Relational Data Model

1. Basics of Relational Databases:

Relational Data Model and its Commercial Dominance

- **Reason**: The relational model uses tables (which are intuitive and familiar) to represent data. This simplicity, combined with its theoretical robustness, makes it widely accepted.
- **Commercial Dominance**: It has been the most popular choice for databases for over 35 years due to its ease of use and strong foundation.

Why Relational Database Systems Became Popular

- **Familiarity**: The concept of tables, rows, and columns is already familiar to most people because tables are used in many other fields (like spreadsheets, reports, etc.).
- **Simplicity**: Tables are simple to understand, which makes them user-friendly.
- Relational databases consist of **tables** with two main parts:
  - **Heading** (definition) that includes the table's name and the column names.
  - **Body** (content) which contains the actual data, where each **row** represents a real-world entity (like a student in a university).

Example: Student Table

- **Heading**: Could include columns like Student Number (StdNo), Name, Address, GPA, Major, etc.
- **Body**: Contains the actual student data, where each row corresponds to a student.
- In large databases, these tables can have **dozens of columns** and **thousands of rows**.

Column Naming Conventions

- Column names should be clear and descriptive. To make it easier to identify which table a column belongs to, a **table abbreviation** is often used.
  - For example, `StdNo` means "Student Number" from the student table, where `Std` is the abbreviation for the table.

Understanding Relationships Among Tables

- In relational databases, it's crucial to understand how **tables are related** to one another. This is done by **matching values** between different tables.
- Example:
    - In a **Student Table**, there's a `StdNo` column.
    - In an **Enrollment Table**, there's also a `StdNo` column, which corresponds to the student who enrolled in a course.
    - By matching these values (`StdNo`), you can establish a **relationship** between the tables.

Graphical Depiction of Table Relationships

- To illustrate relationships between tables, you can use diagrams where arrows show how rows in one table match with rows in another table.
    - Example: A student with a specific `StdNo` could be enrolled in several courses, which are recorded in the enrollment table.
- However, as the number of tables and rows grows, it becomes difficult to follow these relationships visually.

Combining Multiple Tables

- To extract meaningful data from a database, you often need to **combine multiple tables** by matching values.
    - Example:
        - Combine the **Student Table** and **Enrollment Table** by matching `StdNo`.
        - Combine the **Enrollment Table** and **Offering Table** by matching `OfferNo` (the course number).
    - This process is called a **join**, which is crucial in SQL and relational databases.

Alternative Terminology for Relational Databases

- Different groups of people use different terms for the same concepts in relational databases:
    - **Table-oriented terminology** (e.g., table, row, column) appeals to end-users who are familiar with tables and spreadsheets.
    - **Set-oriented terminology** (e.g., relation, tuple, attribute) is used by academics and researchers in the field.

    - **Record-oriented terminology** (e.g., file, record, field) is used by information systems professionals.
- In practice, these terms can be **mixed** together.

Relational Databases and SQL as a Standard

- **Relational databases** have been dominant in the commercial world for several reasons:
    - The **simplicity** of using tables to organize data.
    - A **strong theoretical foundation** from decades of academic research.
    - Many **commercial and open-source products** (like Oracle and PostgreSQL) have implemented this model over the years.
    - The **SQL standard** has been around for over 35 years and provides a consistent way to work with relational databases.
- SQL makes it easy to query and manipulate the data stored in relational databases.

2. Integrity Rules:

Key Concepts Defined

- **Null Values**: These represent missing or unknown values in a table. Null values may occur when a certain piece of data is not available or not applicable.
- **Primary Key**: A unique identifier for each row in a table. It ensures that no two rows have the same value. In cases where multiple columns together form a unique identifier, it's known as a **composite key**.
- **Foreign Key**: A column or a combination of columns that refer to a primary key in another table. It establishes relationships between tables.

Two Major Integrity Rules

1. **Entity Integrity**:
    - Every table must have a primary key.
    - Primary keys cannot have null values.
    - Ensures that every entity (like a person, object, or event) is uniquely identifiable.
    - Example: In a student table, the student number must be unique and non-null, ensuring every student has a unique identifier.
2. **Referential Integrity**:
    - Foreign keys must match primary key values in related tables.
    - Some foreign keys may allow null values, such as cases where a relationship hasn’t been established yet (e.g., a course without an assigned faculty member).
    - Ensures valid relationships between tables.

- Example: Every student number in the enrollment table must exist in the student table to ensure that every enrollment corresponds to a valid student.

Example Analysis: Integrity Violations

- **Entity Integrity Violations**: If a primary key column in any row has a null value, entity integrity is violated.
- **Referential Integrity Violations**: If a foreign key in a table does not match any primary key in the related table, referential integrity is violated.

University Database Example

- **Primary Keys**:
  - **Student Number (StdNo)** in the student table.
  - **Offer Number** in the offering table.
  - A combination of **Student Number** and **Offer Number** in the enrollment table.
- **Foreign Keys**:
  - **Student Number** in the enrollment table refers to the **Student Number** in the student table.
  - **Offer Number** in the enrollment table refers to the **Offer Number** in the offering table.

Database Diagram Notation

- **P**: Marks the primary key of a table.
- **F**: Marks a foreign key in a table.
- **Solid lines**: Indicate that foreign key values cannot be null.
- **Dashed lines**: Indicate that foreign key values may be null.
- **PF**: Indicates that a column is both a primary key and a foreign key.

Key Takeaways

- **Primary Keys**: Ensure that each entity in a table is unique and identifiable.
- **Foreign Keys**: Ensure relationships between tables are valid by matching with primary keys.
- **Entity Integrity**: No part of a primary key can have a missing value.
- **Referential Integrity**: Foreign keys must refer to valid primary keys, but they may allow null values in certain situations.

3. Basic SQL CREATE TABLE statement:

1. Purpose of the `CREATE TABLE` Statement

The `CREATE TABLE` statement defines the structure of a table in a relational database, specifying the columns, data types, and constraints. This is important because every database starts with the definition of its schema, and creating tables is the foundation of database management.

2. Why Do DBMS Vendors Provide Visual Interfaces for `CREATE TABLE`?

Vendors provide visual tools to make creating and modifying tables easier and more user-friendly. Writing SQL manually can be error-prone, so visual tools increase productivity and reduce the chances of syntax errors. These tools generate the SQL commands automatically based on user inputs.

3. Impact of a Missing Comma

In SQL, each column definition within the `CREATE TABLE` statement must be separated by a comma. A missing comma results in a syntax error that may be difficult to interpret. For example, missing a comma between two column definitions can prevent the table from being created, halting database operations.

4. Syntax of `CREATE TABLE`

The basic syntax for creating a table starts with the keywords `CREATE TABLE`, followed by the table name, and a list of column definitions enclosed in parentheses. Each column must have a name and a data type, and you can optionally specify constraints like `PRIMARY KEY`, `NOT NULL`, etc. Pay attention to detail, as even minor syntax errors, like misspelling a keyword or missing a comma, can cause the query to fail.

5. Column Definitions and Data Types

Each column in a table requires a data type to indicate the kind of data that will be stored. Here are some common data types:

- `CHAR(N)`: Fixed-length character string. Use it for values like state abbreviations (`CA`, `TX`), where the length is fixed.
- `VARCHAR(N)`: Variable-length character string. Use it for values like names, where the length varies.
- `INTEGER`: Stores whole numbers. Useful for age, IDs, etc.
- `FLOAT`: Stores floating-point numbers for precision, like scientific data or interest rates.
- `DECIMAL(W, R)`: Stores fixed-point numbers like currency, where `W` is the total width and `R` is the number of decimal places.
- `DATE`, `TIME`, `TIMESTAMP`: Stores date and time values. Useful for logging events like registration dates.

6. Portability Across DBMS

While the `CREATE TABLE` syntax is standardized by SQL, not all parts of it are portable across different Database Management Systems (DBMS). Each DBMS may have slight variations in how it implements SQL, including proprietary data types and constraints.

7. Visual Interfaces for Increased Productivity

Because manually writing SQL can be time-consuming and error-prone, many DBMS vendors have developed graphical interfaces. These tools let you create tables visually, then generate the `CREATE TABLE` SQL statements automatically in the background. This simplifies development and reduces syntax-related issues.

4. Integrity Constraint Syntax:

Integrity Constraints in `CREATE TABLE`

- The `CREATE TABLE` statement can include various integrity constraints to ensure data integrity. The key types of constraints are:
    1. **Primary Key**: Uniquely identifies each record in a table. A primary key must contain unique values and cannot be NULL.
    2. **Foreign Key**: Ensures referential integrity by linking to a primary key in another table. This constraint maintains the relationship between two tables.
    3. **Unique Constraint**: Ensures that all values in a column are unique, preventing duplicates. This can apply to candidate keys that are not primary keys.
    4. **Required Constraint**: Enforced using the `NOT NULL` keyword, indicating that a value must be provided for a column; NULL values are not allowed.
    5. **Check Constraint**: Specifies a condition that must be met for a column, such as a range of acceptable values.

Placement of Constraints

- **In-Line Constraints**: These are defined within the column definition and are typically used for single-column constraints (e.g., `NOT NULL`).
- **External Constraints**: These are specified after all column definitions and are used for multi-column constraints (e.g., composite primary keys).

Check Constraints

- Check constraints ensure data validity based on specified conditions. They can be defined inline or externally, with limitations on using columns from other tables.

<u>Quiz:</u>

Concept Quiz for Module 3

Question 1
A CHECK constraint involves
1 point

- a reference to a parent table.
- a reference to a child table.
- **conditions with comparison operators and logical operators involving one or more columns of the same table.**
- uniqueness for one or more columns of the same table.

Question 2
In a column specification of the CREATE TABLE statement, you must specify
1 point

- **the column name and data type.**
- inline constraints.
- the default value.
- table constraints.

Question 3
In a column specification, a default value specification is required.
1 point

- True
- **False**

Question 4
How is a M-N relationship represented in a relational database?
1 point

- through foreign keys in more than one table
- through referential integrity constraints in more than one table
- **through an associative table containing a combined primary key consisting of multiple foreign keys**
- through a foreign key referencing the primary key of the same table

Question 5
What statements are true about null values? Multiple answers are possible.
1 point

- **A null value indicates the absence of a value.**
- **A null value may mean that the actual value is unknown.**
- **A null value may mean that actual value does not apply to the specified row.**
- A null value indicates a default value.

Question 6
What is the difference between the primary key for a table and candidate keys for the same table?
1 point

- All candidate keys are primary keys.
- All candidate keys not accepting null values are primary keys.
- **The primary key is chosen among candidate keys that do not allow null values.**
- The primary key is a randomly selected candidate key.

Question 7
Relationships in relational databases are represented
1 point

- by entity integrity constraints.
- by uniqueness constraints.
- **by foreign keys and associated referential integrity constraints.**
- by linking tables.

Question 8
How is a 1-M self-referencing relationship represented in a relational database?
1 point

- **a foreign key that references the primary key of the same table**
- a foreign key that references the primary key of a different table
- an associative table with a combined primary key
- two foreign keys that reference each other

Question 9
Names are required for constraints in the CREATE TABLE statement.
1 point

- True
- **False**

Question 10
What keyword(s) indicates a candidate key that is not the primary key?
1 point

- PRIMARY KEY
- FOREIGN KEY
- CHECK
- **UNIQUE**

Queries:

SELECT * FROM film;

SELECT first_name, last_name FROM actor WHERE first_name = 'John' AND last_name = 'Doe';

SELECT COUNT(*) AS total_customers FROM customer;

SELECT title, release_year FROM film ORDER BY release_year DESC;

SELECT title, rating FROM film WHERE rating = 'PG-13';

SELECT * FROM category;

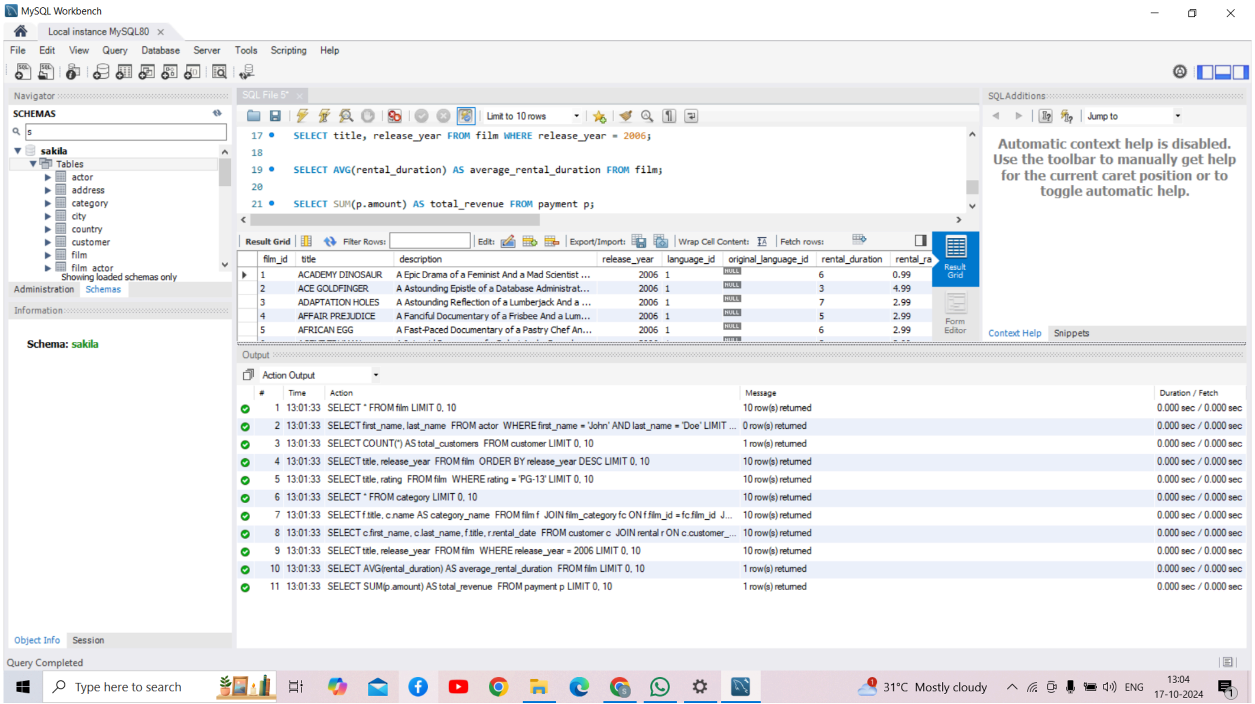
SELECT f.title, c.name AS category_name FROM film f JOIN film_category fc ON f.film_id = fc.film_id JOIN category c ON fc.category_id = c.category_id;

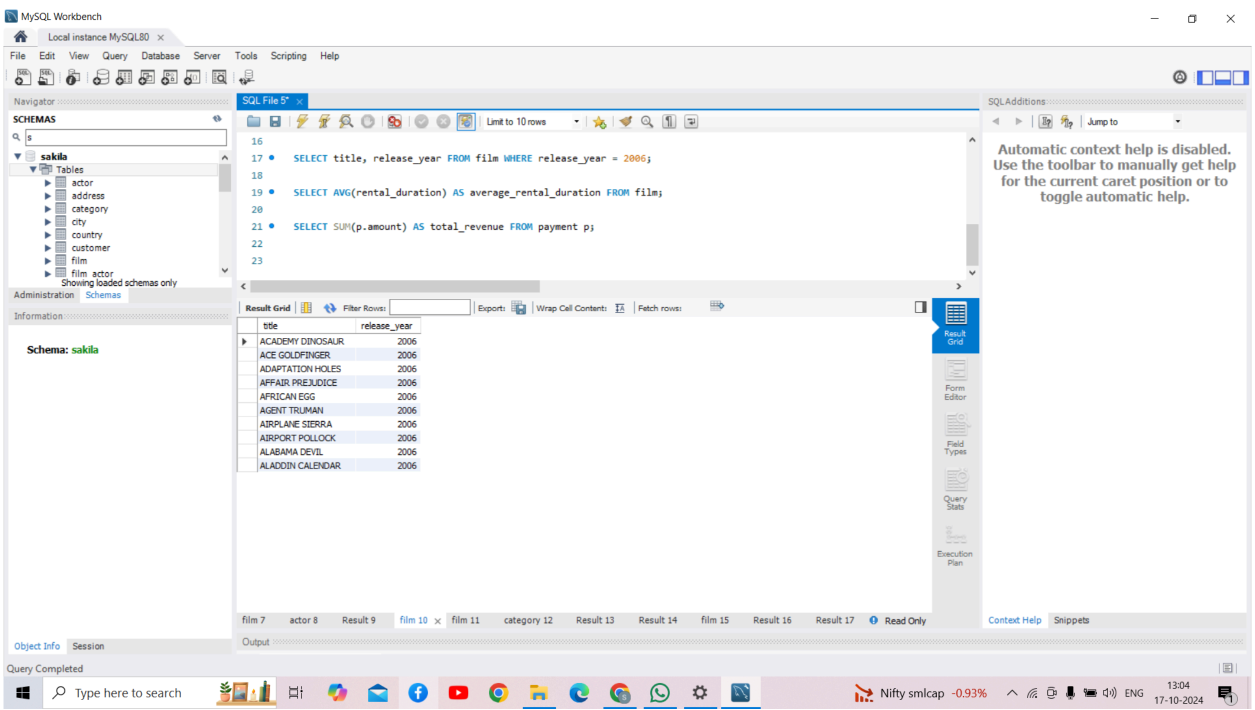
SELECT c.first_name, c.last_name, f.title, r.rental_date FROM customer c JOIN rental r ON c.customer_id = r.customer_id JOIN inventory i ON r.inventory_id = i.inventory_id JOIN film f ON i.film_id = f.film_id;

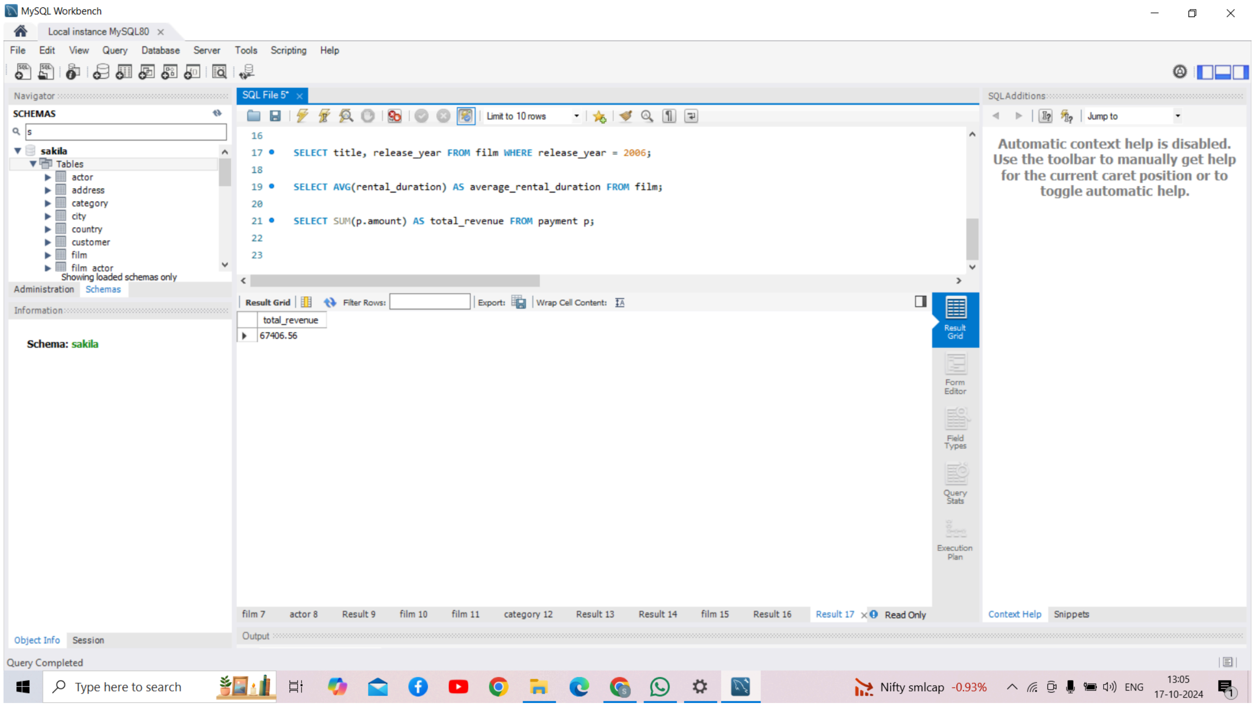
SELECT title, release_year FROM film WHERE release_year = 2006;

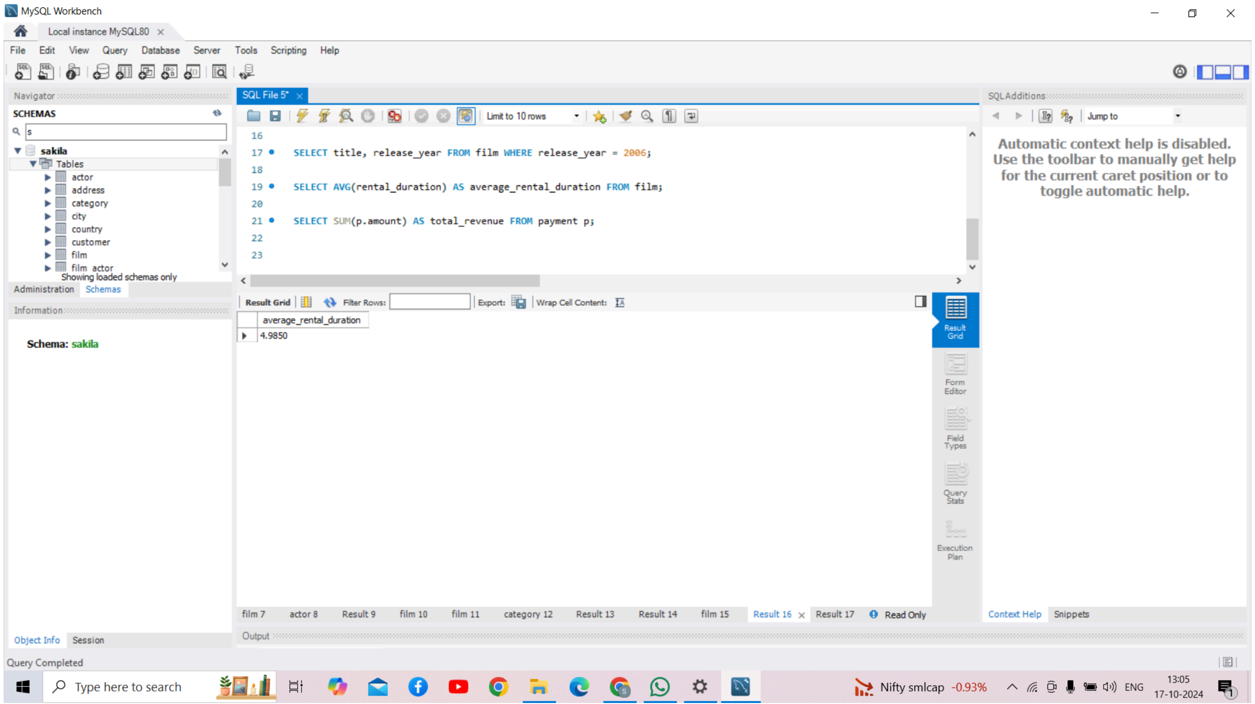
SELECT AVG(rental_duration) AS average_rental_duration FROM film;

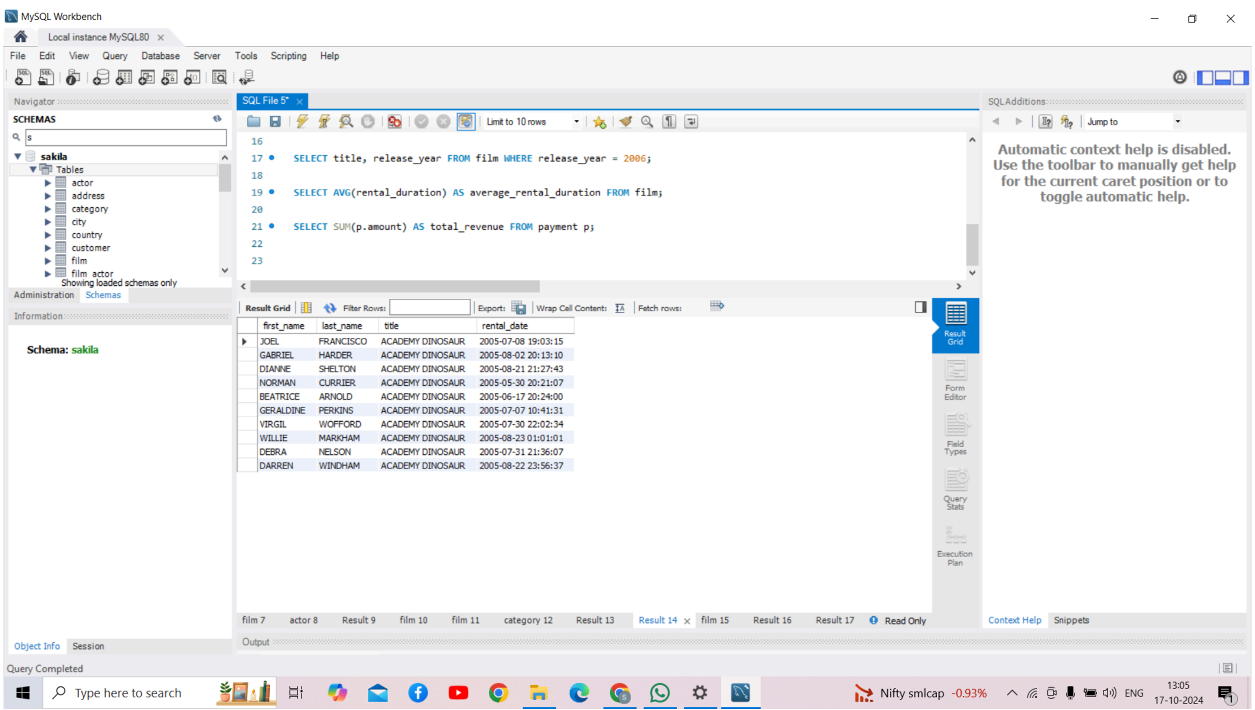
SELECT SUM(p.amount) AS total_revenue FROM payment p;


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
Showing loaded schemas only
Administration Schemas
Information
Schema: sakila
SQL File 5*
Limit to 10 rows
17 SELECT title, release_year FROM film WHERE release_year = 2006;
18
19 SELECT AVG(rental_duration) AS average_rental_duration FROM film;
20
21 SELECT SUM(p.amount) AS total_revenue FROM payment p;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content: Fetch rows:
film_id title description release_year language_id original_language_id rental_duration rental_ra
1 ACADEMY DINOSAUR A Epic Drama of a Feminist And a Mad Scientist ... 2006 1 NULL 6 0.99
2 ACE GOLDFINGER A Astounding Epistle of a Database Administrat... 2006 1 NULL 3 4.99
3 ADAPTATION HOLES A Astounding Reflection of a Lumberjack And a ... 2006 1 NULL 7 2.99
4 AFFAIR PREJUDICE A Fanciful Documentary of a Frisbee And a Lum... 2006 1 NULL 5 2.99
5 AFRICAN EGG A Fast-Paced Documentary of a Pastry Chef An... 2006 1 NULL 6 2.99
Result Grid
Form Editor
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 13:01:33 SELECT * FROM film LIMIT 0, 10 10 row(s) returned 0.000 sec / 0.000 sec
2 13:01:33 SELECT first_name, last_name FROM actor WHERE first_name = 'John' AND last_name = 'Doe' LIMIT ... 0 row(s) returned 0.000 sec / 0.000 sec
3 13:01:33 SELECT COUNT(*) AS total_customers FROM customer LIMIT 0, 10 1 row(s) returned 0.000 sec / 0.000 sec
4 13:01:33 SELECT title, release_year FROM film ORDER BY release_year DESC LIMIT 0, 10 10 row(s) returned 0.000 sec / 0.000 sec
5 13:01:33 SELECT title, rating FROM film WHERE rating = 'PG-13' LIMIT 0, 10 10 row(s) returned 0.000 sec / 0.000 sec
6 13:01:33 SELECT * FROM category LIMIT 0, 10 10 row(s) returned 0.000 sec / 0.000 sec
7 13:01:33 SELECT f.title, c.name AS category_name FROM film f JOIN film_category fc ON f.film_id = fc.film_id J... 10 row(s) returned 0.000 sec / 0.000 sec
8 13:01:33 SELECT c.first_name, c.last_name, f.title, r.rental_date FROM customer c JOIN rental r ON c.customer_... 10 row(s) returned 0.000 sec / 0.000 sec
9 13:01:33 SELECT title, release_year FROM film WHERE release_year = 2006 LIMIT 0, 10 10 row(s) returned 0.000 sec / 0.000 sec
10 13:01:33 SELECT AVG(rental_duration) AS average_rental_duration FROM film LIMIT 0, 10 1 row(s) returned 0.000 sec / 0.000 sec
11 13:01:33 SELECT SUM(p.amount) AS total_revenue FROM payment p LIMIT 0, 10 1 row(s) returned 0.000 sec / 0.000 sec
Object Info Session
Query Completed
Type here to search
31°C Mostly cloudy
ENG
13:04
17-10-2024


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
Showing loaded schemas only
Administration Schemas
Information
Schema: sakila
SQL File 5*
Limit to 10 rows
16
17 SELECT title, release_year FROM film WHERE release_year = 2006;
18
19 SELECT AVG(rental_duration) AS average_rental_duration FROM film;
20
21 SELECT SUM(p.amount) AS total_revenue FROM payment p;
22
23
Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:
title release_year
ACADEMY DINOSAUR 2006
ACE GOLDFINGER 2006
ADAPTATION HOLES 2006
AFFAIR PREJUDICE 2006
AFRICAN EGG 2006
AGENT TRUMAN 2006
AIRPLANE SIERRA 2006
AIRPORT POLLOCK 2006
ALABAMA DEVIL 2006
ALADDIN CALENDAR 2006
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
film 7 actor 8 Result 9 film 10 film 11 category 12 Result 13 Result 14 film 15 Result 16 Result 17 Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Object Info Session
Query Completed
Type here to search
Nifty smlcap -0.93%
ENG
13:04
17-10-2024


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SQL File 5*
SELECT title, release_year FROM film WHERE release_year = 2006;
SELECT AVG(rental_duration) AS average_rental_duration FROM film;
SELECT SUM(p.amount) AS total_revenue FROM payment p;
total_revenue
67406.56
Schema: sakila
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Query Completed


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
Showing loaded schemas only
Administration Schemas
Information
Schema: sakila
SQL File 5*
Limit to 10 rows
16
17 SELECT title, release_year FROM film WHERE release_year = 2006;
18
19 SELECT AVG(rental_duration) AS average_rental_duration FROM film;
20
21 SELECT SUM(p.amount) AS total_revenue FROM payment p;
22
23
Result Grid Filter Rows: Export: Wrap Cell Content:
average_rental_duration
4.9850
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
film 7 actor 8 Result 9 film 10 film 11 category 12 Result 13 Result 14 film 15 Result 16 Result 17 Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Object Info Session
Output
Query Completed
Type here to search
Nifty smlcap -0.93%
ENG
13:05
17-10-2024


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
Showing loaded schemas only
Administration Schemas
Information
Schema: sakila
SQL File 5*
Limit to 10 rows
16
17 SELECT title, release_year FROM film WHERE release_year = 2006;
18
19 SELECT AVG(rental_duration) AS average_rental_duration FROM film;
20
21 SELECT SUM(p.amount) AS total_revenue FROM payment p;
22
23
Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:
first_name last_name title rental_date
JOEL FRANCISCO ACADEMY DINOSAUR 2005-07-08 19:03:15
GABRIEL HARDER ACADEMY DINOSAUR 2005-08-02 20:13:10
DIANNE SHELTON ACADEMY DINOSAUR 2005-08-21 21:27:43
NORMAN CURRIER ACADEMY DINOSAUR 2005-05-30 20:21:07
BEATRICE ARNOLD ACADEMY DINOSAUR 2005-06-17 20:24:00
GERALDINE PERKINS ACADEMY DINOSAUR 2005-07-07 10:41:31
VIRGIL WOFFORD ACADEMY DINOSAUR 2005-07-30 22:02:34
WILLIE MARKHAM ACADEMY DINOSAUR 2005-08-23 01:01:01
DEBRA NELSON ACADEMY DINOSAUR 2005-07-31 21:36:07
DARREN WINDHAM ACADEMY DINOSAUR 2005-08-22 23:56:37
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
film 7 actor 8 Result 9 film 10 film 11 category 12 Result 13 Result 14 film 15 Result 16 Result 17 Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Object Info Session
Output
Query Completed
Type here to search
Nifty smlcap -0.93%
ENG
13:05
17-10-2024