Answers 3.3.

1)

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

2a)

First SQL command:

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

Then, to retrieve only the newly inserted rows for verification:

```
SELECT * FROM category WHERE name IN ('Thriller', 'Crime', 'Mystery', 'Romance', 'War');
```

category_id	name	last_update
17	Thriller	2025-01-03 17:47:58.036689
18	Crime	2025-01-03 17:47:58.036689
19	Mystery	2025-01-03 17:47:58.036689
20	Romance	2025-01-03 17:47:58.036689
21	War	2025-01-03 17:47:58.036689
22	Thriller	2025-01-03 17:48:40.710627
23	Crime	2025-01-03 17:48:40.710627

24	Mystery	2025-01-03 17:48:40.710627
25	Romance	2025-01-03 17:48:40.710627
26	War	2025-01-03 17:48:40.710627

2b)

The CREATE TABLE statement includes several constraints to ensure the data in the category table is accurate, consistent, and reliable. Here's what each constraint does and why it's important:

- **category_id**

Constraints:

1. **NOT NULL:** Ensures every category has an ID, which is essential for uniquely identifying rows.
2. **DEFAULT nextval**(..): Automatically generates a unique ID for each new row, preventing duplicate IDs and reducing manual errors.
3. **PRIMARY KEY**: Guarantees that the category_id column is unique and serves as the main identifier for the table.

These constraints make category_id a reliable **unique identifier**, critical for linking this table to other tables in the database.

- **name:**

Constraints:

1. **NOT NULL**: Ensures every category has a name, preventing rows with missing or incomplete information.
2. **COLLATE** : Ensures text is handled consistently, especially during sorting or comparisons.

These constraints ensure that each category is properly labeled and easily searchable.

- **last_update:**

Constraints:

1. **NOT NULL:** Ensures the column always has a timestamp, preventing missing values.
2. **DEFAULT now**(): Automatically records the current date and time when a new row is added, making data entry easier and more consistent.

This ensures accurate tracking of when each category was last modified, which is useful for auditing and updates.

-Why Are Constraints Important?

Constraints enforce rules on the data to:

- Prevent errors (like blank or duplicate entries).
- Maintain the integrity and reliability of the data.
- Ensure the table is well-structured and works efficiently with other tables in the database.

By applying these constraints, the category table becomes robust, easy to use, and reliable for storing and retrieving movie genre information.

Step 3)

SQL Query:

SELECT film_id, title

FROM film

WHERE title = 'African Egg';

film_id	title
5	African Egg

Now that we know the film_id for "African Egg" is 5, and we want to find the category_id for the genre Thriller and we run the following query:

SELECT category_id, name

FROM category

WHERE name = 'Thriller';

Outcome :

category_id	**name**
17	Thriller
22	Thriller

I decide to choose category_id = 17, and run the UPDATE command:

UPDATE film_category

SET category_id = 17

WHERE film_id = 5;

Then to verify the UPDATE, I run

SELECT *

FROM film_category

WHERE film_id = 5;

Which results correctly :

film_id	category_id	last_update
5	17	2025-01-03 18:33:00.14494

Step 4)

To identify Mystery category:

SELECT category_id, name

FROM category

WHERE name = 'Mystery';

The query output shows that there are two rows for the Mystery category with category_id values of 19 and 24. To delete the Mystery category, I decided whether to delete both rows.

To do this, I run first a command :

SELECT *

FROM film_category

WHERE category_id IN (19, 24);

This in order to check if any movies are associated with these categories, run :

SELECT *

FROM film_category

WHERE category_id IN (19, 24);

Which shows there are no dependencies, so I run this DELETE command:

DELETE FROM category

WHERE category_id IN (19, 24);

To verify the deletion, I run :

SELECT *

FROM category

WHERE name = 'Mystery';

5)

Using SQL for steps 1 to 4 offers significant advantages over Excel, especially for large datasets and complex operations. SQL is specifically designed to handle relational databases efficiently, making it ideal for tasks like querying specific records, updating data in multiple tables and enforcing constraints such as primary keys. SQL allows for dynamic updates and joins across tables, which would be impossible to replicate in Excel without significant manual effort or advanced knowledge of Excel formulas. On the other hand, Excel might seem more intuitive for small datasets because of its visual interface, which allows users to manipulate data manually. However, Excel is more prone to errors, lacks scalability, and makes tracking changes difficult, especially when dealing with large datasets or collaborative workflows. While, SQL provides the additional advantage of automation and reproducibility, where queries can be saved, reused, and applied consistently, ensuring data integrity.