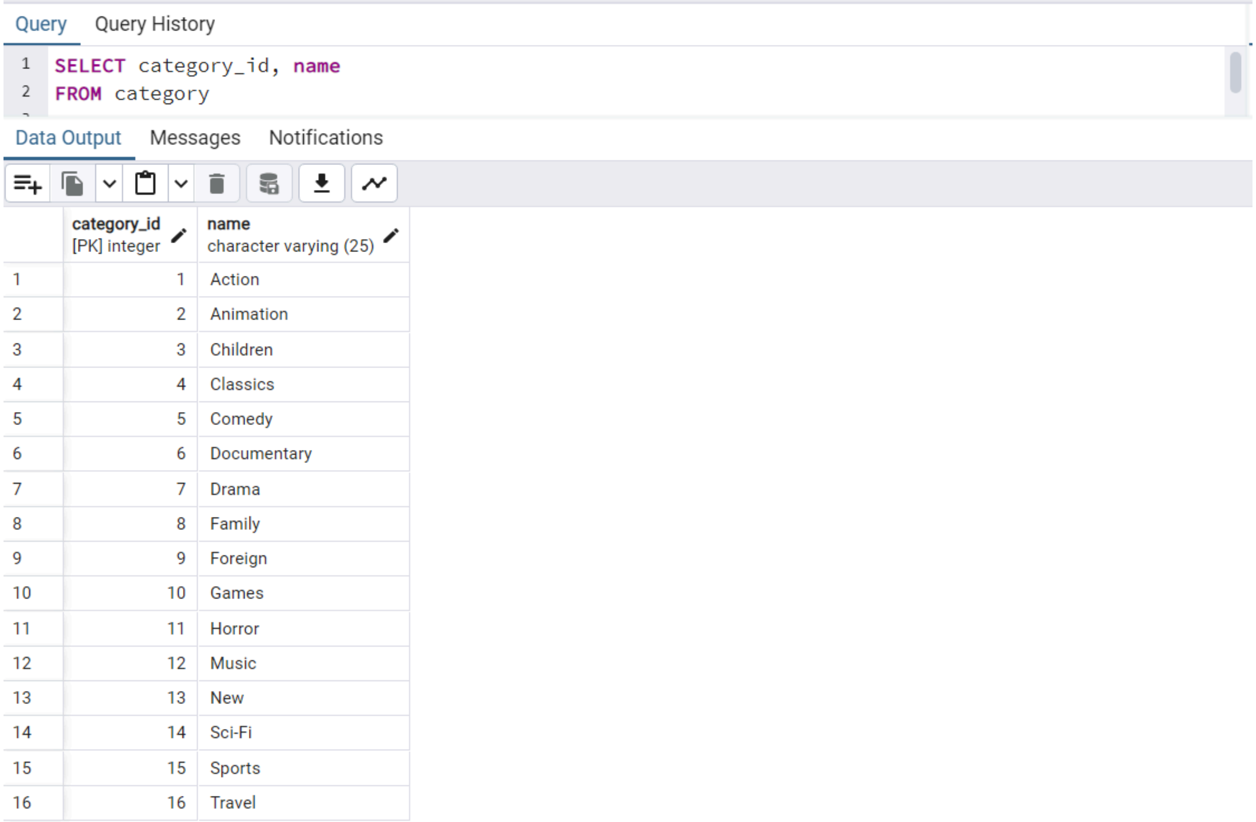
3.3 - SQL Data Analysis
Katherine Lecce

Step 1:

Your first task is to find out what film genres already exist in the category table:

→Write a SELECT command to find out what film genres exist in the category table.

→Copy-paste the output into your answers document or write the answers out—it's up to you. Make sure to include the category ID for each genre.



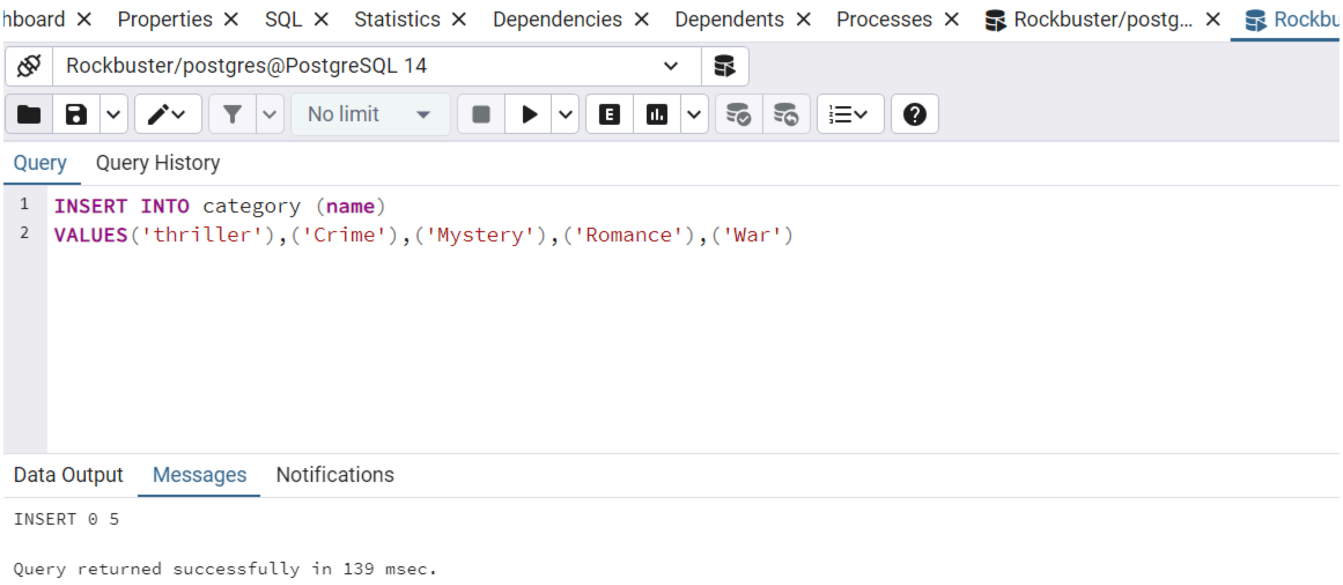
```sql
SELECT category_id, name
FROM category
```

	category_id [PK] integer	name character varying (25)
1	1	Action
2	2	Animation
3	3	Children
4	4	Classics
5	5	Comedy
6	6	Documentary
7	7	Drama
8	8	Family
9	9	Foreign
10	10	Games
11	11	Horror
12	12	Music
13	13	New
14	14	Sci-Fi
15	15	Sports
16	16	Travel

Step 2:

You're ready to add some new genres! Write an `INSERT` statement to add the following genres to the category table: Thriller, Crime, Mystery, Romance, and War:

- Copy-paste your `INSERT` commands into your answers document.



```sql
INSERT INTO category (name)
VALUES('thriller'),('Crime'),('Mystery'),('Romance'),('War')
```

```
INSERT 0 5

Query returned successfully in 139 msec.
```

→ The `CREATE` statement below shows the constraints on the category table. Write a short paragraph explaining the various constraints that have been applied to the columns.

```
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)
);
```

These constraints are important for maintaining that the code we are looking for in the data are correct and allow it to follow certain rules so that the table pulls only the information we need with the rule which helps us organize and filter out data when looking at large sums of data values in a file. In this example.

The **NOT NULL** text ensures that all data is present and prevents the output from being incomplete - meaning the ‘category_id’ row must contain information from that subset.

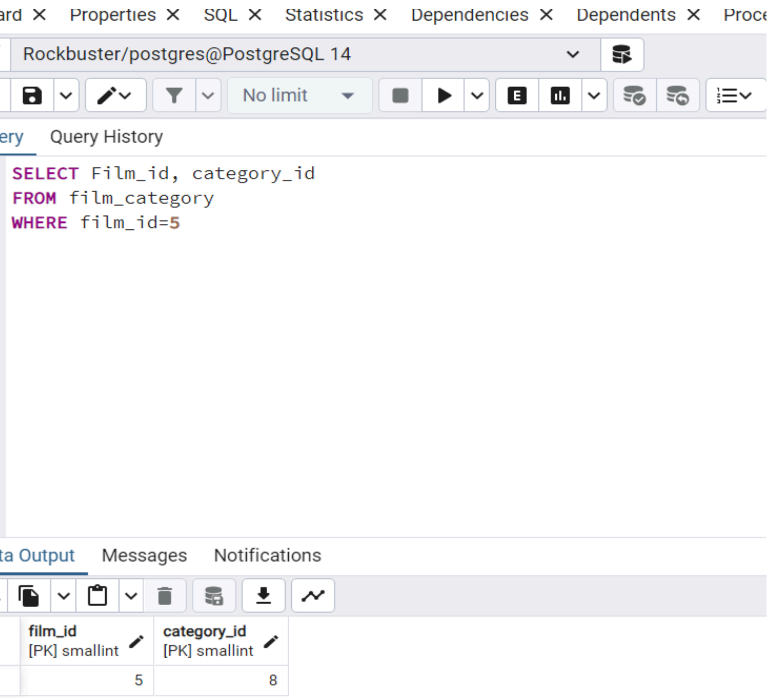
The **DEFAULT** text allows the statement to catch any information that might not be specified by valuable to the quarry output. For this example - the quarry will pull values from the ‘category_id_seq’ to generate a value for any missing values from the ‘category_id’ column.

The **‘Primary Key’** text allows the quarry to find all of the information in the rows needed (in this case ‘category_id’) and creates a unique id for this row and allows the program to pull a small amount of information from a large amount of data available in a file. This ensures again that our quarry is validated in the system and is unique in our findings.

Step 3:

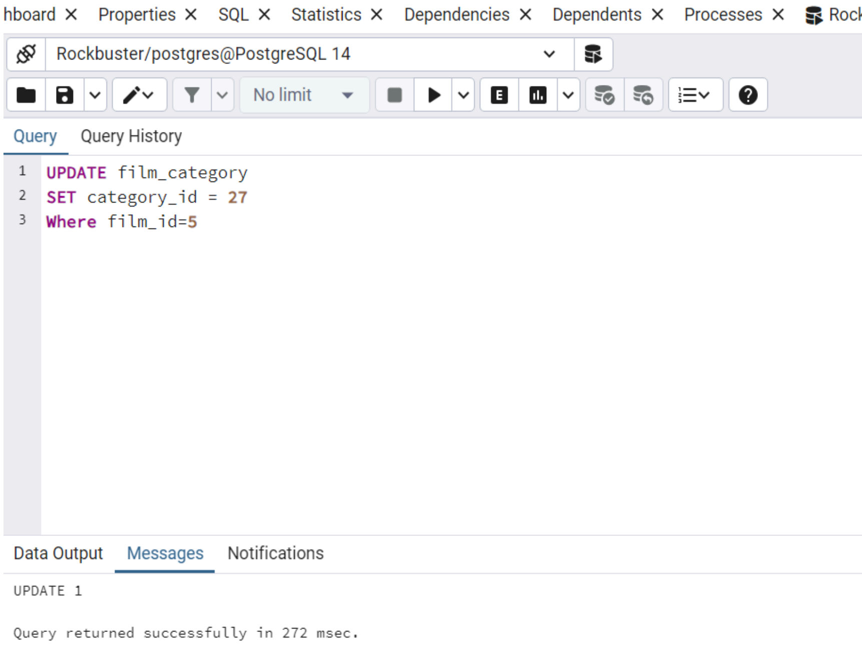
The genre for the movie *African Egg* needs to be updated to thriller. Work through the steps below to make this change:

- Write the `SELECT` statement to find the film_id for the movie *African Egg*.



```sql
SELECT Film_id, category_id
FROM film_category
WHERE film_id=5
```

film_id [PK] smallint	category_id [PK] smallint
5	8



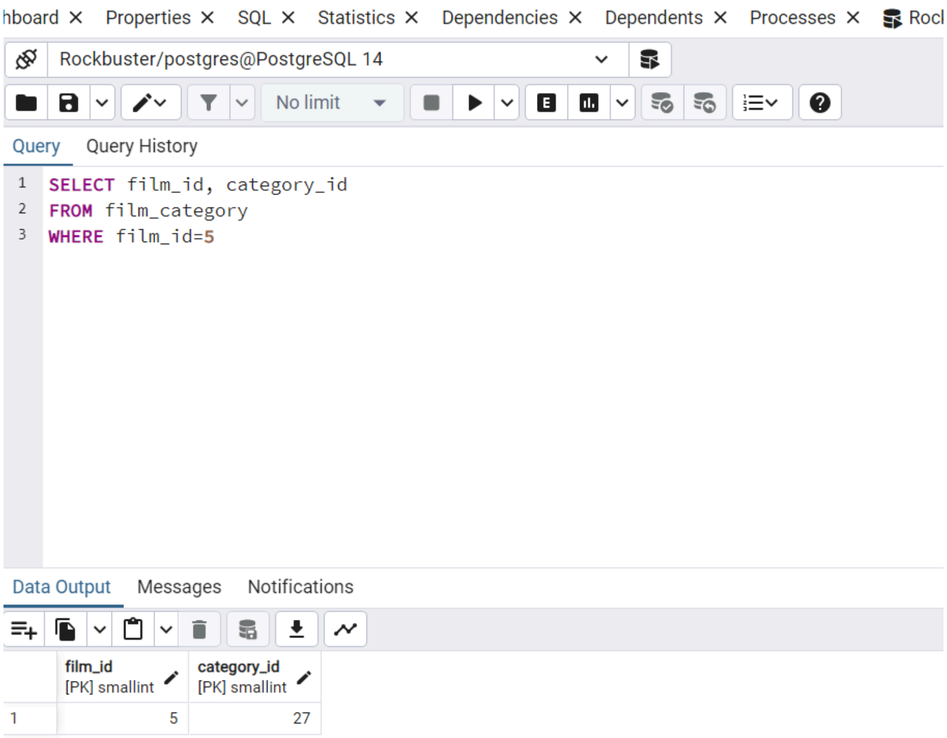
```sql
UPDATE film_category
SET category_id = 27
Where film_id=5
```

```
UPDATE 1

Query returned successfully in 272 msec.
```

Once you have the film_ID and category_ID, write an `UPDATE` command to change the category in the film_category table (not

Double Check:



Rockbuster/postgres@PostgreSQL 14

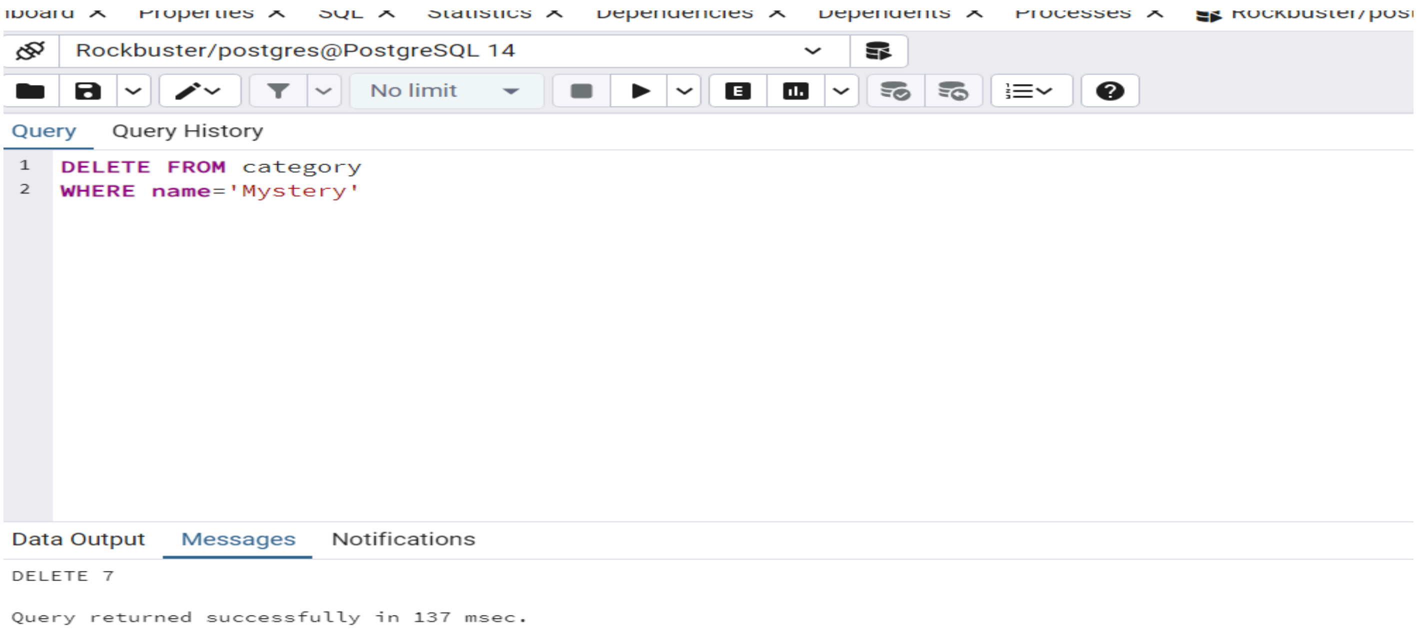
No limit

```
SELECT film_id, category_id
FROM film_category
WHERE film_id=5
```

	film_id [PK] smallint	category_id [PK] smallint
1	5	27

Step 4:

Since there aren’t many movies in the mystery category, you and your manager decide to remove it from the category table. Write a `DELETE` command to do so and copy-paste it into your answers document.



```sql
DELETE FROM category
WHERE name='Mystery'
```

```
DELETE 7

Query returned successfully in 137 msec.
```

Step 5:

Based on what you've learned so far, think about what it would be like to complete steps 1 to 4 with Excel instead of SQL. Are there any pros and cons to using SQL? Write a paragraph explaining your answer.

With every new application, there is a learning curve when learning the functionality of the application, the hardest part for me initially is learning the different language that we are now using the filter to get the exact answers we're looking for from the large amount of data. Though it is a trial and error process, I feel that SQL is less overwhelming since you aren't seeing the whole scale of data in the file we uploaded for Rockbuster in the 3rd section. I also find SQL more reliable when checking the consistency of the data we are looking for with the quarry tool while sometimes when copying and pasting different information from excel causes errors and Excel application sometimes alters the original data skewing the accurasing of our analysis.

It's a great application for filtering large amounts of data since there is no lag time compared to uploading, using, transforming large data sets on the Excel spreadsheet. I think SQL is great for filtering data but not as helpful when we want to look at data in a visual aspect. I find it more helpful to use Excel's features such as creating different filters, charts, and using the pivot table tool to look at different parts of data as a whole.