Lab 003: Solution

COMP1048: Databases and Interfaces (2024-2025)

Matthew Pike & Yuan Yao

Table of contents

1 Lab Overview

This week's lab builds on the previous one, using an extended version of the database. You will work with a pre-populated database and perform more advanced queries to answer the questions below. You can find the database file on Moodle. Write SQL queries for each question.

2 Exercises

Before starting, familiarise yourself with the database schema. Type `.schema` and press Enter to display the schema of the database.

1. Write a query that shows the movie title, year of release, and the leading actor for each movie. Include all movies, even if they don't have a leading actor. Arrange the output by year of release, first in ascending order, and then by movie title in ascending order.

year	movie_title	leading_actor
...	...	...

Solution

```sql
SELECT
    movYear AS 'year',
    movTitle AS 'movie_title',
    actName AS 'leading_actor'
FROM
    Movie LEFT JOIN Actor ON Movie.actID = Actor.actID
ORDER BY
    movYear ASC,
    movTitle ASC;
```

2. Write a query that shows the total sales for each region (for all movies). Arrange the output by region name in ascending order. Your result should appear as follows:

region	total_sales
...	...

Solution

```
SELECT
    region AS 'region',
    SUM(movSales) AS 'total_sales'
FROM
    MovieSales
GROUP BY
    region
ORDER BY
    region ASC;
```

3. Write a query that shows the sales number and region for all movies. If a movie has sales data from multiple regions, each movie-region pairing should appear on a separate row. Include all movies in the output at least once, even if they have no sales data. Arrange the output by movie title in ascending order. Your result should appear as follows:

movie_title	region	sales
...	...	...

Solution

```
SELECT
    movTitle AS 'movie_title',
    region AS 'region',
    movSales AS 'sales'
FROM
    Movie LEFT JOIN MovieSales ON (movId)
ORDER BY
    movTitle ASC;
```

4. Write a query that shows the total sales number (summed over all regions) for each movie. Arrange the output by sales, in descending order. Include all movies, even if they have no sales data. Your result should appear as follows:

movie_title	sales
...	...

Solution

```sql
SELECT
    movTitle AS 'movie_title',
    SUM(movSales) AS 'sales'
FROM
    Movie LEFT JOIN MovieSales ON Movie.movID = MovieSales.movID
GROUP BY
    Movie.movID
ORDER BY
    sales DESC;
```

5. Write a query that shows the total sales for all movies (across all regions) that an actor has starred in. Include all actors, even if they have not appeared in any films. Arrange the output by sales, in descending order. Your result should appear as follows:

actor_name	sales
...	...

Solution

```sql
SELECT
    actName AS 'actor_name',
    SUM(movSales) AS 'sales'
FROM
    Actor LEFT JOIN Movie
        ON Actor.actID = Movie.actID
    LEFT JOIN MovieSales
        ON Movie.movID = MovieSales.movID
GROUP BY
    Actor.actId
ORDER BY
    sales DESC;
```

6. Write a query that increases the sales number of all movies released in Europe by 10%, rounded to the nearest integer. Your query should produce no output.

Solution

```sql
UPDATE
    MovieSales
SET
    movSales = ROUND(movSales * 1.1)
WHERE
    region = 'Europe';
```

7. Write a query that shows the income for each movie. The income is the total sales number (summed over all regions) multiplied by the movie price, rounded to two decimal places. Arrange the output by year of release, first in ascending order, then by movie title in ascending order. You do not need to include movies with no sales in your result. Your result should appear as follows:

year	movie_title	income
...	...	...

Solution

```
SELECT
    movYear AS 'year',
    movTitle AS 'movie_title',
    ROUND(SUM(movSales) * movPrice, 0) AS 'income'
FROM
    Movie NATURAL JOIN MovieSales
GROUP BY
    movId
ORDER BY
    movYear ASC,
    movTitle ASC;
```

8. Write a query that determines the most popular actor in each region. The actor's popularity in one region is measured by the **average sales** of all movies in that region that the actor has appeared in. Arrange the output by region name in ascending order. Your result should appear as follows:

region	actor_name	average_sales
...	...	...

> **Hint**
>
> In SQL, `GROUP BY` can be used to group rows that have the same value in more than one column. For example, the following query will group all rows that have the same value in the `year` and `title` columns.
>
> ```
> SELECT year, title, SUM(sales) AS total_sales
> FROM movies
> GROUP BY year, title;
> ```

Solution

```
SELECT
    region AS 'region',
    actName AS 'actor_name',
```

```
    MAX(avgSales) AS 'average_sales'
FROM(
    SELECT
        region,
        actName,
        AVG(movSales) AS 'avgSales'
    FROM
        Movie NATURAL JOIN MovieSales NATURAL JOIN Actor
    GROUP BY
        region, actId
)
GROUP BY
    region;
```

9. Write a query that removes all movies without any sales data. The output should not show anything.

Solution

```
DELETE FROM
    Movie
WHERE
    movId NOT IN (SELECT movId FROM MovieSales);
```

3 Submitting Your Lab Work

This week, as with last week, you will submit your lab work through a Moodle quiz. For each question, submit an SQL statement via the quiz. You'll receive immediate feedback on each submission (correct or incorrect) and can resubmit as many times as needed before the deadline, without penalty.

Submit all answers before the deadline, as late submissions will not be accepted, and a mark of zero will be recorded for the lab. Unlike other labs, your grade for this lab will be based on the number of correct answers submitted, so it's important to attempt all questions and resubmit until they are correct. There is no penalty for multiple attempts.

This lab contributes 1% of your overall module grade.