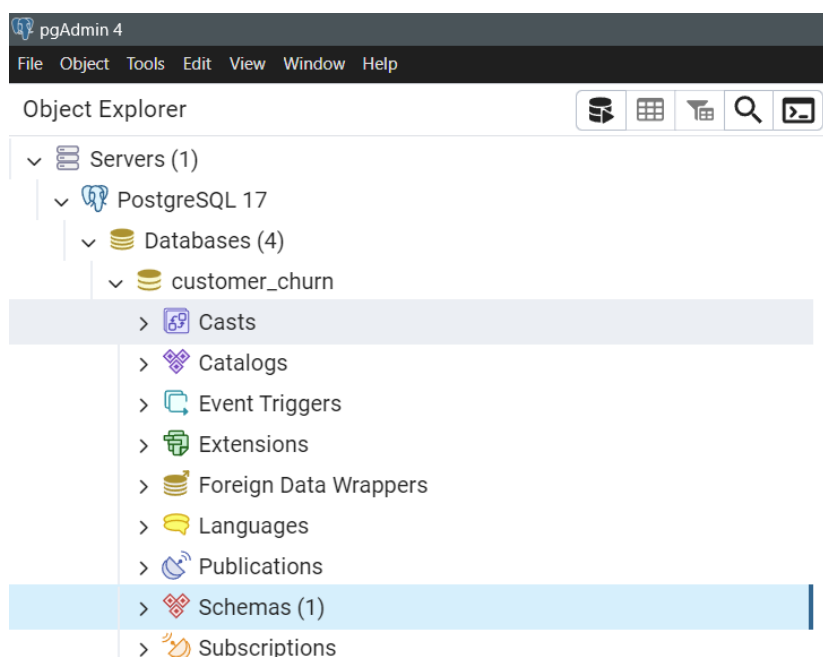
- Defined primary keys and foreign keys for each table to establish relationships and ensure data integrity.

➤ Schema Design:

- Utilized MS Excel to visually design and finalize the database schema, ensuring clarity and alignment with project requirements.

Steps under PostgreSQL

i. Creating the Database



ii. Creating tables in the database

a) Customer_Details Table

The screenshot displays a PostgreSQL client window titled "customer_churn/postgres@PostgreSQL 17*". The interface includes a toolbar with icons for file operations, query execution, and settings. The "Query" tab is active, showing the following SQL command:

```
1 CREATE TABLE customer_details (  
2     customerID VARCHAR(50) PRIMARY KEY,  
3     gender VARCHAR(10),  
4     SeniorCitizen BOOLEAN,  
5     partner VARCHAR(5),  
6     dependents VARCHAR(5)  
7 );  
8
```

Below the query editor, the "Messages" tab is selected, displaying the execution results:

```
CREATE TABLE  
  
Query returned successfully in 75 msec.
```

The status bar at the bottom indicates "Total rows: Query complete 00:00:00.075".

b) Creating Subscription_Details Table

The screenshot shows a PostgreSQL client window titled "customer_churn/postgres@PostgreSQL 17*". The interface includes a toolbar with icons for file operations, query execution, and data viewing. The "Query" tab is active, displaying the following SQL statement:

```
1 CREATE TABLE subscription_details (  
2     customerID VARCHAR(50),  
3     tenure INT,  
4     contract VARCHAR(50),  
5     PaperlessBilling VARCHAR(5),  
6     PaymentMethod VARCHAR(50),  
7     FOREIGN KEY (customerID) REFERENCES  
8     customer_details(CustomerID)  
9 );
```

Below the query editor, the "Messages" tab is active, showing the execution result:

```
CREATE TABLE  
  
Query returned successfully in 85 msec.
```

At the bottom of the window, a status bar displays:

- Total rows:
- Query complete 00:00:00.085
- CRLF
- Ln 8, Col 3

c) Creating Services Table

The screenshot shows a PostgreSQL client window titled "customer_churn/postgres@PostgreSQL 17*". The interface includes a toolbar with icons for file operations, query execution, and settings. The "Query" tab is active, displaying the following SQL code:

```
1 CREATE TABLE services(  
2     customerID VARCHAR(50),  
3     PhoneService VARCHAR(5),  
4     MultipleLines VARCHAR(25),  
5     InternetService VARCHAR(25),  
6     OnlineSecurity VARCHAR(25),  
7     DeviceProtection VARCHAR(25),  
8     TechSupport VARCHAR(25),  
9     StreamingTV VARCHAR(25),  
10    StreamingMovies VARCHAR(25),  
11    FOREIGN KEY (CustomerID) REFERENCES  
12    customer_details(customerID)  
13 );
```

Below the query editor, the "Messages" tab is active, showing the execution result:

```
CREATE TABLE  
  
Query returned successfully in 549 msec.
```

The status bar at the bottom indicates "Total rows: Query complete 00:00:00.549" and "CRLF Ln 12, Col 3".

d) Creating Billing_Details Table

The screenshot shows a PostgreSQL client window titled "customer_churn/postgres@PostgreSQL 17*". The interface includes a toolbar with icons for file operations, query execution, and navigation. Below the toolbar, there are tabs for "Query" and "Query History". The "Query" tab is active, displaying a SQL script to create a table named "billing_details". The script defines columns for "customerID" (VARCHAR(50)), "MonthlyCharges" (FLOAT8), "TotalCharges" (FLOAT8), and "Churn" (VARCHAR(5)), and includes a foreign key constraint referencing the "customerID" column in the "customer_details" table. Below the query editor, there are tabs for "Data Output", "Messages", and "Notifications". The "Messages" tab is active, showing the message "CREATE TABLE" and "Query returned successfully in 71 msec.". At the bottom of the window, a status bar displays "Total rows: Query complete 00:00:00.071" and "CRLF Ln 7, Col 3".

```
1 CREATE TABLE billing_details(  
2     customerID VARCHAR(50),  
3     MonthlyCharges FLOAT8,  
4     TotalCharges FLOAT8,  
5     Churn VARCHAR(5),  
6     FOREIGN KEY (customerID) REFERENCES  
7     customer_details(customerID)  
8 );
```

CREATE TABLE

Query returned successfully in 71 msec.

Total rows: Query complete 00:00:00.071

CRLF Ln 7, Col 3

iii. Importing CSV files to the respective tables:

a) Importing customer_details

The screenshot shows the DBeaver interface with the 'customer_churn/postgres@PostgreSQL 17*' database selected. The 'Query' tab is active, displaying a SQL command to copy data from a CSV file into the 'customer_details' table. The command is as follows:

```
1 COPY customer_details(customerID,gender,SeniorCitizen,partner,dependents)
2 FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\customer_details.csv'
3 DELIMITER ','
4 CSV HEADER;
```

The 'Messages' tab is selected, showing the execution result: 'COPY 7043' and 'Query returned successfully in 154 msec.' The status bar at the bottom indicates 'Total rows: Query complete 00:00:00.154' and 'Ln 4, Col 12'.

b) Importing services

I forgot to add OnlineBackup Column

The screenshot shows the DBeaver interface with the 'customer_churn/postgres@PostgreSQL 17*' database selected. The 'Query' tab is active, displaying a SQL command to add a new column to the 'services' table. The command is as follows:

```
6 ALTER TABLE services
7 ADD COLUMN OnlineBackup VARCHAR(25);
```

The 'Messages' tab is selected, showing the execution result: 'ALTER TABLE' and 'Query returned successfully in 102 msec.' The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.102'.

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17* X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

```

1 COPY
  services(customerid,phoneservice,multiplelines,onlinebackup,internetsevice,onlinesecurity,deviceprotection,t
  echsupport,streamingtv,streamingmovies)
2 FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\services.csv'
3 DELIMITER ','
4 CSV HEADER;

```

Data Output Messages Notifications

COPY 7043

Query returned successfully in 156 msec.

Total rows: 1 Query complete 00:00:00.156 CRLF Ln 4, Col 12

c) Importing Billing_Details

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17* X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

```

1 COPY billing_details(customerid, monthlycharges, totalcharges,churn)
2 FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\billing_details.csv'
3 DELIMITER ','
4 CSV HEADER;

```

Data Output Messages Notifications

COPY 7043

Query returned successfully in 356 msec.

d) Importing Subscription_Details

Dashboard × Properties × Statistics × Dependents × Processes × customer_churn/postgres@PostgreSQL 17* ×

customer_churn/postgres@PostgreSQL 17

Query Query History

```
1 COPY subscription_details(customerid,tenure,contract,paperlessbilling,paymentmethod)
2 FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\subscription_details.csv'
3 DELIMITER ','
4 CSV HEADER;
```

Data Output Messages Notifications

COPY 7043

Query returned successfully in 255 msec.

Total rows: 1 Query complete 00:00:00.255 CRLF Ln 4, Col 12

Now applying queries to the database

Ques 1: Total number of Churned customers.

Dashboard × Properties × Statistics × Dependents × Processes × customer_churn/postgres@PostgreSQL 17* ×

customer_churn/postgres@PostgreSQL 17

Query Query History

```
1 -- Count the no. of churned customers
2 SELECT COUNT(customerid) as ChurnedCustomers
3 FROM billing_details
4 WHERE churn = 'Yes';
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of

churnedcustomers
1869

Total rows: 1 Query complete 00:00:00.274

Ques 2: List of customers with having Internet Service.

Dashboard X Properties X Statistics X Dependents X Processes X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

1 -- List customers with having Internet Service.

2 ✓ SELECT customerid, internetservice

3 FROM services

4 WHERE internetservice = 'Yes';

Data Output Messages Notifications

SQL

	customerid character varying (50)	internetservice character varying (25)
1	5575-GNVDE	Yes
2	3668-QPYBK	Yes
3	7795-CFOCW	Yes

Total rows: 2019 Query complete 00:00:00.327

Ques 3: Average monthly charges of all customers.

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17*

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

```
1 -- Calculate average monthly charges.
2 SELECT
3     ROUND(AVG(monthlycharges),2) AS AVGMonthlyCharges
4 FROM billing_details;
```

Data Output Messages Notifications

ERROR: function round(double precision, integer) does not exist
LINE 3: ROUND(AVG(monthlycharges),2) AS AVGMonthlyCharges
 ^
HINT: No function matches the given name and argument types. You might need to add explicit type casts.

SQL state: 42883
Character: 47

Total rows: 1 Query complete 00:00:00.108

This shows typecast error because monthlycharges have float constraint

Modify the monthlycharges and totalcharges

ALTER TABLE billing_details

ALTER COLUMN MonthlyCharges TYPE INTEGER USING MonthlyCharges::INTEGER;

ALTER TABLE billing_details

ALTER COLUMN TotalCharges TYPE INTEGER USING TotalCharges::INTEGER;

```

12  ALTER TABLE billing_details
13  ALTER COLUMN MonthlyCharges TYPE INTEGER USING MonthlyCharges::INTEGER;
14
15  ALTER TABLE billing_details
16  ALTER COLUMN TotalCharges TYPE INTEGER USING TotalCharges::INTEGER;
17

```

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 90 msec.

Total rows: 1 Query complete 00:00:00.090

Dashboard X Properties X Statistics X Dependents X Processes X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

```

1  -- Calculate average monthly charges.
2  SELECT
3      ROUND(AVG(monthlycharges),2) AS AVGMonthlyCharges
4  FROM billing_details; |
5
6

```

Data Output Messages Notifications

SQL

Sh

	avgmonthlycharges numeric	
1	64.76	

Total rows: 1 Query complete 00:00:00.138

Ques 4: Churn Rate by Contract Type.

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17* X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

1 -- Churn Rate by Contract Type.

2 SELECT

3 contract,

4 COUNT(customerid) AS TotalCustomers,

5 SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,

6 ROUND(SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END) * 100 / COUNT(customerid),2)

7 FROM subscription_details

8 JOIN billing_details USING(Customerid)

9 GROUP BY contract;

Data Output Messages Notifications

SQL

Showing rows: 1 to 3 Page No: 1 of

	contract character varying (50)	totalcustomers bigint	churnedcustomers bigint	round numeric
1	One year	1473	166	11.00
2	Month-to-month	3875	1655	42.00
3	Two year	1695	48	2.00

Total rows: 3 Query complete 00:00:05.220

Ques 5: Top 3 Customers with the Highest Total Charges

Dashboard × Properties × Statistics × Dependents × Processes × customer_churn/postgres@PostgreSQL 17* ×

customer_churn/postgres@PostgreSQL 17

No limit

E

II.

Query

Query History

1 -- Top 3 Customers with the Highest Total Charges

2 SELECT

3 customerid,

4 totalcharges

5 FROM billing_details

6 ORDER BY totalcharges DESC

7 LIMIT 3;

Data Output

Messages

Notifications

Showing rows: 1 to 3

Page No:

1

of

	customerid character varying (50)	totalcharges integer
1	2889-FPWRM	8685
2	7569-NMZYQ	8672
3	9739-JLPQJ	8670

Total rows: 3 Query complete 00:00:00.275

Ques 6: Churn Rate by Internet Service type

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17 X

customer_churn/postgres@PostgreSQL 17

Query Query History

```
1 -- Churn Rate by Internet Service Type
2 SELECT
3     internetService,
4     COUNT(customerid) AS TotalCustomers,
5     SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
6     ROUND(SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END)*100 / COUNT(customerid), 2) AS ChurnRate
7 FROM services
8 JOIN billing_details USING (customerid)
9 GROUP BY internetService;
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	internetService character varying (25)	totalcustomers bigint	churnedcustomers bigint	churnrate numeric
1	No	3498	1461	41.00
2	No internet service	1526	113	7.00
3	Yes	2019	295	14.00

Total rows: 3 Query complete 00:00:00.278 CRLF

Ques 7: Correlation Between Monthly Charges and Churn

Analyze if higher monthly charges lead to churn by categorizing customers into charge ranges.

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17 X

customer_churn/postgres@PostgreSQL 17

Query Query History

```
1 -- Correlation Between Monthly Charges and Churn
2 -- Analyze if higher monthly charges lead to churn by categorizing customers into charge ranges.
3 SELECT
4     CASE
5         WHEN MonthlyCharges < 50 THEN '<$50'
6         WHEN MonthlyCharges BETWEEN 50 AND 100 THEN '$50-$100'
7         ELSE '>$100'
8     END AS ChargeRange,
9     COUNT(customerID) AS TotalCustomers,
10    SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
11    ROUND(SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 100.0 / COUNT(customerID), 2) AS ChurnRate
12 FROM billing_details
13 GROUP BY ChargeRange
14 ORDER BY ChurnRate DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	chargeRange text	totalcustomers bigint	churnedcustomers bigint	churnrate numeric
1	\$50-\$100	3967	1292	32.57
2	>\$100	837	228	27.24
3	<\$50	2239	349	15.59

Total rows: 3 Query complete 00:00:05.193 CRLF Ln 14, Col 25

Ques 8: Services Combination Leading to High Churn

Identify combinations of services (e.g., Phone, Internet) with the highest churn rate.

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17* X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

```
1 -- Services Combination Leading to High Churn
2 -- Identify combinations of services (e.g., Phone, Internet) with the highest churn rate.
3 SELECT
4     PhoneService,
5     InternetService,
6     COUNT(customerID) AS TotalCustomers,
7     SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
8     ROUND(SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 100.0 / COUNT(customerID), 2) AS ChurnRate
9 FROM services
10 JOIN billing_details USING (customerID)
11 GROUP BY PhoneService, InternetService
12 ORDER BY ChurnRate DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	phoneservice character varying (5)	internetservice character varying (25)	totalcustomers bigint	churnedcustomers bigint	churnrate numeric
1	Yes	No	3099	1320	42.59
2	No	No	399	141	35.34
3	Yes	Yes	1736	266	15.32
4	No	Yes	283	29	10.25
5	Yes	No internet service	1526	113	7.40

Total rows: 5 Query complete 00:00:00.348 CRLF Ln 12,

Ques 9: Customer Segmentation by Tenure and Churn

Segment customers into tenure ranges and analyze their churn behavior.

Dashboard X Properties X Statistics X Dependents X Processes X customer_churn/postgres@PostgreSQL 17* X

customer_churn/postgres@PostgreSQL 17

No limit

Query Query History

```
1 -- Customer Segmentation by Tenure and Churn
2 -- Segment customers into tenure ranges and analyze their churn behavior.
3
4 SELECT
5     CASE
6         WHEN tenure <= 12 THEN '0-12 months'
7         WHEN tenure BETWEEN 13 AND 24 THEN '13-24 months'
8         WHEN tenure BETWEEN 25 AND 36 THEN '25-36 months'
9         ELSE '36+ months'
10    END AS TenureGroup,
11    COUNT(customerID) AS TotalCustomers,
12    SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
13    ROUND(SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 100.0 / COUNT(customerID), 2) AS ChurnRate
14 FROM subscription_details
15 JOIN billing_details USING (customerID)
16 GROUP BY TenureGroup
17 ORDER BY ChurnRate DESC;
```


≡+

▼

▼

SQL

Showing rows: 1 to 4

	<div>tenuregroup</div> <div>text</div> <div></div>	<div>totalcustomers</div> <div>bigint</div> <div></div>	<div>churnedcustomers</div> <div>bigint</div> <div></div>	<div>churnrate</div> <div>numeric</div> <div></div>
1	0-12 months	2186	1037	47.44
2	13-24 months	1024	294	28.71
3	25-36 months	832	180	21.63
4	36+ months	3001	358	11.93