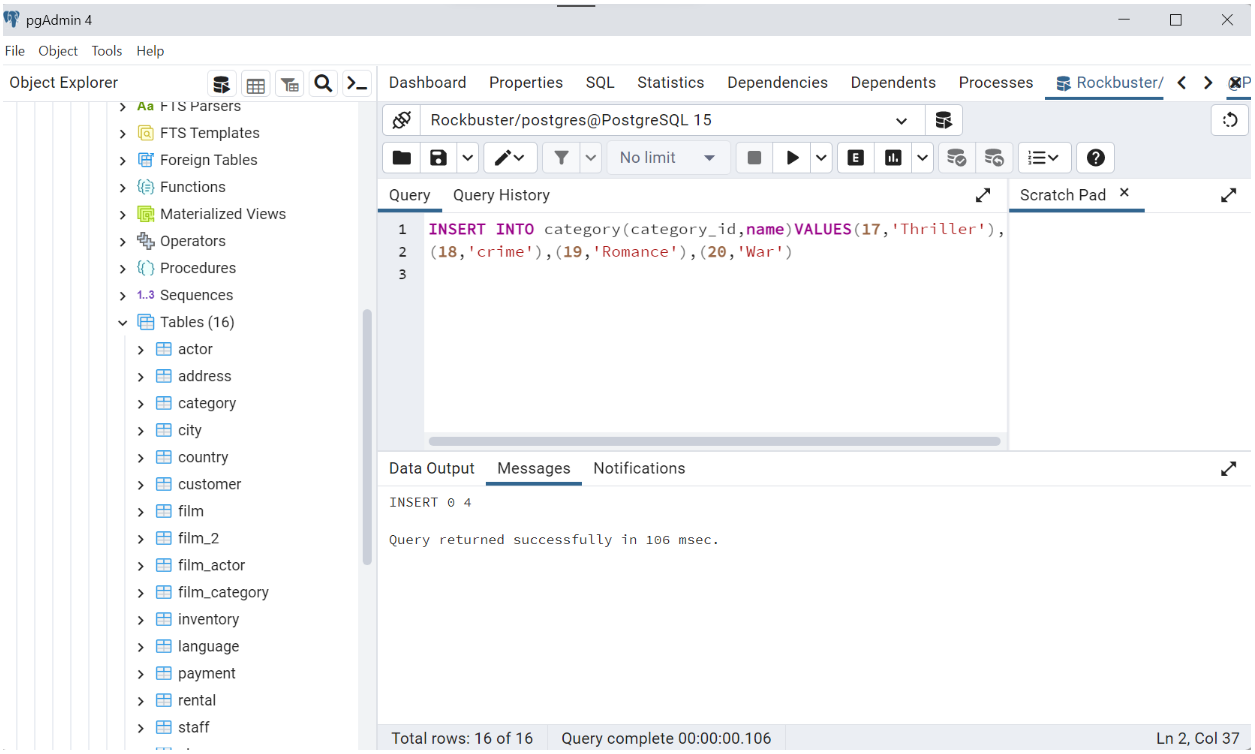
EXERCISE 3:

STEP 1:

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

STEP 2:



```
INSERT INTO category(category_id,name)VALUES(17,'Thriller'),
(18,'crime'),(19,'Romance'),(20,'War')
```

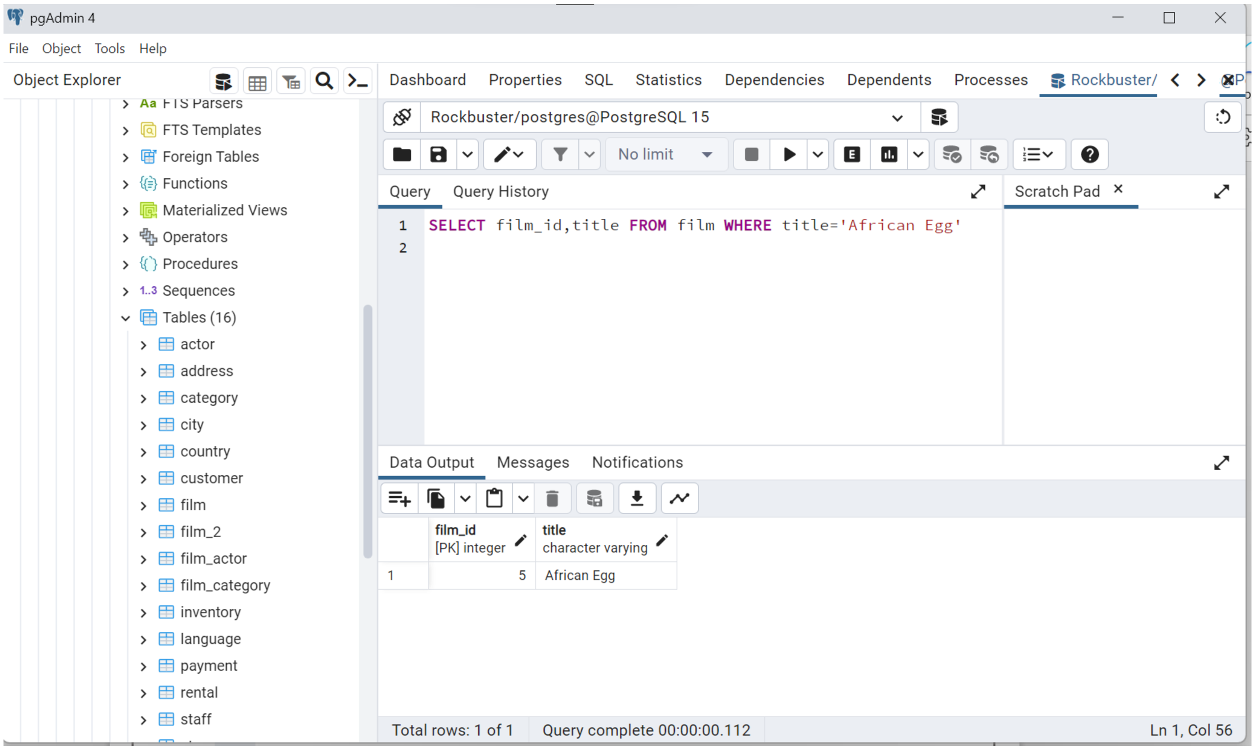
```
INSERT 0 4

Query returned successfully in 106 msec.
```

Write a short paragraph explaining the various constraints that have been applied to the columns. What do these constraints do exactly? Why are they important?

```
  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 ‘NOT NULL’ constraint enforces the rule that the ‘category_id’ column must not contain empty or missing entries. By using the ‘PRIMARY KEY’ constraint, each value in the ‘category_id’ column becomes a unique identifier and cannot be duplicated. The combination of these two constraints ensures that the ‘category_id’ column is an essential location for finding specific information because it will always have a value and be unique for each row in the table. Additionally, the ‘DEFAULT’ constraint specifies a default value to be automatically entered if no other value is provided. This constraint is crucial for maintaining data integrity and preventing the occurrence of missing values.

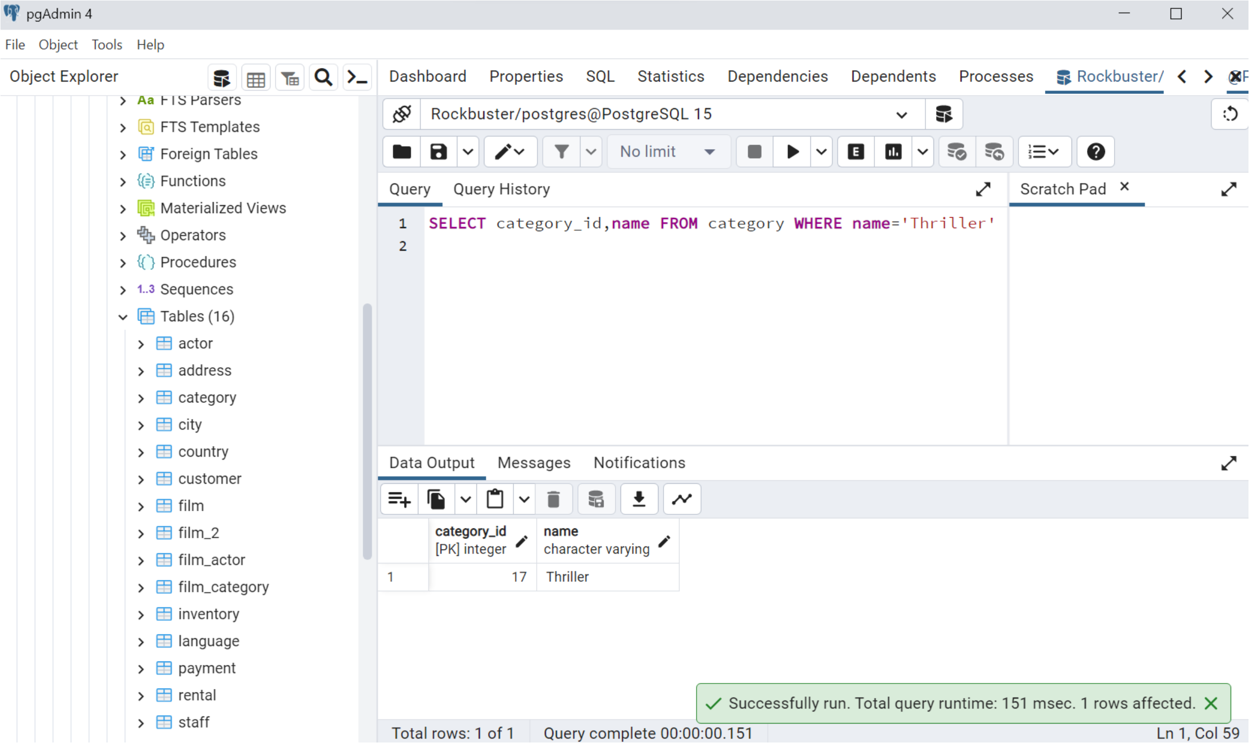
STEP3:



```
SELECT film_id,title FROM film WHERE title='African Egg'
```

	film_id [PK] integer	title character varying
1	5	African Egg

Total rows: 1 of 1 Query complete 00:00:00.112 Ln 1, Col 56

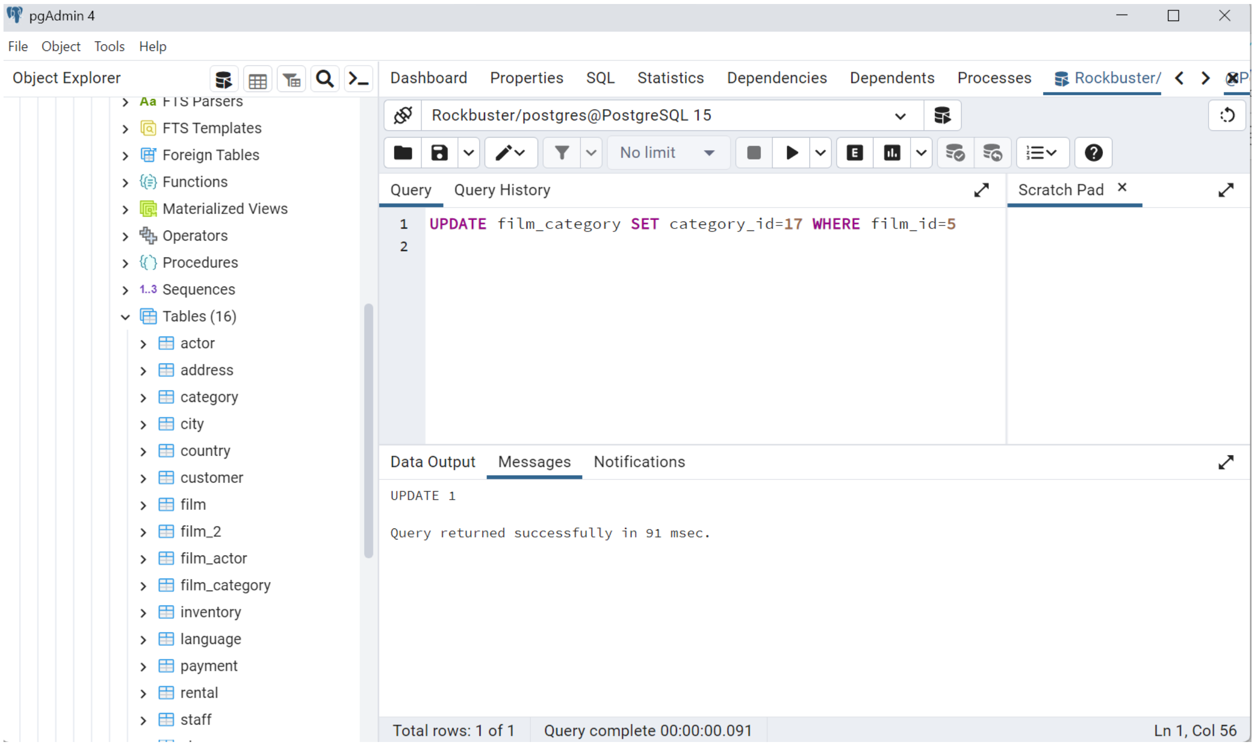


```
SELECT category_id,name FROM category WHERE name='Thriller'
```

	category_id [PK] integer	name character varying
1	17	Thriller

✓ Successfully run. Total query runtime: 151 msec. 1 rows affected.

Total rows: 1 of 1 Query complete 00:00:00.151 Ln 1, Col 59

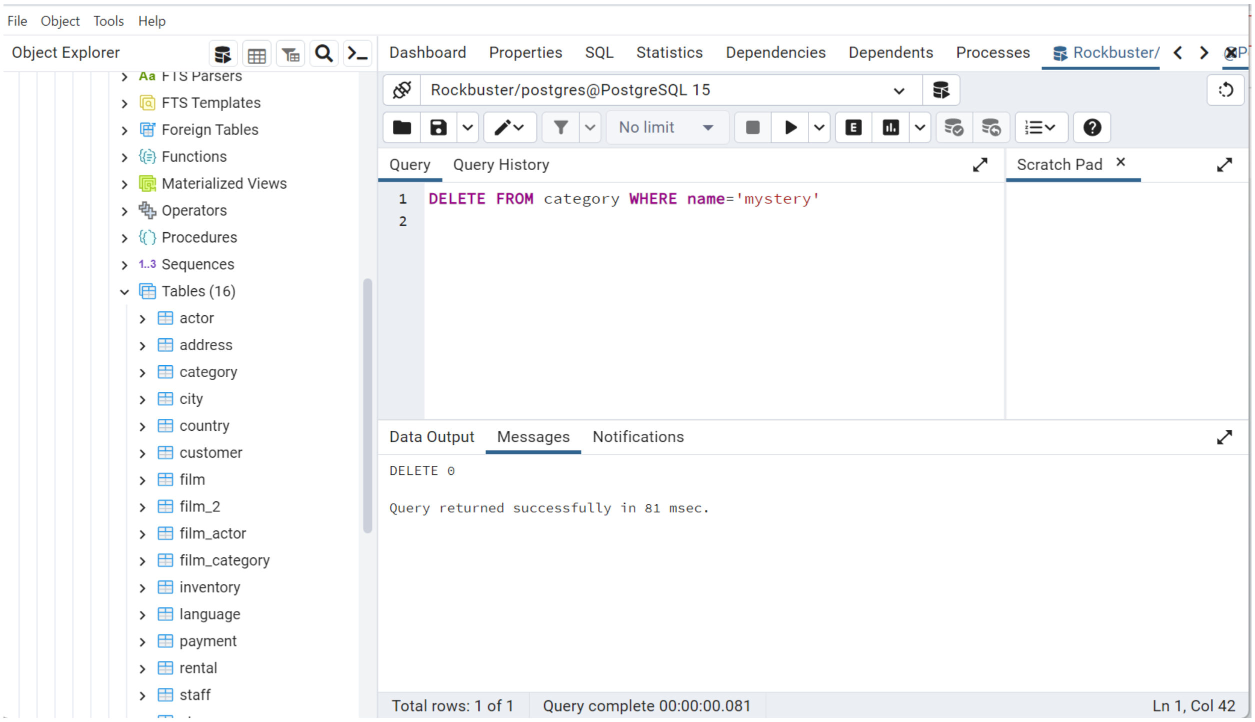


```sql
UPDATE film_category SET category_id=17 WHERE film_id=5
```

```
UPDATE 1

Query returned successfully in 91 msec.
```

Total rows: 1 of 1 Query complete 00:00:00.091 Ln 1, Col 56

STEP4:



Rockbuster/postgres@PostgreSQL 15

```
DELETE FROM category WHERE name='mystery'
```

```
DELETE 0

Query returned successfully in 81 msec.
```

Total rows: 1 of 1 Query complete 00:00:00.081 Ln 1, Col 42

STEP 5:

Yes, there are several pros and cons to using SQL instead of Excel for data analysis and management. Here are some key points to consider:

Pros of using SQL:

Scalability and Performance: SQL databases are designed for handling large volumes of data efficiently. They are optimized for complex queries and can handle millions of records without significant performance issues. Excel, on the other hand, can become slow and cumbersome with large datasets.

Data Integrity and Consistency: SQL databases provide robust data integrity mechanisms, such as constraints, foreign key relationships, and transaction support. This ensures that the data remains consistent and accurate over time. In Excel, data integrity relies on manual input and formula calculations, which can be prone to errors and inconsistencies.

Simultaneous Access and Collaboration: SQL databases allow multiple users to access and manipulate data simultaneously, facilitating collaborative work on shared datasets. In Excel, concurrent editing can lead to conflicts and version control challenges.

Data Security: SQL databases offer advanced security features, including user authentication, role-based access control, and encryption. This helps protect sensitive data and ensures that only authorized users can access and modify it. Excel files, especially when shared via email or cloud storage, are more susceptible to unauthorized access and data breaches.

Advanced Querying and Analysis: SQL provides a powerful querying language that allows you to perform complex operations, aggregations, and joins on datasets. It offers flexibility and control over data manipulation, making it easier to extract meaningful insights. While Excel provides basic data manipulation and analysis capabilities, SQL offers more advanced functionalities.

Cons of using SQL:

Learning Curve: SQL has a steeper learning curve compared to Excel. It requires understanding the syntax, query structure, and relational database concepts. Excel, on the other hand, has a more user-friendly interface and is widely known, making it more accessible to casual users.

Lack of Flexibility for Ad Hoc Analysis: SQL databases are structured and require predefined schema and table designs. This can limit the flexibility for ad hoc analysis or exploring data on the fly. Excel, with its grid-like structure, allows users to quickly manipulate and analyze data without strict schema requirements.

Visualization Capabilities: Excel offers a wide range of built-in charting and visualization tools, allowing users to create visual representations of data with ease. SQL, being primarily a query language, doesn't provide the same level of graphical capabilities. However, SQL data can be connected to external visualization tools to create advanced visualizations.

Spreadsheet-Specific Features: Excel has specific features like macros, pivot tables, and extensive formula capabilities that make it suitable for certain types of analysis and reporting. SQL, being a database language, may not have the same level of specialized functionality in these areas.

Ultimately, the choice between SQL and Excel depends on the specific requirements of your data analysis and management tasks. SQL is more suitable for handling large datasets, complex querying, and collaborative data work, while Excel excels in ad hoc analysis, visualization, and spreadsheet-specific tasks.