

Lab : Populating a Data Warehouse using PostgreSQL

In this lab you will:

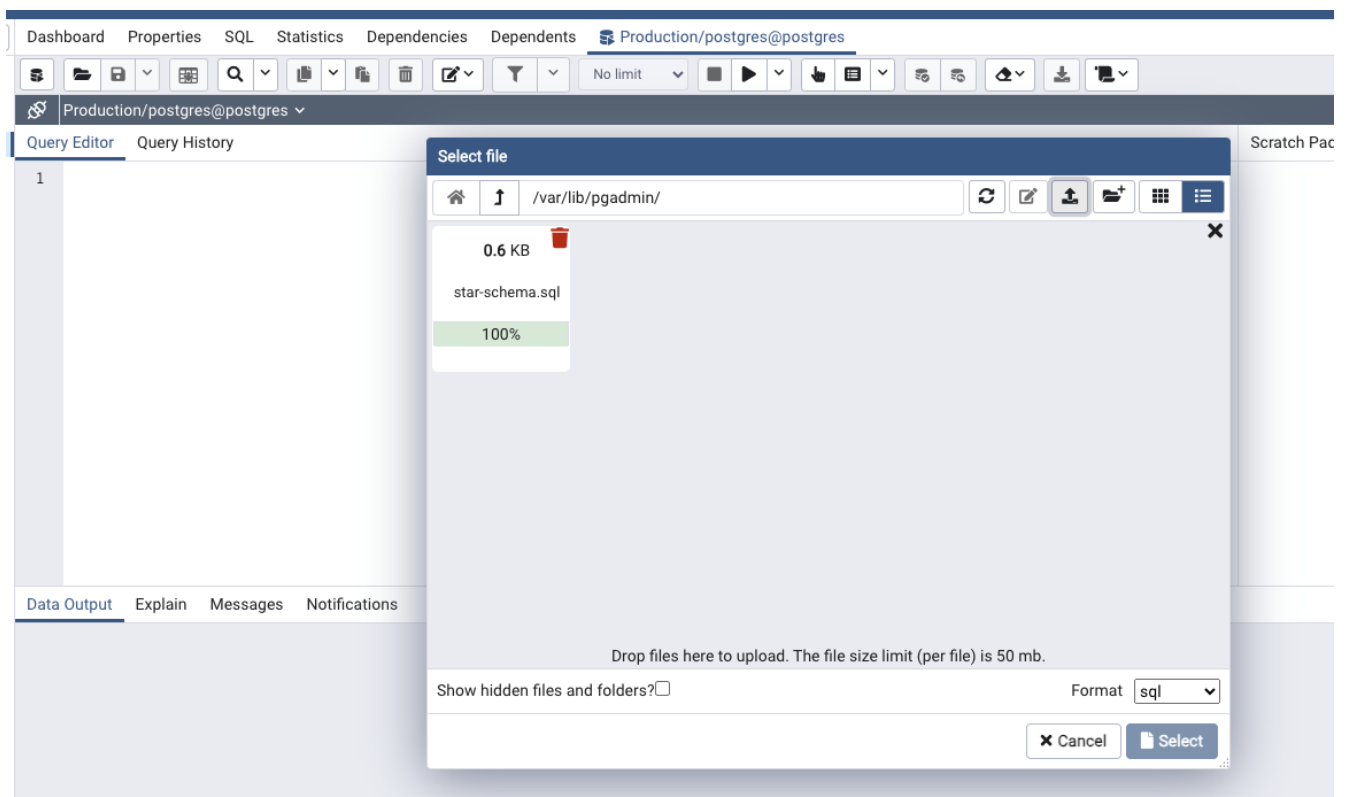
Create production related database and tables in a PostgreSQL instance.
Populate the production data warehouse byloading the tables from Scr

- Using an Instance of pgAdmin (Task A)

Task B: Create tables

Now, that you have your PostgreSQL service active and have created the Production database using pgAdmin, let's go ahead and create a few tables to populate the database and store the data that we wish to eventually upload into it.

1. In the top of the page go to Query tool" and then click on Open File. Next a new page pops up called Select File. Click on Upload icon as shown in the screenshot.



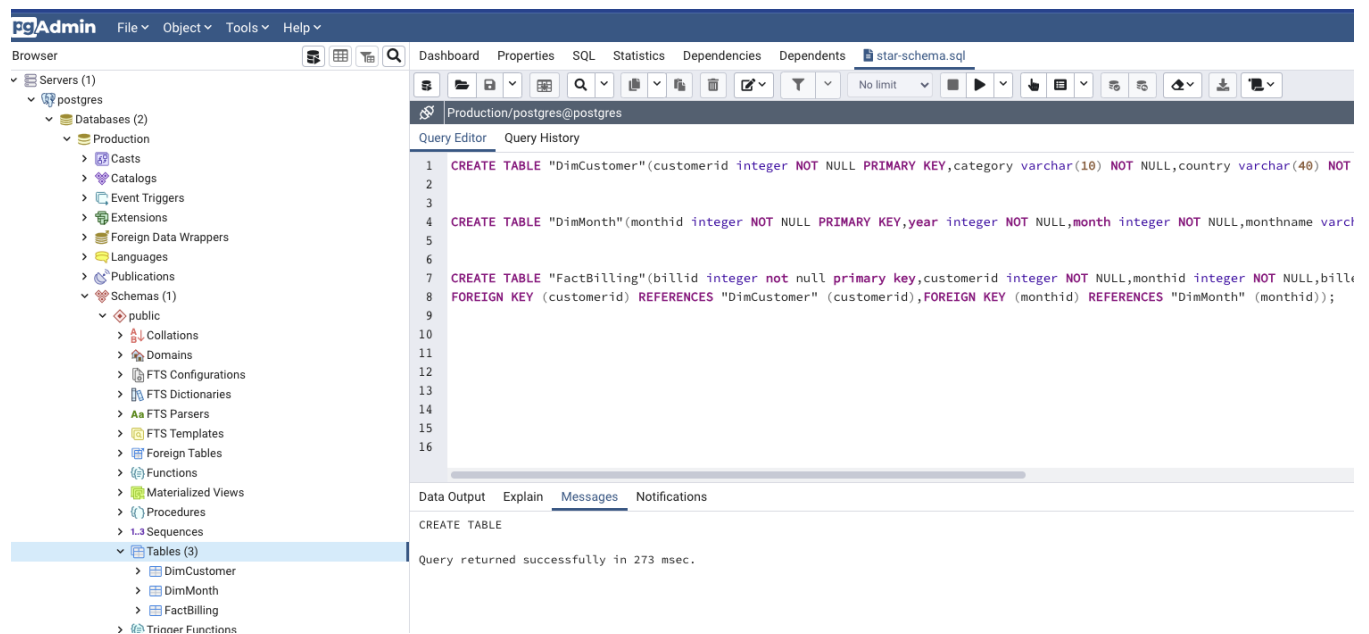
- From the Query tool here we can use the upload icon in the select File Option to upload a sql file under 50 mb

```
1 CREATE TABLE "DimCustomer"(customerid integer NOT NULL PRIMARY KEY,category varchar(10) NOT NULL,country varchar(40) NOT
2
3
4 CREATE TABLE "DimMonth"(monthid integer NOT NULL PRIMARY KEY,year integer NOT NULL,month integer NOT NULL,monthname varchar
5
6
7 CREATE TABLE "FactBilling"(billid integer not null primary key,customerid integer NOT NULL,monthid integer NOT NULL,bille
8 FOREIGN KEY (customerid) REFERENCES "DimCustomer" (customerid),FOREIGN KEY (monthid) REFERENCES "DimMonth" (monthid));
9
10
11
12
13
14
15
16
```

CREATE TABLE

Query returned successfully in 273 msec.

- After clicking “X” on the upload sql file pop-up box, you can then select the uploaded sql file and execute the commands to create your tables
- Note that the query editor is available after selecting a Database with which to run the sql schema file
- Next, right-click on the Production database and click on Refresh option from the dropdown.
 - After the database is refreshed the 3 tables(DimCustomer, DimMonth,FactBilling) are created under the Databases > Production > Schema > Public > Tables.



Task C: Load tables

- With the sql files similar to the schema/table creation, you can then simply insert data

Practice exercises

- Use PGAdmin QUERY Tool for some regular SQL type exploration

Problem 1: Using the PostgreSQL tool, find the count of rows in the table FactBilling

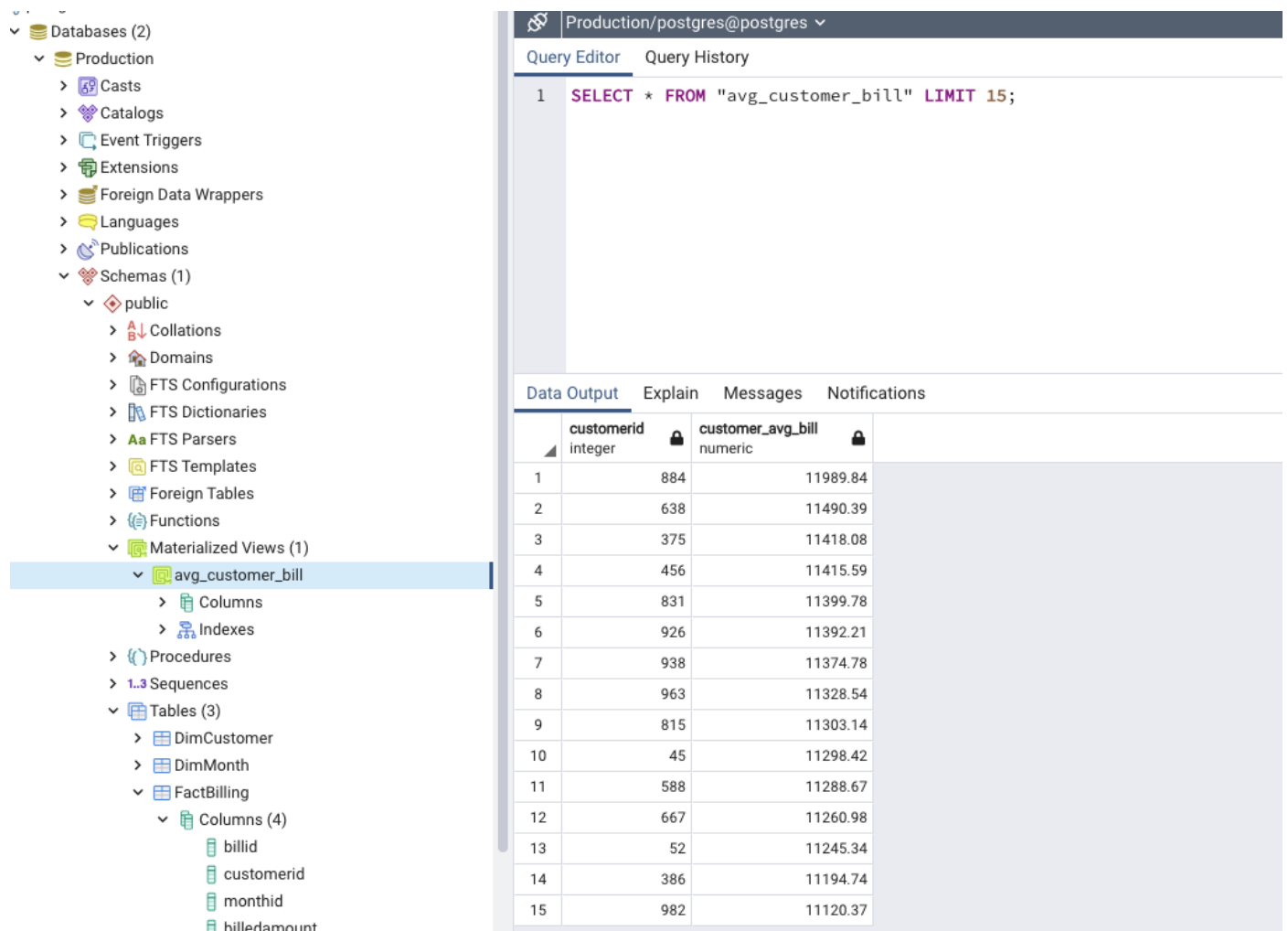
```
SELECT COUNT(*) FROM "FactBilling";
```

- Yes ... the encapsulation is required within the query tool for the particular database

Count
132000

Problem 2: Using the PostgreSQL tool, create a simple MQT named avg_customer_bill with fields customerid and averagebillamount.

```
CREATE MATERIALIZED VIEW avg_customer_bill
AS
SELECT
    customerid, ROUND(AVG(billedamount), 2) AS customer_avg_bill
FROM "FactBilling"
GROUP BY customerid
ORDER BY customer_avg_bill DESC;
```



The screenshot displays the PostgreSQL tool interface. On the left, the 'Databases (2)' tree shows the 'Production' database with various schemas. The 'Materialized Views (1)' section is expanded, showing the newly created 'avg_customer_bill' view. The main panel shows the 'Query Editor' with the following query:

```
1 SELECT * FROM "avg_customer_bill" LIMIT 15;
```

Below the query editor, the 'Data Output' tab is active, displaying a table with 15 rows of data. The table has two columns: 'customerid' (integer) and 'customer_avg_bill' (numeric). The data is sorted in descending order of the average bill amount.

	customerid integer	customer_avg_bill numeric
1	884	11989.84
2	638	11490.39
3	375	11418.08
4	456	11415.59
5	831	11399.78
6	926	11392.21
7	938	11374.78
8	963	11328.54
9	815	11303.14
10	45	11298.42
11	588	11288.67
12	667	11260.98
13	52	11245.34
14	386	11194.74
15	982	11120.37

Problem 3: Refresh the newly created MQT

```
REFRESH MATERIALIZED VIEW avg_customer_bill;
```

Problem 4: Using the newly created MQT find the customers whose average billing is more than 11000.

```
SELECT * FROM "avg_customer_bill" WHERE customer_avg_bill > 11000;
```

Production/postgres@postgres ▾	
Query Editor	Query History
<pre>1 SELECT * FROM "avg_customer_bill" WHERE customer_avg_bill > 11000; 2 3</pre>	
Data Output	Explain Messages Notifications
customerid integer	customer_avg_bill numeric
1	884
2	638
3	375
4	456
5	831
6	926
7	938
8	963
9	815
10	45
11	588
12	667
13	52
14	386
15	982
16	810
17	740
18	416
19	206
20	371
21	201
22	458
23	194