Lab 003: Solution

COMP1048: Databases and Interfaces (2024-2025)

Matthew Pike & Yuan Yao

Table of contents

1 Lab Overview

This week, you are not required to `CREATE` or `INSERT` any data into the database. Instead, you will be provided with a pre-populated database. Please download the database file from the DBI Moodle page and save it to your computer. You will then need to open the database file using the SQLite command line tool via the CLI, just as you did last week.

> **Improving Output Readability**
>
> You can improve the readability of your output by using the `.headers on` command, which will display the column names. Additionally, you can use the `.mode column` command to display the output in a table format, but note that this may truncate the output if it is too wide.

Before you begin, take a moment to familiarise yourself with the database schema. You can do this by typing `.schema` and pressing enter, which will display the structure of the database.

2 Exercises

1. Display the movie title, genre, and year for all movies. Results should be sorted by year in ascending order. Your output should look like this:

Movie_Title	Genre	Year_of_Release
...	...	...

Solution

```
SELECT
    movTitle    AS "Movie_Title",
    movGenre    AS "Genre",
    movYear     AS "Year_of_Release"
FROM
    Movie
ORDER BY
    movYear ASC;
```

2. Display all movies that are comedy or drama that were released after 2000. Your output should look like this:

Movie_Title	Year_of_Release
...	...

Solution

```
SELECT
    movTitle    AS "Movie_Title",
    movYear     AS "Year_of_Release"
FROM
    Movie
WHERE
    movGenre IN ('Comedy', 'Drama') AND movYear > 2000;
```

3. Display the movie title and year for all movies released in which Barry Nelson or Daniel Craig acted. Your output should look like this:

Movie_Title	Year_of_Release
...	...

Solution

```
SELECT
    movTitle    AS "Movie_Title",
    movYear     AS "Year_of_Release"
FROM
    Movie
WHERE
    actId IN (
        SELECT actId
        FROM Actor
        WHERE actName IN ('Barry Nelson', 'Daniel Craig')
    );
```

4. Calculate the average price of all movies. Your output should contain one row with a single column containing the average price. Your output should look like this:

Average_Price
...

Solution

```
SELECT
    AVG(movPrice) AS "Average_Price"
FROM
    Movie;
```

5. Calculate the average movie price for each genre. Average prices should be rounded to two decimal places. Your output should look like this:

Genre	Average_Price
...	...

Solution

```
SELECT
    movGenre    AS "Genre",
    ROUND(AVG(movPrice), 2) AS "Average_Price"
FROM
    Movie
GROUP BY
    movGenre;
```

6. Calculate the number of movies for each genre. Order the results in descending order according to the number of movies. Your output should look like this:

Genre	Number_of_Movies
...	...

Solution

```
SELECT
    movGenre    AS "Genre",
    COUNT(movId) AS "Number_of_Movies"
FROM
    Movie
GROUP BY
    movGenre
ORDER BY
    COUNT(movId) DESC;
```

7. If a genre contains more than two movies, list all movies that belong to that genre. Your output should look like this:

Movie_Title	Genre
...	...

Solution

```
SELECT
    movTitle    AS "Movie_Title",
    movGenre    AS "Genre"
FROM
    Movie
WHERE
    movGenre IN (
        SELECT movGenre
        FROM Movie
        GROUP BY movGenre
        HAVING COUNT(movId) > 2
    );
```

8. List all movies that share a genre with any movie that Barry Nelson acted in. Your output should look like this:

Movie_Title	Genre
...	...

Solution

```
SELECT
    movTitle    AS "Movie_Title",
    movGenre    AS "Genre"
FROM
    Movie
WHERE
```

```
    movGenre IN (
        SELECT movGenre
        FROM Movie
        WHERE actId IN (
            SELECT actId
            FROM Actor
            WHERE actName = 'Barry Nelson'
        )
    );
```

9. Generate a headline for each movie that includes the movie title, genre, and year. The headline should be in the format: “Movie Title(Genre) - Year”. An example headline would be: “Die Hard with a Vengeance(Action)-1999”. Results should be returned as a single column containing the headline. Results should be ordered by the length of the headline in descending order. Your output should look like this:

Headline
...

Solution

```
SELECT
    movTitle || '(' || movGenre || ')-' || movYear AS "Headline"
FROM
    Movie
ORDER BY
    LENGTH(movTitle || '(' || movGenre || ')-' || movYear) DESC;
```

10. Generate a report that summates the total price of all movies for each genre. The final row of the report should contain the total price for all movies. Your output should look like this:

Genre	Total_Price
...	...
Total	...

Solution

```
SELECT
    movGenre    AS "Genre",
    SUM(movPrice) AS "Total_Price"
FROM
    Movie
GROUP BY
    movGenre
UNION
SELECT
```

```
    'Total' AS "Genre",
    SUM(movPrice) AS "Total_Price"
FROM
    Movie;
```

3 Submitting your lab work

This week, we will be using a Moodle quiz to submit your lab work. For each question, you will need to submit an SQL statement through the quiz. You will receive immediate feedback (whether correct or incorrect) on each submission, and you can resubmit your answers as many times as you like before the deadline, without penalty.

You must submit your answers before the deadline. Late submissions will not be accepted, as the solution will automatically be released once the quiz closes, and you will receive a mark of zero for this lab.

Unlike other lab submissions, your grade for this lab will be based on the number of correct answers you submit, so it is important to attempt all questions and resubmit until you get them right. There is no penalty for the number of attempts you make.

This lab contributes 1% of your overall module grade.