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

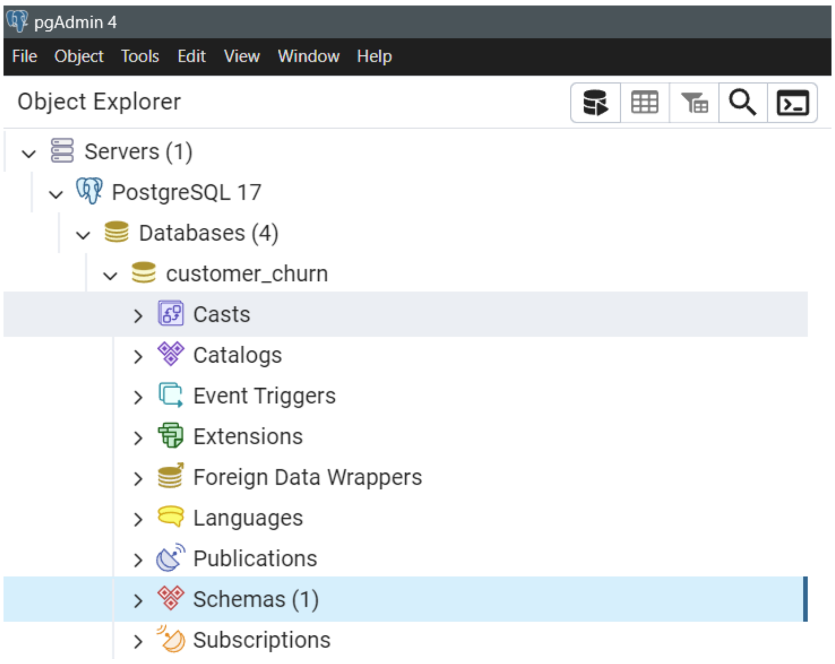
- Each row represents a customer, each column contains customer's attributes described on the column Metadata.
- The raw data contains 7043 rows (customers) and 21 columns (features).
- The "Churn" column is our target.

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.

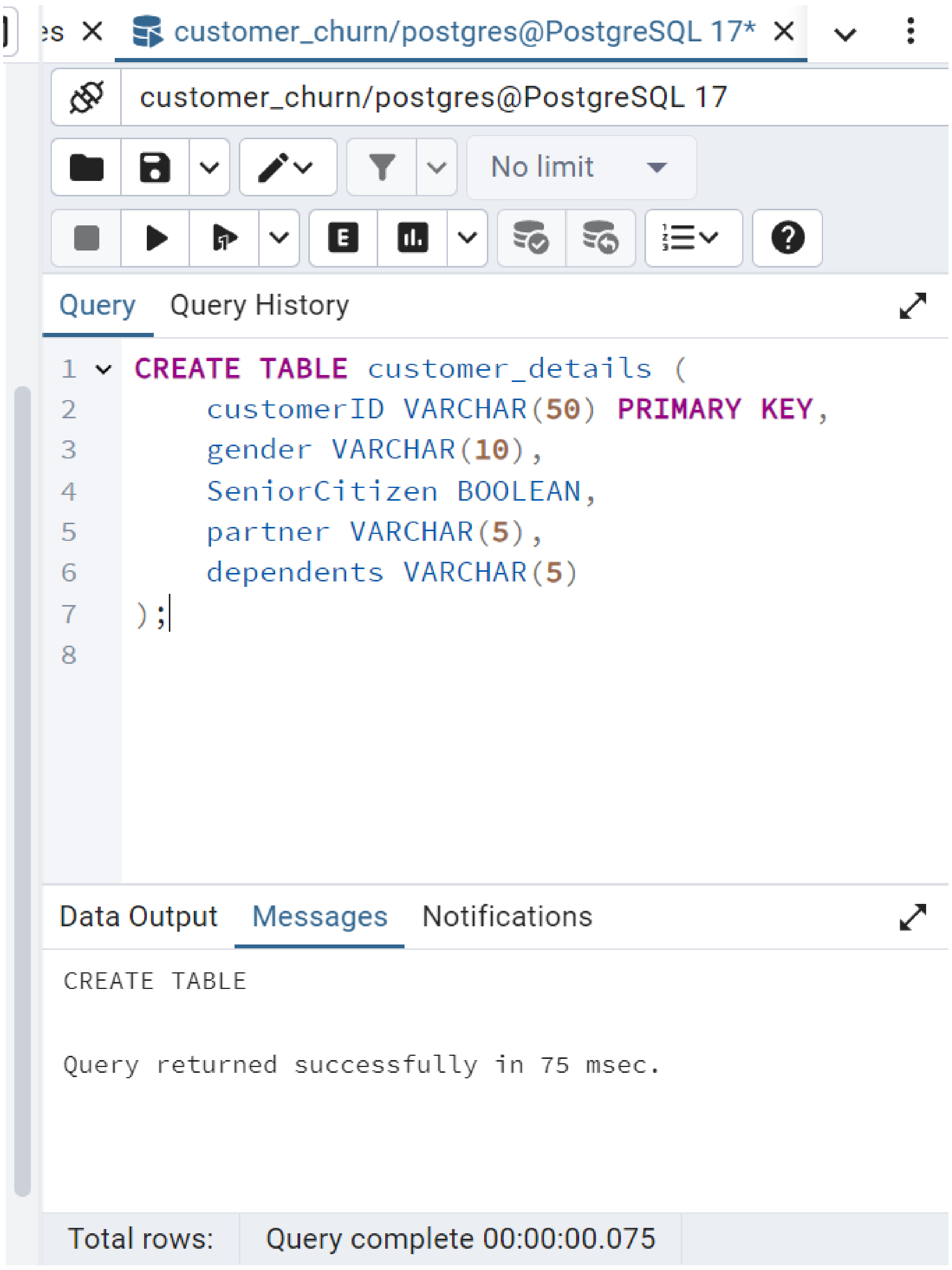
Steps under PostgreSQL

i. Creating the Database



ii. Creating tables in the database

a) Customer_Details Table

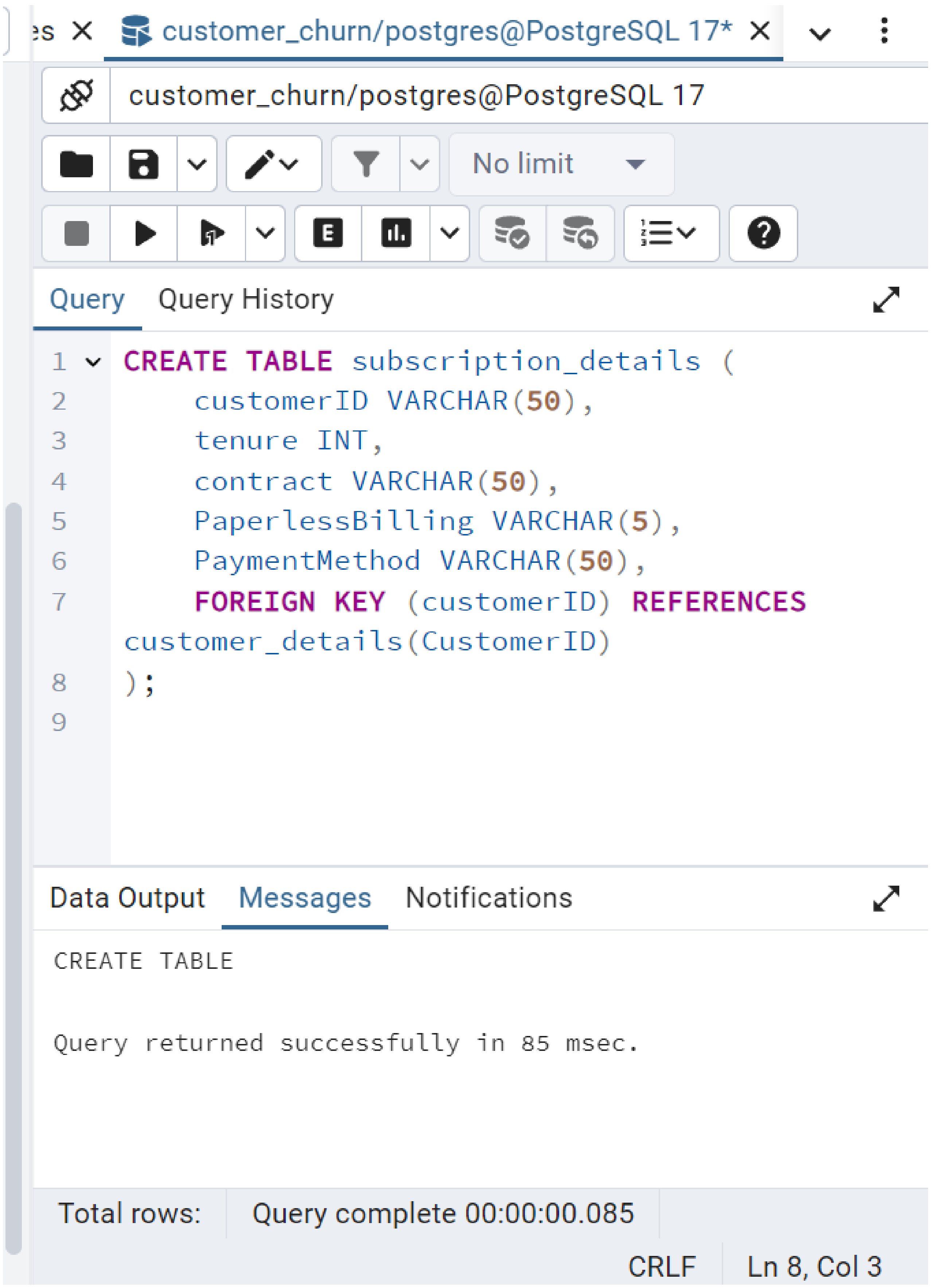

customer_churn/postgres@PostgreSQL 17*
customer_churn/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications

```
CREATE TABLE customer_details (
    customerID VARCHAR(50) PRIMARY KEY,
    gender VARCHAR(10),
    SeniorCitizen BOOLEAN,
    partner VARCHAR(5),
    dependents VARCHAR(5)
);
```

```
CREATE TABLE

Query returned successfully in 75 msec.
```

Total rows: Query complete 00:00:00.075

b) Creating Subscription_Details Table


customer_churn/postgres@PostgreSQL 17*
customer_churn/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications

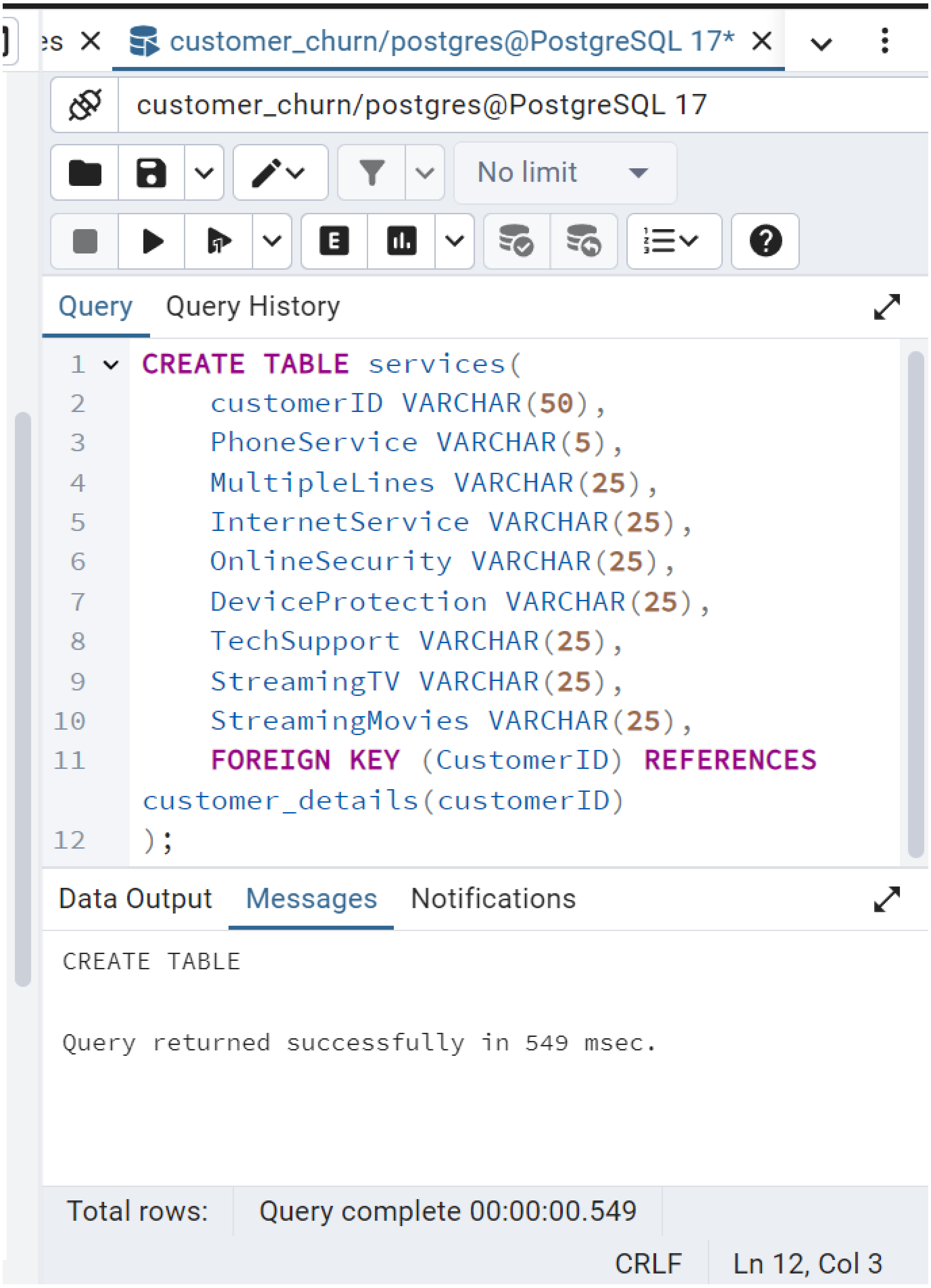
```sql
CREATE TABLE subscription_details (
    customerID VARCHAR(50),
    tenure INT,
    contract VARCHAR(50),
    PaperlessBilling VARCHAR(5),
    PaymentMethod VARCHAR(50),
    FOREIGN KEY (customerID) REFERENCES customer_details(CustomerID)
);
```

```
CREATE TABLE

Query returned successfully in 85 msec.
```

Total rows: Query complete 00:00:00.085

CRLF   Ln 8, Col 3

c) **Creating Services Table**


es
customer_churn/postgres@PostgreSQL 17*
customer_churn/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications

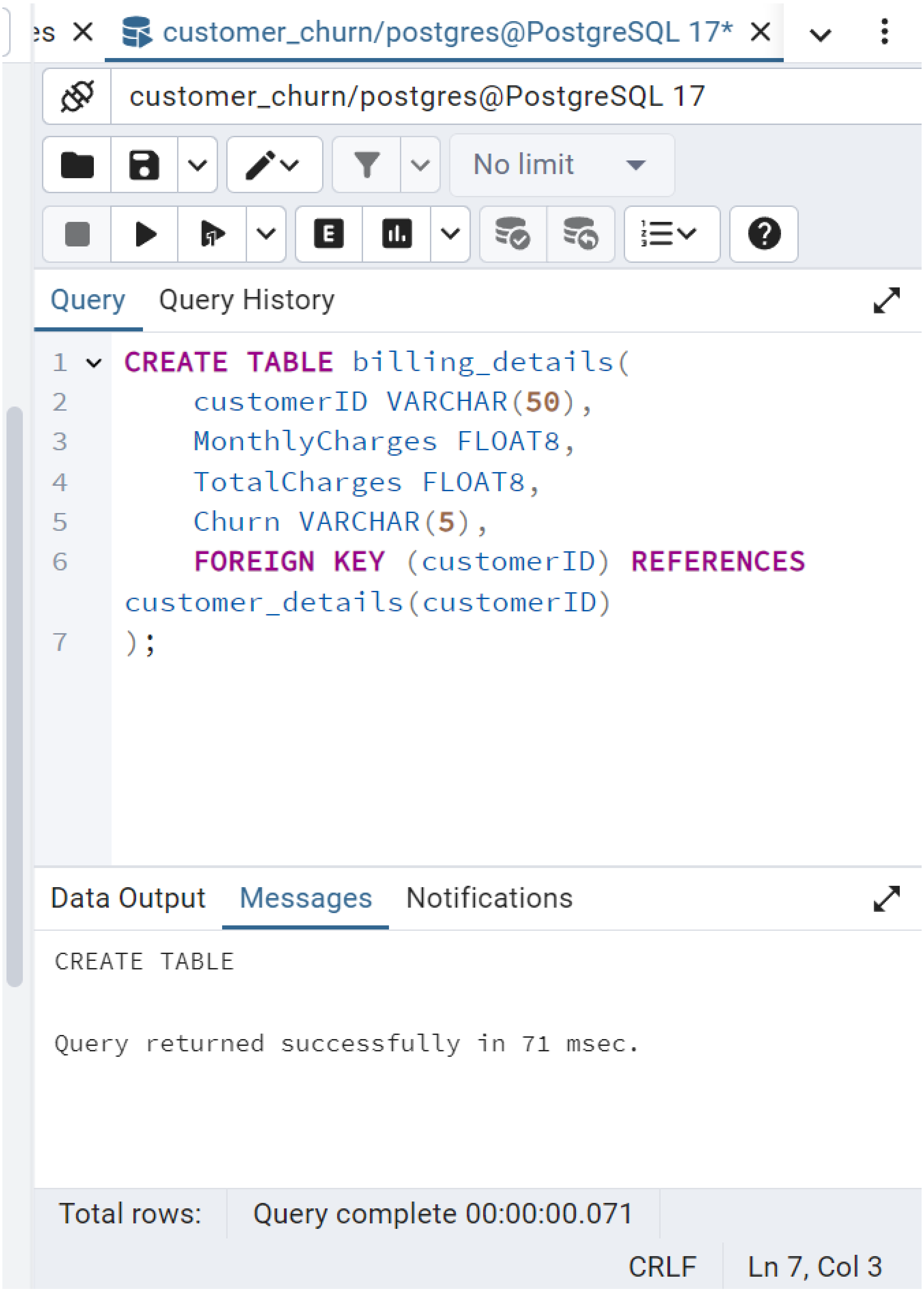
```
CREATE TABLE services(
	customerID VARCHAR(50),
	PhoneService VARCHAR(5),
	MultipleLines VARCHAR(25),
	InternetService VARCHAR(25),
	OnlineSecurity VARCHAR(25),
	DeviceProtection VARCHAR(25),
	TechSupport VARCHAR(25),
	StreamingTV VARCHAR(25),
	StreamingMovies VARCHAR(25),
	FOREIGN KEY (CustomerID) REFERENCES
customer_details(customerID)
);
```

```
CREATE TABLE

Query returned successfully in 549 msec.
```

Total rows: Query complete 00:00:00.549

CRLF Ln 12, Col 3

d) Creating Billing_Details Table


customer_churn/postgres@PostgreSQL 17*
customer_churn/postgres@PostgreSQL 17
No limit
Query
Query History
Data Output
Messages
Notifications

```
CREATE TABLE billing_details(
     customerID VARCHAR(50),
     MonthlyCharges FLOAT8,
     TotalCharges FLOAT8,
     Churn VARCHAR(5),
     FOREIGN KEY (customerID) REFERENCES
customer_details(customerID)
);
```

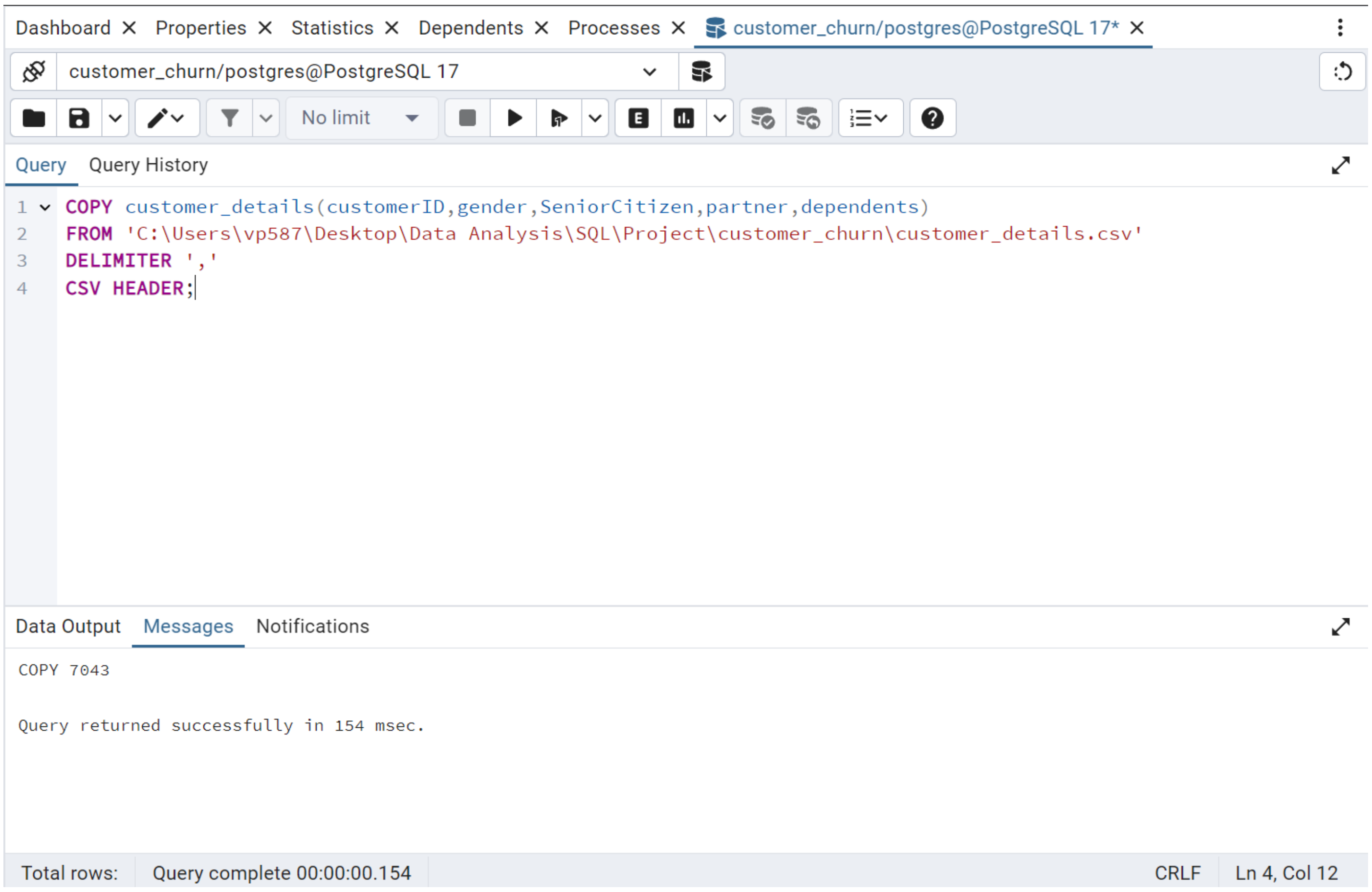
```
CREATE TABLE

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



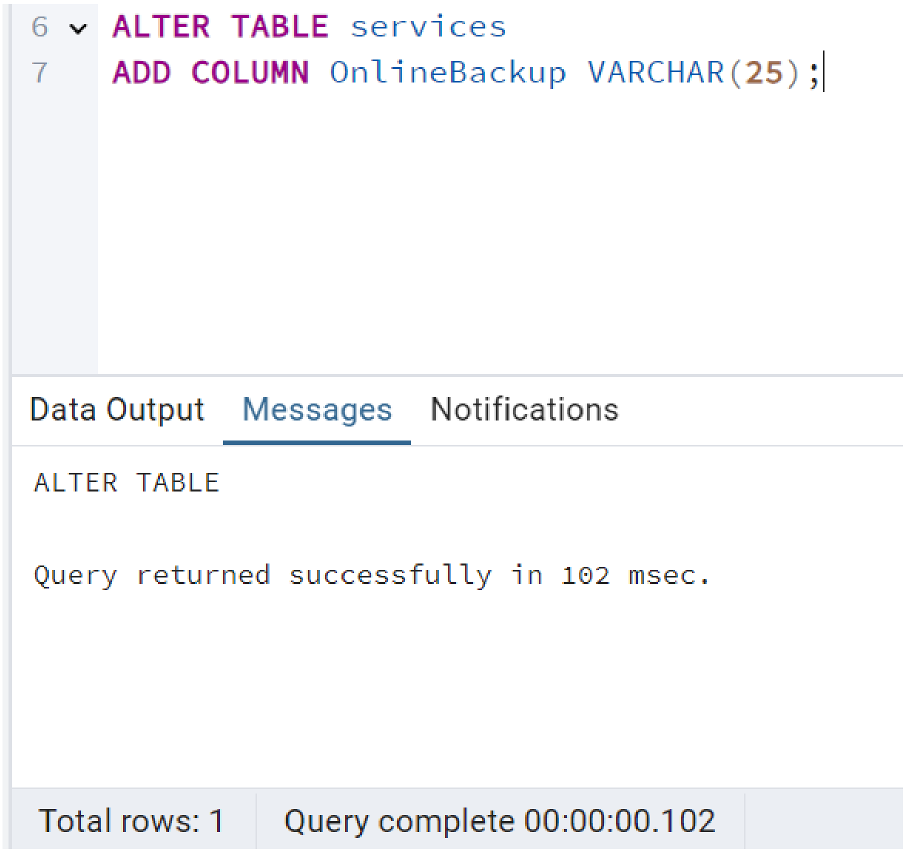
```sql
COPY customer_details(customerID,gender,SeniorCitizen,partner,dependents)
FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\customer_details.csv'
DELIMITER ','
CSV HEADER;
```

```
COPY 7043

Query returned successfully in 154 msec.
```

Total rows: | Query complete 00:00:00.154 | CRLF | Ln 4, Col 12

b) Importing services

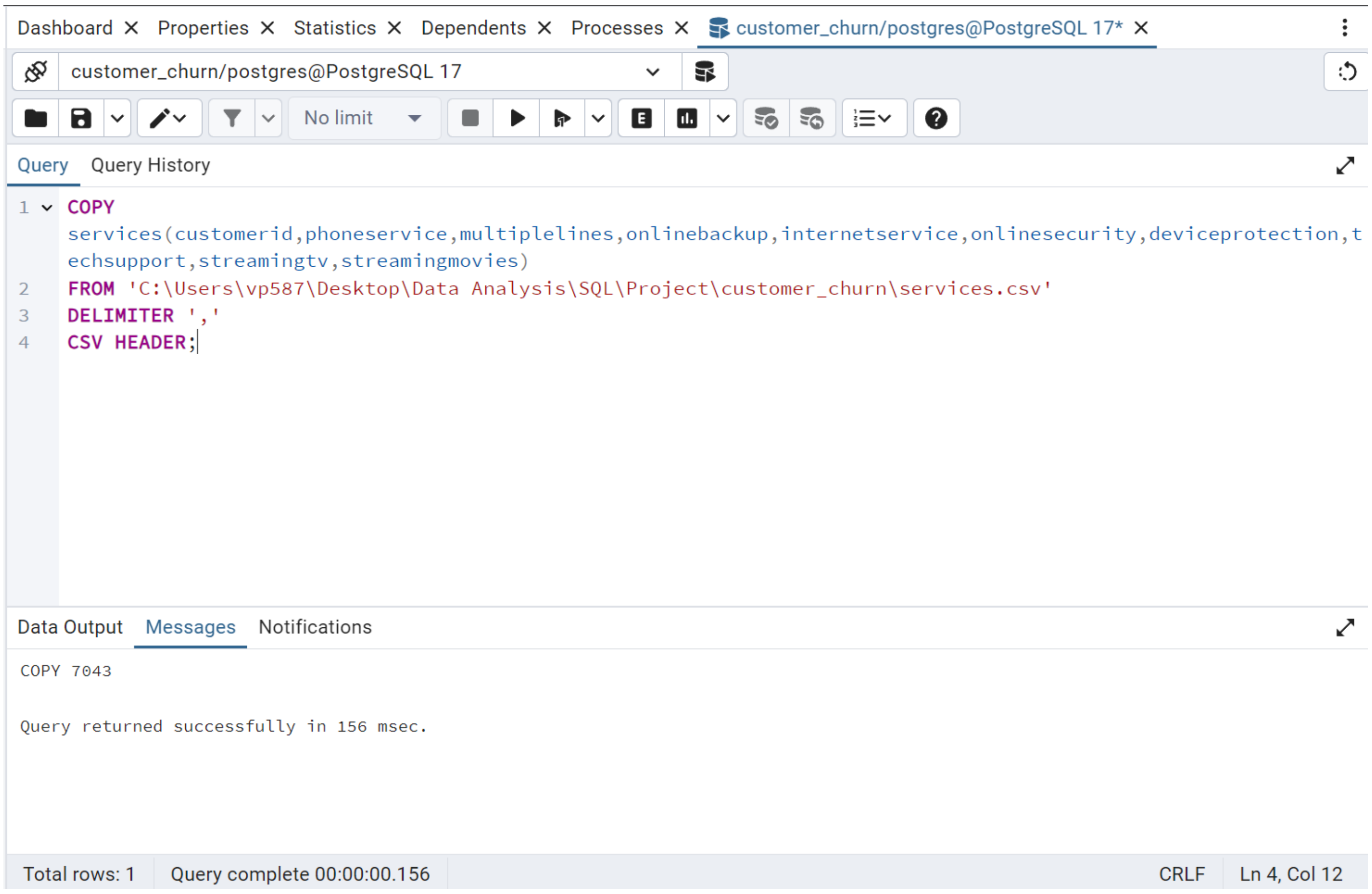
I forgot to add OnlineBackup Column



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

```
ALTER TABLE

Query returned successfully in 102 msec.
```

Total rows: 1 | Query complete 00:00:00.102

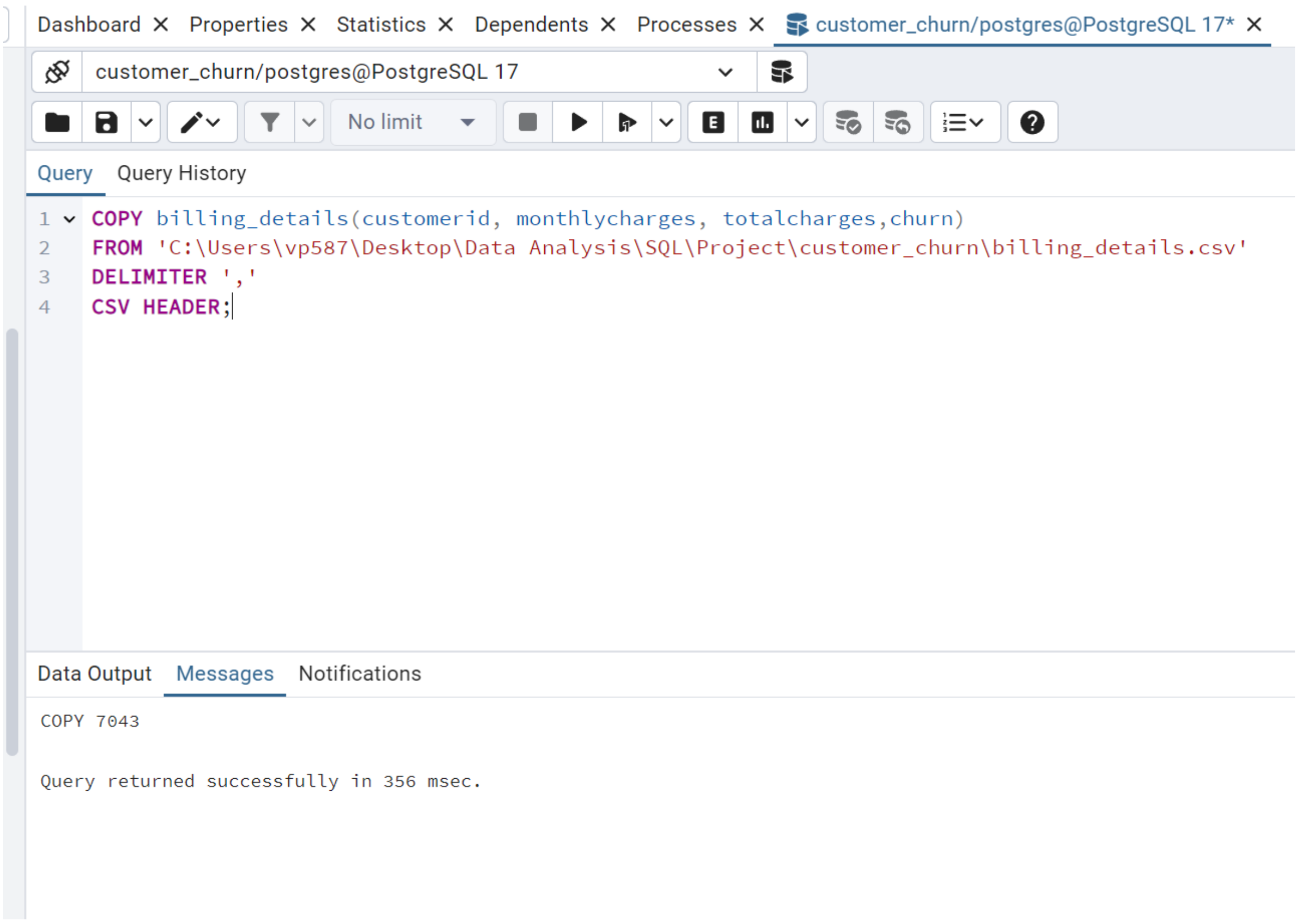


```
COPY
services(customerid,phoneservice,multiplelines,onlinebackup,internetservice,onlinesecurity,deviceprotection,techsupport,streamingtv,streamingmovies)
FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\services.csv'
DELIMITER ','
CSV HEADER;
```

```
COPY 7043

Query returned successfully in 156 msec.
```

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

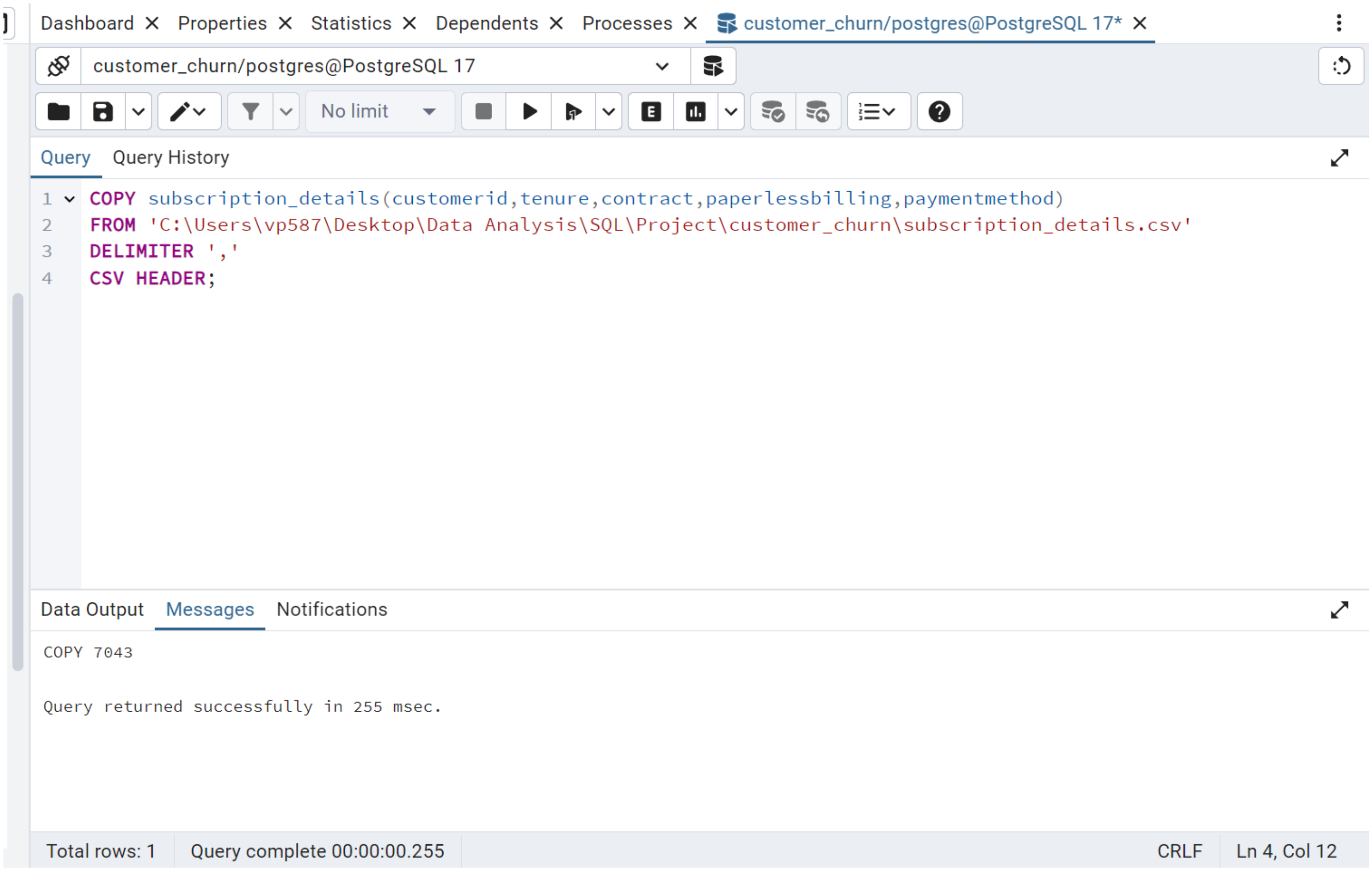
c) Importing Billing_Details



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

```
COPY 7043

Query returned successfully in 356 msec.
```

d) Importing Subscription_Details



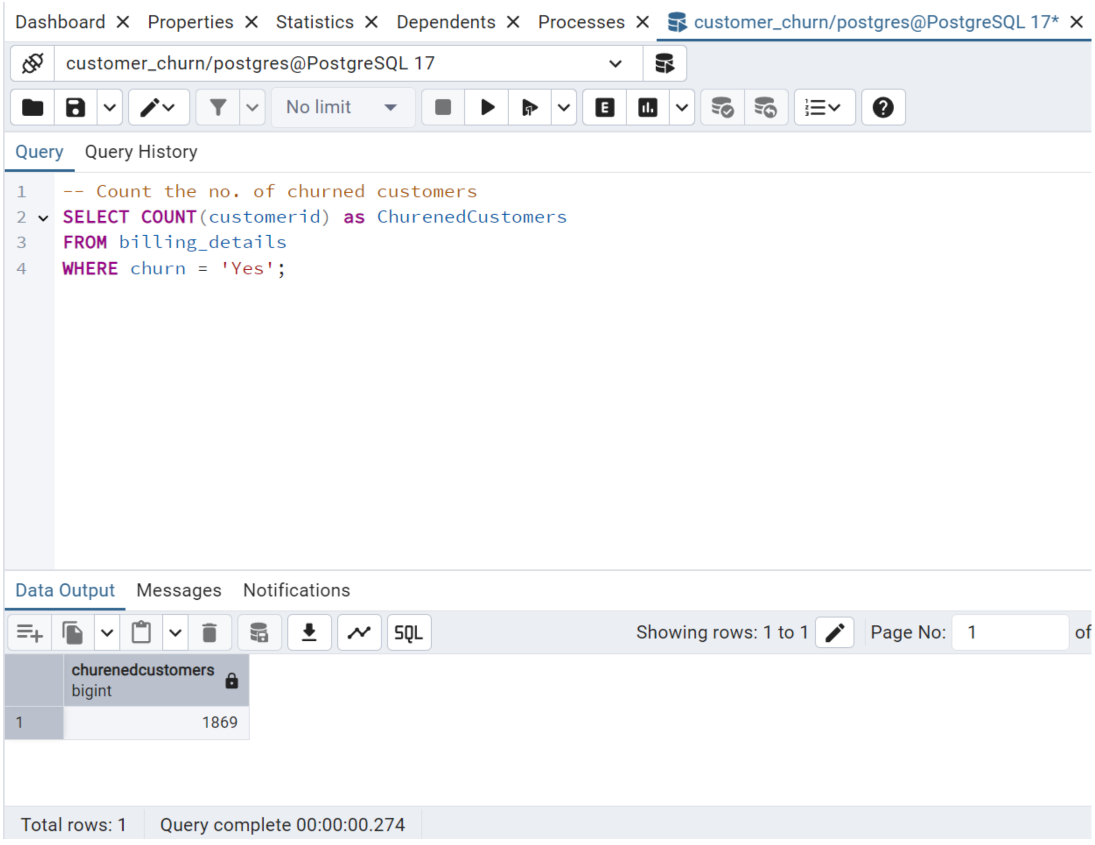
```
COPY subscription_details(customerid,tenure,contract,paperlessbilling,paymentmethod)
FROM 'C:\Users\vp587\Desktop\Data Analysis\SQL\Project\customer_churn\subscription_details.csv'
DELIMITER ','
CSV HEADER;
```

```
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.

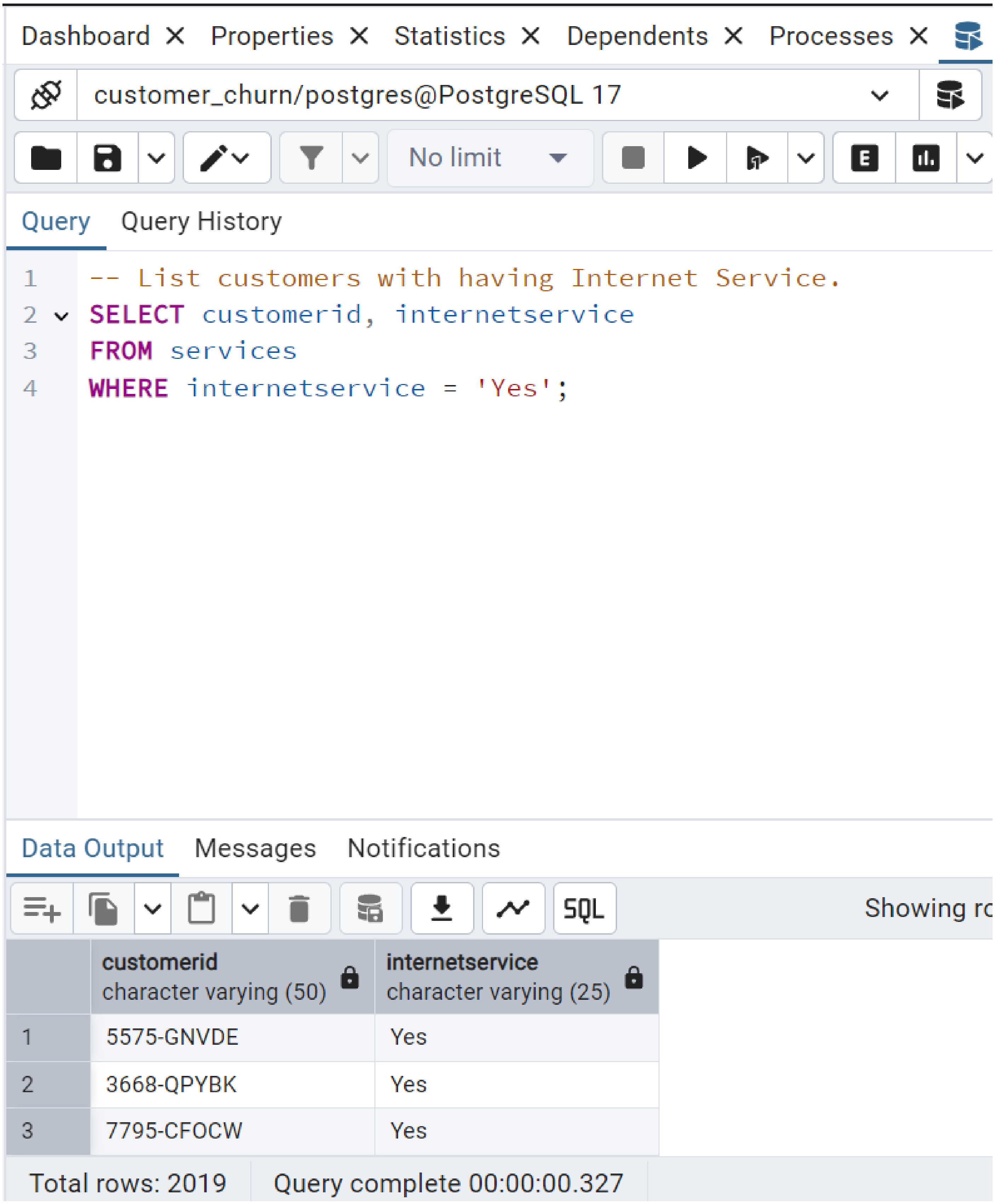


```
-- Count the no. of churned customers
SELECT COUNT(customerid) as ChurenedCustomers
FROM billing_details
WHERE churn = 'Yes';
```

	churenedcustomers bigint
1	1869

Total rows: 1   Query complete 00:00:00.274

Ques 2: List of customers with having Internet Service.

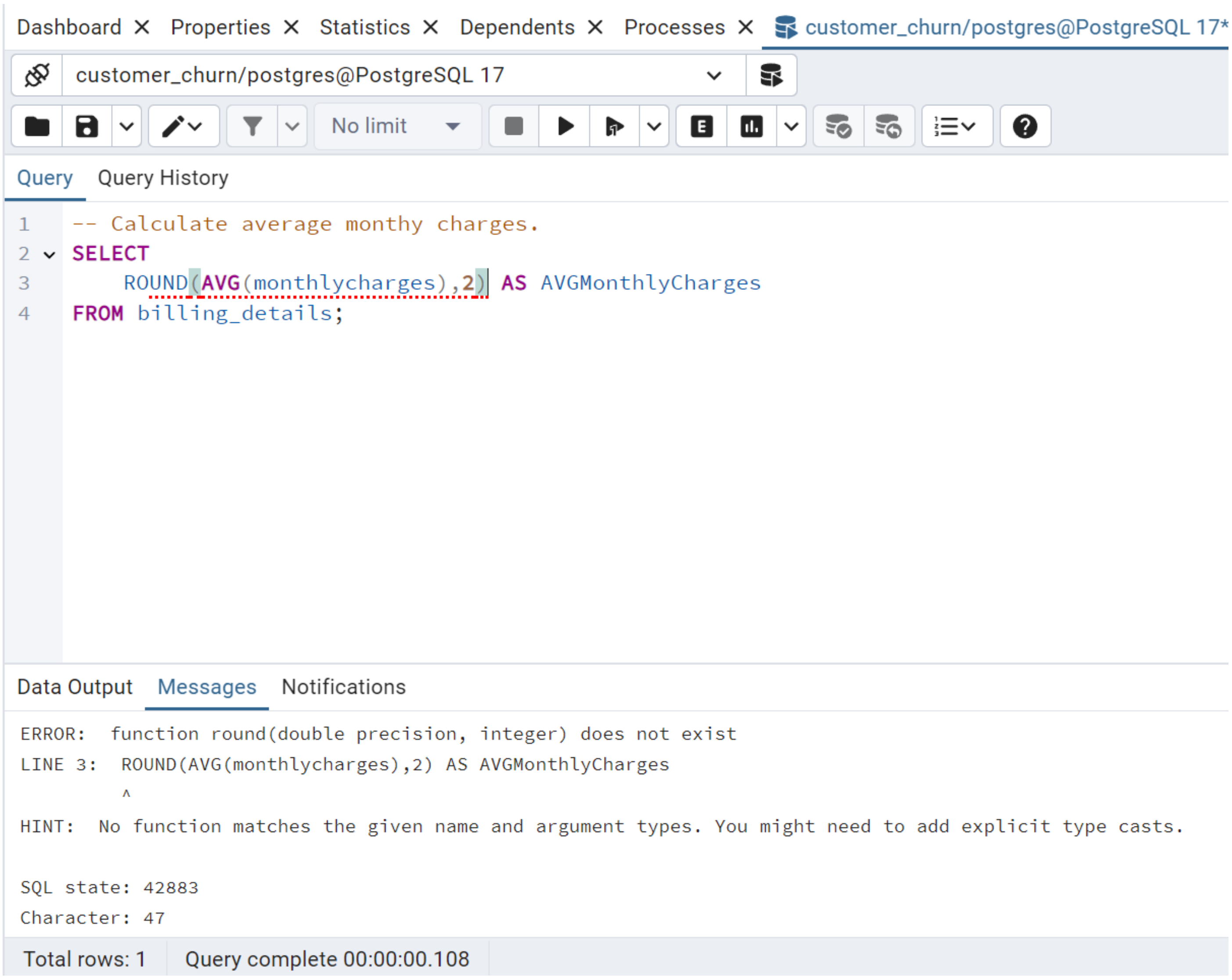


```sql
-- List customers with having Internet Service.
SELECT customerid, internetservice
FROM services
WHERE internetservice = 'Yes';
```

	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 monthy charges of all customers.



```
-- Calculate average monthy charges.
SELECT
    ROUND(AVG(monthlycharges),2) AS AVGMonthlyCharges
FROM billing_details;
```

```
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

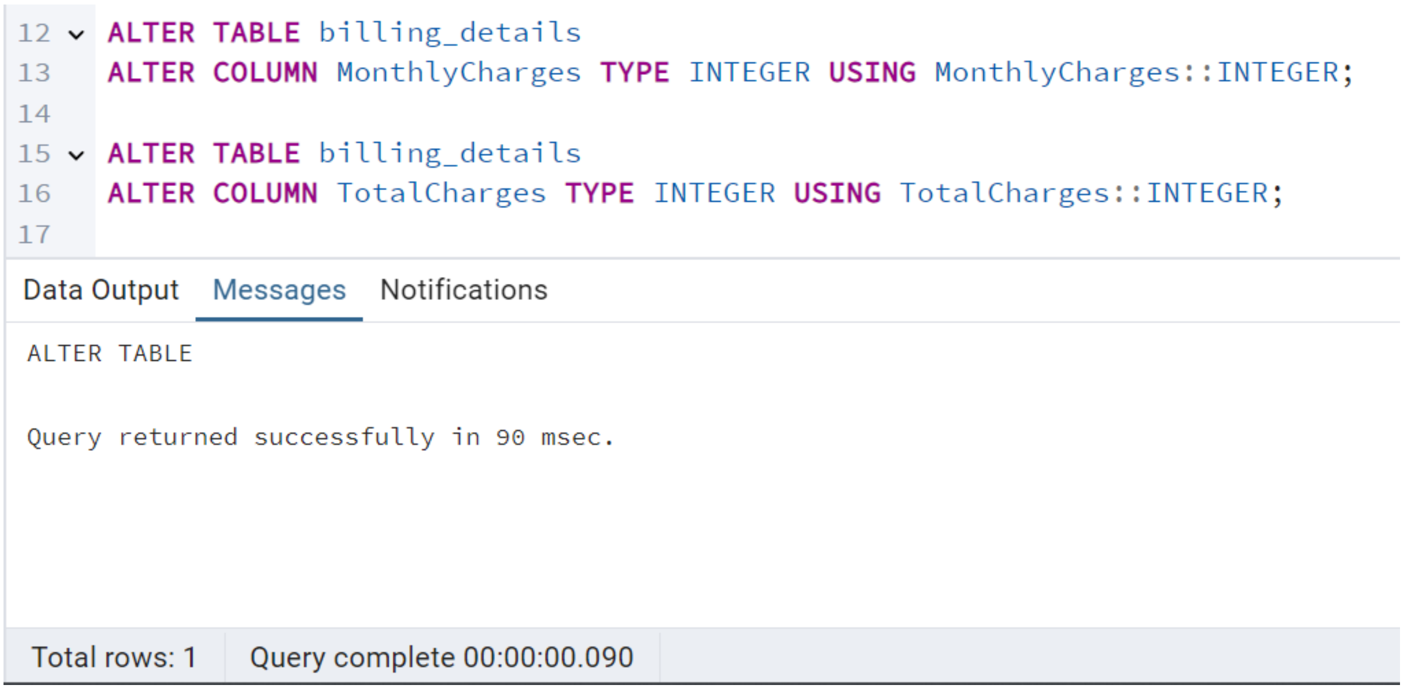
Modify the monthlycharges and totalcharges

ALTER TABLE billing_details

ALTER COLUMN MonthlyCharges TYPE INTEGER USING MonthlyCharges::INTEGER;

ALTER TABLE billing_details

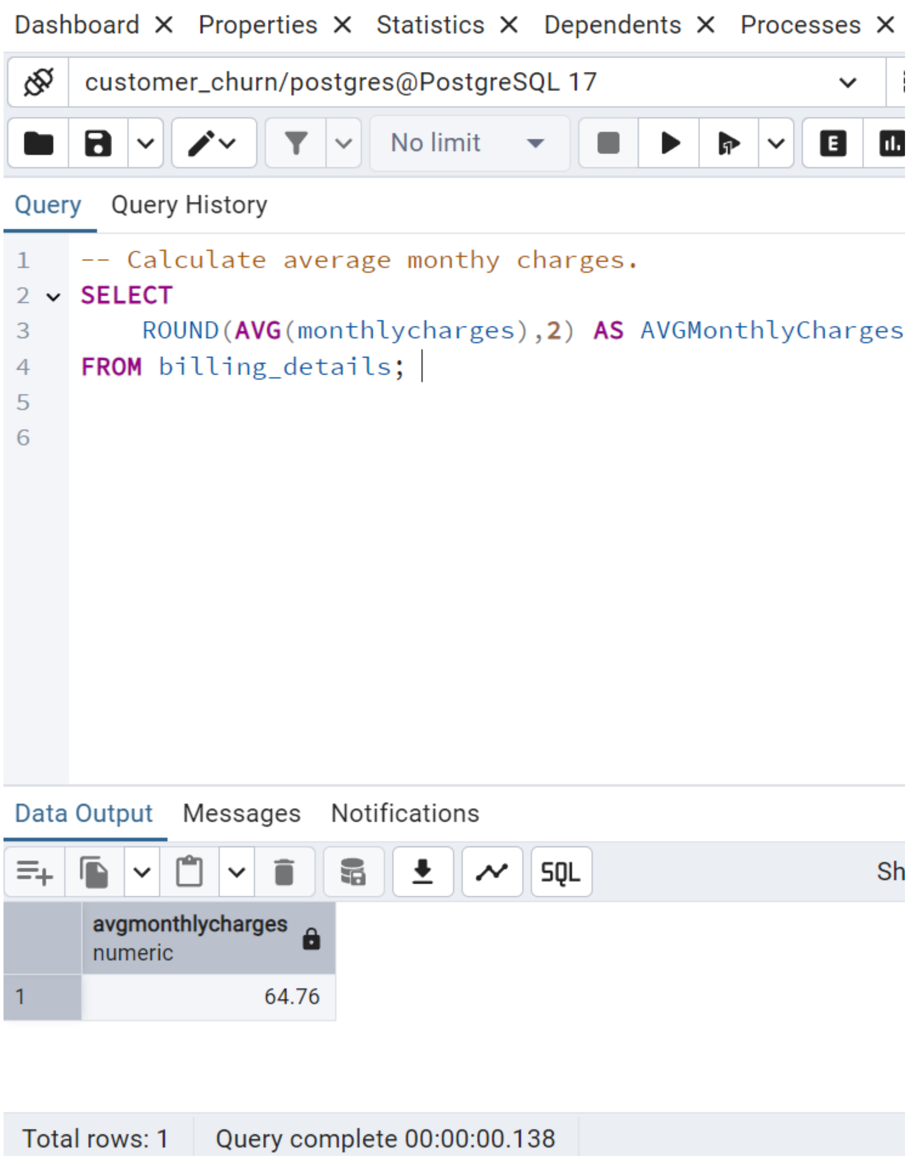
ALTER COLUMN TotalCharges TYPE INTEGER USING TotalCharges::INTEGER;



```
ALTER TABLE billing_details
ALTER COLUMN MonthlyCharges TYPE INTEGER USING MonthlyCharges::INTEGER;

ALTER TABLE billing_details
ALTER COLUMN TotalCharges TYPE INTEGER USING TotalCharges::INTEGER;
```

```
ALTER TABLE

Query returned successfully in 90 msec.
```

Total rows: 1 Query complete 00:00:00.090

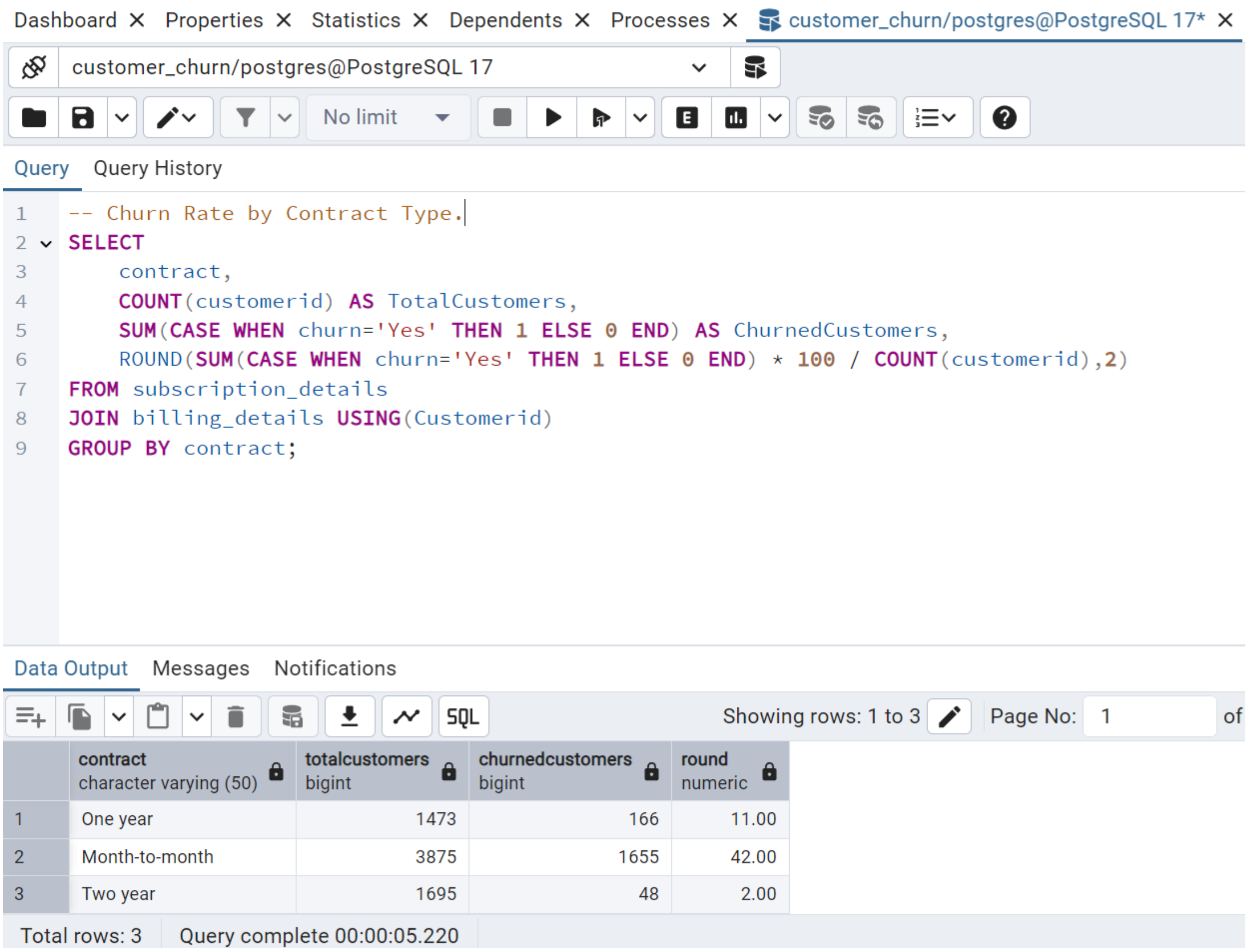


```
-- Calculate average monthy charges.
SELECT
    ROUND(AVG(monthlycharges),2) AS AVGMonthlyCharges
FROM billing_details;
```

	avgmonthlycharges numeric
1	64.76

Total rows: 1 Query complete 00:00:00.138

Ques 4: Churn Rate by Contract Type.



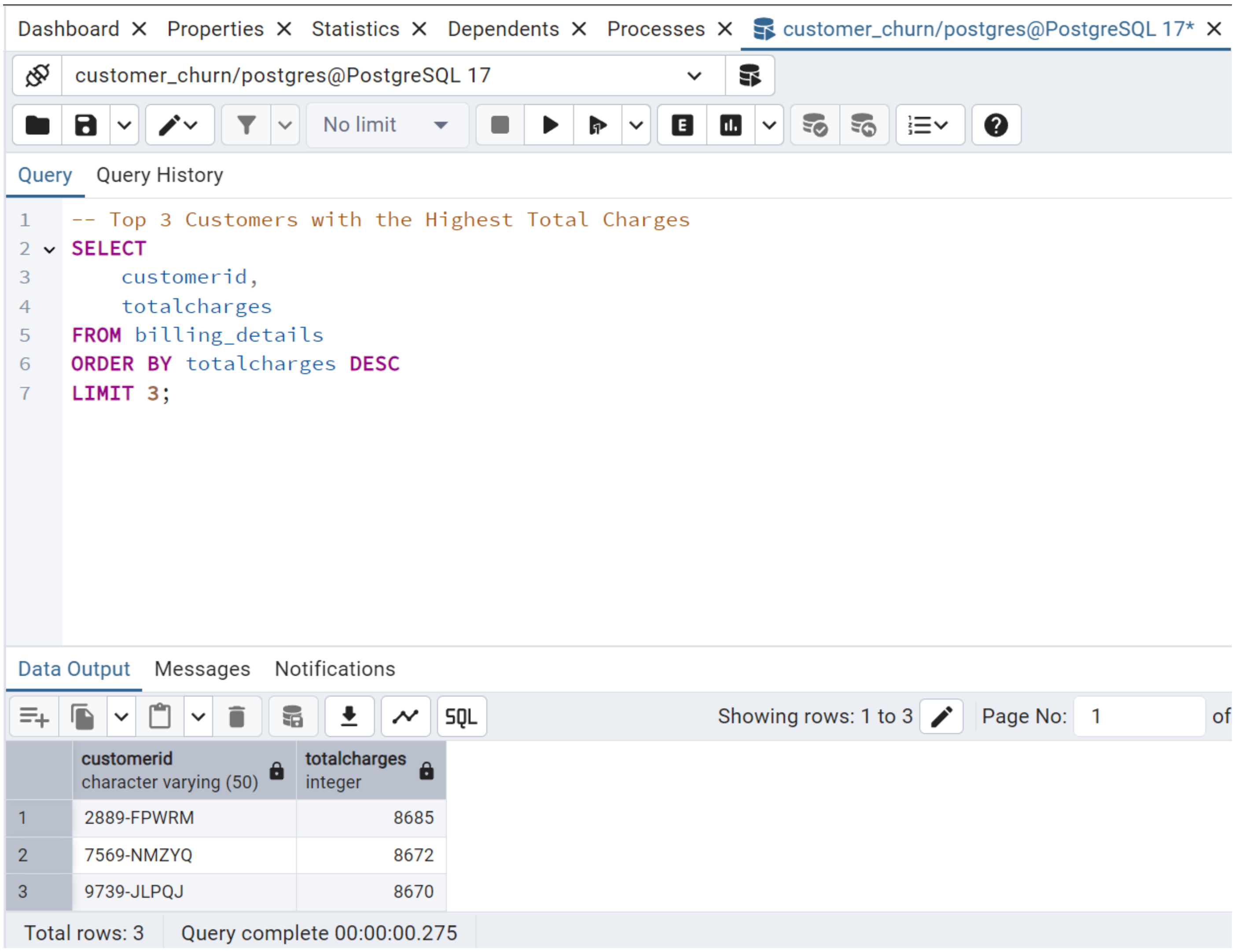
```sql
-- Churn Rate by Contract Type.
SELECT
    contract,
    COUNT(customerid) AS TotalCustomers,
    SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
    ROUND(SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END) * 100 / COUNT(customerid),2)
FROM subscription_details
JOIN billing_details USING(Customerid)
GROUP BY contract;
```

Showing rows: 1 to 3

	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

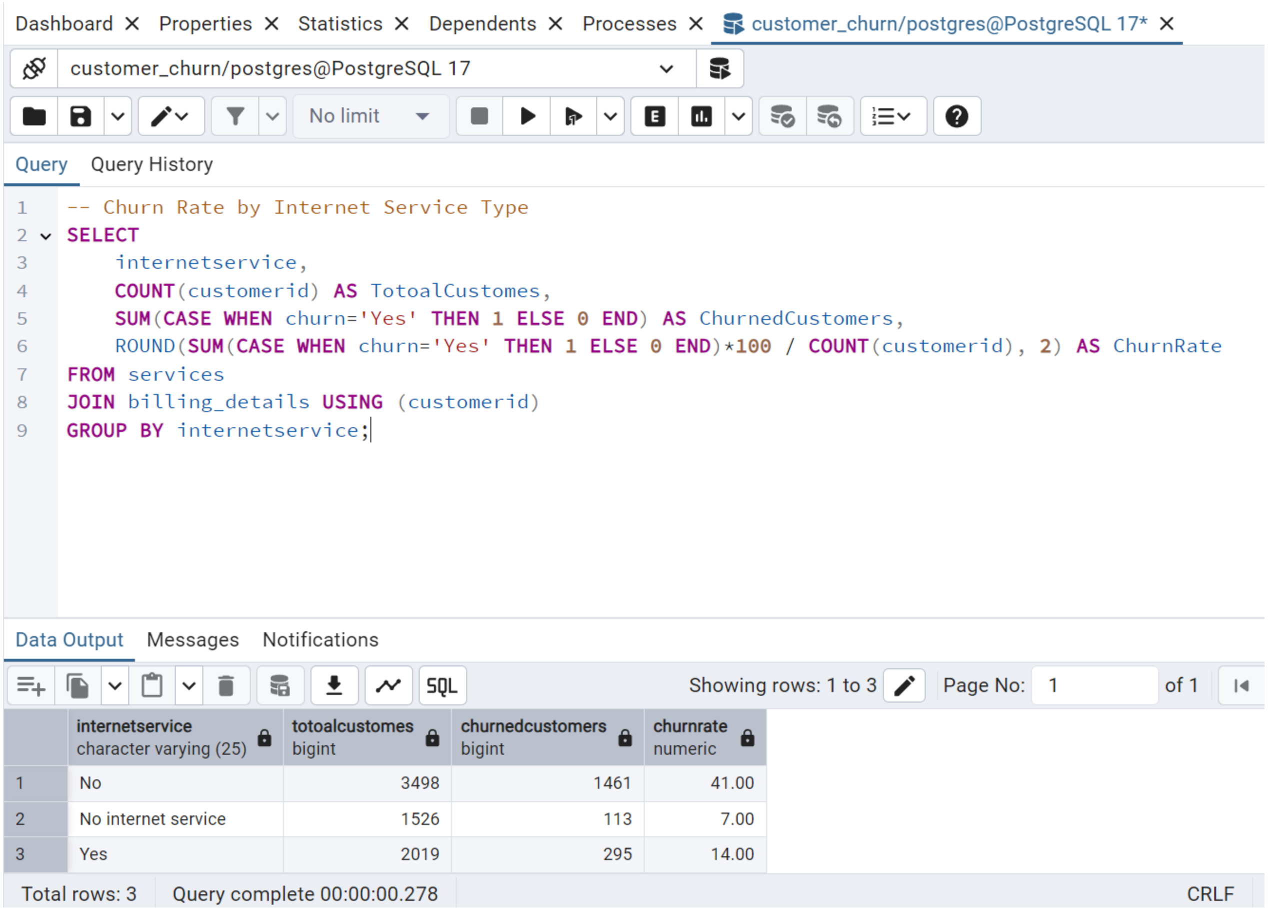


```
-- Top 3 Customers with the Highest Total Charges
SELECT
    customerid,
    totalcharges
FROM billing_details
ORDER BY totalcharges DESC
LIMIT 3;
```

	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

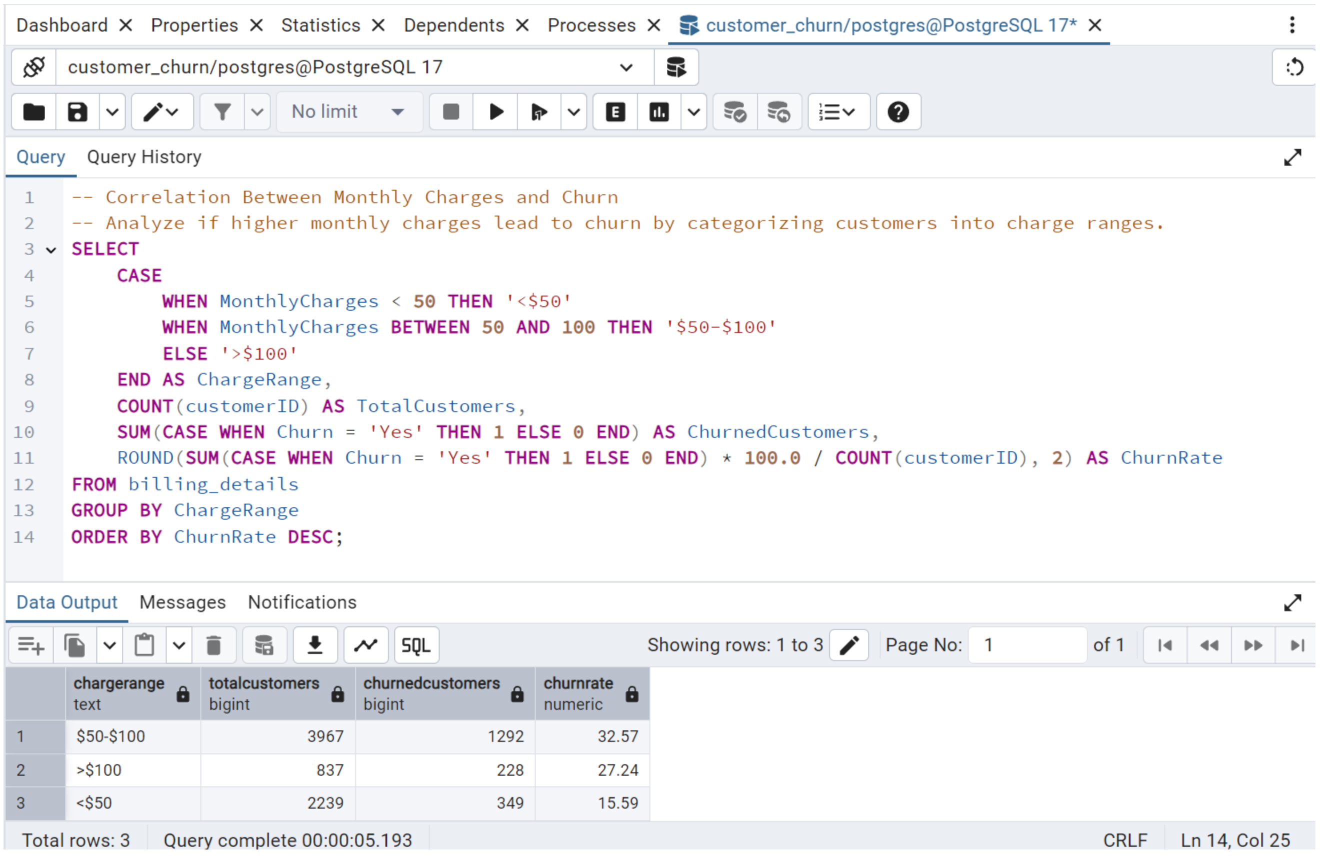

Dashboard ✕ Properties ✕ Statistics ✕ Dependents ✕ Processes ✕ customer_churn/postgres@PostgreSQL 17* ✕
customer_churn/postgres@PostgreSQL 17
No limit
Query Query History
Data Output Messages Notifications
Showing rows: 1 to 3 Page No: 1 of 1

```sql
-- Churn Rate by Internet Service Type
SELECT
    internetservice,
    COUNT(customerid) AS TotoalCustomes,
    SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
    ROUND(SUM(CASE WHEN churn='Yes' THEN 1 ELSE 0 END)*100 / COUNT(customerid), 2) AS ChurnRate
FROM services
JOIN billing_details USING (customerid)
GROUP BY internetservice;
```

	internetservice character varying (25)	totoalcustomes 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.

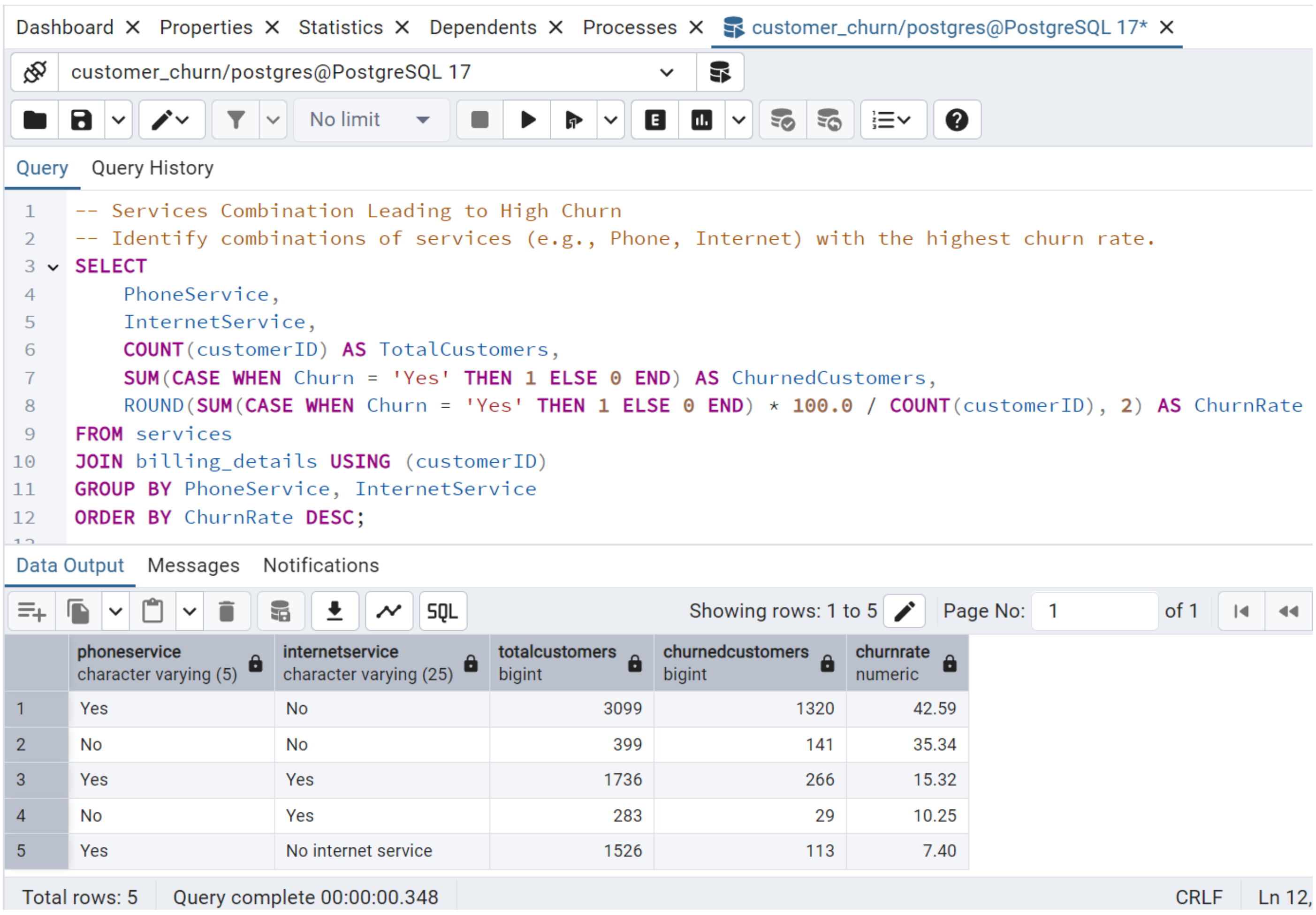

Dashboard ✕ Properties ✕ Statistics ✕ Dependents ✕ Processes ✕ customer_churn/postgres@PostgreSQL 17* ✕
customer_churn/postgres@PostgreSQL 17
No limit
Query Query History
Data Output Messages Notifications
Showing rows: 1 to 3 Page No: 1 of 1

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

	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.



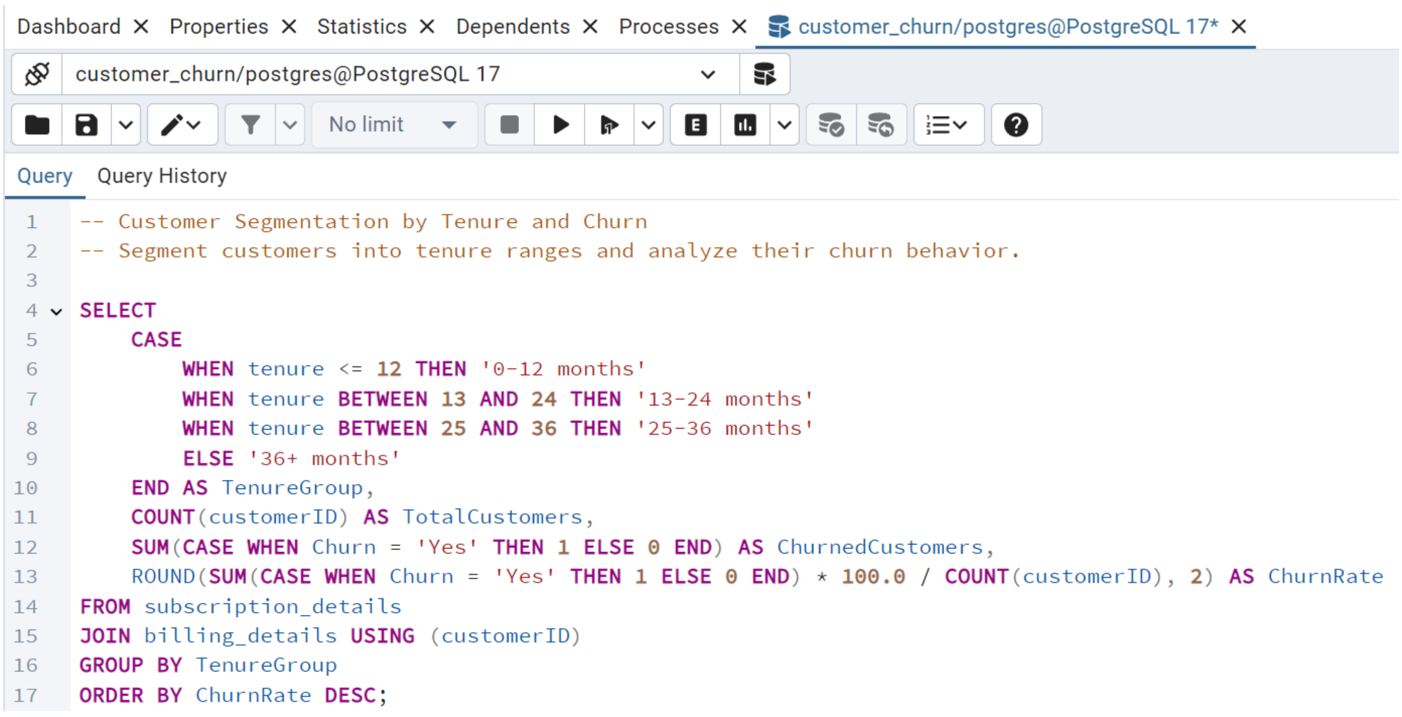
```
-- Services Combination Leading to High Churn
-- Identify combinations of services (e.g., Phone, Internet) with the highest churn rate.
SELECT
    PhoneService,
    InternetService,
    COUNT(customerID) AS TotalCustomers,
    SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
 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

	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

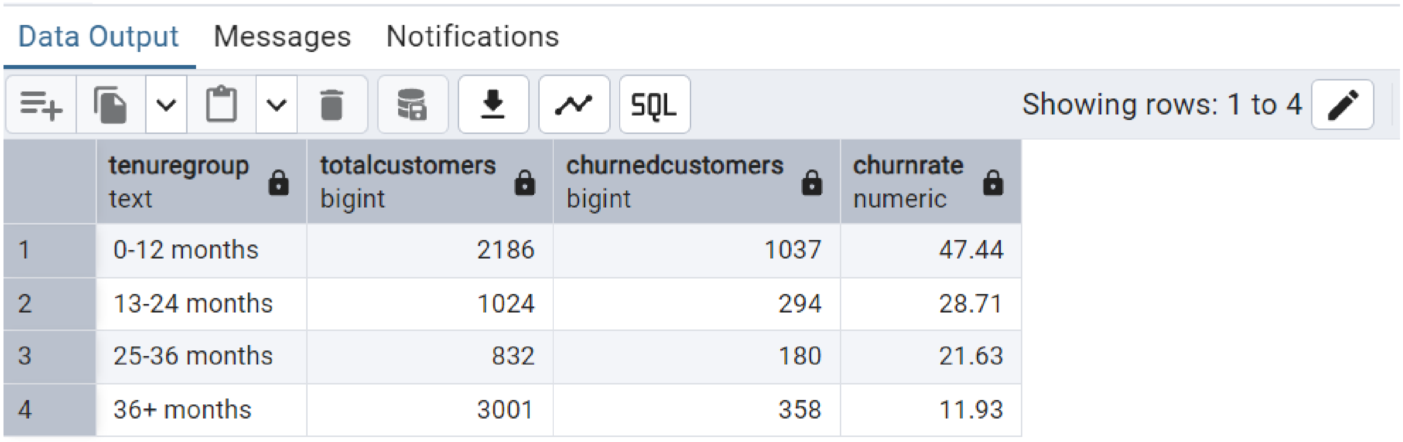
Total rows: 5 Query complete 00:00:00.348 CRLF Ln 12,

Ques 9: Customer Segmentation by Tenure and Churn

Segment customers into tenure ranges and analyze their churn behavior.


Dashboard
Properties
Statistics
Dependents
Processes
customer_churn/postgres@PostgreSQL 17*
customer_churn/postgres@PostgreSQL 17
No limit
Query
Query History

```
-- Customer Segmentation by Tenure and Churn
-- Segment customers into tenure ranges and analyze their churn behavior.

SELECT
    CASE
        WHEN tenure <= 12 THEN '0-12 months'
        WHEN tenure BETWEEN 13 AND 24 THEN '13-24 months'
        WHEN tenure BETWEEN 25 AND 36 THEN '25-36 months'
        ELSE '36+ months'
    END AS TenureGroup,
    COUNT(customerID) AS TotalCustomers,
    SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS ChurnedCustomers,
    ROUND(SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 100.0 / COUNT(customerID), 2) AS ChurnRate
FROM subscription_details
JOIN billing_details USING (customerID)
GROUP BY TenureGroup
ORDER BY ChurnRate DESC;
```



	tenuregroup text	totalcustomers bigint	churnedcustomers bigint	churnrate numeric
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