

### Answers 3.3

Your line manager has told you they'd like to improve the movie search filters on the Rockbuster app and website. After thinking about the best way to do this, you decide to add more film categories so users can browse more genres they're interested in.

#### Directions

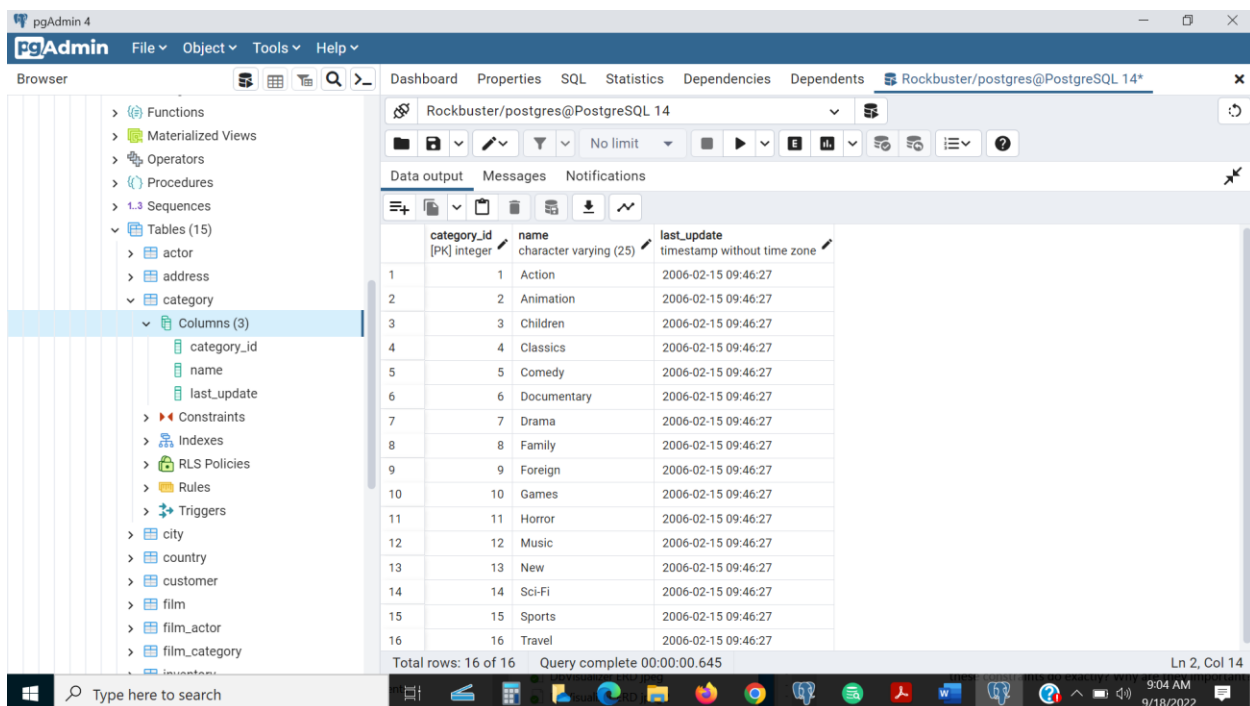
Create a new text document and call it "Answers 3.3." You'll submit your queries, outputs, and written answers in this document at the end of the task.

#### 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
```



The screenshot shows the pgAdmin 4 interface. On the left, the 'Rockbuster' database is selected, and the 'category' table is highlighted under 'Tables (15)'. The 'Columns (3)' for the 'category' table are listed: 'category\_id', 'name', and 'last\_update'. The main pane displays the data for the 'category' table, showing 16 rows. The status bar at the bottom indicates 'Total rows: 16 of 16' and 'Query complete 00:00:00.645'.

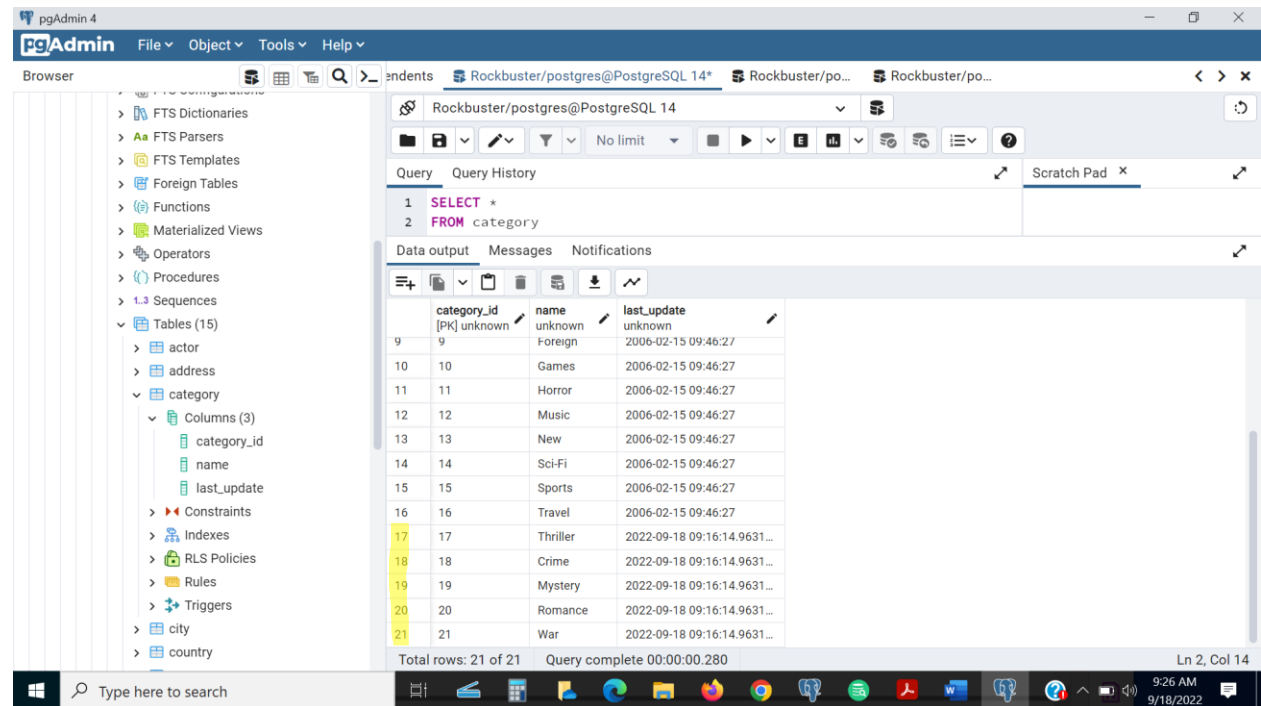
| category_id [PK] integer | name character varying (25) | last_update timestamp without time zone |
|--------------------------|-----------------------------|---|
| 1                        | Action                      | 2006-02-15 09:46:27                     |
| 2                        | Animation                   | 2006-02-15 09:46:27                     |
| 3                        | Children                    | 2006-02-15 09:46:27                     |
| 4                        | Classics                    | 2006-02-15 09:46:27                     |
| 5                        | Comedy                      | 2006-02-15 09:46:27                     |
| 6                        | Documentary                 | 2006-02-15 09:46:27                     |
| 7                        | Drama                       | 2006-02-15 09:46:27                     |
| 8                        | Family                      | 2006-02-15 09:46:27                     |
| 9                        | Foreign                     | 2006-02-15 09:46:27                     |
| 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                     |

#### 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. [Rows 17-21 added](#)

```
INSERT INTO category
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 help keep data organized, specify what type of data a table or column can accept, are typically set when table is created. If constraints are done well, they help with performance and can act as data quality checks.

NOT NULL DEFAULT – ensures that a column can't have any empty or missing values

PRIMARY KEY – gives each record in a table a unique ID. This key cannot contain any null or duplicate values.

- category\_id – primary key, data type integer (Int2), cannot be null
- name – data type text, cannot be null
- last\_update – data type timestamp, cannot be null

### 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'
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'film' table is selected under the 'Columns (13)' section. The main pane displays a SQL query and its results.

Query:

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

Data output:

| film_id | title       | description   | release_year | language_id | rental_duration | rental_rate | length | rep |
|---------|-------------|---------------|--------------|-------------|-----------------|-------------|--------|-----|
| 5       | African Egg | A Fast-Pac... | 2006         | 1           | 6               | 2.99        | 130    | nur |

Total rows: 1 of 1 Query complete 00:00:00.406 Ln 3, Col 28

- 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.

```
SELECT *  
FROM film_category  
WHERE film_id = 5
```

**\*\*category\_id = 8\*\***

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'film' table selected. The 'Columns (13)' for the 'film' table are listed: film\_id, title, description, release\_year, language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating, last\_update, special\_features, and fulltext. The 'film\_id' column is highlighted. The main query window shows the following SQL query:

```
1 SELECT *
2 FROM film_category
3 WHERE film_id = 5
```

The 'Data output' tab shows the results of the query. The table has three columns: film\_id, category\_id, and last\_update. The results are as follows:

| film_id | category_id | last_update         |
|---------|-------------|---------------------|
| 1       | 5           | 2006-02-15 10:07:09 |

The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.106'.

UPDATE film\_category  
SET category\_id = 3  
WHERE film\_id = 5

Updated film category id from 8, which is family, to category id 3 which is children.

The screenshot shows the pgAdmin 4 interface after the update. The left sidebar displays the database structure, with the 'film\_category' table selected. The 'Columns (3)' for the 'film\_category' table are listed: category\_id, name, and last\_update. The 'category\_id' column is highlighted. The main query window shows the following SQL query:

```
1 SELECT *
2 FROM film_category
3 WHERE film_id = 5
```

The 'Data output' tab shows the results of the query. The table has three columns: film\_id, category\_id, and last\_update. The results are as follows:

| film_id | category_id | last_update                |
|---------|-------------|----------------------------|
| 1       | 3           | 2022-09-18 12:46:10.368119 |

The status bar at the bottom indicates 'Total rows: 1 of 1' and 'Query complete 00:00:00.189'.

#### 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.

Based on what I've learned from both EXCEL and SQL, SQL for you can find and update data more easily with SQL than functions in EXCEL. Less time with SQL executing and commands and do not have to scroll through spreadsheets for data, with SQL you can directly access the table for the data.

#### Step 6:

Save your "Answers 3.3" document as a PDF and upload it here for your tutor to review.

#### 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.

If you get this you're a SQL champ!

Items to be corrected identified by red

```
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)
}
```

Correct query:

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