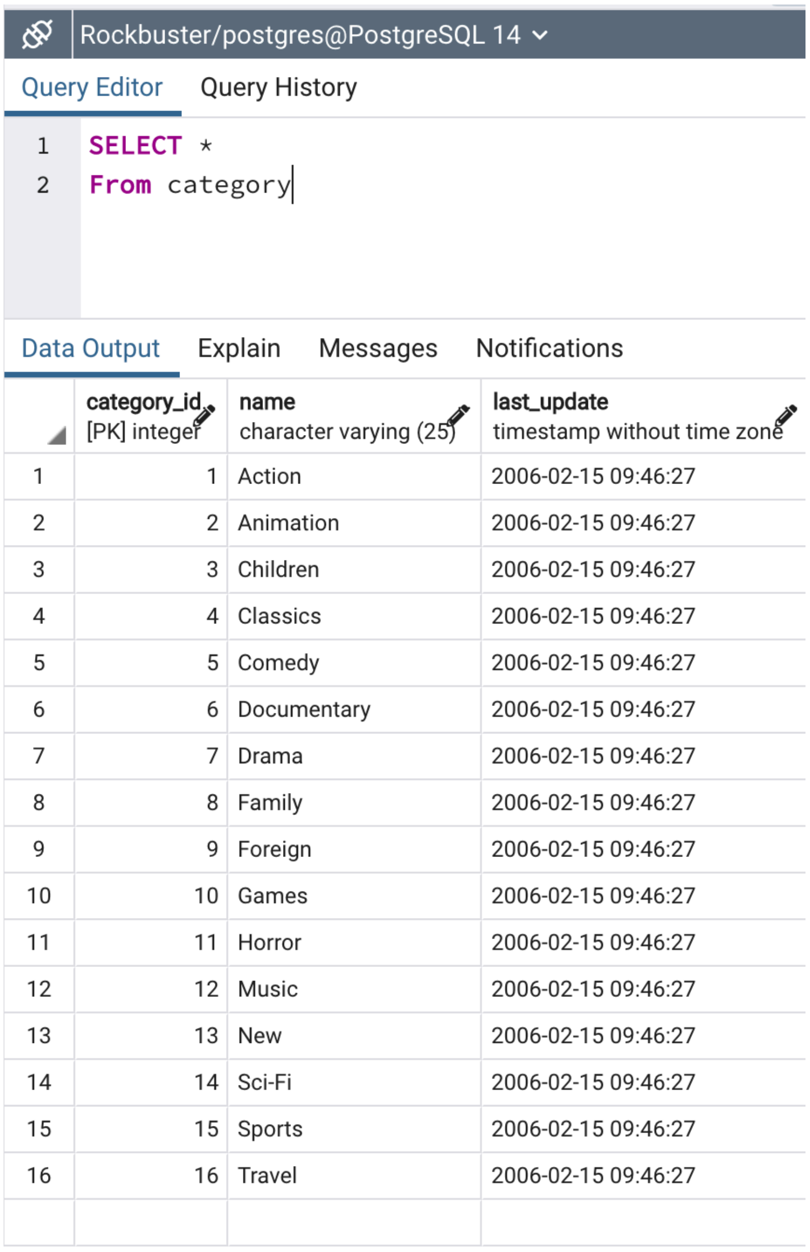
**Juan Ignacio Galvalisi**

## Exercise 3.3: SQL for Data Analyst

### Step 1

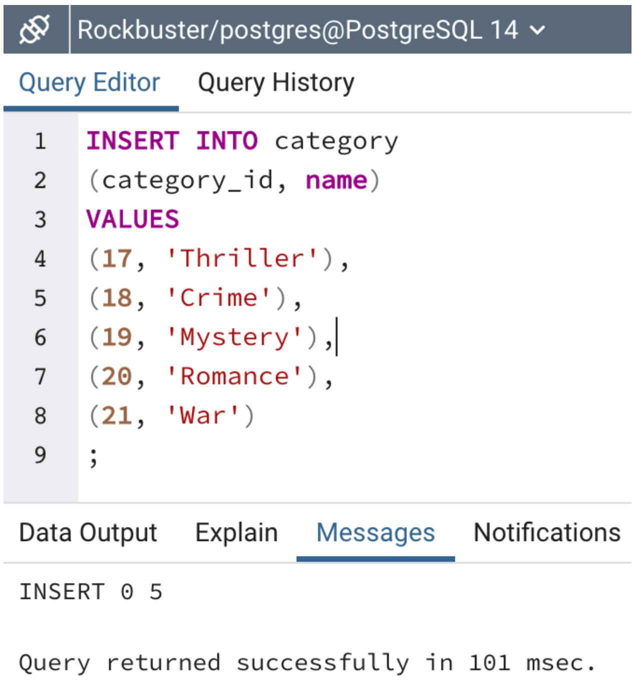


```
SELECT *
From category
```

| | 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 |

### 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:**

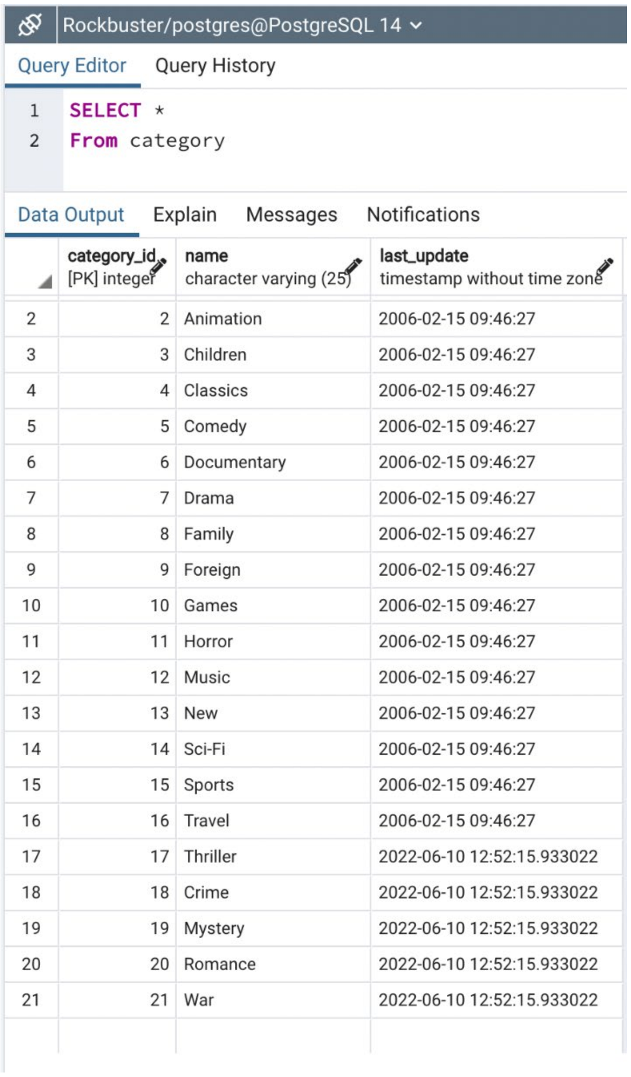

Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

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

Data Output | Explain | Messages | Notifications

```
INSERT 0 5

Query returned successfully in 101 msec.
```


Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```sql
SELECT *
From category
```

Data Output | Explain | Messages | Notifications

| | category_id [PK] integer | name character varying (25) | last_update timestamp without time zone |
|---|---|---|---|
| 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 | 2022-06-10 12:52:15.933022 |
| 18 | 18 | Crime | 2022-06-10 12:52:15.933022 |
| 19 | 19 | Mystery | 2022-06-10 12:52:15.933022 |
| 20 | 20 | Romance | 2022-06-10 12:52:15.933022 |
| 21 | 21 | War | 2022-06-10 12:52:15.933022 |

**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)
);
```

NOT NULL

- It's not allowed to have null values within the columns. If there is a null value, an error message will appear.
    - category_id: must be not null and integer.
    - name: must be not null and text.
    - last_update: must be not null and timestamp.

DEFAULT

- Establishes what value should be if no value is specified.
    - category_id: default as a next value.
    - last_update: set the time zone to the current date and time.

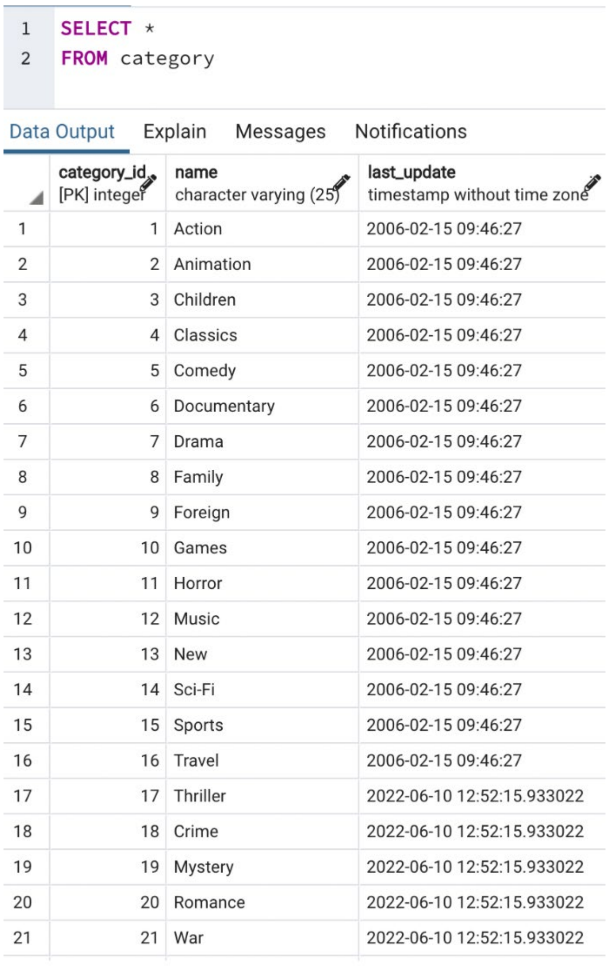
PRIMARY KEY

- category_pkey: Only allowed unique values and it cannot be empty.

## Step 3

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

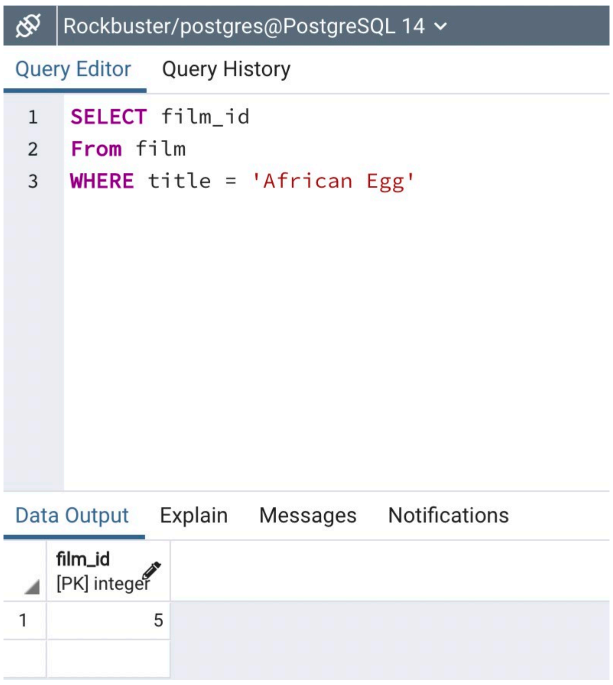
- **Write the SELECT statement to find the film_id for the movie *African Egg*.**



```
SELECT *
FROM category
```

Data Output | Explain | 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 | 2022-06-10 12:52:15.933022 |
| 18 | 18 | Crime | 2022-06-10 12:52:15.933022 |
| 19 | 19 | Mystery | 2022-06-10 12:52:15.933022 |
| 20 | 20 | Romance | 2022-06-10 12:52:15.933022 |
| 21 | 21 | War | 2022-06-10 12:52:15.933022 |



Rockbuster/postgres@PostgreSQL 14

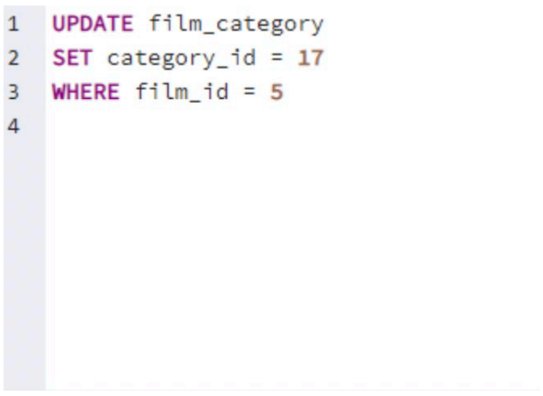
Query Editor | Query History

```
SELECT film_id
From film
WHERE title = 'African Egg'
```

Data Output | Explain | Messages | Notifications

| | film_id [PK] integer |
|---|---|
| 1 | 5 |

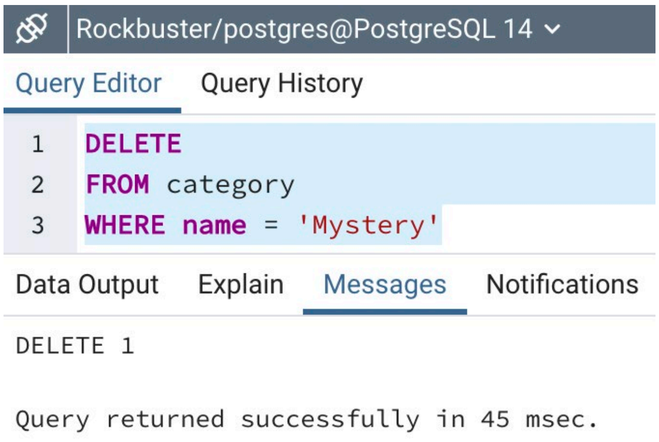
- **Once you have the film_ID and category_ID, write an UPDATE command to change the category in the film_category table (not the category table).**



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

## 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 1

Query returned successfully in 45 msec.
```

## Step 5

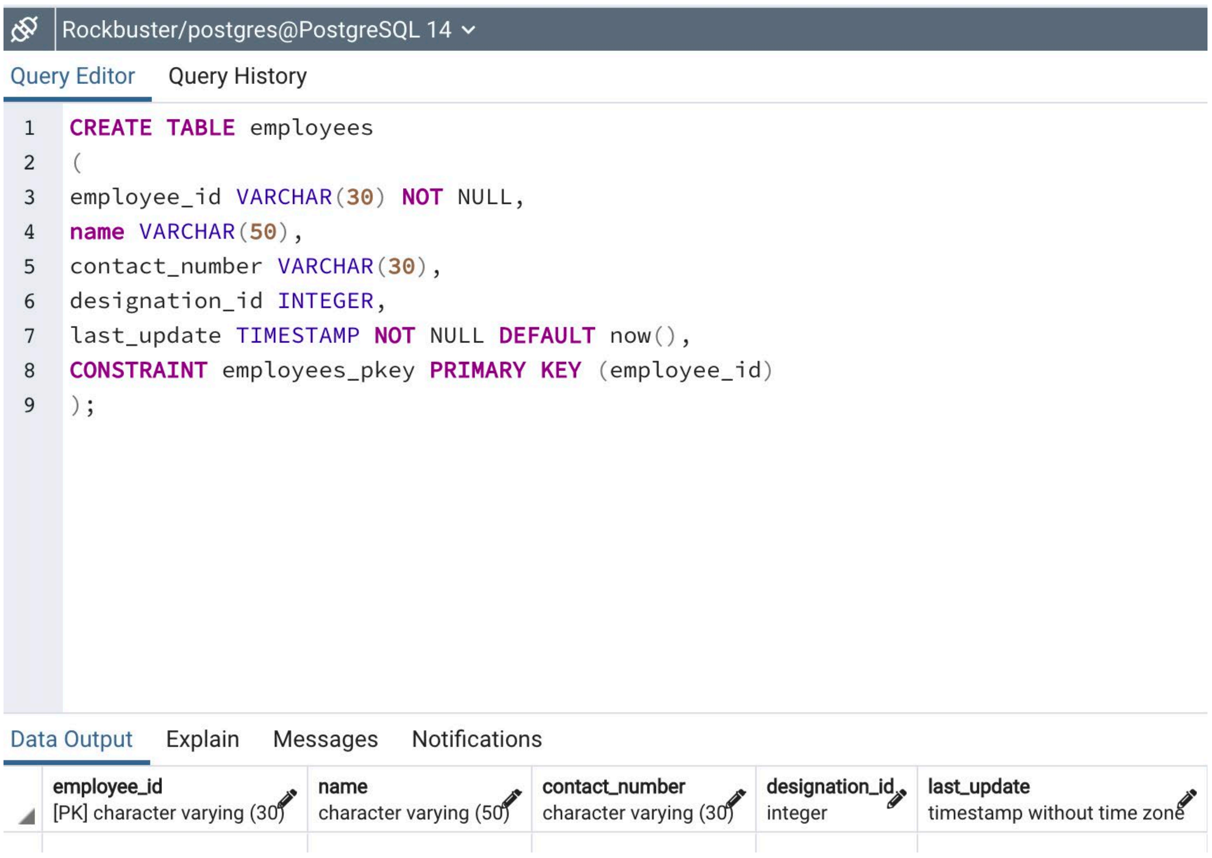
**Based on what you've learned so far, think about what it would be like to complete steps 1 to 4 with Excel instead of SQL. Are there any pros and cons to using SQL? Write a paragraph explaining your answer.**

It's challenging to define choosing only one between the two. Adding new data to our data would have been more tedious in Excel than in SQL, as it was with the new genres. However, updating the category_id of the movie "African Eggs" would have been easier in Excel than in SQL. At the same time, Excel's architecture is more user-friendly than SQL's. To work comfortably with SQL, you must have minimal coding concepts because the software is no longer as intuitive as Excel. If this dataset had been much more extensive, with millions of cells, these tasks would have been easier to perform in SQL than in Excel. It always depends on what you need at that moment, with the available data.

## Bonus Task

**The SQL query below contains some typos. See if you can fix it based on what you've learned so far about SQL and data types; then try running it in pgAdmin 4. If the query works, copy it into your Answers 3.3 document.**

```
CREATE TBL 3EMPLOYEES

 {

employee_id VARINT(30) NOT EMPTY

name VARCHAR(50),

contact_number VARCHAR(30) ,

designation_id INT,

last_update TIMESTAMP NOT NULL DEF now()

CONSTRAIN employee_pkey PRIMARY KEY (employee_id)

}
```


Rockbuster/postgres@PostgreSQL 14

Query Editor | Query History

```
CREATE TABLE employees
(
employee_id VARCHAR(30) NOT NULL,
name VARCHAR(50),
contact_number VARCHAR(30),
designation_id INTEGER,
last_update TIMESTAMP NOT NULL DEFAULT now(),
CONSTRAINT employees_pkey PRIMARY KEY (employee_id)
);
```

Data Output | Explain | Messages | Notifications

| employee_id [PK] character varying (30) | name character varying (50) | contact_number character varying (30) | designation_id integer | last_update timestamp without time zone |
|---|---|---|---|---|