

Step 1:

	category_id [PK] integer	name character varying (25)
1	1	Action
2	2	Animation
3	3	Children
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

Step 2:

The screenshot shows a PostgreSQL client window titled "Rockbuster/postgres@PostgreSQL 15". The interface includes a toolbar with various icons for file operations, query execution, and settings. The "Query" tab is active, displaying the following SQL statement:

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

Below the query editor, the "Data Output" tab is selected, showing the result of the query:

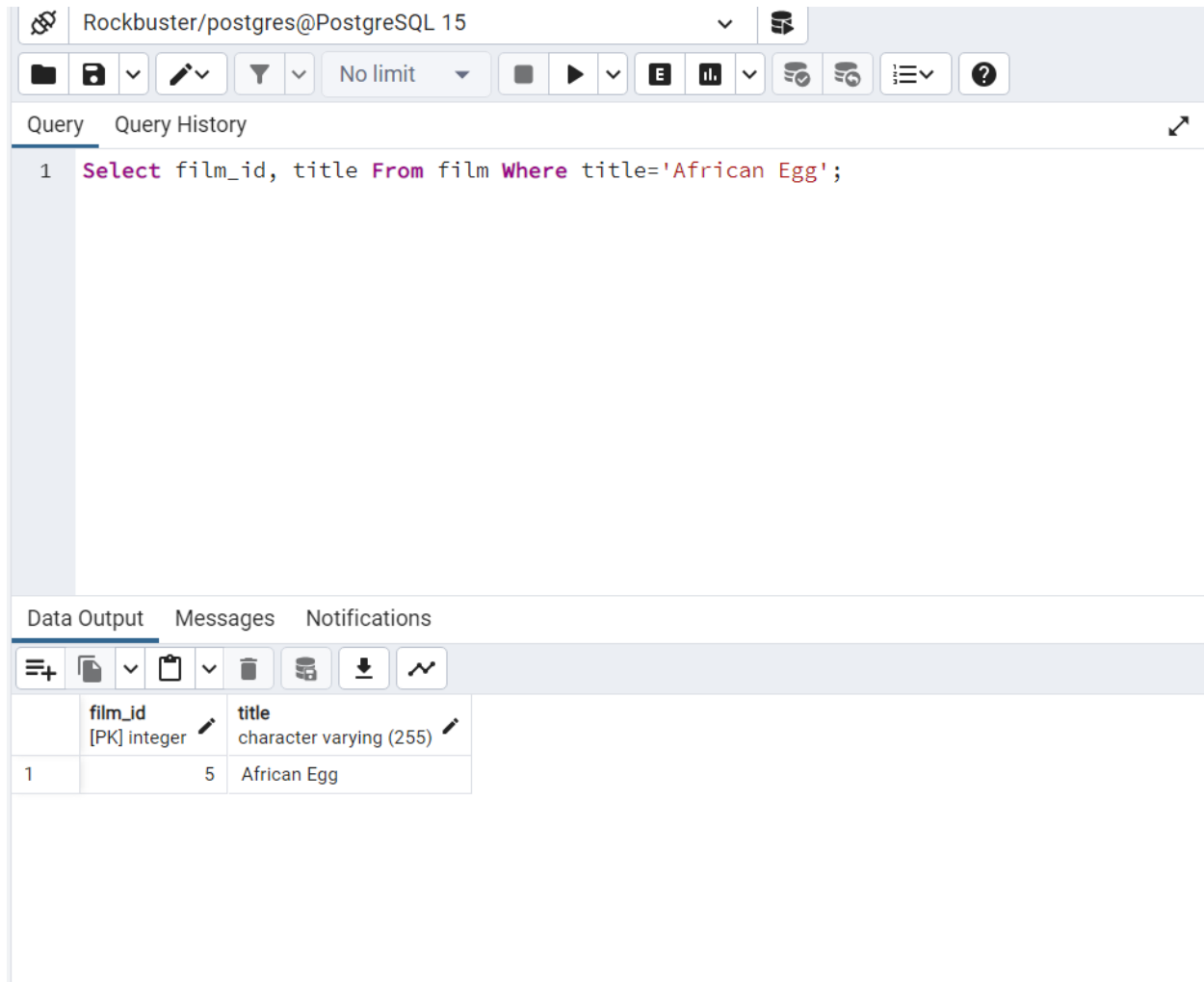
```
INSERT @ 4  
  
Query returned successfully in 62 msec.
```

The status bar at the bottom indicates "Total rows: 16 of 16", "Query complete 00:00:00.062", and "Ln 2, Col 25".

The NOT NULL constraint being applied to category_id column works to create the rule that no missing or empty values are in the category-id column. The category_id column also has the PRIMARY KEY

constraint applied to it. This rule is important because the `category_id` column is the primary key which is used to find specific information and means that this column has a different value for each row in the table.

Step 3:



The screenshot shows a PostgreSQL query editor interface. At the top, the connection is identified as 'Rockbuster/postgres@PostgreSQL 15'. Below the connection bar is a toolbar with various icons for file operations, query execution, and settings. The 'Query' tab is active, displaying the following SQL query:

```
1 Select film_id, title From film Where title='African Egg';
```

Below the query editor, the 'Data Output' tab is active, showing the results of the query in a table format. The table has two columns: 'film_id' (integer, primary key) and 'title' (character varying (255)). The results show a single row with film_id 5 and title 'African Egg'.

	film_id [PK] integer	title character varying (255)
1	5	African Egg

Query Query History

1 **Select** category_id, name **From** category **WHERE** name= 'Thriller';

Data Output Messages Notifications



	category_id [PK] integer	name character varying (25)
1	17	Thriller

Query Query History



```
1 Update film_category
2 Set category_id=17 Where film_id=5;
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 101 msec.

Step 4:

Query	Query History
1	<code>DELETE from category WHERE name='Mystery';</code>

Data Output	Messages	Notifications
<code>DELETE 0</code>		
<code>Query returned successfully in 49 msec.</code>		

Step 5:

If I used excel rather than SQL to complete the processes above it would have taken more time to complete the process. This is because there wouldn't be a single command that can be used in excel to complete these steps so I would have to manually delete and ctrl f to find values. However if I was working with a smaller set of data then excel would be preferred.