

### 3.3: SQL FOR DATA ANALYSTS

#### 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.

The screenshot shows a SQL query editor with a query window and a data output window. The query window contains the following SQL command:

```
1 SELECT category_id, name FROM category
```

The data output window displays the results of the query as a table with two columns: **category\_id** (integer) and **name** (character varying (25)). The results are as follows:

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

Total rows: 16 of 16 Query complete 00:00:00.206

#### STEP 2

Write an INSERT statement to add the following genres to the category table: Thriller, Crime, Mystery, Romance, and War.

The screenshot shows a SQL query editor with a query window and a data output window. The query window contains the following SQL commands:

```
1 SELECT category_id, name FROM category
2
3 INSERT INTO category(name)
4 VALUES ('Thriller'),
5 ('Crime'),
6 ('Mystery'),
7 ('Romance'),
8 ('War')
9
10 SELECT * FROM category
```

The data output window displays the results of the query as a table with three columns: **category\_id** (integer), **name** (character varying (25)), and **last\_update** (timestamp without time zone). The results are as follows:

category_id	name	last_update
10	Games	2006-02-15 09:46:27
11	Horror	2006-02-15 09:46:27
12	Music	2006-02-15 09:46:27
13	New	2006-02-15 09:46:27
14	Sci-Fi	2006-02-15 09:46:27
15	Sports	2006-02-15 09:46:27
16	Travel	2006-02-15 09:46:27
17	Thriller	2023-02-05 23:20:20.749831
18	Crime	2023-02-05 23:20:20.749831
19	Mystery	2023-02-05 23:20:20.749831
20	Romance	2023-02-05 23:20:20.749831
21	War	2023-02-05 23:20:20.749831

Total rows: 21 of 21 Query complete 00:00:00.077

```
CREATE TABLE category
(
  category_id integer NOT NULL DEFAULT nextval('category_c
category_id_seq'::regclass),
  name text COLLATE pg_catalog."default" NOT NULL,
  last_update timestamp with time zone NOT NULL DEFAULT no
w()),
  CONSTRAINT category_pkey PRIMARY KEY (category_id)
);
```

The CREATE statement 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?

- NOT NULL - A constraint to make sure that there is no missing value in every column
- PRIMARY KEY - Category\_id is the primary key column. The primary key column can't contain any null or duplicate values

### STEP 3

The genre for the movie *African Egg* needs to be updated to thriller.

- Write the SELECT statement to find the film\_id for the movie *African Egg*.

13		14	SELECT category_ID, film_ID	9		10	SELECT film_id, title
14		15	FROM film_category	10		11	FROM film
15		16	WHERE film_ID='5'	11		12	WHERE title='African Egg'
16				12			
Data Output Messages Notifications				Data Output Messages Notifications			
<div> <div>+</div> <div>≡</div> <div>⌵</div> <div>📋</div> <div>🗑️</div> <div>🔄</div> <div>⬇️</div> <div>📶</div> </div>				<div> <div>+</div> <div>≡</div> <div>⌵</div> <div>📋</div> <div>🗑️</div> <div>🔄</div> <div>⬇️</div> <div>📶</div> </div>			
	category_id		film_id		film_id		title
	[PK] smallint		[PK] smallint		[PK] integer		character varying (255)
1	8		5	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).

18	UPDATE film_category	22	SELECT*FROM film_category
19	SET category_id=17	23	WHERE film_id=5
20	WHERE film_id=5		
21			
22			
Data Output Messages Notifications		Data Output Messages Notifications	
UPDATE 1		<div> <div>+</div> <div>≡</div> <div>⌵</div> <div>📋</div> <div>🗑️</div> <div>🔄</div> <div>⬇️</div> <div>📶</div> </div>	
Query returned successfully in 666 msec.		<div> <div>+</div> <div>≡</div> <div>⌵</div> <div>📋</div> <div>🗑️</div> <div>🔄</div> <div>⬇️</div> <div>📶</div> </div>	
	film_id		category_id
	[PK] smallint		[PK] smallint
1	5	17	last_update
			timestamp without time zone
			2023-02-06 00:02:27.78945

## 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.

```
25 DELETE
26 FROM category
27 WHERE name='Mystery'
```

Data Output

Messages

Notifications

DELETE 1

Query returned successfully in 363 msec.

	category_id [PK] integer	name character varying (25)	last_update timestamp without time zone
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	2023-02-05 23:20:20.749831
18	18	Crime	2023-02-05 23:20:20.749831
19	20	Romance	2023-02-05 23:20:20.749831
20	21	War	2023-02-05 23:20:20.749831

## STEP 5

Considering the small size of the *category* and *film\_category* tables, Excel still seems to be easier to manipulate, edit and delete data.

SQL would be more beneficial for datasets with more rows where more scrolling would be required in Excel.

## BONUS TASK

The SQL query 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.

```
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)
}
```

```
31 CREATE TABLE employees
32 (
33   employee_ID SERIAL PRIMARY KEY,
34   name VARCHAR(50),
35   contact_number VARCHAR(30),
36   designation_id INT,
37   last_update TIMESTAMP NOT NULL DEFAULT NOW()
38 )
```

Data Output

Messages

Notifications

CREATE TABLE

Query returned successfully in 79 msec.