Data Immersion 3.3

Answers 3.3

Step 1

Query Query History

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

Data Output Messages Notifications

	category_id [PK] integer	name character varying (25)
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

Total rows: 16 of 16 Query complete 00:00:01.463

Step 2

Query Query History

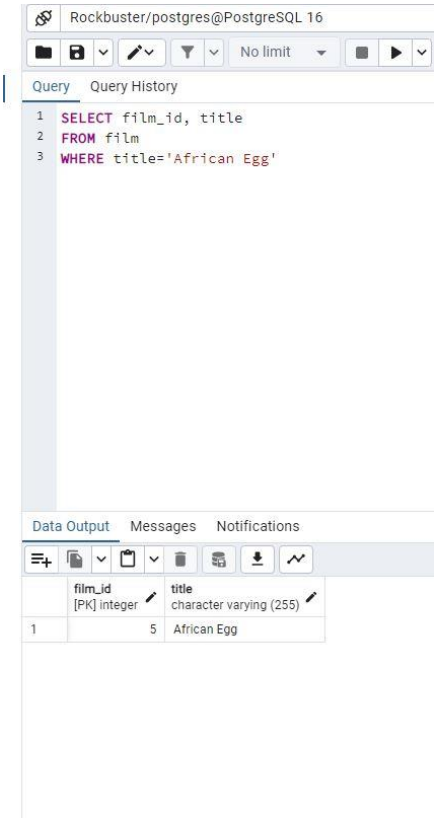
```
INSERT INTO category (name)
VALUES ('Thriller'), ('Crime'), ('Romance'),
('Mystery'), ('War')
```

Data Output Messages Notifications

	category_id [PK] integer	name character varying (25)
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
17	17	Thriller
18	18	Crime
19	19	Romance
20	20	Mystery
21	21	War

In CREATE TABLE statement for the category table, each column is subject to specific constraints to enforce data integrity. The category_id column is constrained to be an integer that can never be null, ensuring that each record must have an identifier. Additionally, it automatically gets a unique, sequentially generated default value, making it ideal for a primary key. The **name column** must be a text value and cannot be null, guaranteeing every category has a descriptive label. The **last_update** column, which records when the record was last modified, is required to have a value (not null) and defaults to the time of data insertion. The primary key constraint applied to **category_id** solidifies its role as a unique identifier within the table. These constraints are fundamental as they ensure the presence and uniqueness of critical data, which is crucial for the accurate identification of records and relational database operations.

Step 3



Query Query History

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

Data Output Messages Notifications

	film_id [PK] smallint	category_id [PK] smallint
1	5	26

Query Query History

```
UPDATE film_category
SET category_id=27
WHERE film_id=5
```

Data Output Messages Notifications

	film_id [PK] smallint	category_id [PK] smallint
1	5	26

Data Output Messages Notifications

	film_id [PK] smallint	category_id [PK] smallint
1	5	27

Step 4

Query Query History

```
1 DELETE FROM category
2 WHERE NAME= 'mystery'
3
```

Data Output Messages Notifications

```
DELETE 0

Query returned successfully in 83 msec.
```

Step 5

Performing database operations like selecting, updating, and deleting records in SQL compared to Excel presents a set of pros and cons. SQL databases are designed for efficient data management and complex querying, particularly with large data sets. They offer robustness in terms of transaction management, concurrency control, and data integrity. For instance, SQL queries can swiftly handle millions of records with complex filters and joins, something that would be extremely cumbersome and potentially unfeasible in Excel.

On the other hand, I definitely feel Excel provides a more user-friendly interface, with drag-and-drop features, easy data entry, and powerful visualisation tools right out of the box, making it more accessible for users without programming experience. Excel is excellent for analysing smaller datasets and creating charts or graphs for presentations.