

3.3: SQL for Data Analysts

Answers 3.3.

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.

SELECT *

FROM category name

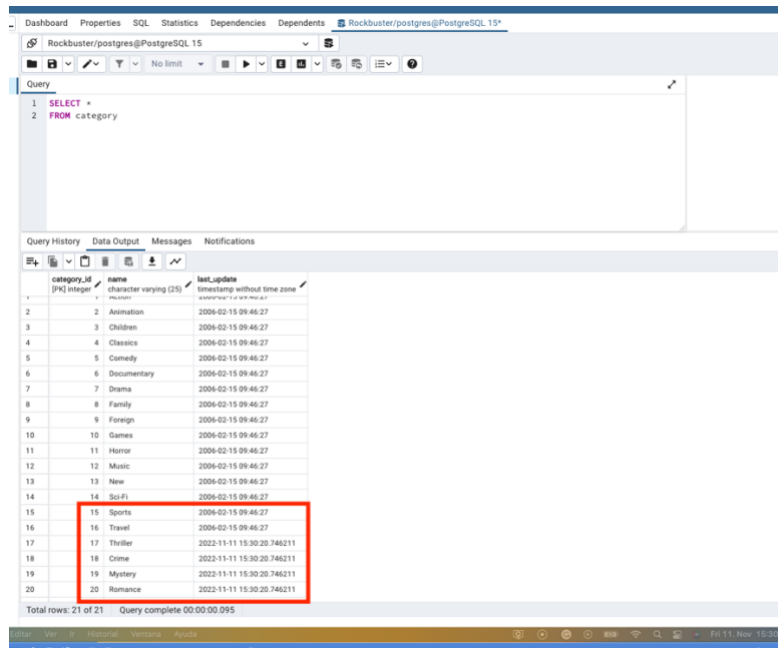
- Copy-paste the output into your answers document or write the answers out—it's up to you. Make sure to include the category ID for each genre.

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

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.



The screenshot shows a PostgreSQL query editor with a query window containing the following SQL:

```
1 SELECT *
2 FROM category
```

Below the query window, the 'Data Output' tab displays the results of the query. The results are shown in a table with the following columns: category_id, name, and last_update. The table contains 20 rows of data. The last row, with category_id 20 and name 'Romance', is highlighted with a red box.

category_id	name	last_update
1	Animation	2006-02-15 09:46:27
2	Children	2006-02-15 09:46:27
3	Classics	2006-02-15 09:46:27
4	Comedy	2006-02-15 09:46:27
5	Documentary	2006-02-15 09:46:27
6	Drama	2006-02-15 09:46:27
7	Family	2006-02-15 09:46:27
8	Foreign	2006-02-15 09:46:27
9	Games	2006-02-15 09:46:27
10	Horror	2006-02-15 09:46:27
11	Music	2006-02-15 09:46:27
12	Nature	2006-02-15 09:46:27
13	Sci-Fi	2006-02-15 09:46:27
14	Sports	2006-02-15 09:46:27
15	Taxel	2006-02-15 09:46:27
16	Thriller	2006-02-15 09:46:27
17	Crime	2006-02-15 09:46:27
18	Mystery	2006-02-15 09:46:27
19	Romance	2006-02-15 09:46:27
20	War	2006-02-15 09:46:27

INSERT INTO category (name)

VALUES ('Thriller'), ('Crime'), ('Mystery'), ('Romance'), ('War');

- 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)
);
```

Constraints are crucial in keeping the data organized and ensuring that the values in each column are consistently formatted. They can also help to ensure that values in a column are unique, not null, or even check that the values meet certain conditions.

This constraint contains 'NOT NULL DEFAULT' and ensures that a column can't have any empty or missing.

2. Category_id - The data type should be an integer, and it cannot be null

3. Name - The data type should be in text, and it cannot be null

4. Last_update - The data type should be the timestamp with time zone and cannot be null, and finally, 5. Primary key: The primary key gives each record in a table a unique ID.

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 film

WHERE title = 'African Egg'

Query

```
1 SELECT *
2 FROM film
3 WHERE title = 'African Egg'
4
```

Query History Data Output Messages Notifications

								
	film_id [PK] integer	title character varying (255)	description text	release_year integer				
1	5	African Egg	A Fast-Pac...	2006				

-
- Query:
- **SELECT** category_id

FROM film_category

WHERE film_id = 5

The screenshot shows a PostgreSQL query editor interface. At the top, the connection is 'Rockbuster/postgres@PostgreSQL 15'. Below the connection bar is a toolbar with various icons. The main area is titled 'Query' and contains the following SQL query:

```
1 SELECT category_id
2 FROM film_category
3 WHERE film_id = 5
4
```

Below the query editor is a tabbed interface with 'Query History', 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with one row of results. The table has two columns: 'category_id' and 'smallint'. The first row has the values '1' and '8'.

category_id	smallint
1	8

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_id= 5

Category_id= 8

Thriller ID= 17

The screenshot shows a PostgreSQL client interface with a top navigation bar (Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes) and a user profile (Rockbuster/). The main area is titled 'Query' and contains the following SQL code:

```
1 UPDATE film_category
2 SET category_id = 17
3 WHERE film_id = 5;
4
5 SELECT category_id
6 FROM film_category
7 WHERE film_id = 5
8
9
```

Below the query editor, there are tabs for 'Query History', 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with one row:

category_id
17

A green box highlights the 'category_id' column header and the value '17' in the first row. A green arrow points from the 'WHERE film_id = 5' clause in the query to the highlighted row in the results table.

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.

DELETE

FROM category

WHERE name = 'Mystery';

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.

I can see the advantage of using SQL to query data. This can be achieved faster, mainly if I use many tables simultaneously. The disadvantages are that you must learn all the queries and know the structure of the database to complete the task faster than Excel. In Excel, these types of tasks (searching IDs, Changing Ids, Adding values, etc.) can also be done quickly by using filters and advance search/ replace.

