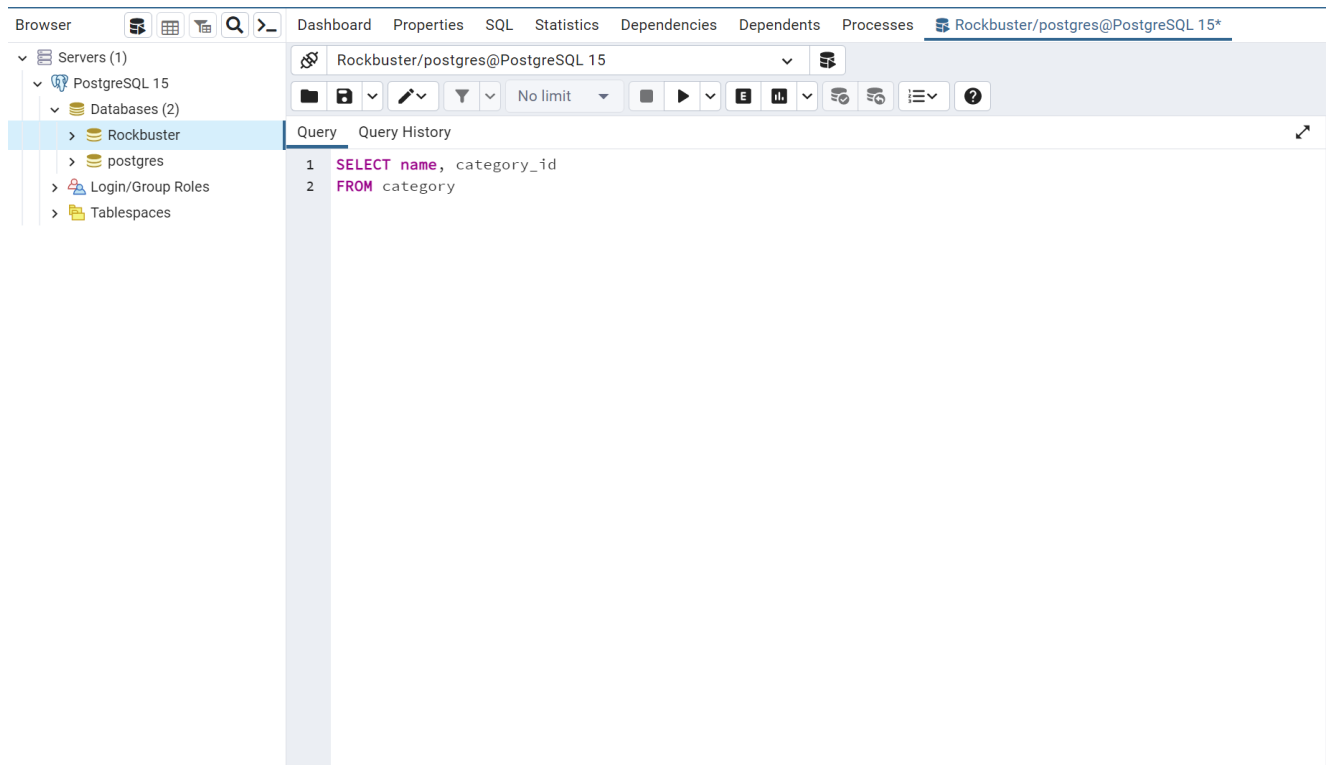


Step 1: Writing a SELECT command to find out the genres present in the category table.

SQL Statement



The screenshot shows a PostgreSQL client interface with a sidebar on the left containing a tree view of servers and databases. The main area displays a SQL query in a text editor. The query is:

```
1 SELECT name, category_id
2 FROM category
```

The interface includes a top menu bar with options like Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, and Processes. The current connection is identified as 'Rockbuster/postgres@PostgreSQL 15'.

Output

name	category_id
Action	1
Animation	2
Children	3
Classics	4
Comedy	5
Documentary	6
Drama	7
Family	8
Foreign	9

Games	10
Horror	11
Music	12
New	13
Sci-Fi	14
Sports	15
Travel	16

Step 2: Evaluating INSERT and CREATE TABLE Statements

QueryQuery History

1

INSERT INTO category(name)

2

VALUES ('Thriller')

QueryQuery History

1

INSERT INTO category(name)

2

VALUES ('Crime')

QueryQuery History

1

INSERT INTO category(name)

2

VALUES ('Mystery')

QueryQuery History

1

INSERT INTO category(name)

2

VALUES ('Romance')

QueryQuery History

1

INSERT INTO category(name)

2

VALUES ('War')

```

CREATE TABLE category
(
  category_id integer NOT NULL DEFAULT
nextval('category_category_id_seq'::regclass),
  name text COLLATE pg_catalog."default" NOT NULL,
  last_update timestamp with time zone NOT NULL DEFAULT now(),
  CONSTRAINT category_pkey PRIMARY KEY (category_id)
);

```

The SQL statement above was written to provide constraints on the columns within the Category table.

- The “NOT NULL DEFAULT” constraint ensures that there are no null values entered under the category_id column of the Category table if null values are copied over from the original table.
- The “NOT NULL DEFAULT” in the 6th line ensures that the last_update column of the Category table does not allow for the transfer of null values that may exist in the original last_update column.
- The “PRIMARY KEY” constraint ensures that the category_id column serves as the primary key, turning all the values in the category_id column into a primary key that acts as a unique identifier for an entire row in the table.

In essence, the above statement ensures that the category_id column can, indeed, serve as the primary key by preventing null values from the category_id.

Step 3: Inserting New Movie Title

Select Statement to Find Film ID for Title

Query
Query History

```

1 SELECT (film_id)
2 FROM film
3 WHERE title='African Egg'

```

Data Output
Messages
Notifications

	film_id [PK] integer
1	5

Select Statement to Find Category ID for the “Thriller” genre

The screenshot shows a SQL query editor with a 'Query' tab selected. The query is as follows:

```
1 SELECT (category_id)
2 FROM category
3 WHERE name='Thriller'
```

Below the query editor, there is a 'Data Output' tab showing the results of the query. The results are displayed in a table with one column, 'category_id', and one row with the value '17'.

category_id
17

SQL statement updating the category id in the film_category table to 17 (the code for Thriller movies) for African Egg.

The screenshot shows a SQL query editor with a 'Query' tab selected. The query is as follows:

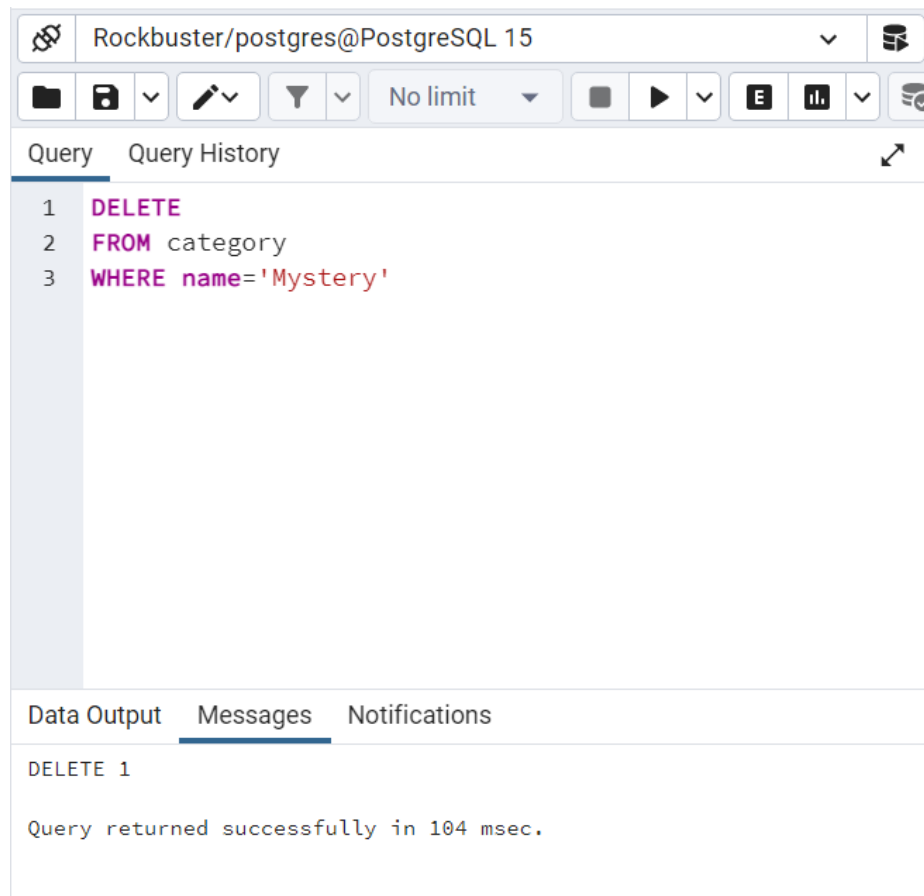
```
1 UPDATE film_category
2 SET category_id=17
3 WHERE film_id=5
```

Below the query editor, there is a 'Data Output' tab showing the results of the query. The results are displayed in a table with one column, 'category_id', and one row with the value '17'.

category_id
17

Step 4: Deleting A Category

SQL Statement to Delete all Columns in the Category Table where "Mystery" is as an option from the name column



The screenshot shows a PostgreSQL client window titled "Rockbuster/postgres@PostgreSQL 15". The interface includes a toolbar with icons for file operations, query execution, and settings. The "Query" tab is active, displaying the following SQL statement:

```
1 DELETE
2 FROM category
3 WHERE name='Mystery'
```

Below the query editor, the "Data Output" tab is selected, showing the result of the query:

```
DELETE 1
```

Below the data output, a message indicates the query was successful:

```
Query returned successfully in 104 msec.
```

Step 5: Reflection on Excel vs SQL

If I had carried out the above tasks in Excel, I would have had to begin by creating a Pivot Table to uncover the genres present. I also would have had to have pulled all pivot table data from their respective source table(s) by going directly into said table(s). Much of the rest of the work would have been carried out in the Pivot Tables by making use of functions like filters, the find and replace tool, etc. One of the major advantages of using SQL in this particular use case is the ease of navigating between different tables without having to physically navigate to and from the actual tables. It's easier to visualize. The main con is that one has to have a fairly strong command of the SQL language to take advantage of the confusion- and time-savings.