WGU D211 PA

Andrew Shrestha

Part I

A. Interactive Dashboard

https://public.tableau.com/app/profile/andrew.shrestha/viz/Dashboard2_16343507815340/Dashboard1?publish=yes

1. Datasets Utilized in the Dashboard

Churn_Clean.CSV (Uploaded)

&

Telco_customer_churn.CSV(Uploaded)
https://www.kaggle.com/yeanzc/telco-customer-churn-ibm-dataset

2. Installations

No Installations are necessary when viewing/ interacting with the Tableau created Dashboard. All that is needed is to follow the links above in web browser of your choice.

3. Navigating Interactive Dashboard

Churn Category Option

- Ability to view all chart statistics with regards to whether a customer churned or not. This is done by only highlighting the Churn option across the whole dashboard, making it easier to visualize the relevant information.

Gender Category Option

- Ability to view gender segmentation in the Gender Churn Ratio graphic. The gender is segmented into three categories: Male, Female, and Non-Binary. This also utilizes Tableau's highlighting feature with the relevant information.

State Category Option

- Ability to view State specific/group specific revenue in the Monthly Revenue by U.S State graphic. Highlighting feature is utilized, as it will only show the specific state of interest along with the Revenue values associated with this.

Clickable Options

- Although not as interactive, the interactive dashboard also includes clickable options stating the top reasons for customer churn. This can enhance insights into why this is the case and give companies a better understanding on solid solutions they are able to fix.

Combination of Category Options

- Ability to combine the category view from the above three options to narrow down specifics in terms of further filtering information.

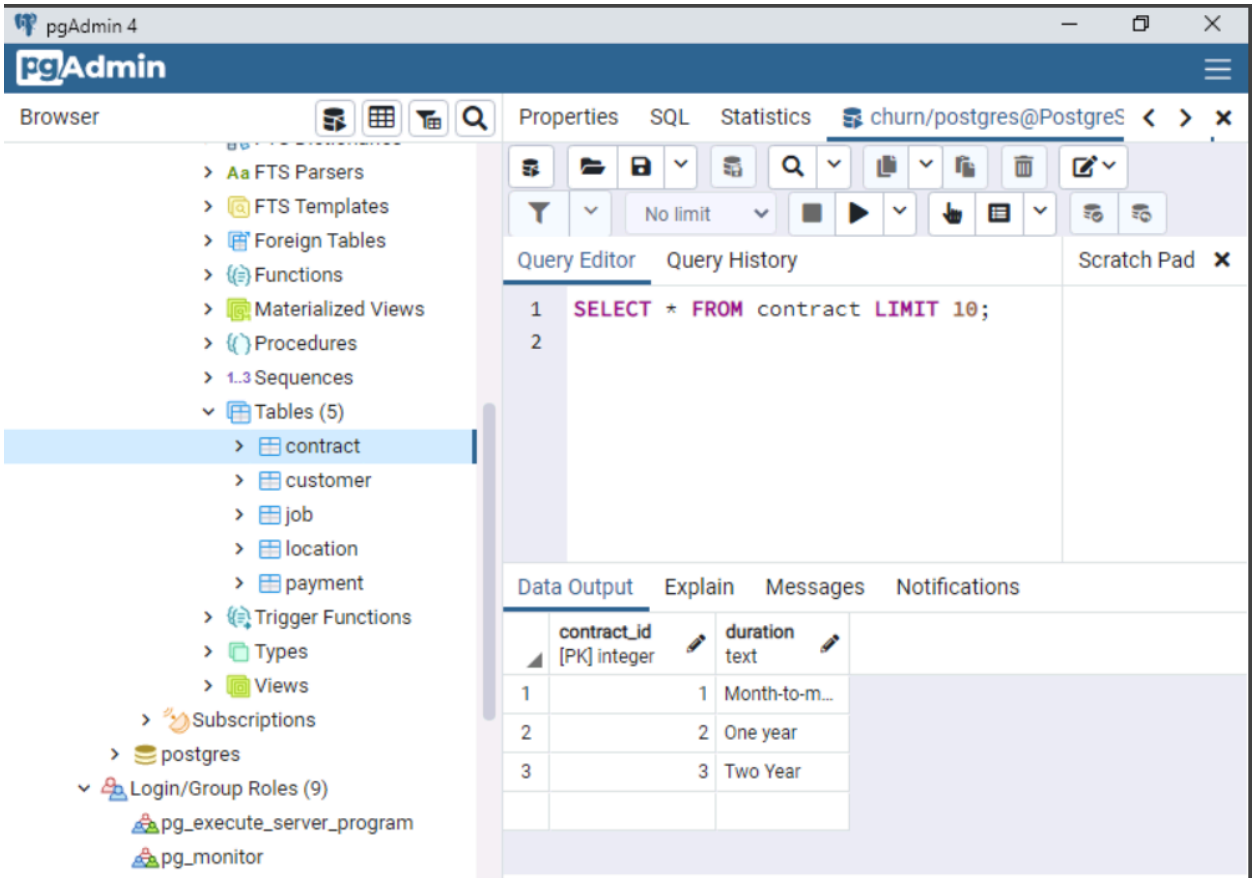
4. Copy of SQL Codes Supporting Dashboard

Loading the Database Tables to get Quick Summary of Data

With the use of pgadmin, we are able to execute SQL Commands to get a better grasp of the data that we are working on with regards to the Churn Data Base tables given.

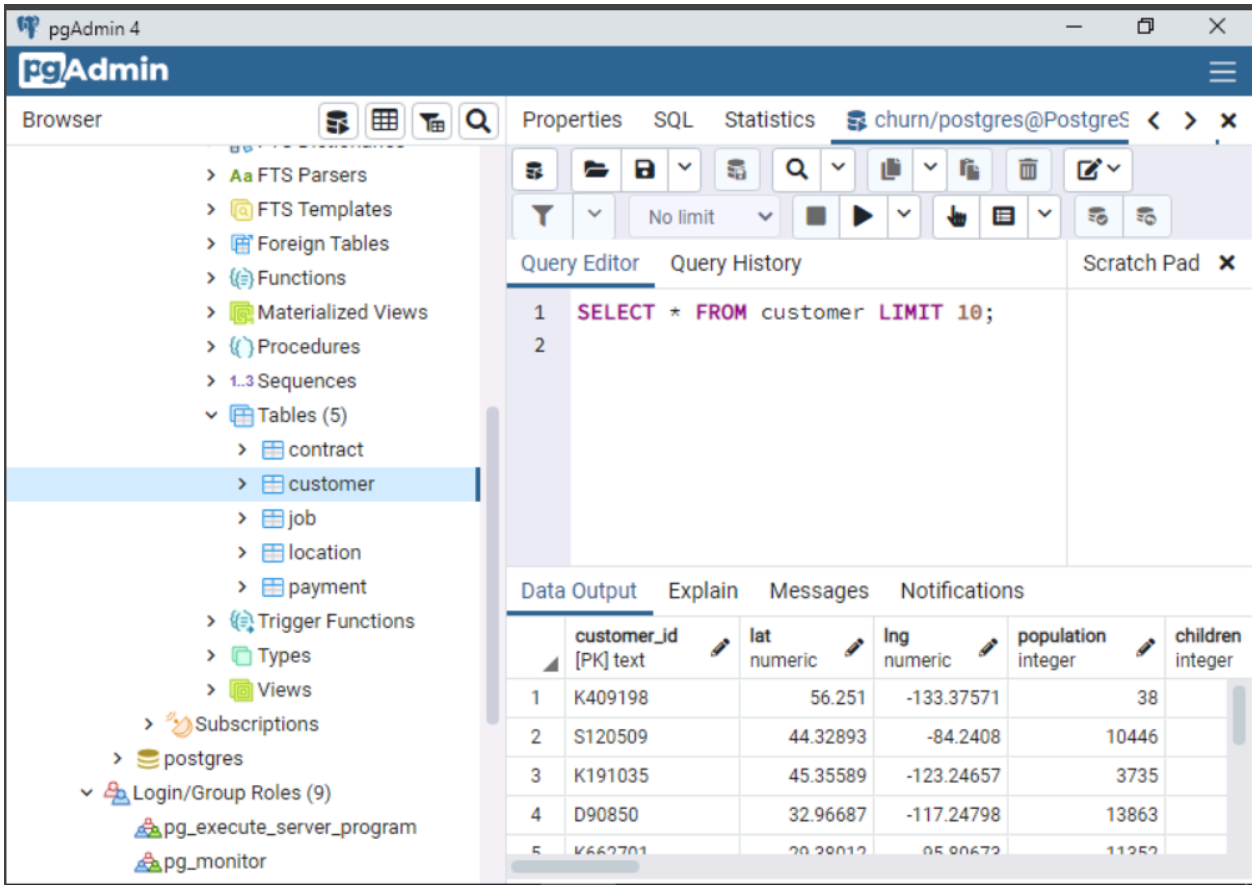
1) Contract Table

```
SELECT * FROM contract LIMIT 10;
```


pgAdmin 4
pgAdmin
Browser
Properties
SQL
Statistics
churn/postgres@PostgreS
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (5)
contract
customer
job
location
payment
Trigger Functions
Types
Views
Subscriptions
postgres
Login/Group Roles (9)
pg_execute_server_program
pg_monitor
No limit
Query Editor
Query History
Scratch Pad
SELECT * FROM contract LIMIT 10;
Data Output
Explain
Messages
Notifications
contract_id [PK] integer
duration text
1 1 Month-to-m...
2 2 One year
3 3 Two Year

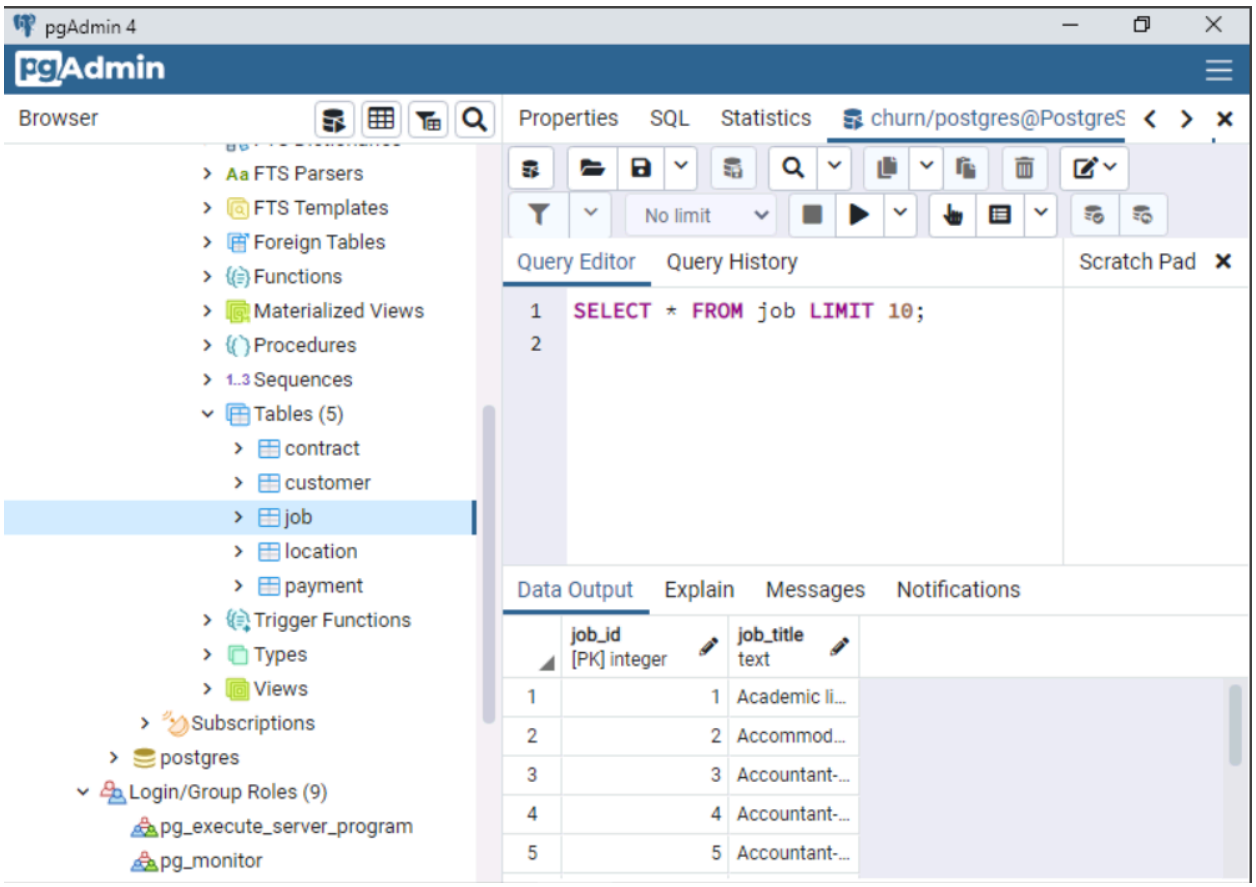
2) Customer Table

SELECT * FROM customer LIMIT 10;



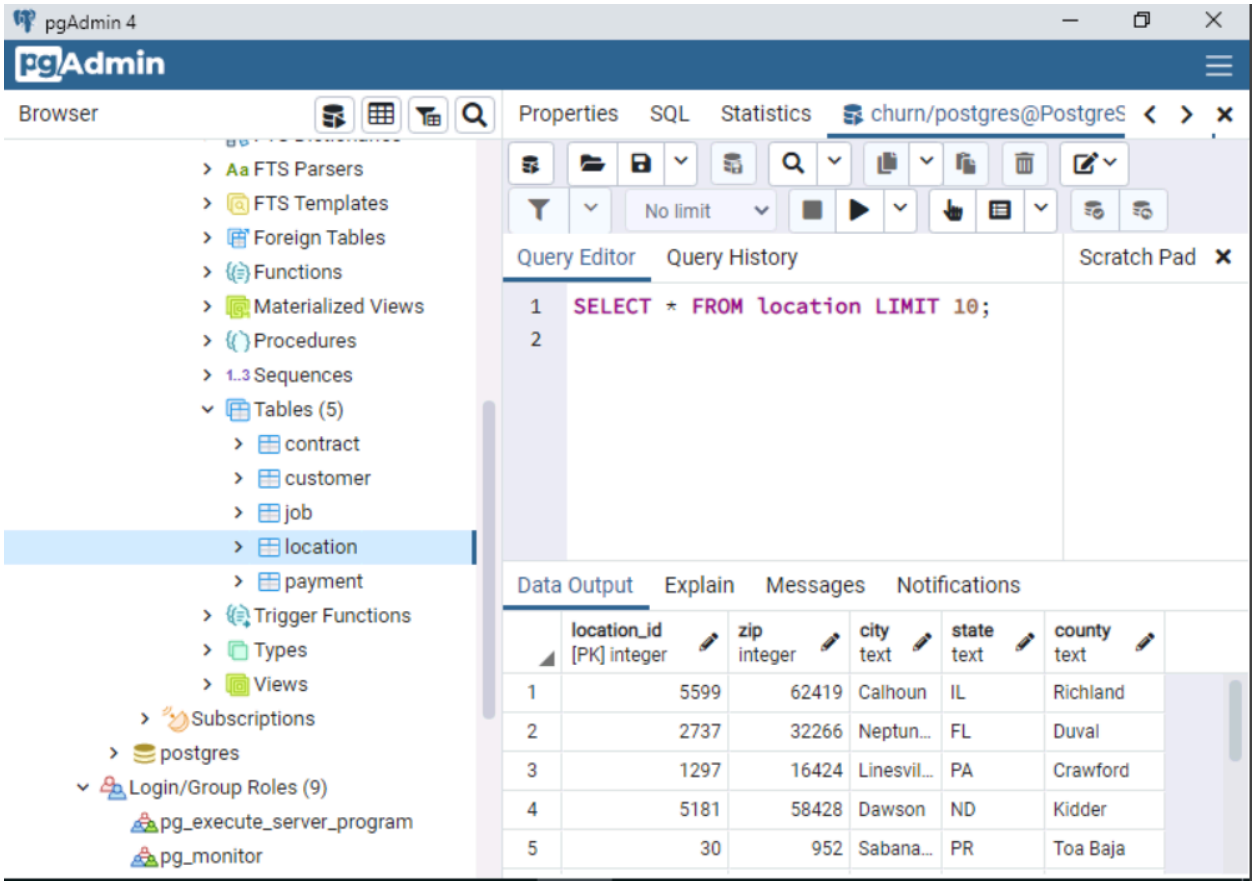
3) Job Table

SELECT * FROM job LIMIT 10;



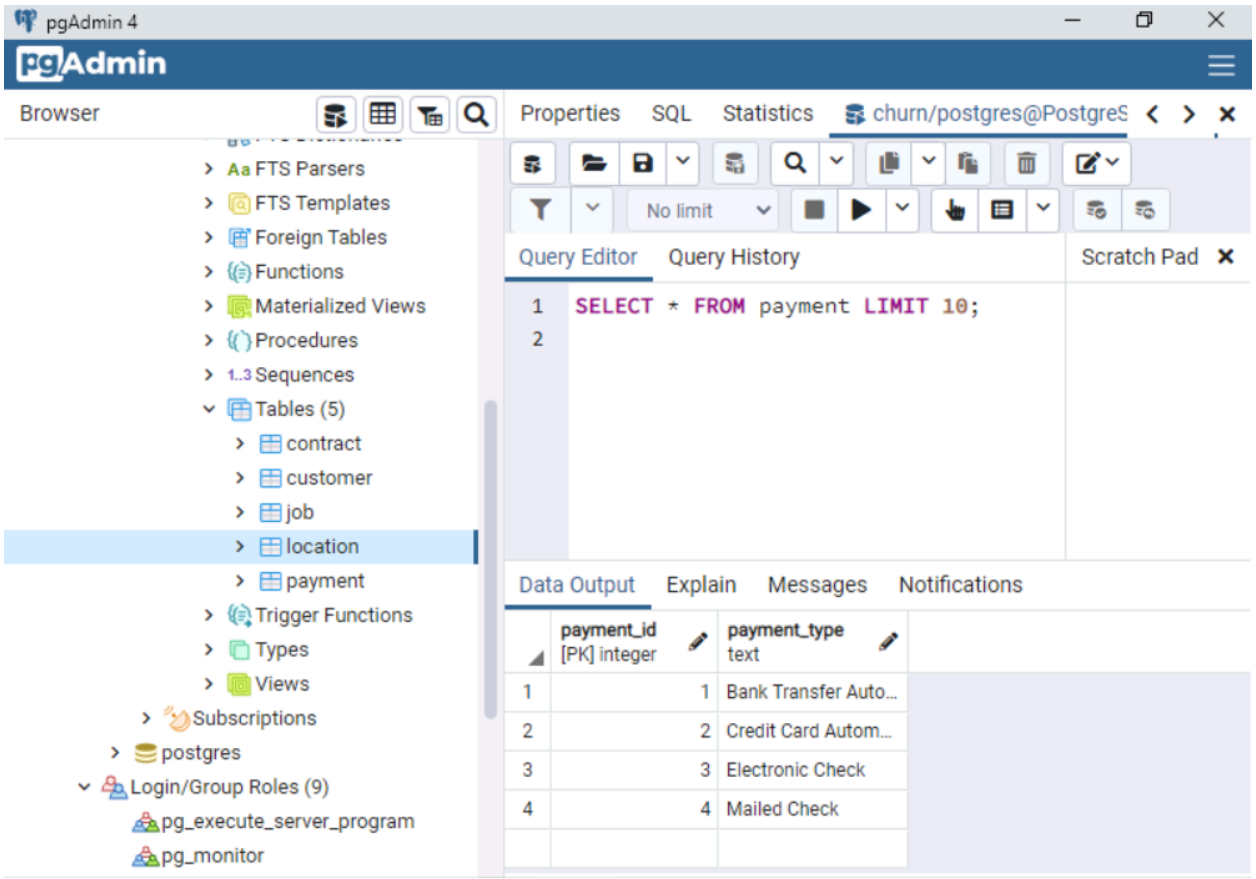
4) Location Table

SELECT * FROM location LIMIT 10;



5) Payment Table

SELECT * FROM payment LIMIT 10;



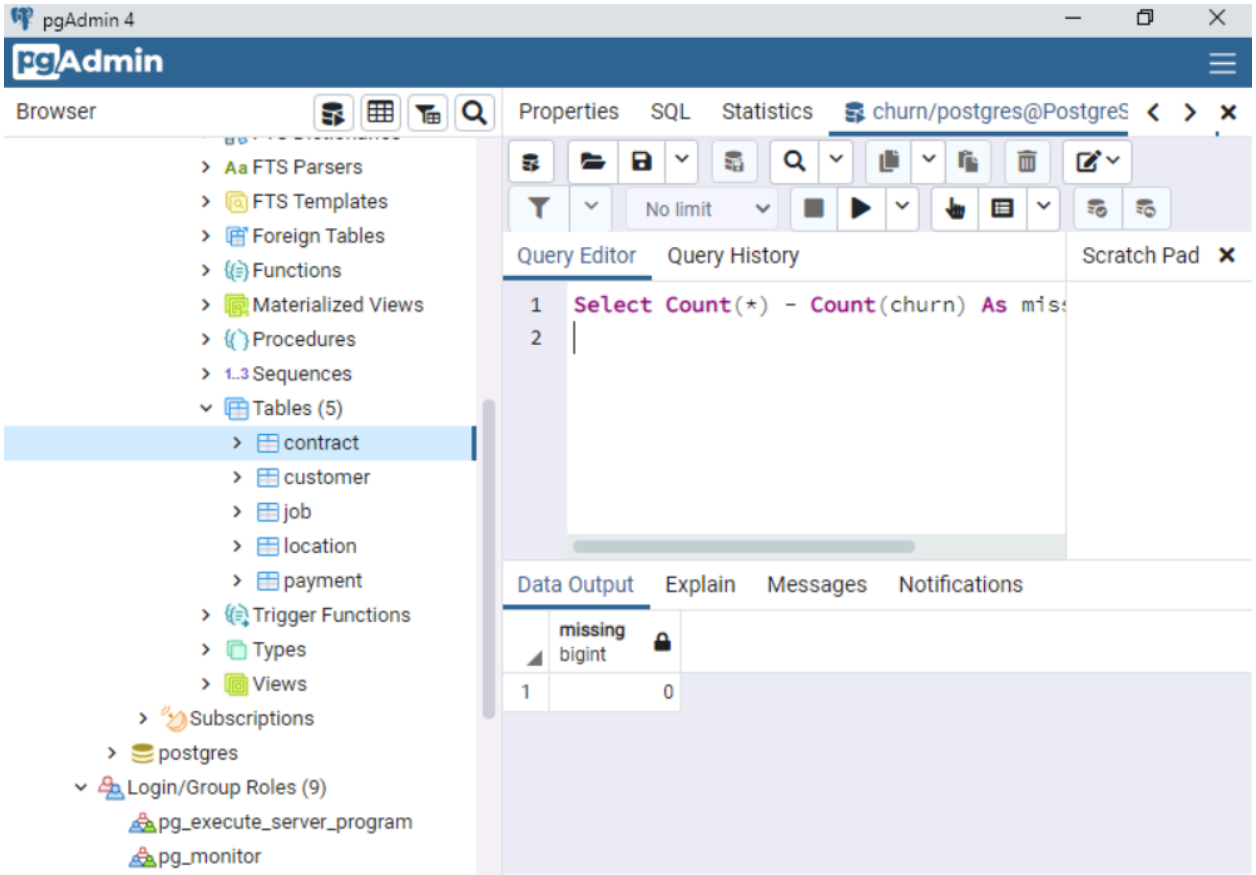
Ensuring Missing Variables are Checked and Considered

1) Contract Table

For the Contract table, since there are only 3 data points, it is easy for us to eyeball that there are not missing data in the dataset for the associated variables. Thus, no SQL Query is needed to check this.

2) Customer Table

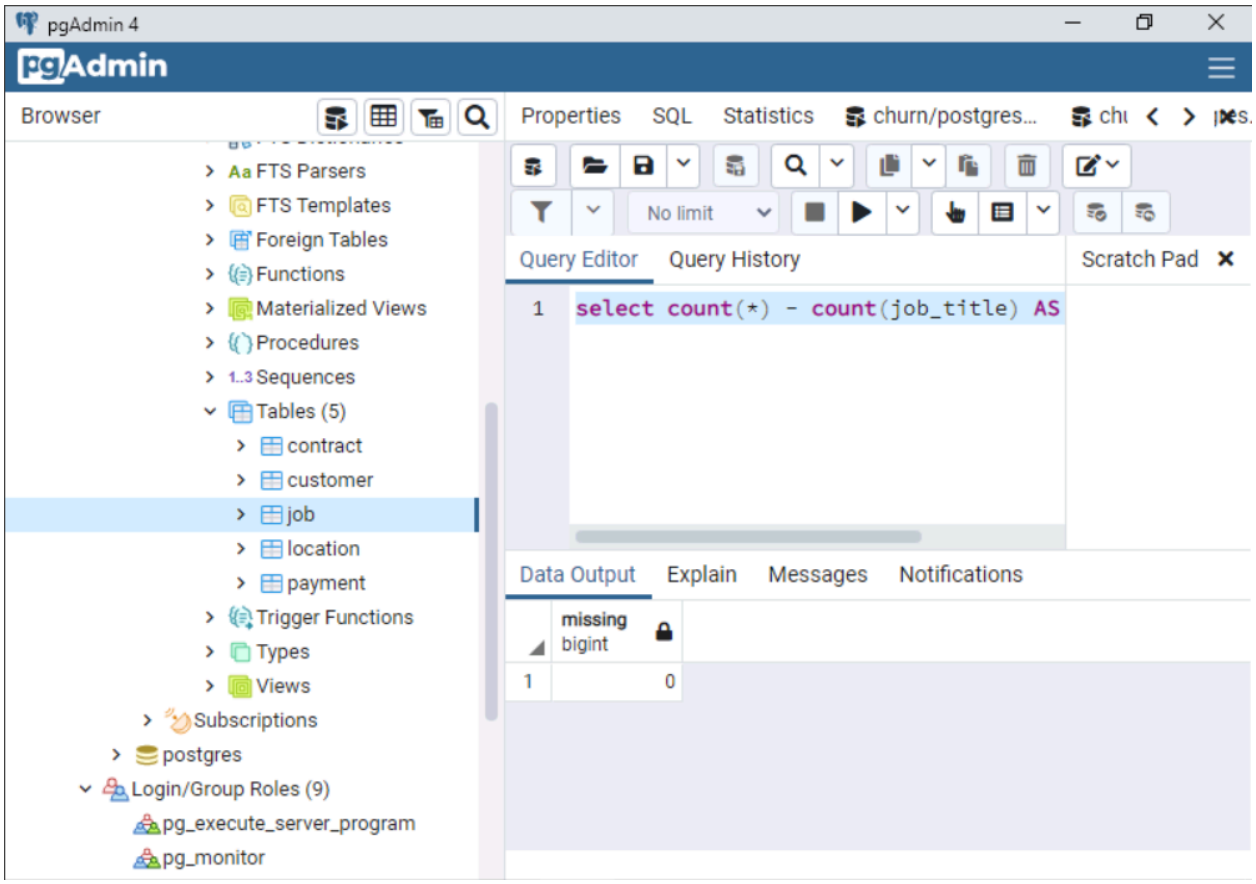
SELECT COUNT(*) – COUNT(churn) AS missing FROM customer;



From the picture above, we see that out of the 10,000 observations, there are currently no missing values for this table.

3) Job Table

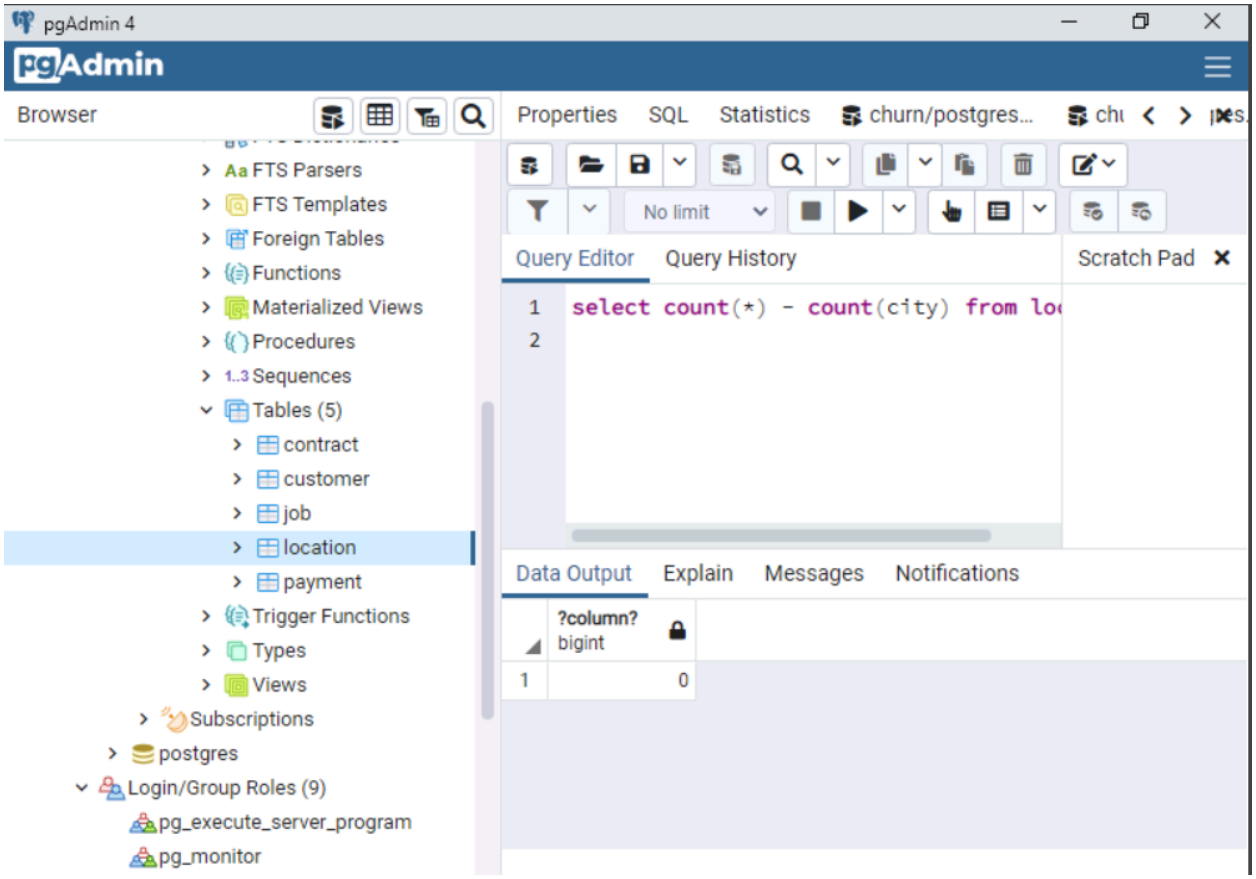
SELECT COUNT(*) – COUNT(job_title) AS missing FROM job;



From the picture above, we see that out of the 639 observations, there are currently no missing values for this table.

4) Location Table

SELECT COUNT(*) – COUNT(city) AS missing FROM location;



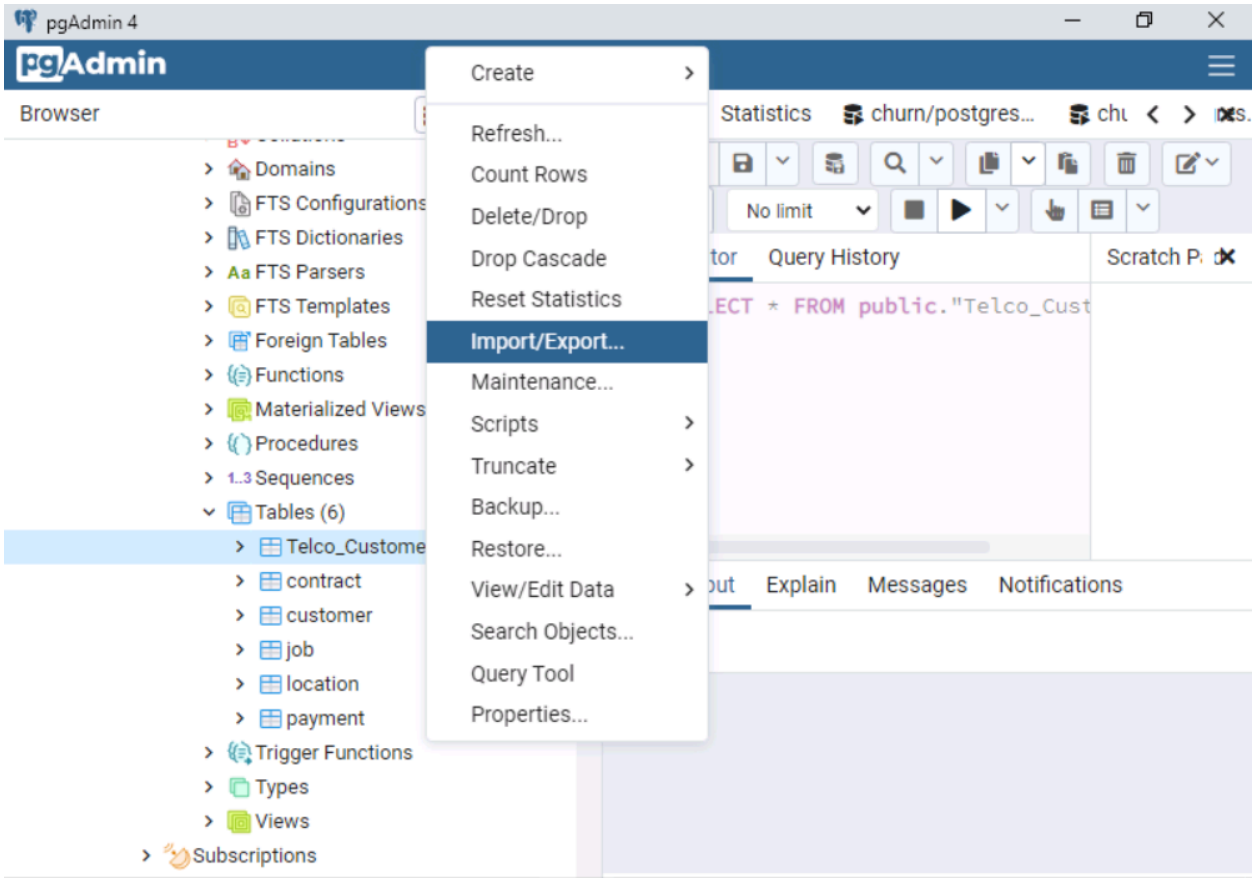
From the picture above, we see that out of the 8583 observations, there are currently no missing values for this table.

5) Payment Table

For the Payment table, since there are only 4 data points, it is easy for us to eyeball that there are not missing data in the dataset for the associated variables. Thus, no SQL Query is needed to check this.

Importing our secondary data set "Telco_Customer_Churn" to the database

1) This was done via using the 'create table' option under the table section of the pgadmin, then completing the columns and needed and manually importing the CSV file.



2)

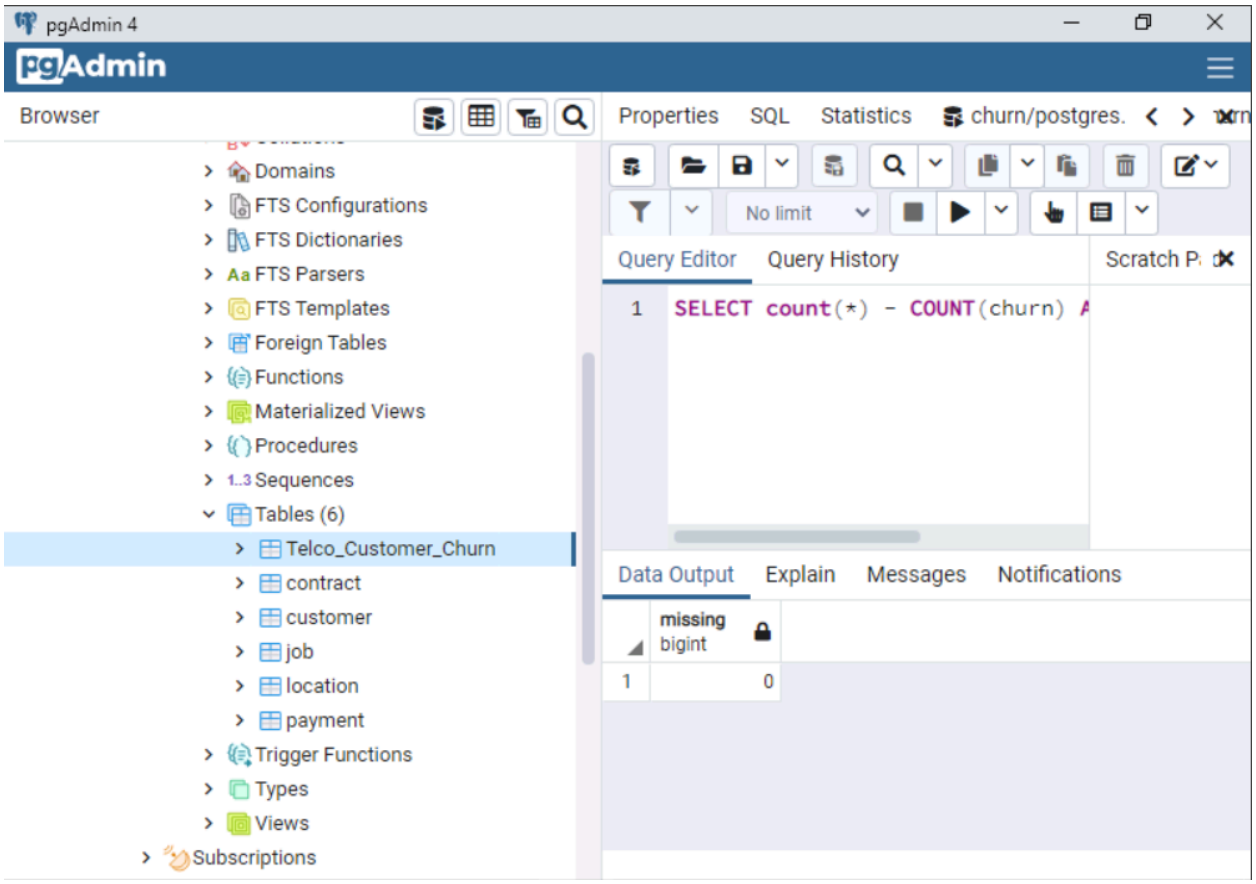
SELECT * FROM Telco_Customer_Churn LIMIT 10;

	index	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
2	1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No
3	2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
4	3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No
5	4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No
6	5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No
7	6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes
8	7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	No
9	8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No
10	9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes

3)

We then check for missing values for consistency purposes and ensuring accurate data.

SELECT (*) -COUNT (churn) AS missing FROM Telco_Customer_Churn;



Join the two tables together via finding a primary key

SELECT customer.gender, Telco_Customer_Churn.gender FROM customer INNER JOIN Telco_Customer_Churn ON customer.gender = Telco_Customer_Churn.gender;

```
select customer.gender, telco_customer_churns.gender
from customer
inner join telco_customer_churns
on customer.gender = telco_customer_churns.gender;
```

	gender	gender
1	Male	Male
2	Male	Male
3	Male	Male
4	Male	Male
5	Male	Male

```
Execution finished without errors.
Result: 34392120 rows returned in 4623ms
At line 1:
select customer.gender, telco_customer_churns.gender
from customer
inner join telco_customer_churns
on customer.gender = telco_customer_churns.gender;
```

2) Ensure dimensions are accurate

select count(c.customer_id) AS the_company, count(t.customerID) AS other_companies from customer AS C Inner join telco_customer_churns AS t on c.churn = t.churn;

```
select count(c.customer_id) AS the_company, count(t.customerID) AS other_companies from customer AS C Inner join telco_customer_churns AS t on c.chur
```

	the_company	other_companies
1	42981750	42981750

```
Execution finished without errors.
Result: 1 rows returned in 38882ms
At line 1:
select count(c.customer_id) AS the_company, count(t.customerID) AS other_companies from customer AS C Inner join telco_customer_churns AS t on c.churn =
t.churn;
```

3) Initiating some Comparison Queries between the two tables

select customer_id, contract_id from customer where timezone in ('America/California','America/New_York') limit 10;

```
select customer_id, contract_id from customer where timezone in ('America/California','America/New_York') limit 10;
```

	customer_id	contract_id
1	W303516	One year
2	U335188	Month-to-month
3	M716771	Month-to-month
4	I676080	Two Year
5	J980369	Month-to-month

```
Execution finished without errors.
Result: 10 rows returned in 7ms
At line 1:
select customer_id, contract_id from customer where timezone in ('America/California','America/New_York') limit 10;
```

```
select case when income > 90000 then 'High Earners'
when income > 40000 then 'Middle Earners'
else 'Low Earners'
end as social_standing,
count(customer_id) as totals
from customer
group by social_standing
order by totals asc;
```

```sql
select case when income > 90000 then 'High Earners'
when income > 40000 then 'Middle Earners'
else 'Low Earners'
end as social_standing,
count(customer_id) as totals
from customer
group by social_standing
order by totals asc;
```

	social_standing	totals
1	High Earners	594
2	Middle Earners	3423
3	Low Earners	5983

```
Execution finished without errors.
Result: 3 rows returned in 55ms
At line 1:
select case when income > 90000 then 'High Earners'
when income > 40000 then 'Middle Earners'
else 'Low Earners'
end as social_standing,
count(customer_id) as totals
from customer
group by social_standing
order by totals asc;
```