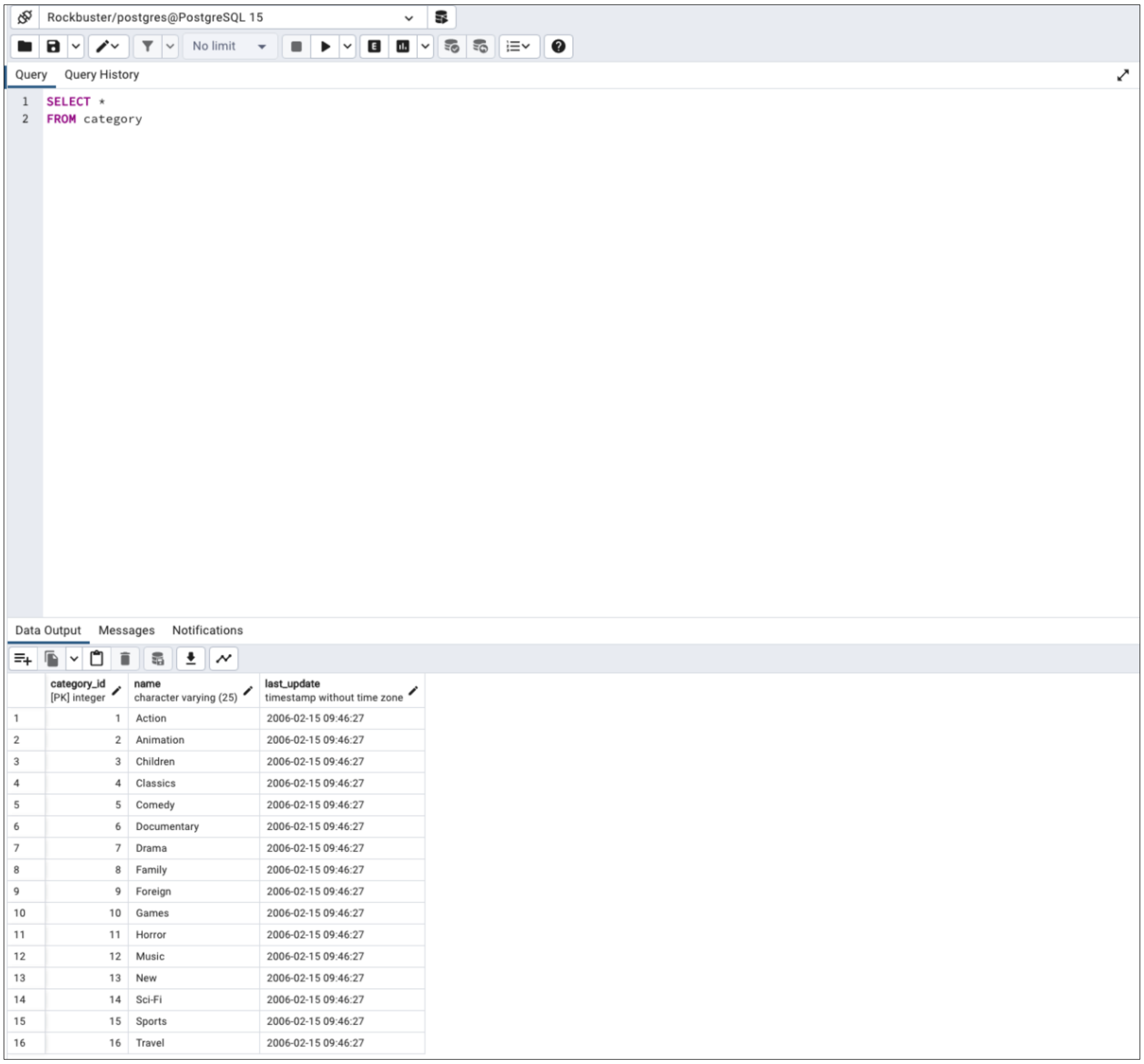
Step 1:

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

- **Open pgAdmin 4, click the Rockbuster database, and open the Query Tool.**
- **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.**

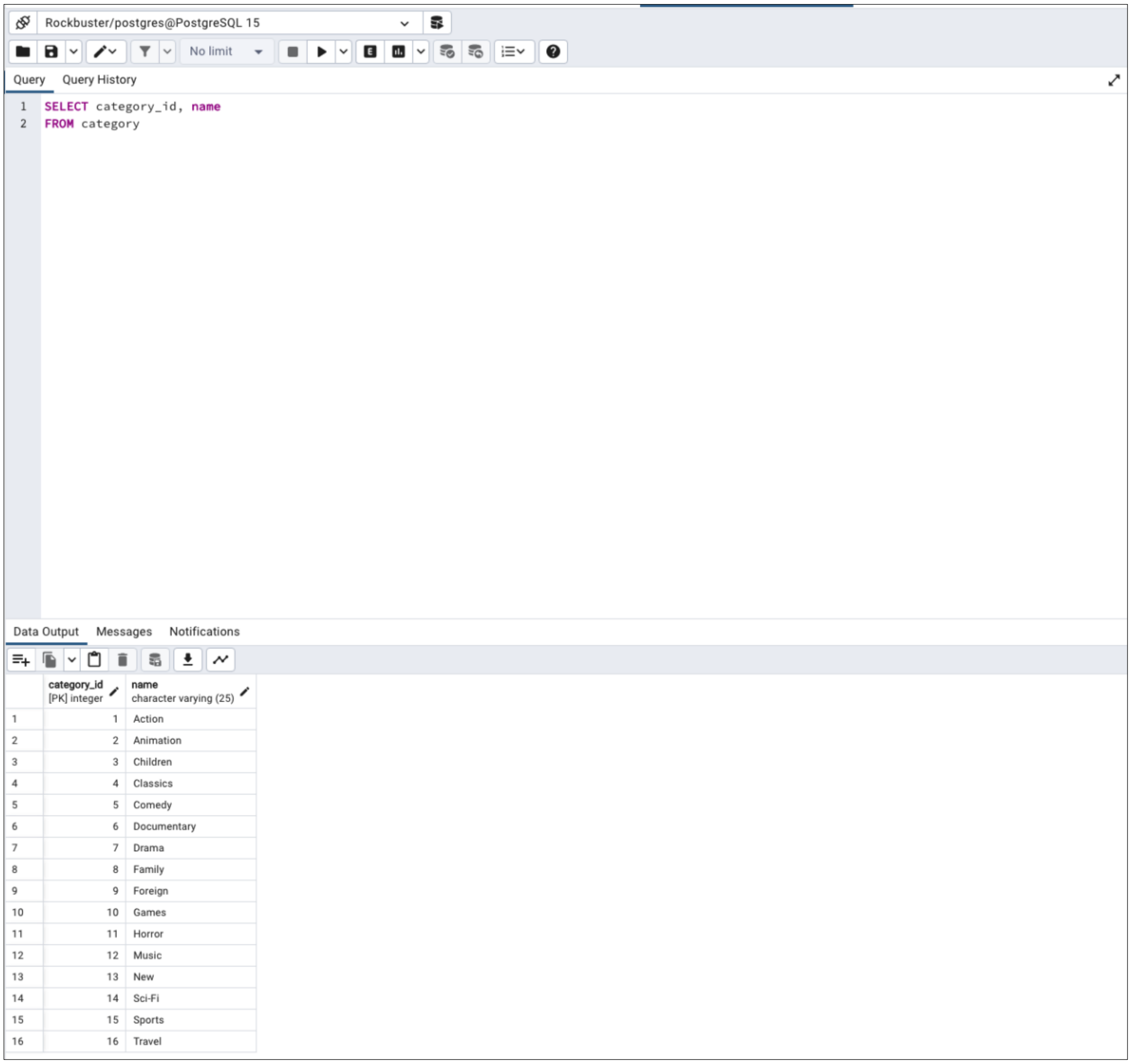


```
SELECT *
FROM category
```

	category_id [PK] integer	name character varying (25)	last_update timestamp without time zone
1	1	Action	2006-02-15 09:46:27
2	2	Animation	2006-02-15 09:46:27
3	3	Children	2006-02-15 09:46:27
4	4	Classics	2006-02-15 09:46:27
5	5	Comedy	2006-02-15 09:46:27
6	6	Documentary	2006-02-15 09:46:27
7	7	Drama	2006-02-15 09:46:27
8	8	Family	2006-02-15 09:46:27
9	9	Foreign	2006-02-15 09:46:27
10	10	Games	2006-02-15 09:46:27
11	11	Horror	2006-02-15 09:46:27
12	12	Music	2006-02-15 09:46:27
13	13	New	2006-02-15 09:46:27
14	14	Sci-Fi	2006-02-15 09:46:27
15	15	Sports	2006-02-15 09:46:27
16	16	Travel	2006-02-15 09:46:27

This query returns **all the columns** from the **category table**, use *, which is shorthand for all the columns.

or



```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

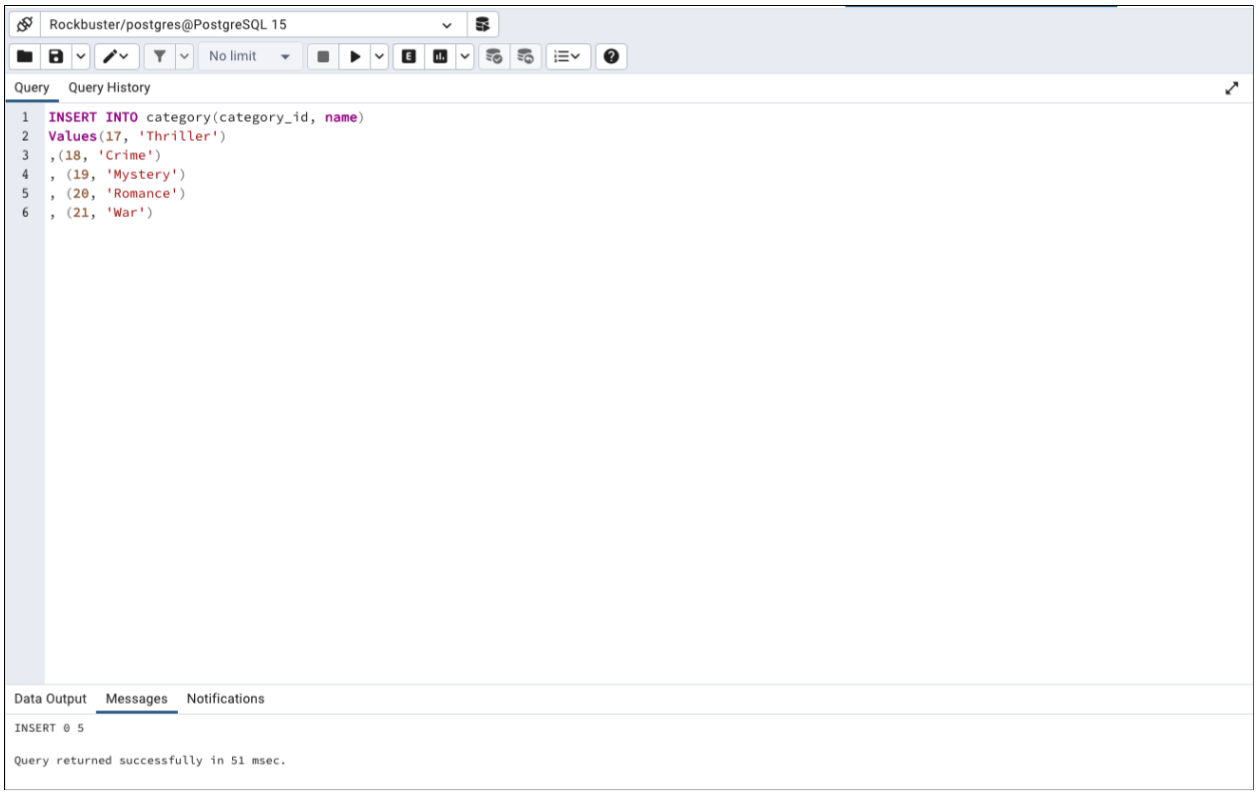
This query is telling the database to return the columns **"category_id"** and **"name"** from the **"category"** table. The column names are also separated by a comma.

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.**

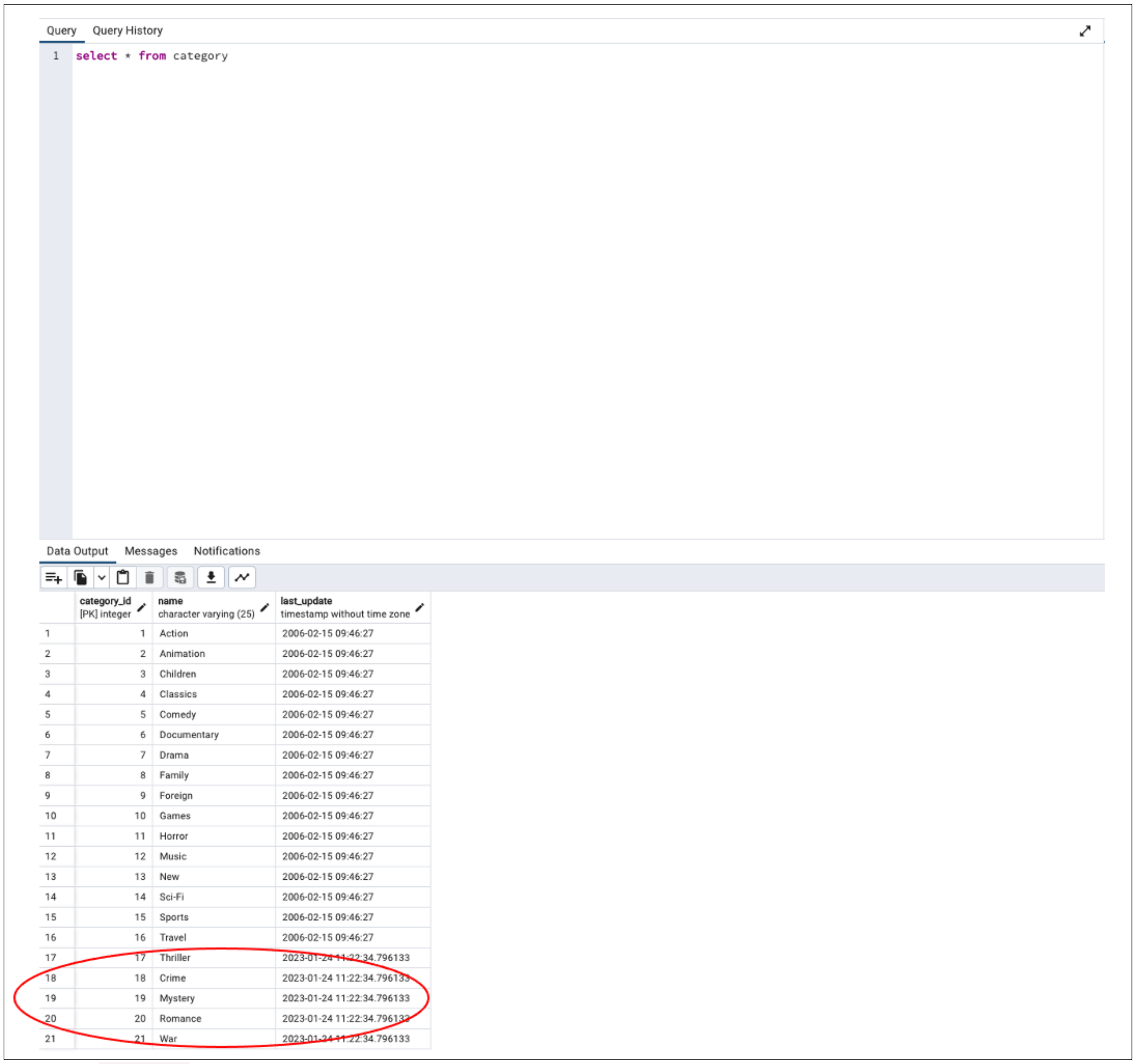
2a)



```
INSERT INTO category(category_id, name)
Values(17, 'Thriller')
,(18, 'Crime')
, (19, 'Mystery')
, (20, 'Romance')
, (21, 'War')
```

```
INSERT 0 5

Query returned successfully in 51 msec.
```


Query   Query History

```
select * from category
```

Data Output   Messages   Notifications

	category_id [PK] integer	name character varying (25)	last_update timestamp without time zone
1	1	Action	2006-02-15 09:46:27
2	2	Animation	2006-02-15 09:46:27
3	3	Children	2006-02-15 09:46:27
4	4	Classics	2006-02-15 09:46:27
5	5	Comedy	2006-02-15 09:46:27
6	6	Documentary	2006-02-15 09:46:27
7	7	Drama	2006-02-15 09:46:27
8	8	Family	2006-02-15 09:46:27
9	9	Foreign	2006-02-15 09:46:27
10	10	Games	2006-02-15 09:46:27
11	11	Horror	2006-02-15 09:46:27
12	12	Music	2006-02-15 09:46:27
13	13	New	2006-02-15 09:46:27
14	14	Sci-Fi	2006-02-15 09:46:27
15	15	Sports	2006-02-15 09:46:27
16	16	Travel	2006-02-15 09:46:27
17	17	Thriller	2023-01-24 11:22:34.796133
18	18	Crime	2023-01-24 11:22:34.796133
19	19	Mystery	2023-01-24 11:22:34.796133
20	20	Romance	2023-01-24 11:22:34.796133
21	21	War	2023-01-24 11:22:34.796133

2b) 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. 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)
);
```

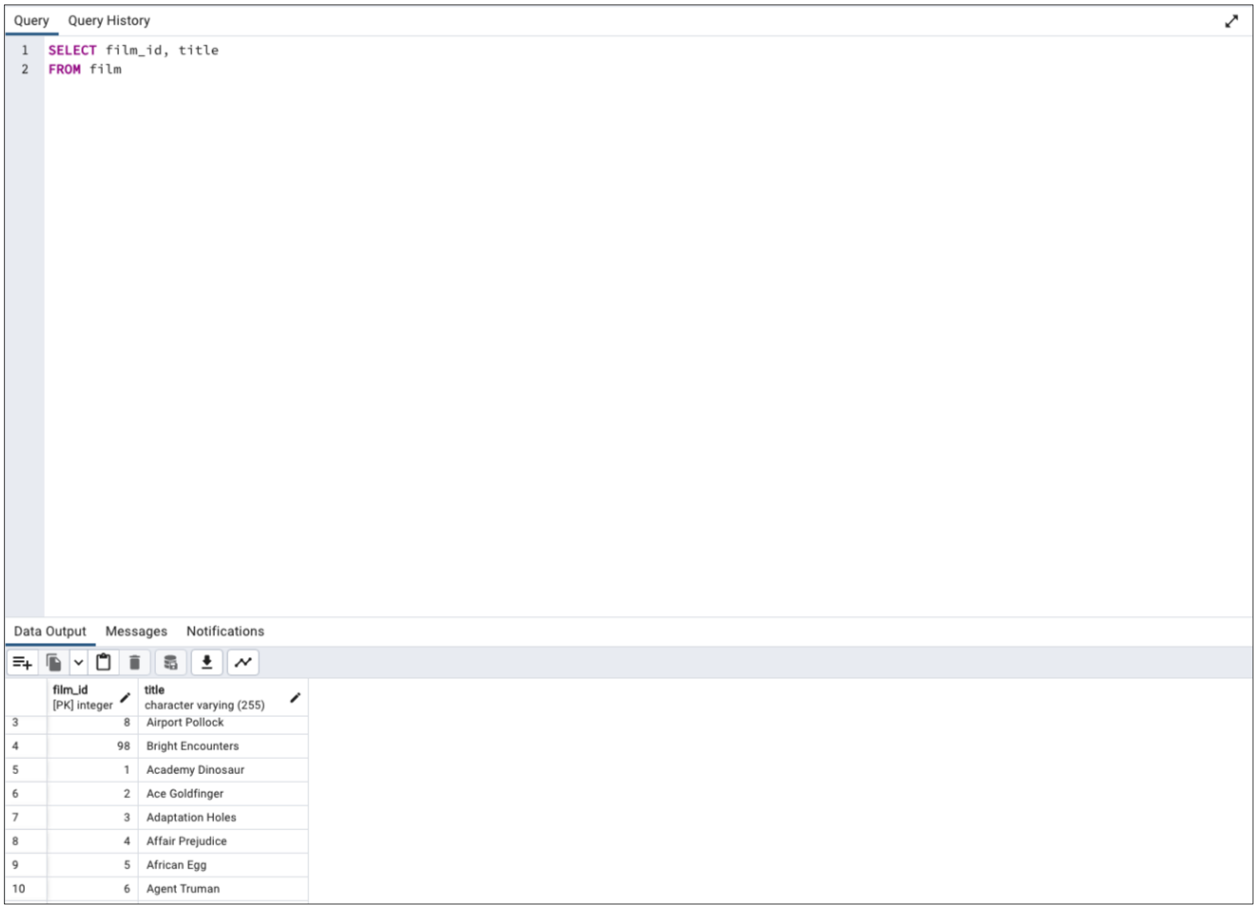
The **NOT NULL constraint** ensures that **the category_id column** cannot have **any empty or missing values**. The primary key constraint gives each record in a table a unique ID. The primary key column cannot contain any null or duplicate values. In the create statement above, the primary key constraint sets the “category_id” column as the primary key when the “category” table is created.

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*.**

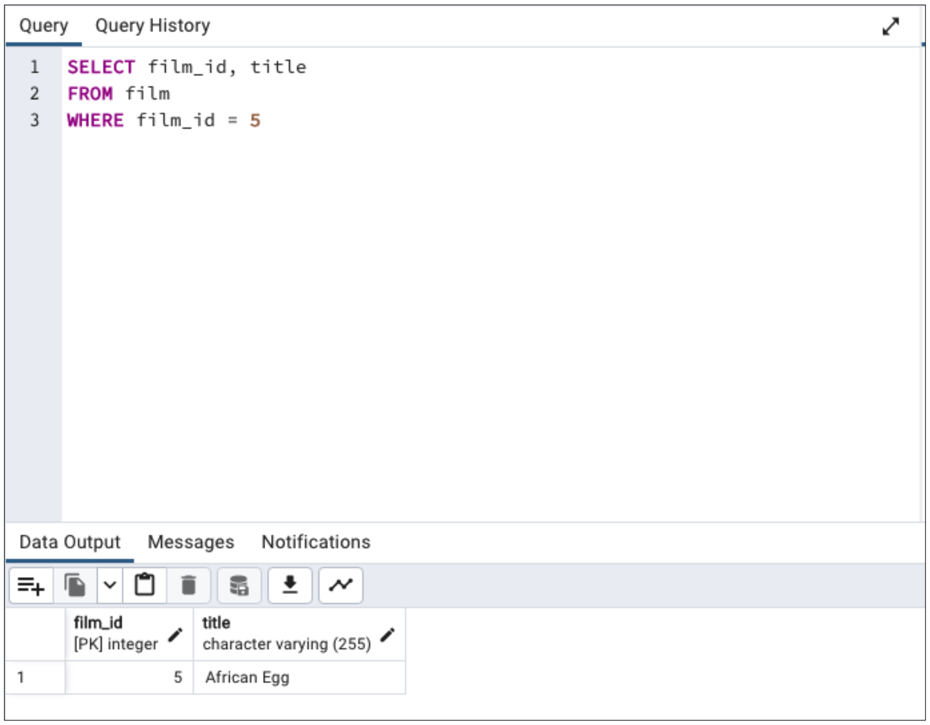
1) First, we will tell the database to return the columns “film_ID” and “title” from the “film” table.



```
SELECT film_id, title
FROM film
```

	film_id [PK] integer	title character varying (255)
3	8	Airport Pollock
4	98	Bright Encounters
5	1	Academy Dinosaur
6	2	Ace Goldfinger
7	3	Adaptation Holes
8	4	Affair Prejudice
9	5	African Egg
10	6	Agent Truman

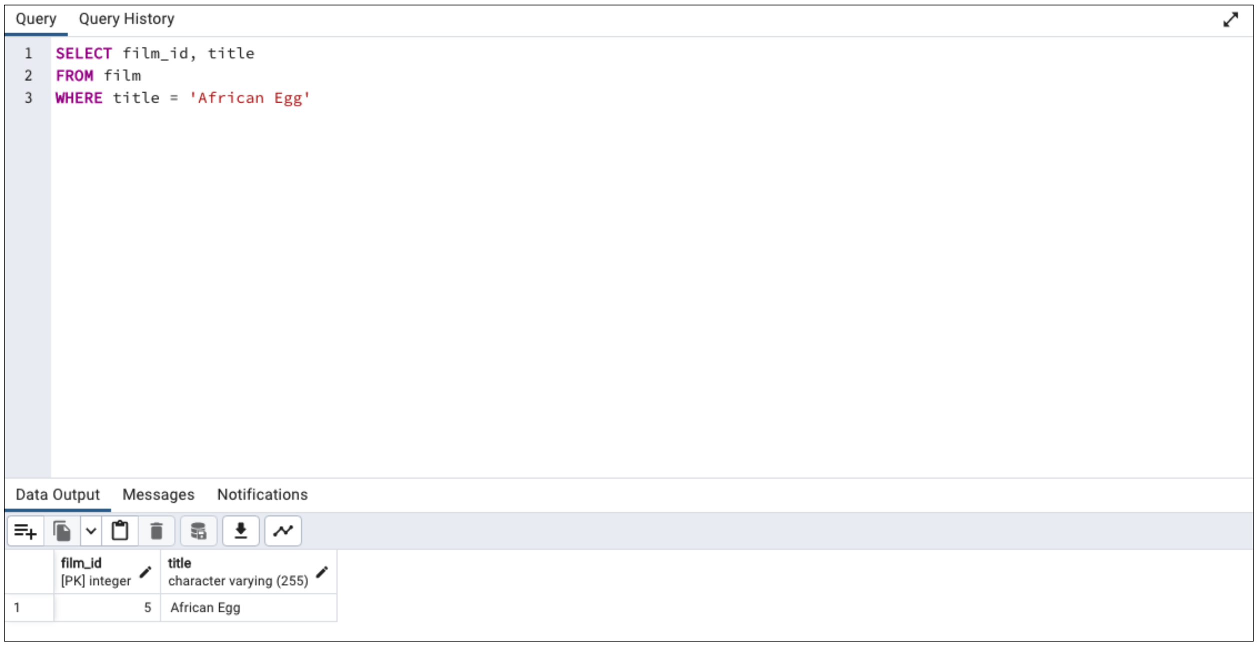
2) Since we know film_id = 5, we will include the WHERE clause to specify which row we want.



```sql
SELECT film_id, title
FROM film
WHERE film_id = 5
```

	film_id [PK] integer	title character varying (255)
1	5	African Egg

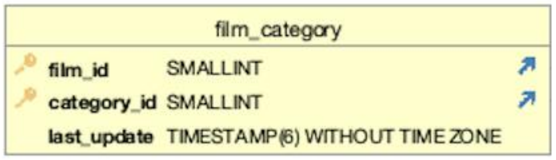
or



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

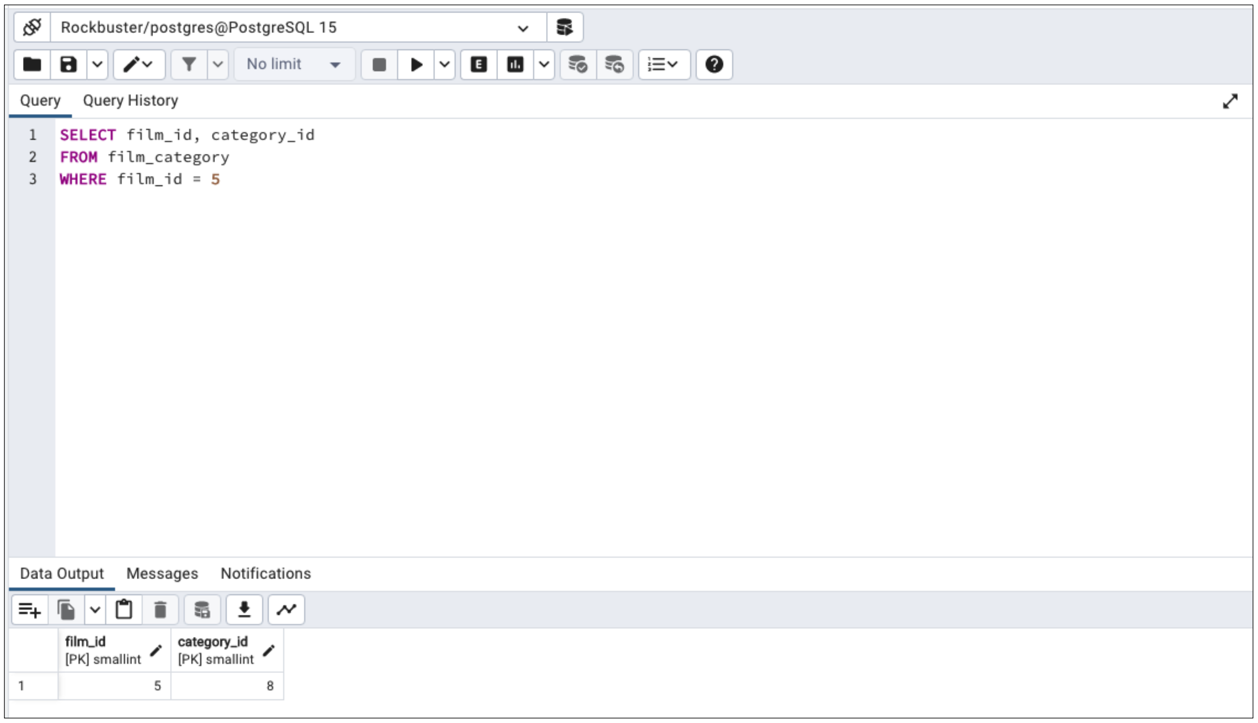
	film_id [PK] integer	title character varying (255)
1	5	African Egg

Once you have the film_ID and category_ID, write an UPDATE command to change the category in the film_category table (not the category table). Copy-paste this command into your answers document.



film_category	
film_id	SMALLINT
category_id	SMALLINT
last_update	TIMESTAMP(6) WITHOUT TIME ZONE

Film_category is a dimension table that provides information about the film category.
Column film_id represents a foreign key to the film table
Column category_id represents a foreign key to the category table
Therefore, we want to tell the database to return columns **"film_id"** and **"category_id"** from the **"film_category" table** and include the WHERE clause to specify that **row 5** from **film_id** is what we want.



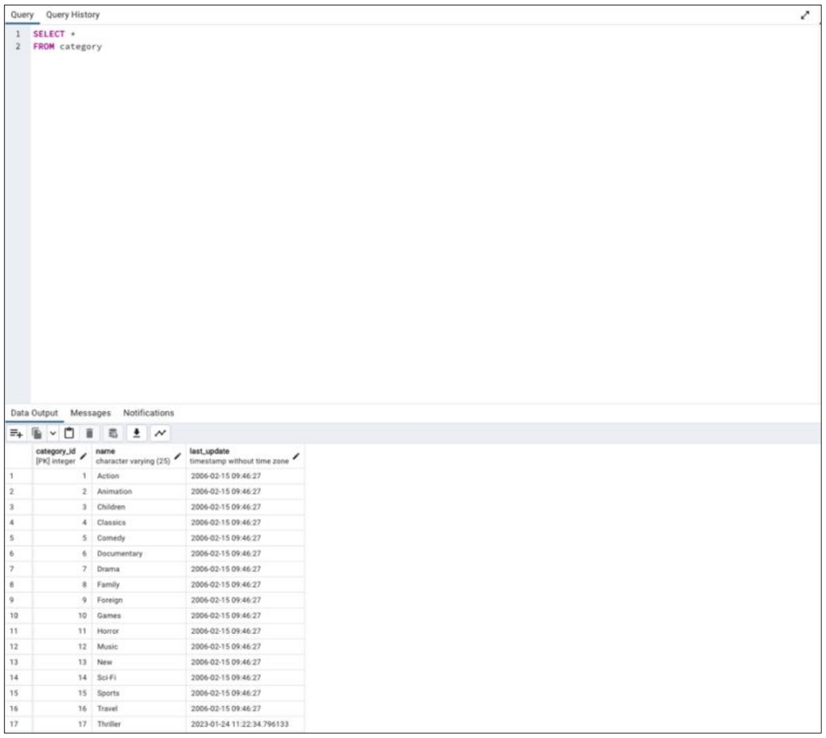
```
SELECT film_id, category_id
FROM film_category
WHERE film_id = 5
```

	film_id [PK] smallint	category_id [PK] smallint
1	5	8

UPDATE command

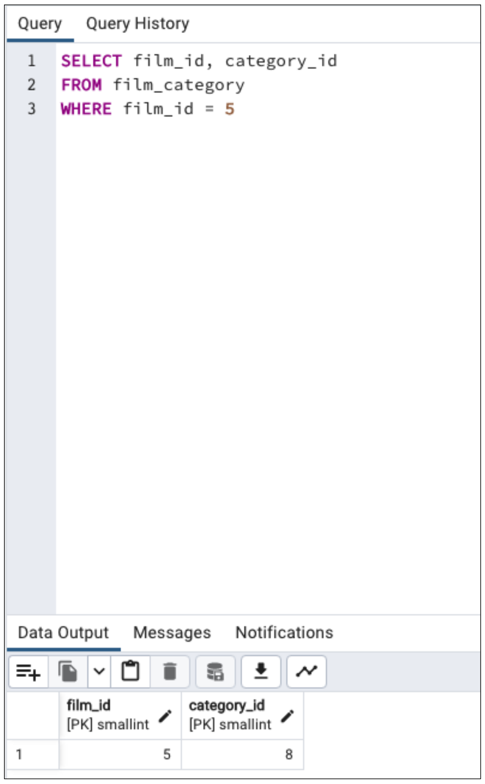
We want to change the movie African Eggs from Family (#8) to Thriller (#17)

Before



```sql
SELECT *
FROM category
```

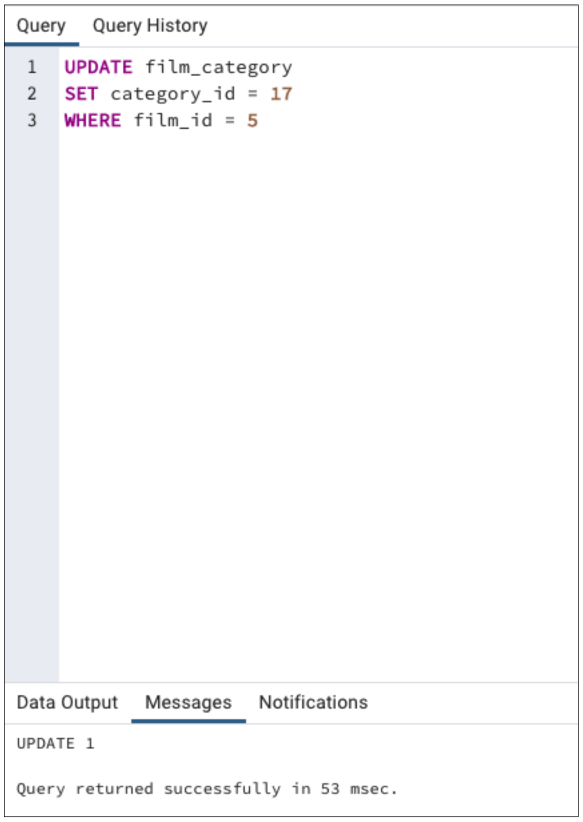
	category_id [PK] integer	name character varying (25)	last_update timestamp without time zone
1	1	Action	2006-02-15 09:46:27
2	2	Animation	2006-02-15 09:46:27
3	3	Children	2006-02-15 09:46:27
4	4	Classics	2006-02-15 09:46:27
5	5	Comedy	2006-02-15 09:46:27
6	6	Documentary	2006-02-15 09:46:27
7	7	Drama	2006-02-15 09:46:27
8	8	Family	2006-02-15 09:46:27
9	9	Foreign	2006-02-15 09:46:27
10	10	Games	2006-02-15 09:46:27
11	11	Horror	2006-02-15 09:46:27
12	12	Music	2006-02-15 09:46:27
13	13	New	2006-02-15 09:46:27
14	14	Sci-Fi	2006-02-15 09:46:27
15	15	Sports	2006-02-15 09:46:27
16	16	Travel	2006-02-15 09:46:27
17	17	Thriller	2023-01-24 11:22:34.796133



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

	film_id [PK] smallint	category_id [PK] smallint
1	5	8

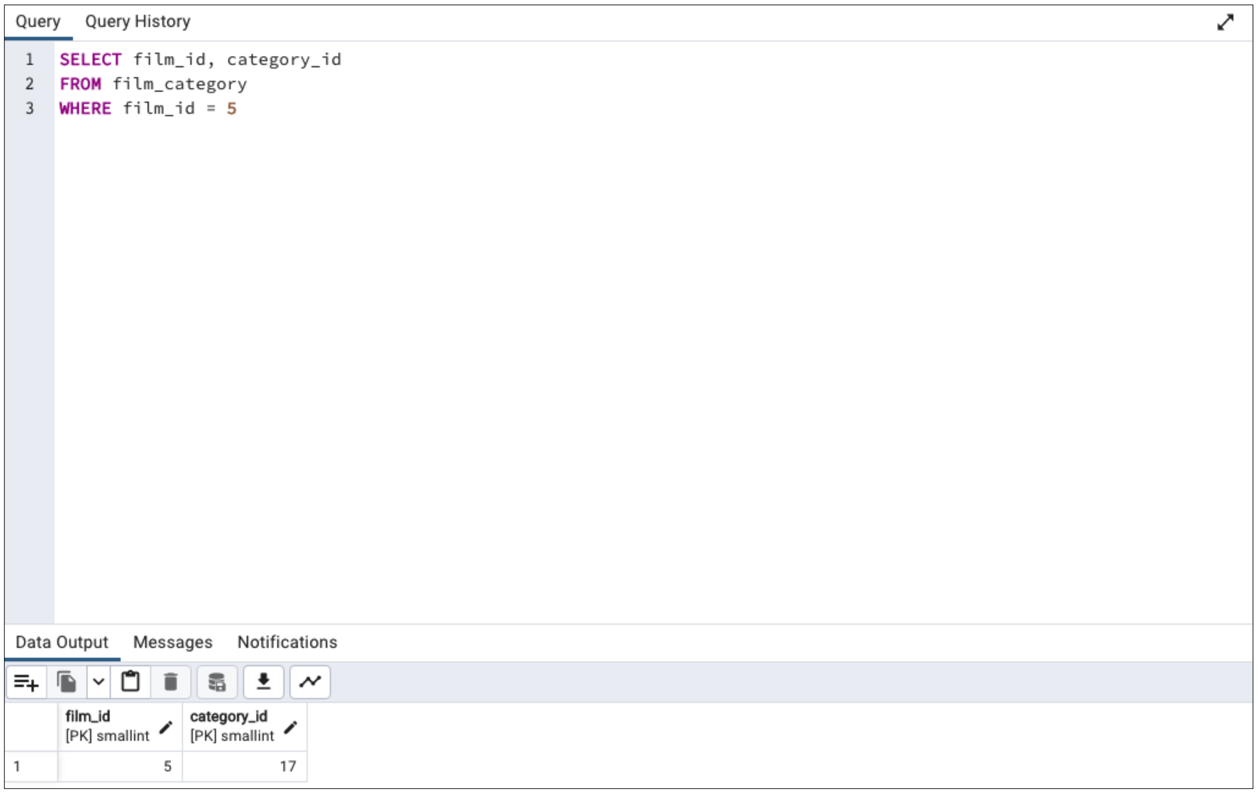
After



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

```
UPDATE 1

Query returned successfully in 53 msec.
```



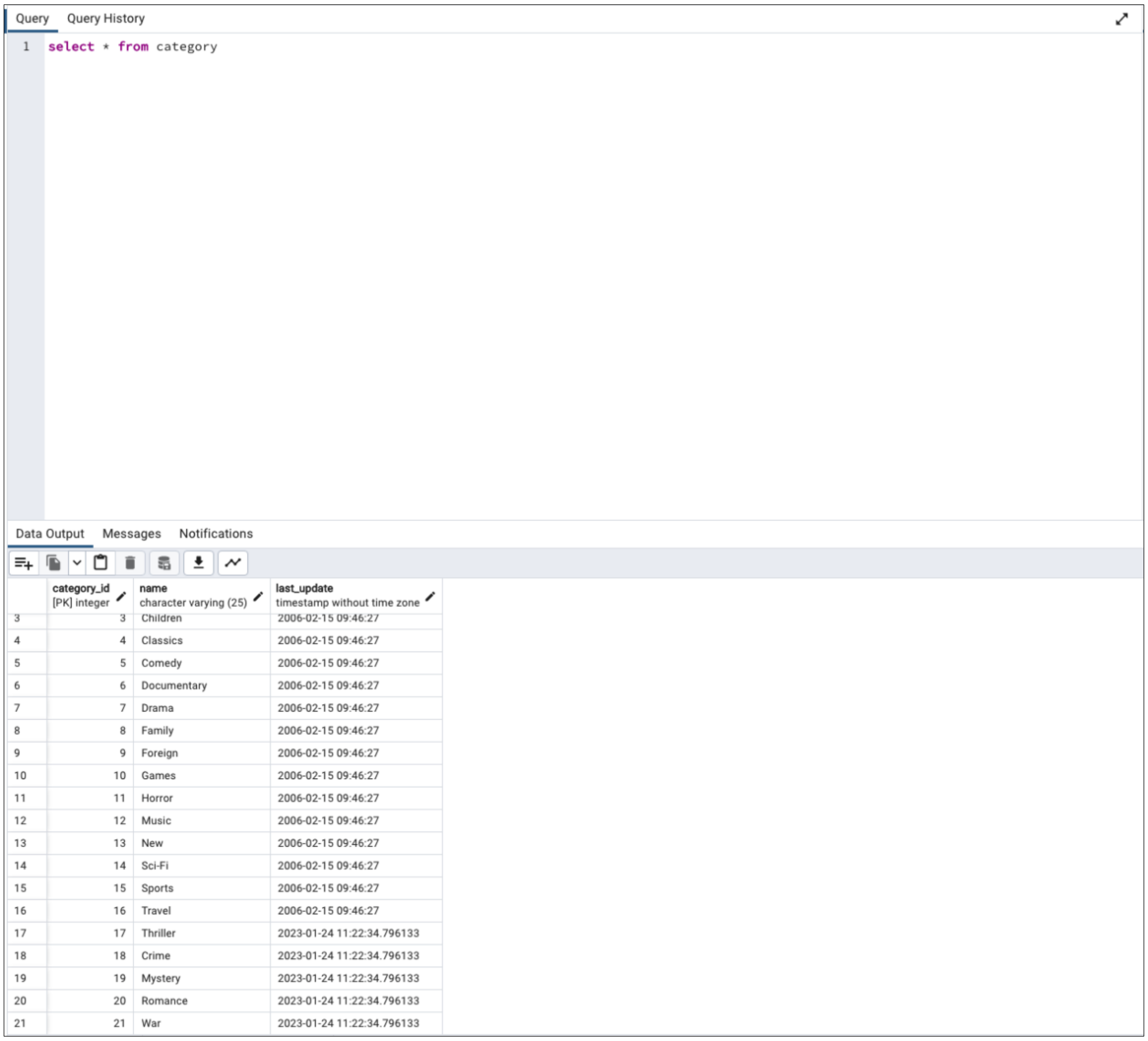
```
SELECT film_id, category_id
FROM film_category
WHERE film_id = 5
```

	film_id [PK] smallint	category_id [PK] smallint
1	5	17

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.

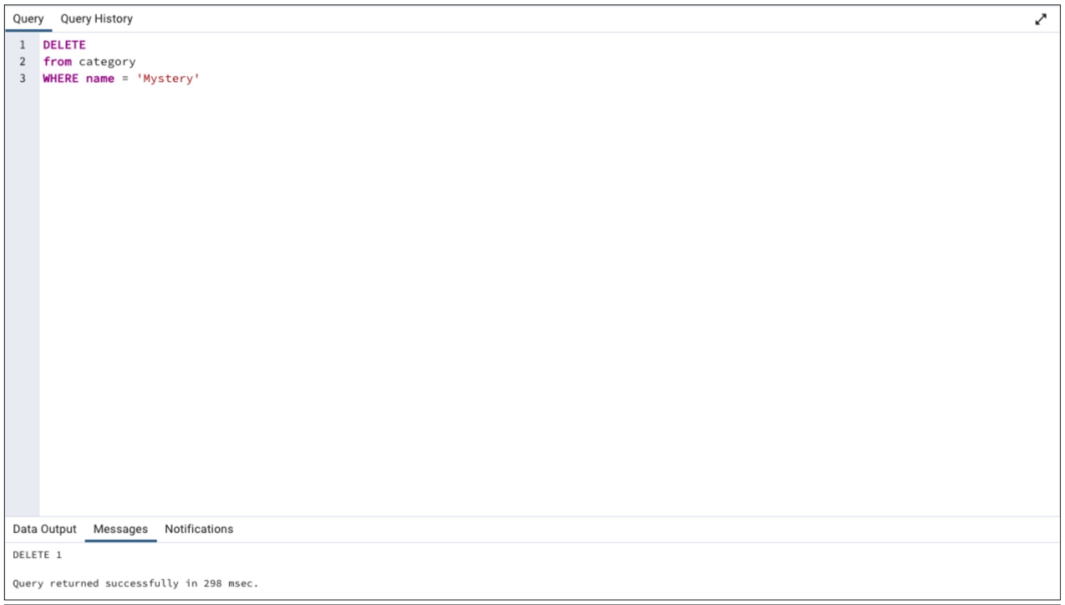
With Mystery



```
select * from category
```

	category_id [PK] integer	name character varying (25)	last_update timestamp without time zone
3	3	Children	2006-02-15 09:46:27
4	4	Classics	2006-02-15 09:46:27
5	5	Comedy	2006-02-15 09:46:27
6	6	Documentary	2006-02-15 09:46:27
7	7	Drama	2006-02-15 09:46:27
8	8	Family	2006-02-15 09:46:27
9	9	Foreign	2006-02-15 09:46:27
10	10	Games	2006-02-15 09:46:27
11	11	Horror	2006-02-15 09:46:27
12	12	Music	2006-02-15 09:46:27
13	13	New	2006-02-15 09:46:27
14	14	Sci-Fi	2006-02-15 09:46:27
15	15	Sports	2006-02-15 09:46:27
16	16	Travel	2006-02-15 09:46:27
17	17	Thriller	2023-01-24 11:22:34.796133
18	18	Crime	2023-01-24 11:22:34.796133
19	19	Mystery	2023-01-24 11:22:34.796133
20	20	Romance	2023-01-24 11:22:34.796133
21	21	War	2023-01-24 11:22:34.796133

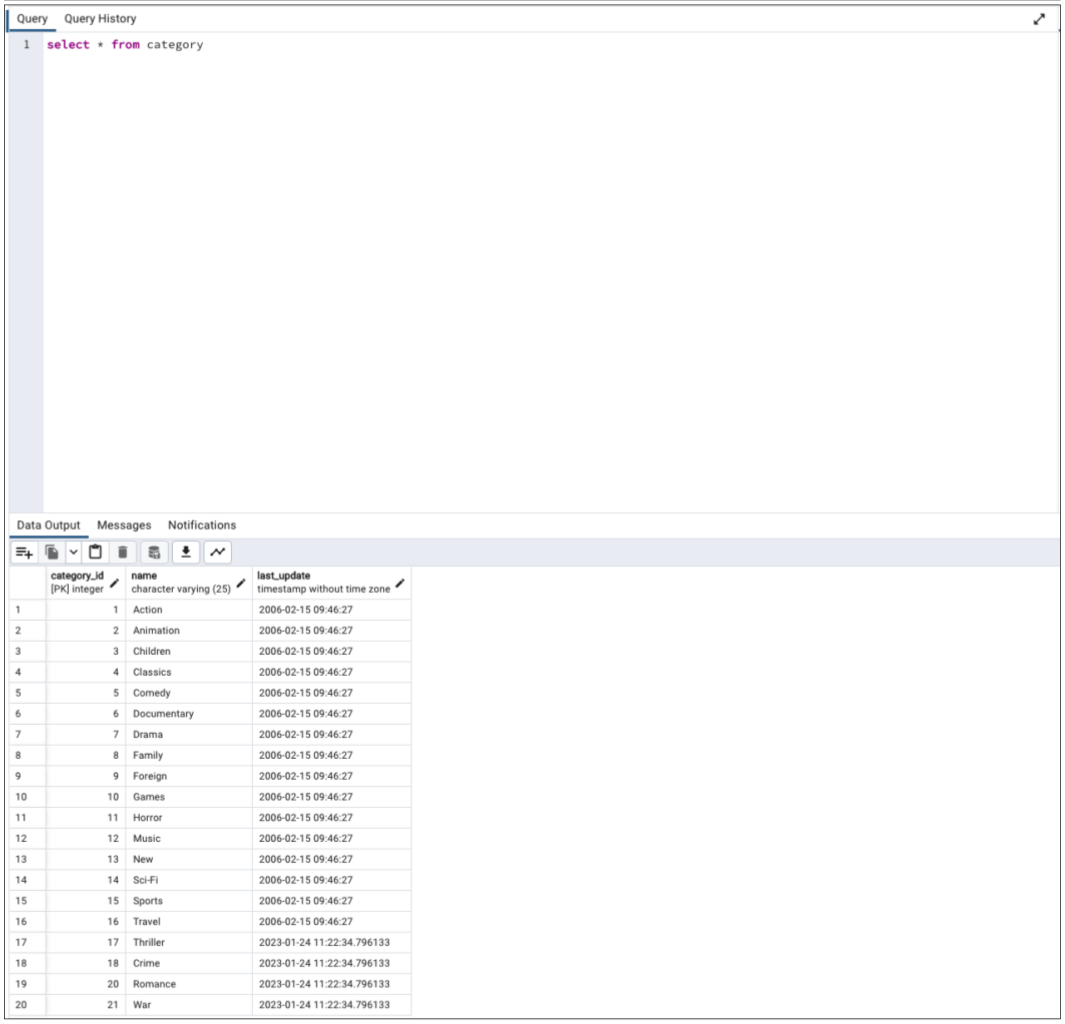
Without Mystery



```
DELETE
from category
WHERE name = 'Mystery'
```

```
DELETE 1

Query returned successfully in 298 msec.
```



```
select * from category
```

	category_id [PK] integer	name character varying (25)	last_update timestamp without time zone
1	1	Action	2006-02-15 09:46:27
2	2	Animation	2006-02-15 09:46:27
3	3	Children	2006-02-15 09:46:27
4	4	Classics	2006-02-15 09:46:27
5	5	Comedy	2006-02-15 09:46:27
6	6	Documentary	2006-02-15 09:46:27
7	7	Drama	2006-02-15 09:46:27
8	8	Family	2006-02-15 09:46:27
9	9	Foreign	2006-02-15 09:46:27
10	10	Games	2006-02-15 09:46:27
11	11	Horror	2006-02-15 09:46:27
12	12	Music	2006-02-15 09:46:27
13	13	New	2006-02-15 09:46:27
14	14	Sci-Fi	2006-02-15 09:46:27
15	15	Sports	2006-02-15 09:46:27
16	16	Travel	2006-02-15 09:46:27
17	17	Thriller	2023-01-24 11:22:34.796133
18	18	Crime	2023-01-24 11:22:34.796133
19	20	Romance	2023-01-24 11:22:34.796133
20	21	War	2023-01-24 11:22:34.796133

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.

The advantages of SQL:

Faster query processing – large amount of data is retrieved quickly and efficiently. Operations such as insertion, deletion, and manipulation of data is done instantly.

No coding skills – SQL does not require any substantial knowledge in coding. The program has some basic keywords such as SELECT, INSERT, and UPDATE that carry out tasks.

The disadvantages of SQL:

Complex interface – SQL can be a difficult interface for some users, which can at times create uncertainty while dealing with the database.

Cost Inefficient – Some versions are costly, which makes it difficult for programmers and analysts to use it.