

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/yeancz/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

- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Procedures
- > 1.3 Sequences
- ▼ Tables (5)
 - > contract
 - > customer
 - > job
 - > location
 - > payment
- > Trigger Functions
- > Types
- > Views
- > Subscriptions
- > postgres
- ▼ Login/Group Roles (9)
 - pg_execute_server_program
 - pg_monitor

Properties SQL Statistics churn/postgres@PostgreS

Query Editor Query History Scratch Pad

```
1 SELECT * FROM contract LIMIT 10;
2
```

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;

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Browser' pane displays a tree view of the database structure. The 'customer' table is selected under the 'Tables (5)' folder. The main pane is divided into three sections: 'Query Editor', 'Query History', and 'Scratch Pad'. The 'Query Editor' contains the SQL query: `SELECT * FROM customer LIMIT 10;`. Below the query editor, the 'Data Output' tab is active, displaying a table with 5 rows and 6 columns: `customer_id` [PK] text, `lat` numeric, `lng` numeric, `population` integer, and `children` integer. The data is as follows:

	customer_id [PK] text	lat numeric	lng numeric	population integer	children integer
1	K409198	56.251	-133.37571	38	
2	S120509	44.32893	-84.2408	10446	
3	K191035	45.35589	-123.24657	3735	
4	D90850	32.96687	-117.24798	13863	
5	K662701	20.28012	85.80672	11252	

3) Job Table

SELECT * FROM job LIMIT 10;

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Browser' pane shows a tree view of the database structure. The 'job' table is selected under the 'Tables (5)' folder. The main area is divided into several tabs: 'Properties', 'SQL', 'Statistics', 'Query Editor', 'Query History', and 'Scratch Pad'. The 'Query Editor' tab is active, showing the SQL query: `SELECT * FROM job LIMIT 10;`. Below the query editor, the 'Data Output' tab is active, displaying the first 5 rows of the 'job' table. The table has two columns: 'job_id' (integer, primary key) and 'job_title' (text).

job_id	job_title
1	Academic li...
2	Accommod...
3	Accountant-...
4	Accountant-...
5	Accountant-...

4) Location Table

SELECT * FROM location LIMIT 10;

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Browser' pane displays a tree view of the database structure. The 'location' table is selected under the 'Tables (5)' category. The main pane is divided into three sections: 'Query Editor', 'Query History', and 'Scratch Pad'. The 'Query Editor' contains the SQL query: `SELECT * FROM location LIMIT 10;`. Below the query editor, the 'Data Output' tab is active, displaying a table with 7 columns: `location_id` [PK] integer, `zip` integer, `city` text, `state` text, and `county` text. The table contains 5 rows of data.

	location_id [PK] integer	zip integer	city text	state text	county text	
1		5599	62419	Calhoun	IL	Richland
2		2737	32266	Neptun...	FL	Duval
3		1297	16424	Linesvil...	PA	Crawford
4		5181	58428	Dawson	ND	Kidder
5		30	952	Sabana...	PR	Toa Baja

5) Payment Table

SELECT * FROM payment LIMIT 10;

The screenshot shows the pgAdmin 4 web interface. On the left, the 'Browser' pane displays a tree view of the database structure. The 'payment' table is selected under the 'Tables (5)' category. The main pane is divided into two sections: the 'Query Editor' and the 'Data Output' section.

Query Editor: The SQL query `SELECT * FROM payment LIMIT 10;` is entered and executed. The 'Query History' tab is also visible.

Data Output: The results of the query are displayed in a table with two columns: `payment_id` (integer, primary key) and `payment_type` (text). The results show the first 10 rows of the `payment` table.

payment_id	payment_type
1	Bank Transfer Auto...
2	Credit Card Autom...
3	Electronic Check
4	Mailed Check

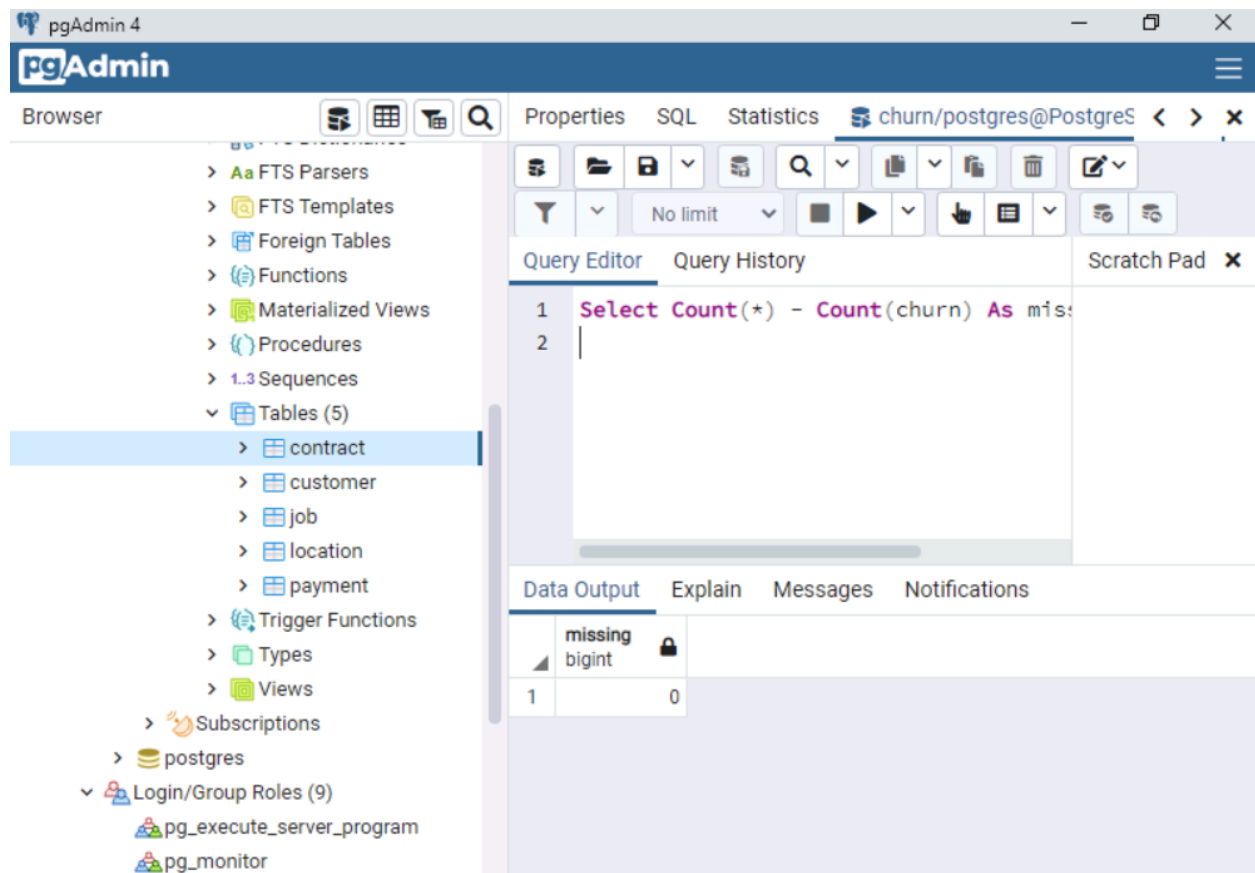
Ensuring Missing Variables are Checked and Considered

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;`



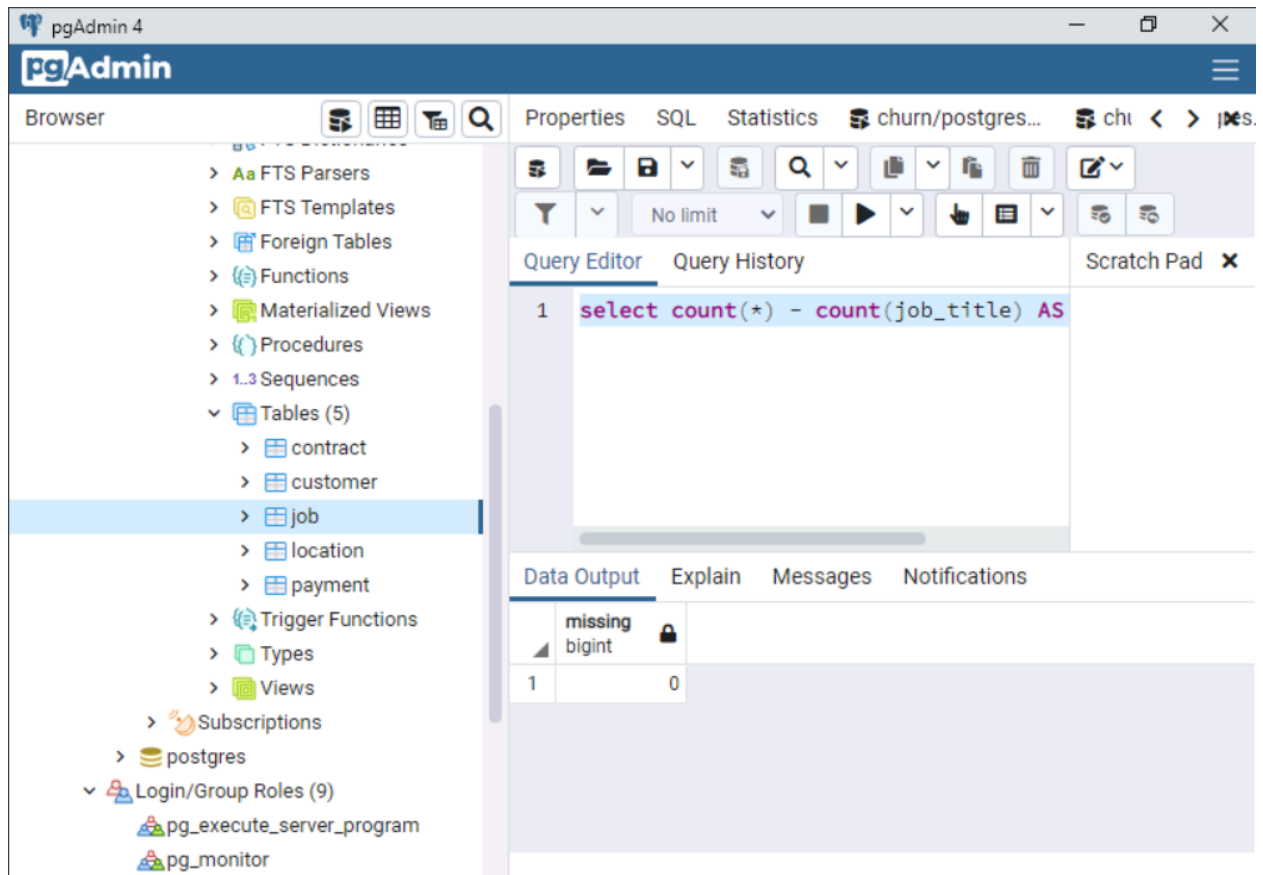
The screenshot shows the pgAdmin 4 web interface. On the left, the 'Browser' pane displays a tree view of the database structure, with 'Tables (5)' expanded and 'contract' selected. The 'Query Editor' pane on the right contains the SQL query: `1 Select Count(*) - Count(churn) As miss`
`2 |`. Below the query editor, the 'Data Output' pane shows the results of the query. The results are displayed in a table with two columns: 'missing' and 'bigint'. The first row shows the value '0'.

missing	bigint
1	0

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

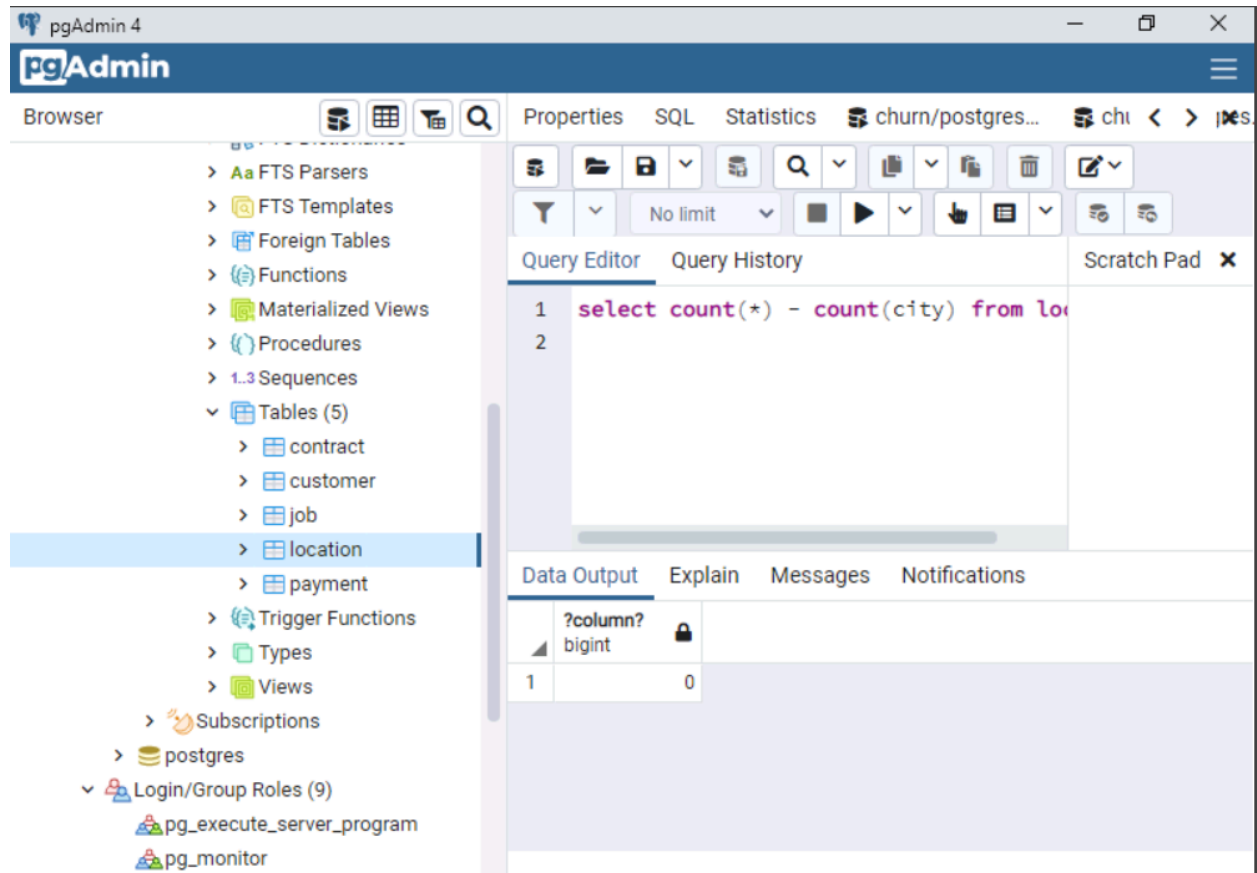
`SELECT COUNT(*) - COUNT(job_title) AS missing FROM job;`



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;



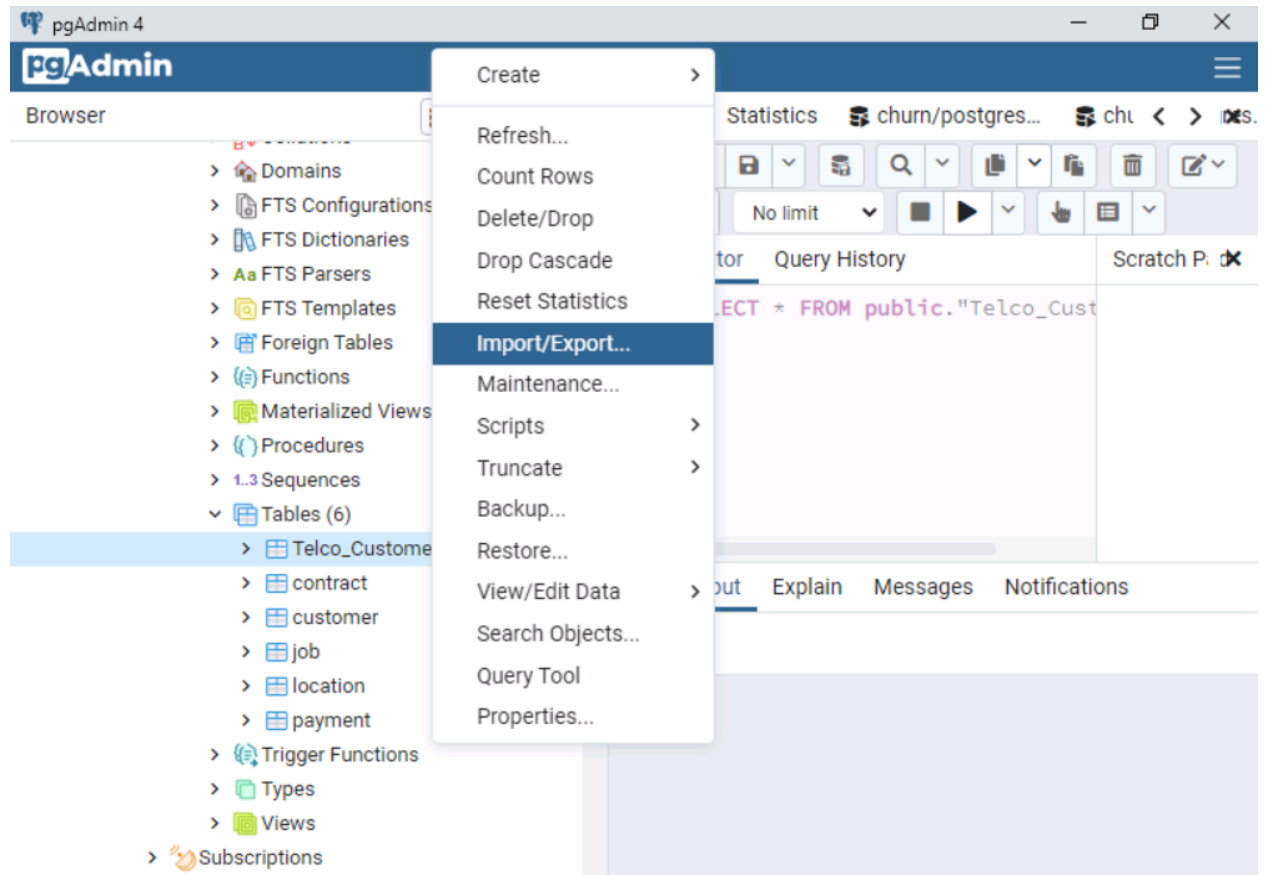
From the picture above, we see that out of the 8583 observations, there are currently no missing values for this table.

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)

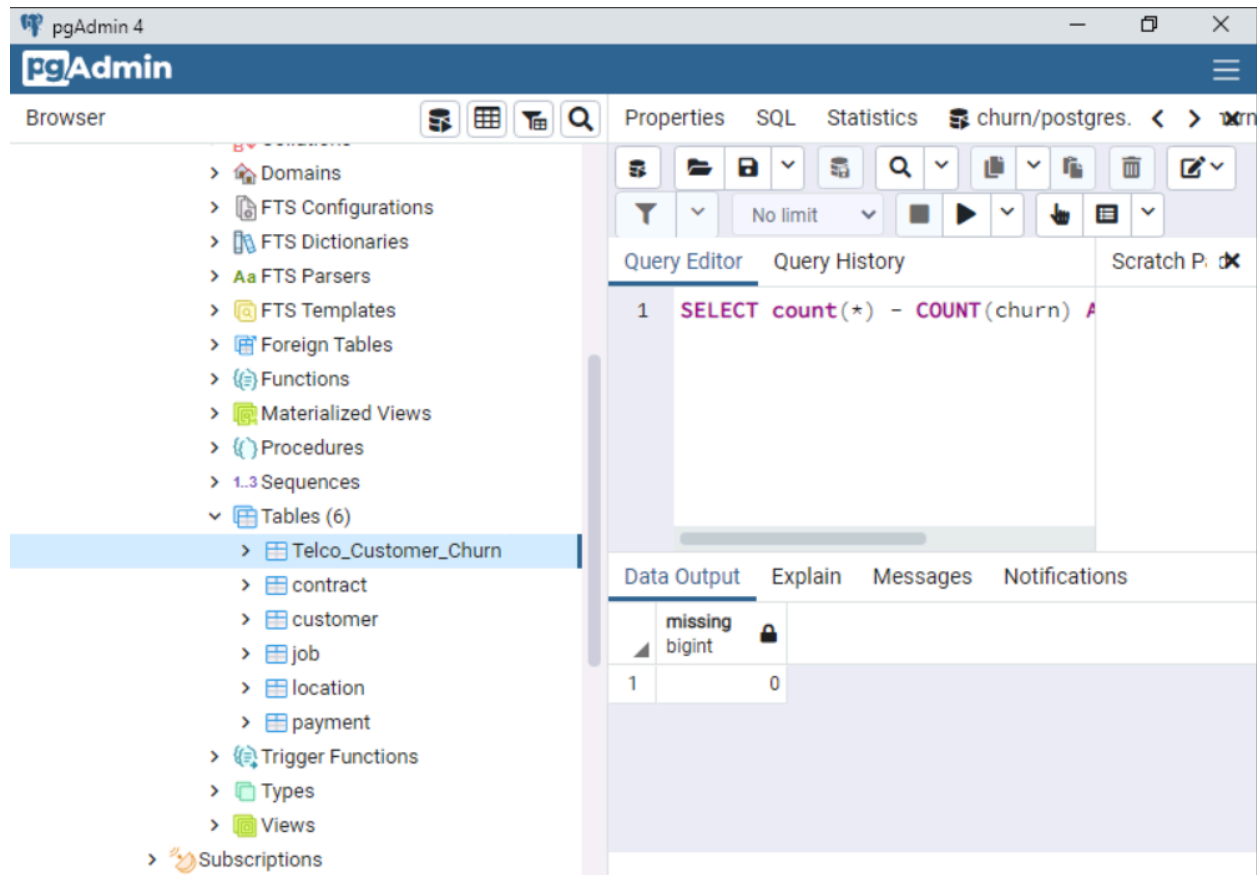
SELECT * FROM Telco_Customer_Churn LIMIT 10;

	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;
```

```

1 select customer.gender, telco_customer_churns.gender
2 from customer
3 inner join telco_customer_churns
4 on customer.gender = telco_customer_churns.gender;
5

```

<

	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;

```

```
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=1
```

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

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

```
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;
```

```

1 select case when income > 90000 then 'High Earners'
2 when income > 40000 then 'Middle Earners'
3 else 'Low Earners'
4 end as social_standing,
5 count(customer_id) as totals
6 from customer
7 group by social_standing
8 order by totals asc;
9

```

<

	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;

