

A
Project
MOVIE THEATER
In
The Partial Fulfillment of
Semester-II
Bachelor of Computer Application (BCA)
2023-24



UNIVERSITY OF KOTA, KOTA

Submitted By:

Stu. Name - Priyal Jain

Stu. Name - Manas Joshi

Stu. Name - Arpit Gayari

Stu. Name - Abhishek Patidar

Name of Supervisor

Ms. Priya Jain

(Assistant Prof.)



Department of Computer Science,
AKLANK COLLEGE, KOTA

CERTIFICATE

This is certify that the project entitled “Movie Theater” being submitted by Priyal Jain,Manas Joshi , Arpit Gayari ,Abhishek Patidar the Partial Fulfillment of Semester-II,Bachelor of Computer Application is a record of detailed work carried out under the guidance , during the academic session 2023-24 and it has been found worthy of acceptance according to the requirement of the University.

During her tenure with us, I found her sincere and hardworking

Ms. Neelima Jain
(HOD)

Ms. Priya Jain
(Supervisor)

Dr. Lalit Kumar Sharma
Principal

External

S.No	TOPICS	PAGE NO.
1.	CERTIFICATE	2
2.	Acknowledgement	4
3.	Abstract	5
4.	Literature Review <ul style="list-style-type: none"> • Data • Database • DBMS • RDBMS • Normalization • ER Diagram • Database schema • SQL • My SQL Workbench • Query 	9 9 10 15 17 20 23 26 27 29
5.	Hardware and Software Requirements <ul style="list-style-type: none"> • Introduction • Types of Hardware and Software 	33 33
6.	Database Design <ul style="list-style-type: none"> • ER diagram of a project • Database Scheme • Normalization techniques applied 	38 40 41
7.	IMPLEMENTATION <ul style="list-style-type: none"> • Software install • Create table • Preparing Report • Insert record • Add foreign key • Queries 	44 54 59 62 67 72
8.	User Interface	82
9.	Future Enhancement	93
10.	Conclusion	95
11.	Reference	96

ACKNOWLEDGEMENT

We feel immensely privileged to extend our heartfelt gratitude and utmost gratitude towards our supervisor, **Prof. Priya Jain ma'am**, for her invaluable guidance and influential mentorship exhibits during the course of this project. Owing to her technical advice, exemplary guidance, persistent mentoring, and constant encouragement towards achieving perfection, it has been possible to take this project forward to this stage.

We are obliged for the valuable information and the helping hand provided by her amidst her busy schedule.

We thank **Prof. Neelima Jain**, Head of the Department, Bachelor of Computer Applications for providing the necessary facilities to complete our work.

We would like to thank our mentor, **Prof. Preeti Sharma**, for assisting us in our project. We would not have been able to do this without the constant support and motivation from all of them.

ABSTRACT

Purpose of the Movie Theater Booking System:

The primary objective of the movie theater booking system is to streamline the process of managing movie schedules, theater information, customer data, and booking transactions. This system is designed to enhance the efficiency of theater operations, ensuring smooth handling of movie showtimes, seat reservations, and customer bookings. Utilizing Database Management Systems (DBMS) and Structured Query Language (SQL), this project aims to create a robust and scalable database that supports the daily operations of a movie theater, offering accuracy, reliability, and user-friendly interfaces.

Scope of the Project:

This project focuses on developing a comprehensive movie theater booking system using industry-standard tools and technologies. The key functionalities covered in this system include:

1. **Movie Management:** Storing and managing movie details such as titles, genres, directors, duration, ratings, and pricing. This allows for efficient scheduling and promotion of movies.
2. **Theater Management:** Handling information related to theater locations, seating capacities, and managing multiple theaters within the system. Ensures accurate seat allocation and theater usage.
3. **Showtime Management:** Scheduling and managing showtimes by linking movies to specific theaters. This includes setting up show dates and times, ensuring that customers can easily find and book available showtimes.
4. **Customer Management:** Storing customer details such as names, email addresses, and phone numbers. This facilitates personalized service and communication, enhancing customer experience.
5. **Booking Management:** Managing booking transactions by linking customers to specific showtimes. This includes tracking the number of seats booked, ensuring that seat reservations are accurately processed.
6. **Reporting and Analytics:** Generating reports and analyzing data related to movie performance, theater occupancy, and customer behavior. This supports better decision-making and operational improvements.

Importance of a Movie Theater Booking System:

1. **Operational Efficiency:** Automating the scheduling, booking, and management processes reduces manual effort and increases the speed of operations.
2. **Customer Satisfaction:** Providing a seamless and convenient booking experience enhances customer satisfaction, leading to repeat business and customer loyalty.
3. **Data Accuracy:** Reducing manual data entry minimizes errors, ensuring that showtimes, bookings, and customer details are accurate and up-to-date.
4. **Scalability:** A well-designed database supports the growth of the business, allowing for the easy addition of new movies, theaters, and showtimes without significant rework.
5. **Revenue Optimization:** By efficiently managing showtimes and seat availability, the system helps maximize theater occupancy and revenue.

Overview of the System:

The movie theater booking system will be developed using the following technologies:

1. **Database Management Systems (DBMS):**
 - **MySQL:** An open-source relational database management system known for its reliability and scalability. MySQL is suitable for handling the complex data requirements of a movie theater booking system, offering efficient data storage and retrieval capabilities.
2. **Structured Query Language (SQL):**
 - **SQL** is used to create, modify, and query the database, ensuring efficient management of movie, theater, showtime, customer, and booking data. SQL's powerful querying capabilities enable complex data operations, such as filtering showtimes by movie and theater, and generating booking reports.
3. **MySQL Workbench:**
 - **MySQL Workbench** is a unified visual tool used for data modeling, SQL development, and database administration. It provides a comprehensive environment for designing the database schema,

managing the database server, and performing data migration tasks. The use of MySQL Workbench ensures that the database is well-structured, secure, and optimized for performance.

By leveraging these technologies, the project aims to develop a robust and user-friendly movie theater booking system that meets the needs of both theater management and customers, providing a seamless booking experience and efficient theater operations.

Literature Survey

Literature Survey

DATA:- "Data" refers to pieces of information, often collected through observation, measurement, or research. It can be in various forms, such as numbers, text, images, videos, or any other type of content that can be analyzed and used to make decisions, gain insights, or build knowledge. Here are a few common contexts where data is used:

1. **Quantitative Data:** Numerical information, such as statistics, measurements, or counts. Examples include sales figures, temperature readings, or survey responses.
2. **Qualitative Data:** Descriptive information, often in text form, that provides insights into characteristics, behaviors, or opinions. Examples include interview transcripts, open-ended survey responses, and observations.
3. **Structured Data:** Organized in a specific format, such as databases or spreadsheets, making it easy to search and analyze. Examples include SQL databases and Excel sheets.
4. **Unstructured Data:** Not organized in a predefined manner and often more complex to process and analyze. Examples include social media posts, emails, and videos.
5. **Big Data:** Extremely large datasets that require advanced methods and technologies for storage, processing, and analysis. Examples include data generated by large-scale online platforms like Google or Facebook.

Data is essential in various fields such as science, business, healthcare, and technology, where it is used to inform decision-making, identify trends, and create models or predictions.

DATABASE

A **database** is an organized collection of data that is stored and accessed electronically. The data in a database is typically structured in a way that makes it easy to manage, retrieve, and manipulate. Databases are used to store a wide variety of information, ranging from simple lists, like contact information, to complex data systems, such as those used in banking, e-commerce, and social media platforms.

Types of Databases

There are various types of databases, each optimized for specific data handling needs:

1. **Relational databases:** Organize data into tables with rows and columns, allowing for complex queries and relationships.
2. **Hierarchical Databases:** Organize data in a tree-like structure with a single root and multiple levels of parent-child relationships, where each child node has only one parent.
3. **Network Databases:** Use a graph structure to represent more complex relationships, allowing each record to have multiple parent and child records, enabling many-to-many relationships.
4. **Distributed databases:** Store data across multiple physical locations, enabling scalability and fault tolerance.
5. **Cloud databases:** Run on cloud computing platforms, offering flexibility, scalability, and managed services.

Database Management System

DBMS: -A **Database Management System (DBMS)** is software that allows users to create, manage, and interact with databases. It provides an interface between users and the data in the database, ensuring that data is stored, retrieved, and manipulated efficiently and securely.

Here are some key points about DBMS:

1. **Definition and Creation of Databases:** DBMS allows users to define the structure of the data and create databases according to specific needs. This includes specifying data types, relationships between data, and constraints.

2. **Data Storage and Retrieval:** It provides mechanisms to store large amounts of data and efficiently retrieve it as needed. This is crucial for handling large datasets and ensuring quick access to information.
3. **Data Manipulation:** Users can insert, update, delete, and retrieve data using a query language such as SQL (Structured Query Language). This allows for flexible and powerful data manipulation and management.
4. **Security and Access Control:** DBMS ensures that only authorized users can access or modify the data. It provides various security mechanisms such as authentication, authorization, and encryption to protect data integrity and privacy.
5. **Backup and Recovery:** To prevent data loss and ensure data availability, DBMS includes features for backing up data and recovering it in case of failures or errors.
6. **Data Integrity and Consistency:** DBMS enforces rules to maintain the accuracy and consistency of data. This includes constraints like primary keys, foreign keys, and unique constraints.
7. **Concurrency Control:** In multi-user environments, DBMS manages concurrent access to data, ensuring that transactions are executed safely and consistently without conflicts.

Types of DBMS:

- **Relational DBMS (RDBMS):** Stores data in tables and uses SQL for data manipulation. Examples include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
- **Object-Oriented DBMS (OODBMS):** Stores data in the form of objects, as used in object-oriented programming. Examples include db4o and Object DB.
- **Hierarchical and Network DBMS:** Older models that store data in tree-like or graph structures. Examples include IBM's IMS (hierarchical) and IDMS (network).

Advantages of DBMS

1. **Data Redundancy Control:** Minimizes duplicate data, leading to efficient storage and easier maintenance.
2. **Data Integrity:** Ensures accuracy and consistency through integrity constraints and validation checks.
3. **Data Security:** Protects sensitive information by controlling access through authentication, authorization, and encryption.
4. **Data Independence:** Allows application programs to remain unaffected by changes in physical data storage.
5. **Efficient Data Access:** Optimizes query processing and indexing for faster data retrieval and resource efficiency.
6. **Backup and Recovery:** Simplifies recovery from failures with regular backups and robust recovery mechanisms.
7. **Concurrent Access:** Enables multiple users to access and manipulate data simultaneously without conflicts.
8. **Improved Decision-Making:** Provides tools for data analysis and reporting, enhancing decision-making capabilities.

Disadvantages of DBMS

1. **Complexity:** DBMS systems can be complex to design, implement, and manage, requiring specialized knowledge and skills.
2. **Cost:** High initial costs for purchasing, licensing, and maintaining a DBMS can be significant, especially for small organizations.
3. **Performance Overhead:** The additional functionality of a DBMS may introduce overhead, potentially leading to slower performance compared to simpler systems.

4. **Large Size:** DBMS systems can consume substantial memory and storage resources, increasing hardware costs.
5. **Security Risks:** Despite robust security features, DBMS systems are prime targets for cyber-attacks; a breach can have severe consequences.
6. **Backup and Recovery Complexity:** Setting up and managing backup and recovery systems can be complex and time-consuming, despite their powerful capabilities.
7. **Potential for Misuse:** Improper use or configuration of a DBMS can result in data loss, corruption, or security vulnerabilities.

Applications of Databases

Databases are used in a wide range of applications, including:

- Movie Theater
- Banking and finance
- Healthcare
- E-commerce
- Social media
- Telecommunications
- Scientific research

Database Languages in DBMS

1. Data Definition Language (DDL)
 - **Purpose:** DDL is used to define and modify the structure of the database and its schema.

- **Common Commands:**

- **CREATE:** Used to create database objects such as tables and indexes.
- **ALTER:** Modifies existing database structures.
- **DROP:** Deletes database objects.
- **TRUNCATE:** Removes all records from a table without affecting its structure.

2. Data Manipulation Language (DML)

- **Purpose:** DML allows users to access and manipulate data stored in the database.

- **Common Commands:**

- **SELECT:** Retrieves data from the database.
- **INSERT:** Adds new records to a table.
- **UPDATE:** Modifies existing records.
- **DELETE:** Removes records from a table.

3. Data Control Language (DCL)

- **Purpose:** DCL is used to control access to data within the database.

- **Common Commands:**

- **GRANT:** Provides specific privileges to users.
- **REVOKE:** Removes specific privileges from users.

4. Transaction Control Language (TCL)

- **Purpose:** TCL manages transactions in the database, ensuring data integrity.

- **Common Commands:**
 - **COMMIT:** Saves all changes made during the current transaction.
 - **ROLLBACK:** Undoes changes made during the current transaction.

Relational Database Management System

RDBMS:- RDBMS stands for **Relational Database Management System**. It is a type of database management system that stores data in a structured format using rows and columns. This structure makes it easy to organize, manage, and retrieve data.

Properties of RDBMS

- Stores data in the form of table.
- Does not require the user to understand its physical implementation.
- Provides information about its content and structure in the system table.
- Support the concept of null values.

Transactions: A sequence of operations performed as a single logical unit of work. Transactions ensure data consistency and integrity, following the ACID properties:

Features of RDBMS

- **ACID support:** Ensures data integrity through Atomicity, Consistency, Isolation, and Durability.
- **Multi-user access:** Allows concurrent access to the database while maintaining data integrity.
- **Data durability:** Ensures data is not lost and remains available.

- **Data consistency:** Maintains data integrity and validity.
- **Data flexibility:** Allows for complex queries and relationships through SQL.
- **Hierarchical relationships:** Enables parent-child relationships between tables.

Popular RDBMS: Some widely used RDBMS include:

- **MySQL:** Open-source RDBMS widely used for web applications.
- **PostgreSQL:** Open-source RDBMS is known for its robustness and advanced features.
- **Oracle Database:** Commercial RDBMS is known for its scalability and enterprise-level features.
- **Microsoft SQL Server:** Commercial RDBMS from Microsoft, widely used in enterprise environments.

RDBMS is widely used in various applications, from small-scale projects to large enterprise systems, due to its ability to efficiently manage and retrieve structured data while ensuring data integrity and security.

Normalization

Normalization: The process of organizing data to reduce redundancy and improve data integrity. This involves dividing a database into two or more tables and defining relationships between them. The main aim is to divide a database into two or more tables and define relationships between the tables to ensure data integrity and avoid anomalies. The process involves applying a series of rules or normal forms to achieve a well-structured database.

The main objectives of normalization are:

1. **Eliminating redundant data:** Normalization removes duplicate data, reducing storage requirements and potential data inconsistencies.
2. **Minimizing data anomalies:** Normalization prevents data anomalies that can occur during insert, update, or delete operations.
3. **Simplifying queries:** Normalized databases have a clear structure, making queries simpler and more efficient.
4. **Improving data integrity:** Normalization ensures data integrity by defining relationships and constraints between tables.

Here are the key normal forms:

1. First Normal Form (1NF)

- **Rule:** Ensure that each table has a primary key and that each column contains atomic (indivisible) values. There should be no repeating groups or arrays.
- **Objective:** Eliminate duplicate columns and create separate tables for related data.
- **Example:**
- **Before:**

OrderID	Product1	Product2	Product3
1	Apple	Banana	Orange

After:

OrderID	Product
1	Apple
1	Banana
1	Orange

2. Second Normal Form (2NF)

- **Rule:** Ensure the database is in 1NF and that all non-key attributes are fully functionally dependent on the primary key.
- **Objective:** Eliminate partial dependency (i.e., where non-key attributes depend on part of a composite primary key).
- **Example:**
 - **Before:**

OrderID	ProductID	ProductName	Price
1	101	Apple	0.5
1	102	Banana	0.3

3. Third Normal Form (3NF)

- ◆ **Rule:** Ensure the database is in 2NF and that all the attributes are functionally dependent only on the primary key.

- ◆ **Objective:** Eliminate transitive dependency (i.e., where non-key attributes depend on other non-key attributes).
- ◆ **Example:**

Before:

OrderID	ProductID	SupplierID	SupplierName
1	101	201	SupplierA

After:

Orders Table:

OrderID	ProductID	SupplierID
1	101	201

Suppliers Table:

SupplierID	SupplierName
201	SupplierA

Benefits of Normalization

- **Reduced Data Redundancy:** Minimizes duplicate data.
- **Improved Data Integrity:** Ensures data consistency and accuracy.
- **Efficient Data Management:** Simplifies data maintenance and updates.
- **Enhanced Query Performance:** Often leads to better-structured queries.

Normalization helps in creating a clear, efficient, and flexible database design, making it easier to manage and scale as the amount of data grows.

ER Diagram

An Entity-Relationship (ER) diagram is a graphical representation of an information system that depicts the relationships between entities in a database. It is used during the database design phase to illustrate the logical structure of the database and to ensure that all necessary components and their relationships are clearly defined.

What is an ER Diagram?

- **Definition:** An ER Diagram is a type of flowchart that illustrates how entities (such as people, objects, or concepts) relate to each other within a system. It is primarily used in database design to model and visualize the logical structure of a database.
- **Components of an ER Diagram**
 1. **Entities:** Objects or concepts that can have data stored about them. Entities are usually nouns, such as "Customer," "Order," or "Product." In the diagram, they are represented by rectangles.
 2. **Attributes:** Characteristics or properties of an entity. Attributes are usually adjectives or properties, such as "Customer Name," "Order Date," or "Product Price." In the diagram, they are represented by ovals and are connected to their respective entities.
 3. **Relationships:** Associations between entities. Relationships are usually verbs or phrases, such as "places," "contain," or "buy." In the diagram, they are represented by diamonds and are connected to the entities they associate.
 4. **Primary Key:** An attribute or a set of attributes that uniquely identifies an entity instance. It is usually underlined in the diagram.
 5. **Foreign Key:** An attribute in one entity that links to the primary key of another entity. It establishes a relationship between the entities.

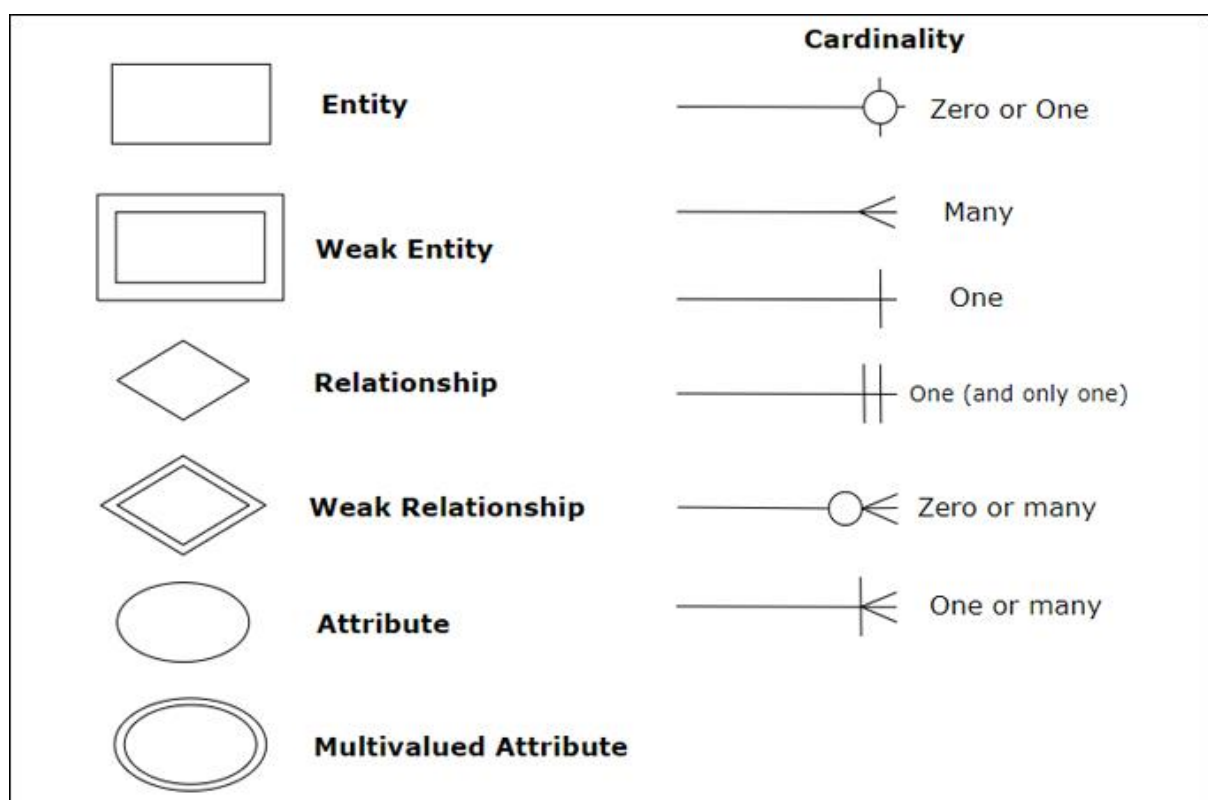
Purpose of ER Diagrams

- **Database Design:** ER diagrams help in modeling and designing relational databases, allowing developers to visualize the structure before implementation.

- **Troubleshooting:** They can be used to analyze existing databases to identify and resolve issues related to logic or deployment.
- **Business Process Analysis:** ER diagrams assist in designing and analyzing databases used in business processes, helping to streamline operations.
- **Education and Research:** They are valuable for planning data structures in educational contexts and for research purposes, where structured data analysis is required.

Symbols Used in ER Diagrams

- **Rectangles:** Represent entities.
- **Ellipses:** Represent attributes.
- **Diamonds:** Represent relationships.
- **Double Ellipses:** Indicate multi-valued attributes.
- **Double Rectangles:** Represent weak entities.
- **Double Diamonds:** Represent weak relationships.
- **Lines:** Connect entities to their attributes and relationships.

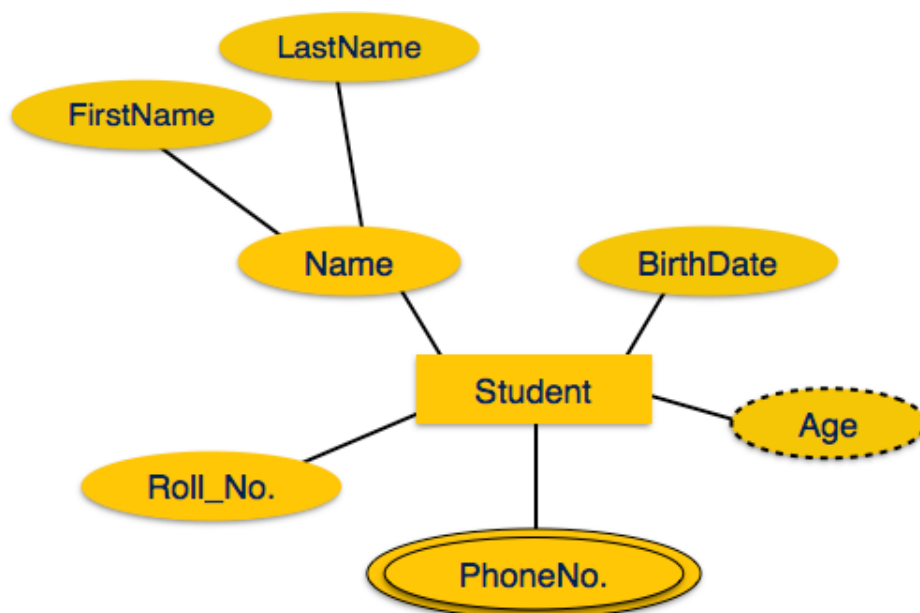


Types of Relationships

- **One-to-One (1:1):** Each entity in the relationship corresponds to one entity in the other set.
- **One-to-Many (1:N):** A single entity in one set can be associated with multiple entities in another set.
- **Many-to-Many (M:N):** Entities in both sets can have multiple associations with each other.

ER Diagram Representation:

NAME	ROLL NO.	AGE
+-----+	+-----+	+-----+
EMILY WHITE	21	18
SANIYA SHARMA	42	17
ADITI GUPTA	1	19
+-----+	+-----+	+-----+



In this diagram:

- The "NAME" entity is connected to the "ROLL NO." entity with a one-to-one (1) relationship because one NAME can have only one roll no.

ER diagrams are valuable tools for database design as they provide a clear and concise way to visualize the structure and relationships within a database, making it easier to implement and maintain.

Benefits of Using ER Diagrams

- **Visualization:** They provide a clear visual representation of data structures, making it easier to understand how entities interact.
- **Simplification:** ER diagrams reduce complexity by breaking down data relationships into understandable components.
- **Standardization:** They offer a standard method for modeling data, facilitating communication among stakeholders during the database design process.

Database Schema

A database schema is a blueprint or architecture of how a database is structured. It defines the organization of data within the database, including tables, columns, relationships, constraints, indexes, and views. The schema provides a logical view of the entire database, outlining how data is stored, accessed, and manipulated. It is essential for ensuring data consistency, integrity, and efficient retrieval.

Key Components of a Database Schema

1. Tables:

- **Definition:** Core components representing entities, composed of rows (records) and columns (attributes).
- **Example:** A "Customers" table with columns like CustomerID, Name, Email, and Phone.

2. Columns and Data Types:

- **Definition:** Columns represent entity attributes, each assigned a specific data type (e.g., INT, VARCHAR).
- **Example:** In a "Movies" table, Title as VARCHAR and ReleaseDate as DATE.

3. Primary Keys:

- **Definition:** Unique identifiers for each record in a table, ensuring no duplicate values.
- **Example:** MovieID serves as the primary key in a "Movies" table.

4. Foreign Keys:

- **Definition:** Columns that create relationships between tables, referencing primary keys in other tables.
- **Example:** CustomerID in "Bookings" referencing CustomerID in the "Customers" table.

5. Indexes:

- **Definition:** Structures that speed up data retrieval by providing quick access to rows in a table.
- **Example:** An index on the Email column in a "Customers" table for fast email searches.

6. Constraints:

- **Definition:** Rules that enforce data accuracy and integrity (e.g., NOT NULL, UNIQUE).
- **Example:** A NOT NULL constraint on the Email column ensures every customer has an email address.

7. Relationships:

- **Definition:** Connections between tables, commonly one-to-one, one-to-many, or many-to-many.

- **Example:** A one-to-one relationship between "name" and "age" where one person can have only one age.

8. Views:

- **Definition:** Virtual tables created by SELECT queries that join and filter data from other tables.
- **Example:** A view showing active customers by joining "Customers" and "Orders."

Types of Database Schemas

1. Logical Schema:

- **Definition:** The logical schema defines the structure of the data and the relationships between them without considering the physical storage. It includes entities, attributes, and relationships.
- **Example:** A logical schema for an e-commerce database might include entities like Customers, Orders, Products, and Payments.

2. Physical Schema:

- **Definition:** The physical schema details how data is stored in the database, including file storage, indexing, and partitioning.
- **Example:** The physical schema might define that the "Orders" table is stored on a particular server and is indexed by OrderID.

3. Conceptual Schema:

- **Definition:** The conceptual schema provides an overview of the entire database architecture, focusing on the high-level entities and their relationships.
- **Example:** A conceptual schema might depict the overall structure of a hospital management system, showing how Patients, Doctors, and Appointments are interrelated.

Tools for Designing Database Schemas

1. ER Diagrams (Entity-Relationship Diagrams):

- **Purpose:** Visual tools for designing database schemas, showing entities, attributes, and relationships.
- **Tools:** MySQL Workbench, Microsoft Visio.

2. Database Design Tools:

- **MySQL Workbench:** A popular tool for visual database design, supporting the creation of ER diagrams, schema validation, and SQL scripting.
- **Microsoft SQL Server Management Studio (SSMS):** A tool for designing and managing SQL Server databases, including schema design and query optimization.
- **Oracle SQL Developer:** A tool for designing and managing Oracle databases, providing features like data modeling and schema comparison.

SQL (Structured Query Language)

SQL stands for Structured Query Language. SQL is a computer language used to interact with relational database systems. SQL is a tool for organizing, managing, and retrieving archived data from a computer database.

When data needs to be retrieved from a database, SQL is used to make the request. The DBMS processes the SQL query retrieves the requested data and returns it to us. Rather, SQL statements describe how a collection of data should be organized or what data should be extracted or added to the database.

Importance of SQL

Universality: Standard language for relational databases; compatible with systems like MySQL, SQL Server, and Oracle.

- **Ease of Use:** Simple, human-readable syntax; easy to learn for non-programmers.
- **Efficient Data Retrieval:** Quickly filter and retrieve specific data from large databases for reporting and insights.
- **Data Manipulation:** Easily insert, update, and delete data, ensuring accuracy and up-to-date information.
- **Data Integrity and Security:** Enforces rules for consistent data and provides mechanisms for user access control.
- **Scalability:** Handles large data volumes efficiently, suitable for businesses of all sizes.
- **Integration:** Can be integrated with various programming languages and tools, enhancing versatility in data management.

History and Development

- SQL was developed in the 1970s by IBM researchers as part of the System R project.
- It became a standard language for relational database management systems (RDBMS) and was adopted by ANSI in 1986 and ISO in 1987.

MY SQL WORKBENCH

MySQL Workbench is a powerful, integrated visual tool designed for database architects, developers, and administrators to work with MySQL databases. It provides a comprehensive set of tools for designing, developing, and managing MySQL databases. MySQL Workbench is widely used due to its intuitive interface, extensive feature set, and seamless integration with MySQL databases, making it an essential tool for anyone working with MySQL.

Key Features of MySQL Workbench

1. Data Modeling & Design:

- **ER Diagrams:** Visually design and map database schemas.
- **Forward/Reverse Engineering:** Generate or update database schemas from ER diagrams.
- **Model Synchronization:** Keep your database design in sync with the actual database.

2. SQL Development:

- **SQL Editor:** Advanced editor with syntax highlighting and auto-completion.
- **Visual Query Builder:** Create SQL queries graphically without writing code.
- **Result Management:** View and export query results in various formats.

3. Database Administration:

- **User & Permission Management:** Manage database users and their access rights.
- **Server Configuration:** Adjust server settings directly from the interface.
- **Data Import/Export:** Easily move data between the database and external files.
- **Backup & Restore:** Perform database backups and restore operations.

4. Performance Monitoring & Tuning:

- **Performance Reports:** Get insights on query performance and server health.
- **Visual Explain Plans:** Analyze query execution visually for optimization.
- **Health Monitoring:** Track server metrics in real-time.

5. Database Migration:

- **Migration Wizard:** Seamlessly migrate databases from other platforms to MySQL.

Uses of MySQL Workbench (Brief Points)

- **Database Design:** Create and manage database schemas using ER diagrams.
- **SQL Development:** Write, edit, and execute SQL queries efficiently.
- **Data Management:** Import, export, and manage database data.
- **User Management:** Control user access and permissions.
- **Server Administration:** Configure and monitor MySQL server settings.
- **Performance Tuning:** Analyze and optimize query performance.
- **Database Migration:** Migrate databases from other platforms to MySQL

Query

Query in DBMS (Database Management System)

A query in DBMS is a request for data or information from a database. It allows users to interact with the database by retrieving, updating, deleting, or inserting data. Queries are written in Structured Query Language (SQL) and serve as the primary method for managing and manipulating the data stored in a relational database.

Purpose of a Query

- **Data Retrieval:** To fetch data based on specified criteria.
- **Data Manipulation:** To update, insert, or delete data.
- **Data Definition:** To define or alter the structure of database tables.
- **Control Access:** To manage permissions and user roles.

Types of Queries in DBMS

1. Basic SQL Queries

These are fundamental types used to interact with the database.

- **SELECT:**
 - Retrieves data from one or more tables.
 - Example: `SELECT * FROM employees;`
- **INSERT:**
 - Adds new data into a table.
 - Example: `INSERT INTO employees (name, age) VALUES ('Alice', 30)`
- **UPDATE:**
 - Modifies existing data in a table.
 - Example: `UPDATE employees SET salary = 50000 WHERE name = 'Alice';`
- **DELETE:**
 - Removes data from a table.
 - Example: `DELETE FROM employees WHERE age > 50;`

2. Aggregate Functions Queries

These are used to perform calculations on multiple rows and return a single value.

- **SUM()**
 - Purpose: Returns the total sum of a numeric column.
 - Example: `SELECT SUM(salary) FROM employees;`

- **AVG()**
 - Purpose: Returns the average value of a numeric column.
 - Example: `SELECT AVG(salary) FROM employees;`

- **COUNT()**
 - Purpose: Returns the number of rows that match a specific condition.
 - Example: `SELECT COUNT(*) FROM employees WHERE department = 'HR';`

- **MAX()**
 - Purpose: Returns the maximum value in a column.
 - Example: `SELECT MAX(salary) FROM employees;`

- **MIN()**
 - Purpose: Returns the minimum value in a column.
 - Example: `SELECT MIN(salary) FROM employees;`

1. Advanced Queries

- **GROUP BY:**

- Groups rows that have the same values into summary rows, often used with aggregate functions.
- Example: `SELECT department, SUM(salary) FROM employees GROUP BY department;`

- **HAVING:**

- Filters the results after the `GROUP BY` clause, similar to `WHERE`, but for aggregate data.
- Example: `SELECT department, SUM(salary) FROM employees GROUP BY department HAVING SUM(salary) > 100000;`

- **ORDER BY:**

- Sorts the result set in ascending or descending order.
- Example: `SELECT * FROM employees ORDER BY salary DESC;`

- **JOIN Queries:** Combines rows from two or more tables based on a related column.

- **INNER JOIN:** Returns records with matching values in both tables.
 - Example: `SELECT employees.name, departments.department_name FROM employees INNER JOIN departments ON employees.department_id = departments.id;`
- **LEFT JOIN:** Returns all records from the left table, and the matched records from the right table.

- Example: `SELECT employees.name, departments.department_name FROM employees LEFT JOIN departments ON employees.department_id = departments.id;`
- **RIGHT JOIN:** Returns all records from the right table, and the matched records from the left table.
 - Example: `SELECT employees.name, departments.department_name FROM employees RIGHT JOIN departments ON employees.department_id = departments.id;`
- **UNION:**
 - Combines the result of two or more `SELECT` statements into a single result set, removing duplicates.
 - Example: `SELECT name FROM employees UNION SELECT name FROM managers;`
- **UNION ALL:**
 - Same as `UNION`, but includes duplicates.
 - Example: `SELECT name FROM employees UNION ALL SELECT name FROM managers;`
- **Subqueries:**
 - A query nested inside another query to retrieve data based on a specific condition.
 - Example: `SELECT name FROM employees WHERE salary > (SELECT AVG(salary) FROM employees);`
- **EXISTS:**
 - Checks if any rows are returned by the subquery.

- Example: `SELECT name FROM employees WHERE EXISTS (SELECT * FROM departments WHERE department_name = 'HR');`
- **BETWEEN:**
 - Used to filter data within a range.
 - Example: `SELECT * FROM employees WHERE salary BETWEEN 30000 AND 60000;`
- **LIKE:**
 - Used for pattern matching with strings.
 - Example: `SELECT * FROM employees WHERE name LIKE 'A%';`

6. Transaction Control Queries

- **COMMIT:**
 - Saves all changes made during the transaction.
 - Example: `COMMIT;`
- **ROLLBACK:**
 - Undoes all changes made during the transaction.
 - Example: `ROLLBACK;`
- **SAVEPOINT:**
 - Sets a point within a transaction to which you can roll back.
 - Example: `SAVEPOINT save1;`

Hardware and Software

Introduction:

Understanding the basics of hardware and software is fundamental to grasping how computer systems operate. Both are integral components of any computing device, from personal computers to large-scale enterprise systems.

Hardware

Hardware refers to the physical components of a computer system that you can touch and see. These components are responsible for executing the tasks defined by the software.

Software

Software refers to the instructions and data that tell the hardware what to do. It is intangible and runs on the hardware to perform specific tasks.

Types of Hardware

- **Central Processing Unit (CPU):**
 - Often referred to as the "brain" of the computer.
 - Executes instructions from software and performs calculations.
 - Components include the arithmetic logic unit (ALU) and control unit (CU).
- **Memory (RAM):**
 - Temporary storage is used by the CPU to store data and instructions currently being used.
 - Volatile memory, meaning it loses its data when the power is turned off.
- **Computer:** A digital machine that processes data, executes programs, and performs a wide range of tasks using both hardware and software.
- **Keyboard:** An input device with keys for typing text and executing commands on a computer.
- **Mouse:** A handheld device used to navigate and interact with a computer's graphical user interface.

- Graphics **Processing Unit (GPU)**:
 - Specialized processor for rendering graphics and images.
 - Important for gaming, video editing, and scientific computations.
- Network **Interface Card (NIC)**:
 - Allows the computer to connect to a network and communicate with other devices.

Types of Software

1. MySQL Workbench:

- **Database Design and Management:** MySQL Workbench is a unified visual tool for database architects, offering features like data modeling, SQL development, and comprehensive administration tools for MySQL databases.
- **Cross-Platform Compatibility:** It is available on Windows, macOS, and Linux, providing a consistent interface for database design and management across different operating systems.

2. Smart Draw :

- **Diagramming and Visualization Tool:** SmartDraw is a powerful software used to create diagrams, flowcharts, and technical drawings, offering templates and automation features.

Conclusion

Understanding the interplay between hardware and software is essential for anyone involved in computing. This synergy enables the wide range of applications and functionalities we rely on in modern computing. Hardware provides the foundation upon which software operates, and software directs the hardware to perform specific tasks, creating a seamless user experience.

Database Design

ER DIAGRAM

An Entity-Relationship (ER) diagram visually represents the entities in a database and the relationships between them. Below is a brief overview of the entities, their attributes, and the relationships for the given tables in a movie booking database system .

Entities and Attributes

- Entity- **Movies**

- **Attributes:**

- **MovieID:** Unique identifier for each movie.
 - **Title:** Name of the movie (Primary Key).
 - **Genre:** Genre of the movie (e.g., Action, Comedy).
 - **Director:** Director of the movie.
 - **Duration:** Length of the movie in minutes.
 - **Rating:** Average rating of the movie.
 - **Price:** Cost of watching the movie.

- Entity- **Theaters**

- **TheaterID:** Unique identifier for each theater.
 - **Name:** Name of the theater (Primary Key).
 - **Location:** Physical location of the theater.
 - **Capacity:** Number of seats available in the theater.
 - **Showtimes:** Scheduled times for movie screenings.

- Entity- **Showtimes**

- **ShowtimeID:** Unique identifier for each showtime (Primary Key).
 - **Title:** Foreign key referencing the Movies table.
 - **Theater_name:** Foreign key referencing the Theaters table.
 - **ShowDateTime:** Date and time when the movie is shown.

- Entity- **Customers**

- **CustomerID:** Unique identifier for each customer (Primary Key).
 - **Name:** Name of the customer.
 - **Email:** Contact email of the customer.
 - **Phone:** Phone number of the customer.

- Entity- **Bookings**

- **BookingID**: Unique identifier for each booking (Primary Key).
- **CustomerID**: Foreign key referencing the Customers table.
- **ShowtimeID**: Foreign key referencing the Showtimes table.
- **SeatsBooked**: Number of seats booked by the customer.

Relationships

1. Movies to Showtimes:

- **Relationship Type**: One-to-Many
- **Details**: A movie can have multiple showtimes in different theaters or at different times.

2. Theaters to Showtimes:

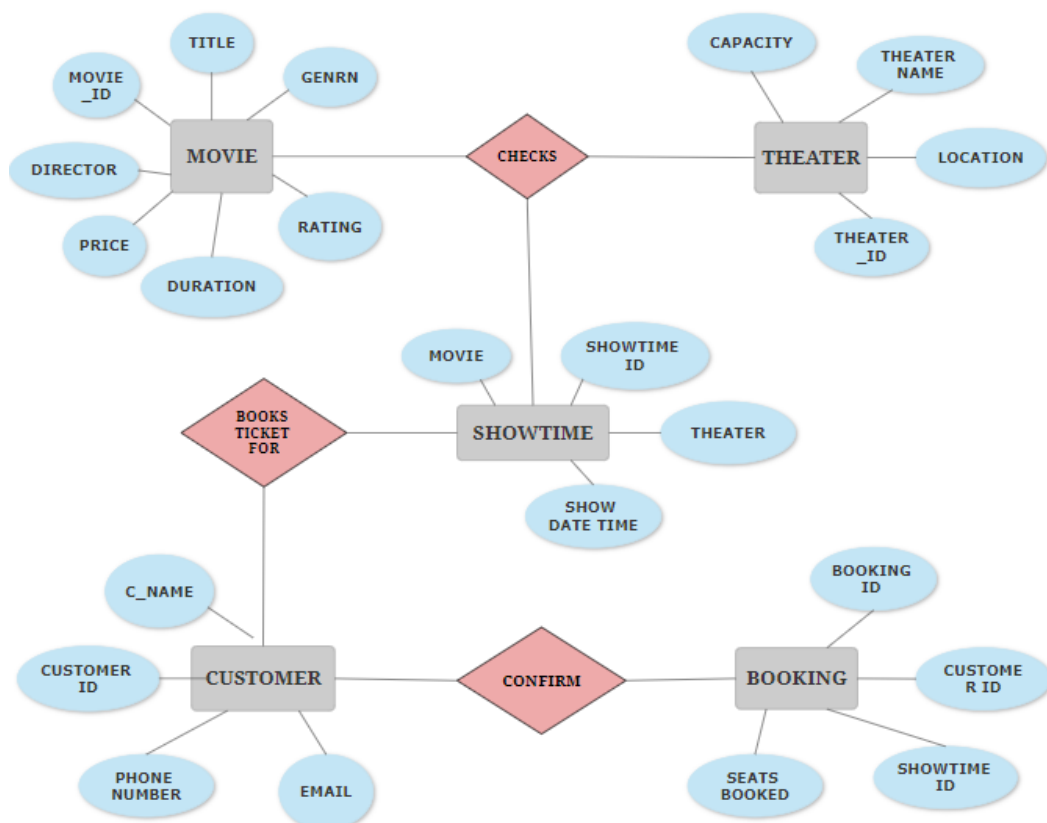
- **Relationship Type**: One-to-Many
- **Details**: A theater can have multiple showtimes for different movies.

3. Showtimes to Bookings:

- **Relationship Type**: One-to-Many
- **Details**: Each showtime can have multiple bookings by different customers.

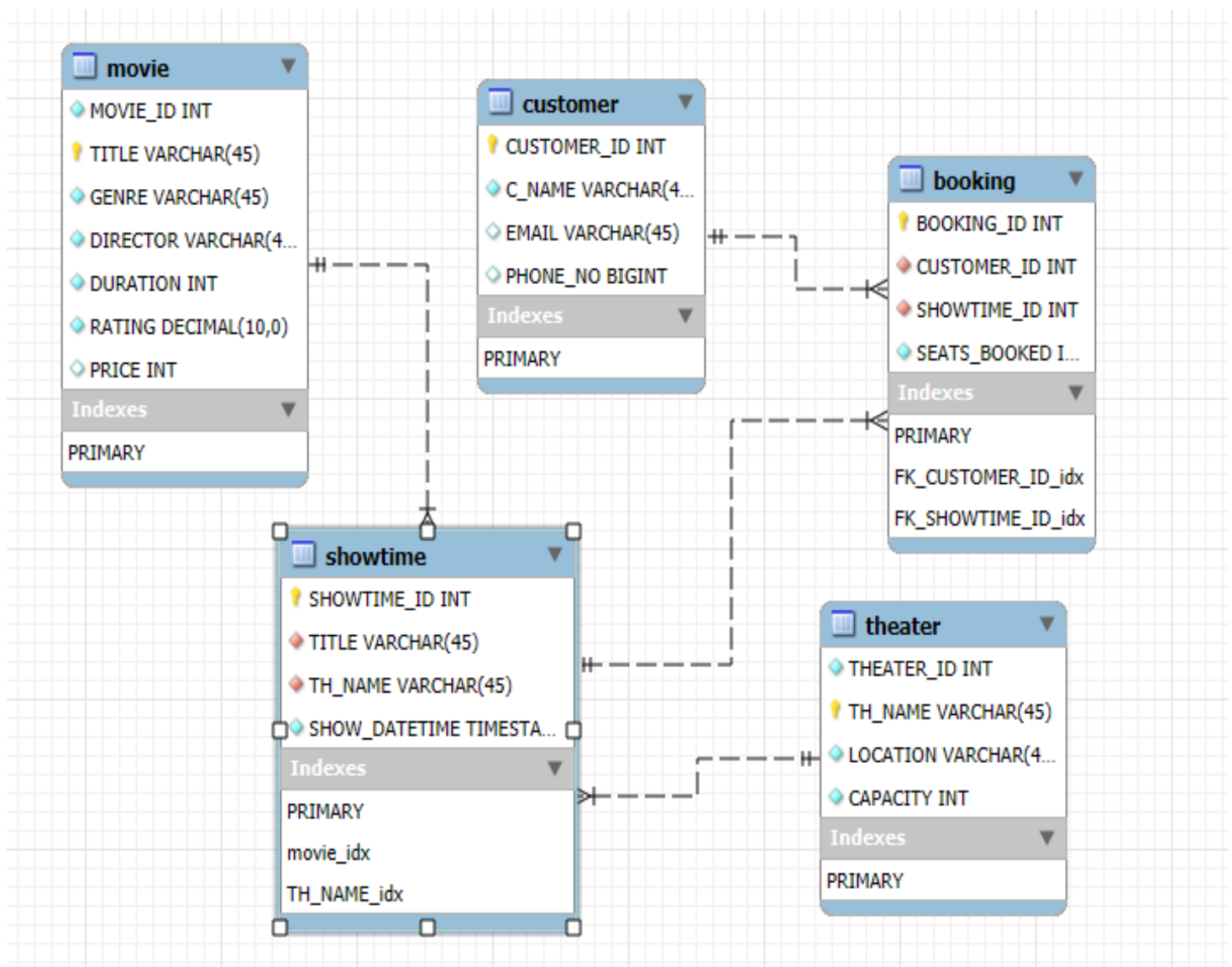
4. Customers to Bookings:

- **Relationship Type**: One-to-Many
- **Details**: A customer can make multiple bookings for different showtimes.



Database Schema

A database schema is a blueprint or logical structure of a database that defines how data is organized and how the relationships among the data are associated. It is a collection of database objects, including tables, views, indexes, procedures, and constraints, and serves as a guide for the database system to interpret and manage data.



Normalization Techniques

To apply normalization techniques to the provided database design, we'll consider the process of organizing the data into normal forms. Here's an explanation of the normalization applied to each table, along with a description of the ER diagram.

Normalization Techniques

First Normal Form (1NF)

Requirements:

- Each table must have a primary key.
- Each column should contain atomic values.
- Each row must be unique.

Application:

- **Movies Table:** Each movie has unique attributes, and there are no repeating groups or arrays. The primary key is `Title`.
- **Theaters Table:** Each theater has unique attributes. The primary key is `Name`.
- **Showtimes Table:** Each showtime has unique attributes, and each showtime is uniquely identified by `ShowtimeID`.
- **Customers Table:** Each customer has unique attributes, with a primary key `CustomerID`.
- **Bookings Table:** Each booking has unique attributes, with a primary key `BookingID`.
-

Second Normal Form (2NF)

Requirements:

- Achieve 1NF.
- Remove partial dependencies; all non-key attributes must depend on the entire primary key.

Application:

- Since all tables have a single primary key, there are no partial dependencies to resolve. Each non-key attribute in every table fully depends on the primary key.

Third Normal Form (3NF)

Requirements:

- Achieve 2NF.
- Remove transitive dependencies; no non-key attribute should depend on another non-key attribute.

Application:

- **Movies Table:** Attributes like Genre, Director, Duration, Rating, and Price depend only on Title.
- **Theaters Table:** Attributes Location and Capacity depend only on Name.
- **Showtimes Table:** ShowDateTime depends only on ShowtimeID.
- **Customers Table:** Name, Email, and Phone depend only on CustomerID.
- **Bookings Table:** SeatsBooked depends only on BookingID.

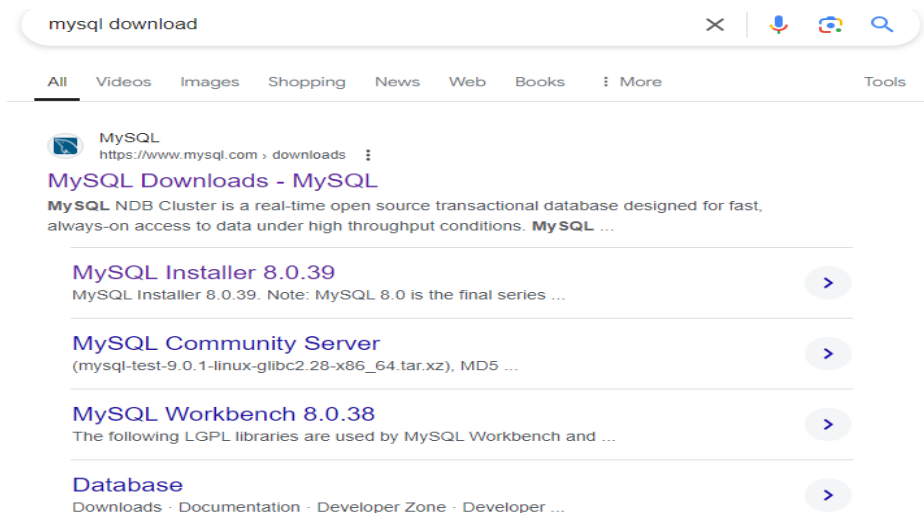
IMPLEMENTATION

Steps to Install MySQL Workbench

Steps to Download and Install MySQL from the Official Website

1. Visit the MySQL Website:

- Open your web browser and go to www.mysql.com.



2. Navigate to Downloads:

- Click on the "Downloads" tab located in the main menu.

3. Select MySQL Community Downloads:

- Scroll down to find the "MySQL Community (GPL) Downloads" section and click on it.



4. Choose MySQL Installer:

- Click on "**MySQL Installer for Windows**". You will see options for the web installer and the full installer.

MySQL Community Downloads

- [MySQL Yum Repository](#)
- [MySQL APT Repository](#)
- [MySQL SUSE Repository](#)
- [MySQL Community Server](#)
- [MySQL NDB Cluster](#)
- [MySQL Router](#)
- [MySQL Shell](#)
- [MySQL Operator](#)
- [MySQL NDB Operator](#)
- [MySQL Workbench](#)
- [MySQL Installer for Windows](#)
- [C API \(libmysqlclient\)](#)
- [Connector/C++](#)
- [Connector/J](#)
- [Connector/NET](#)
- [Connector/Node.js](#)
- [Connector/ODBC](#)
- [Connector/Python](#)
- [MySQL Native Driver for PHP](#)
- [MySQL Benchmark Tool](#)
- [Time zone description tables](#)
- [Download Archives](#)

ORACLE © 2024 Oracle


[Privacy](#) / [Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie Preferences](#)

5. Download

- Click on download where windows (x86,32-bit),MSI Installer 8.0.39

[General Availability \(GA\) Releases](#) [Archives](#) [i](#)

MySQL Installer 8.0.39

 **Note:** MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.


Select Version:

8.0.39

Select Operating System:

Microsoft Windows

Windows (x86, 32-bit), MSI Installer <small>(mysql-installer-web-community-8.0.39.0.msi)</small>	8.0.39	2.1M	Download
		MD5: d8499da0b2c4b5dFa81a5c5185af9238 Signature	
Windows (x86, 32-bit), MSI Installer <small>(mysql-installer-community-8.0.39.0.msi)</small>	8.0.39	303.6M	Download
		MD5: 353c5e5ab9350d0e9ddcb42264229b5d Signature	

 We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

6. Skip Login:

- You may be prompted to log in or sign up. Click on "**No thanks, just start my download**" to proceed without logging in.

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »
using my Oracle Web account


Sign Up »
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can sign up for a free account by clicking the Sign Up link and following the instructions.

[No thanks, just start my download.](#)

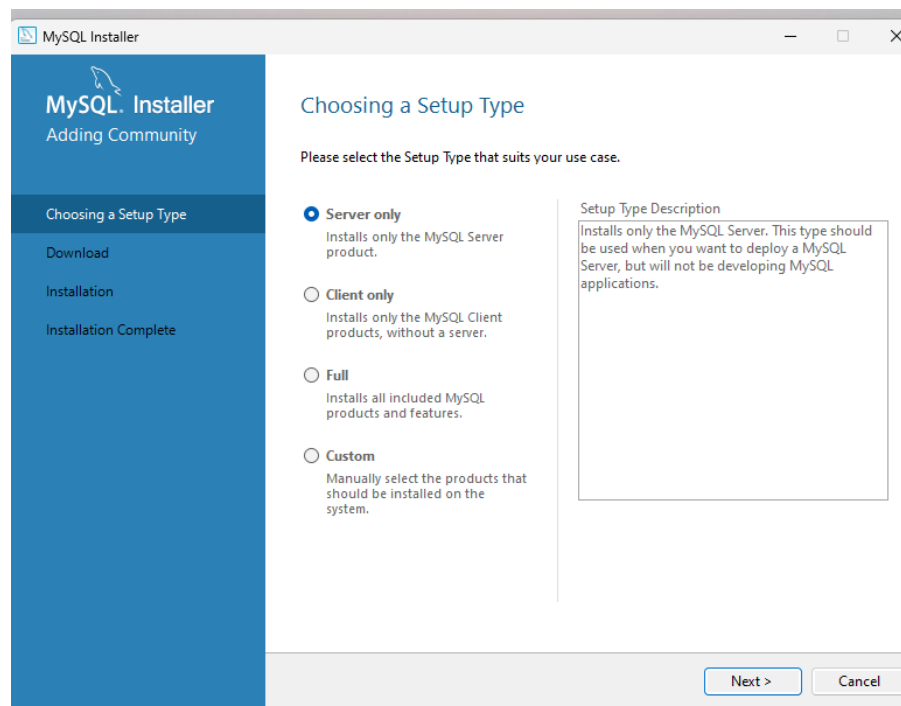
7. Run the Installer:

- Once the download is complete, locate the installer file (usually in your Downloads folder) and double-click it to start the installation process.

Name	Date modified	Type	Size
▼ Today			
 mysql-installer-web-community-8.0.39.0	08-08-2024 11:07	Windows Installer ...	2,160 KB
▼ Last week			

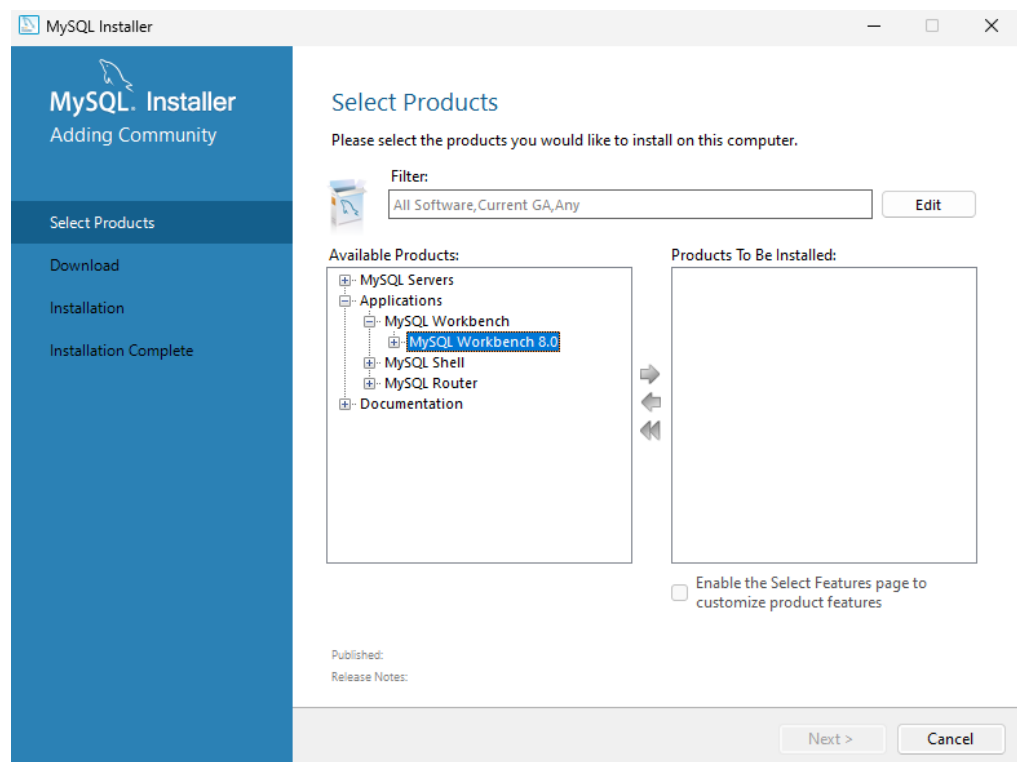
8. Choose Setup Type:

- In the installer, select the setup type. Click on **"Custom"** to install all components. Click **"Next"**.



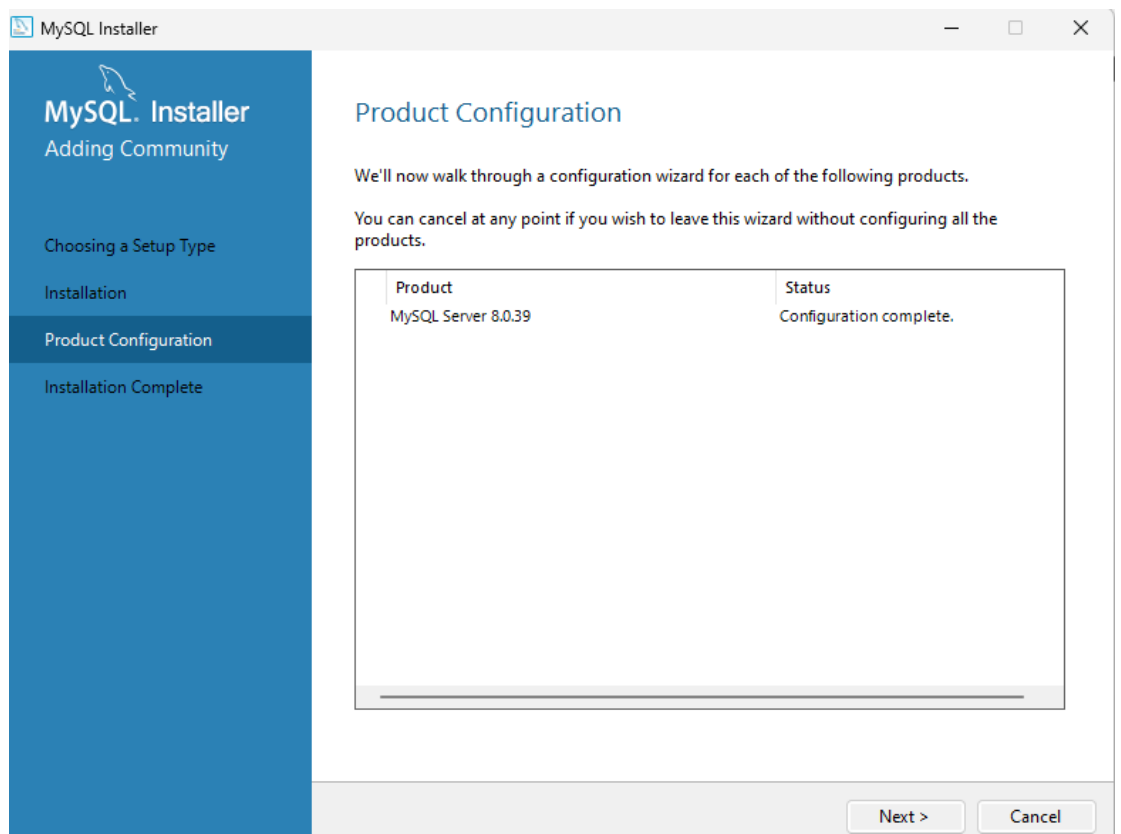
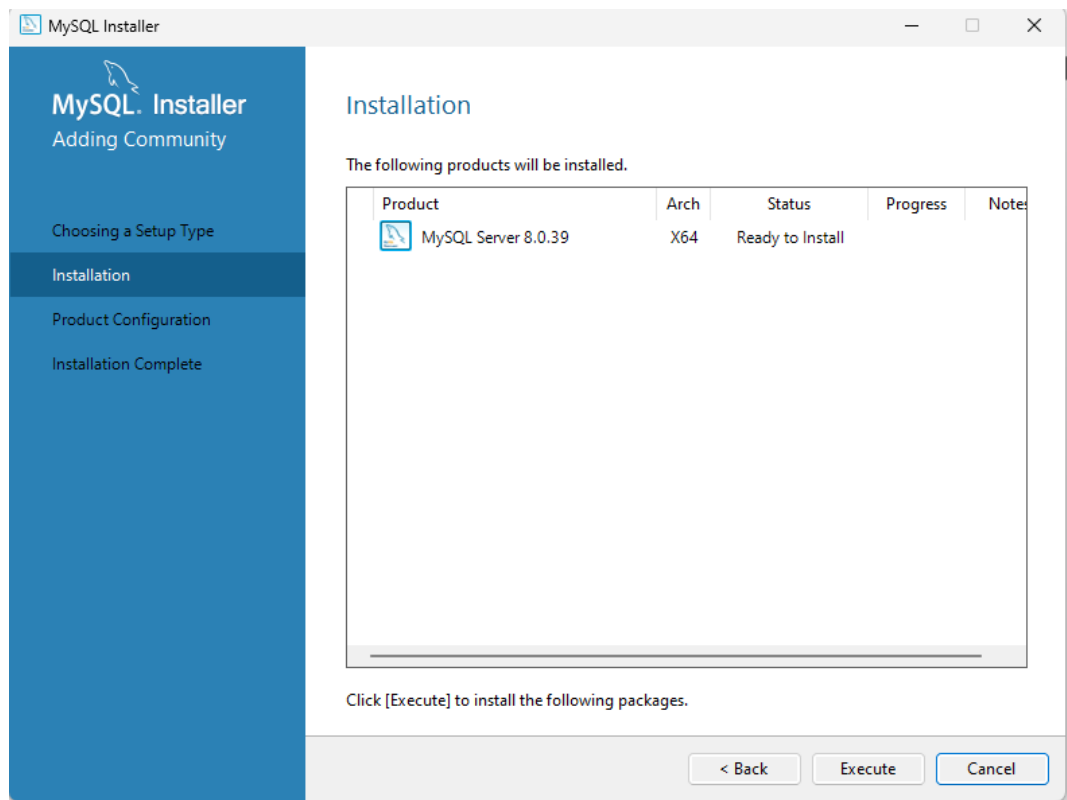
9. Check Requirements:

- The installer will check for necessary requirements. If any are missing, follow the prompts to install them. Click **"Next"** once all requirements are met.



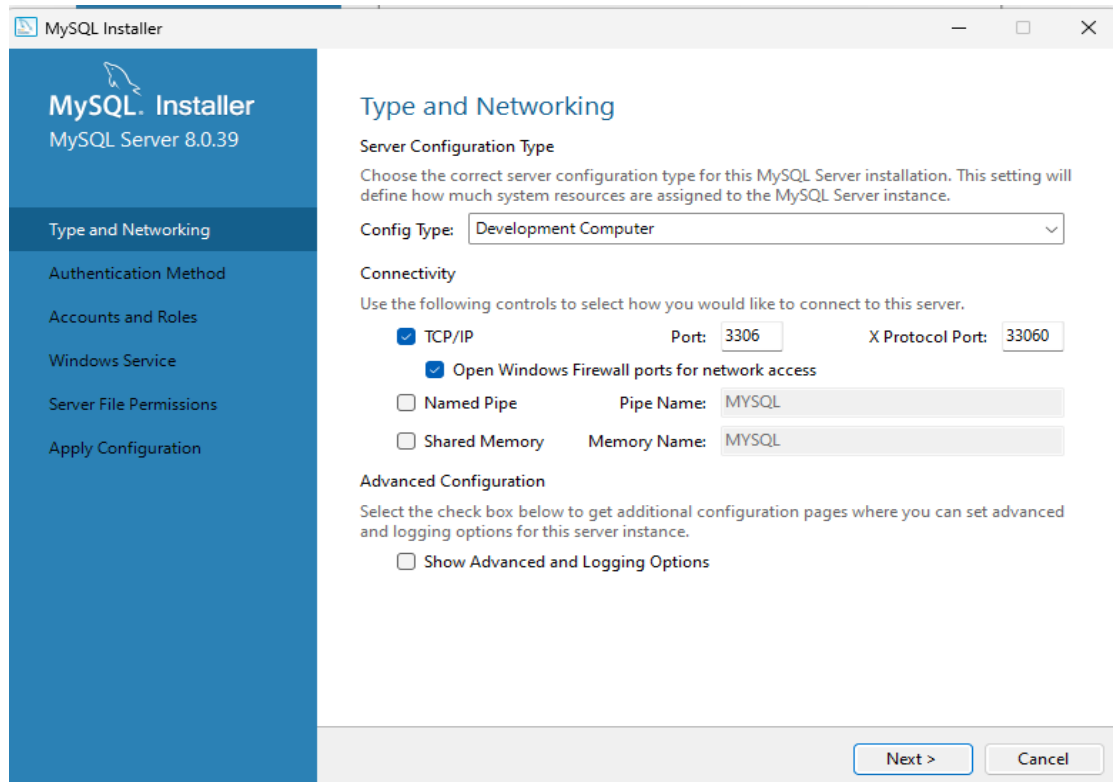
10. Install MySQL Products:

- Click **"Execute"** to begin installing the selected products. Wait for the installation to complete and click **"Next"**.



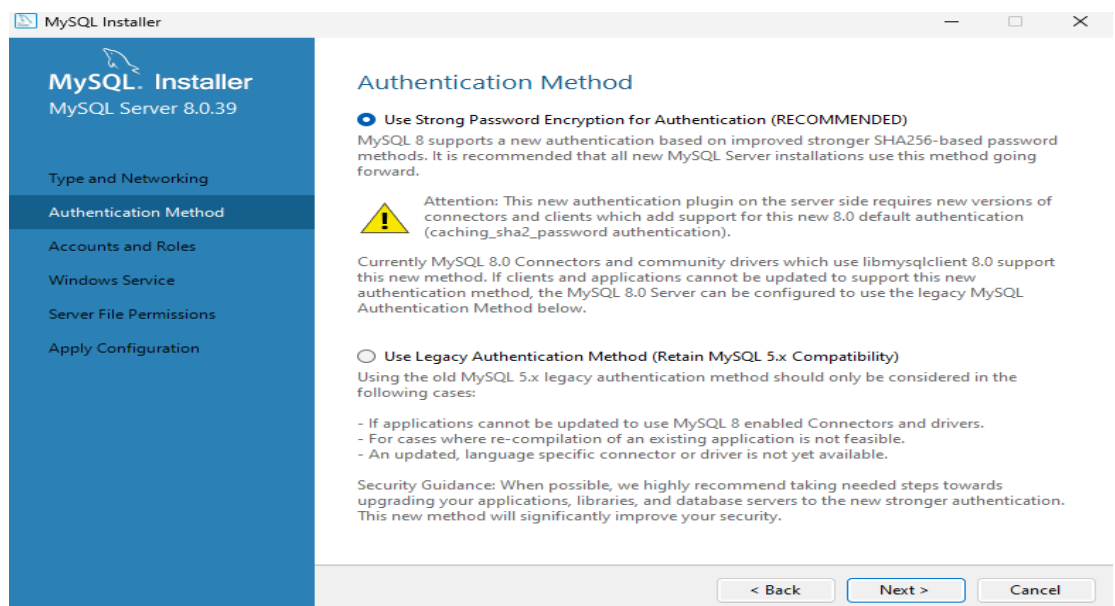
11. Configure MySQL Server:

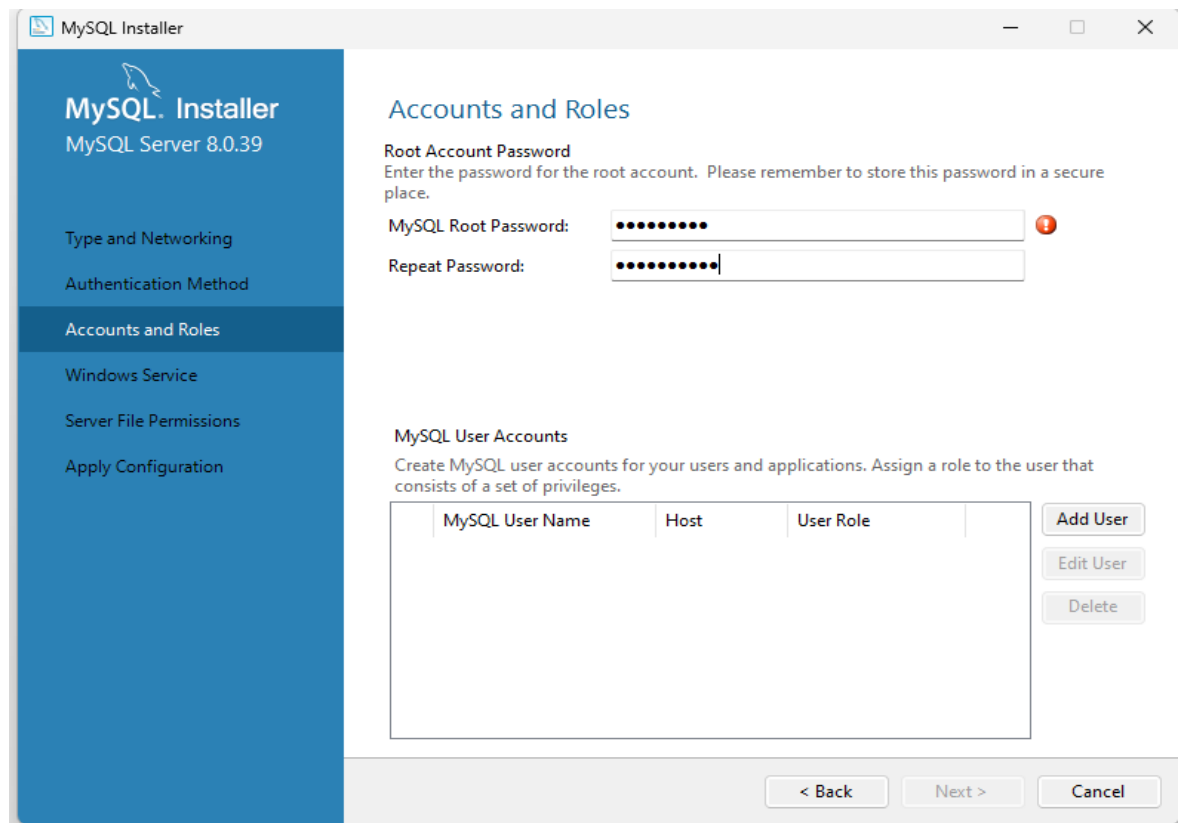
- Choose **"Standalone MySQL Server"** and click **"Next"**. Set the configuration type to **"Development Computer"** and ensure **"TCP/IP"** is selected. Click **"Next"**.



12. Set Root Password:

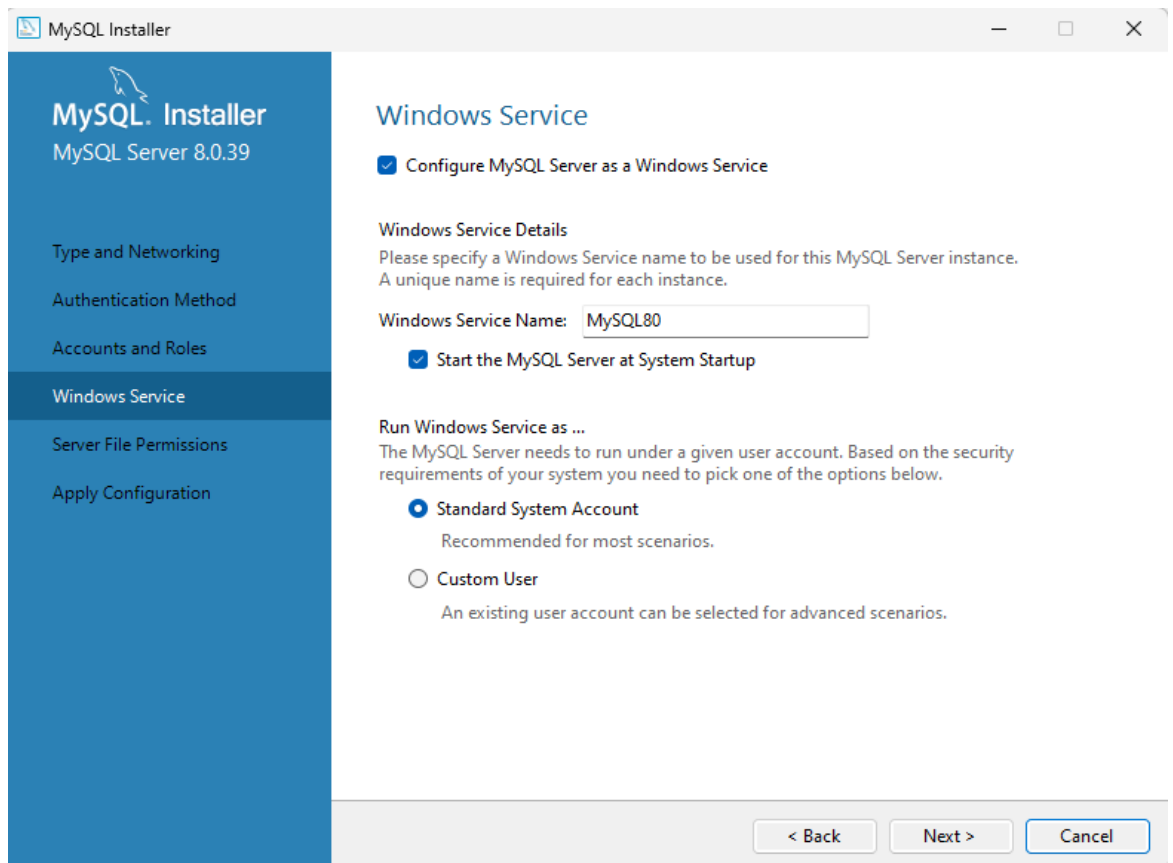
- On the **"Accounts and Roles"** screen, set a password for the root account. Remember this password for future access. Click **"Next"**.





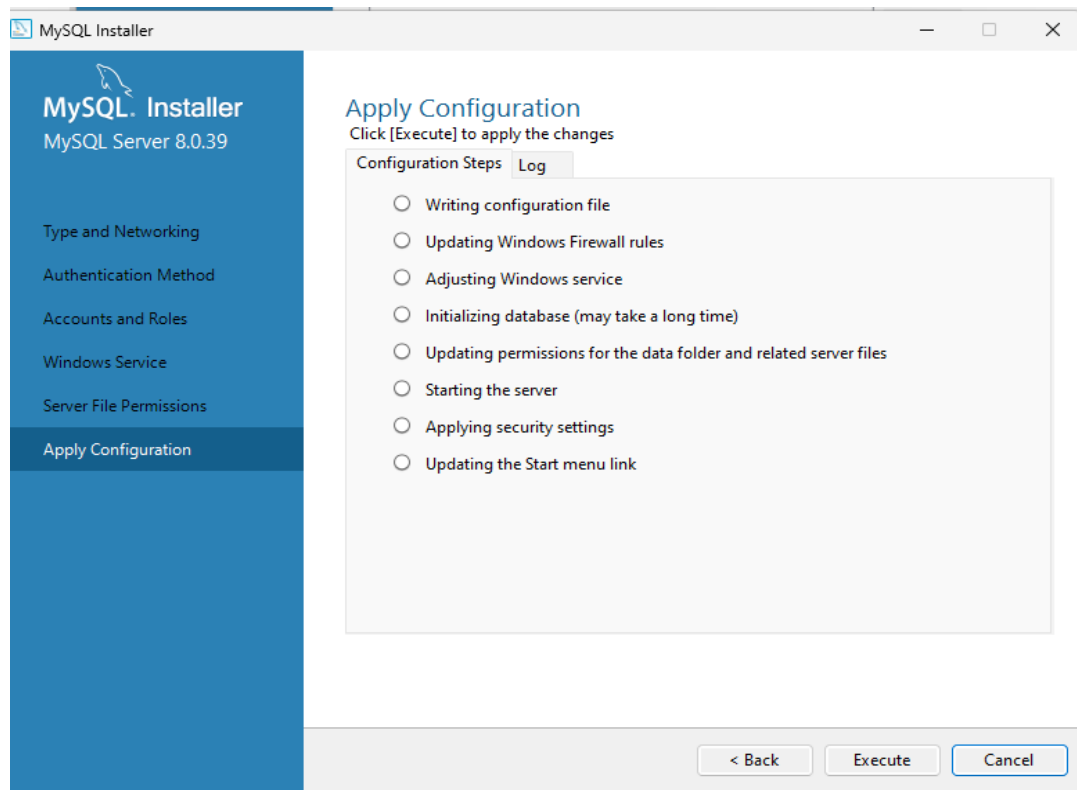
13. Windows Service Configuration:

- Keep the default settings for the Windows service and click "Next".



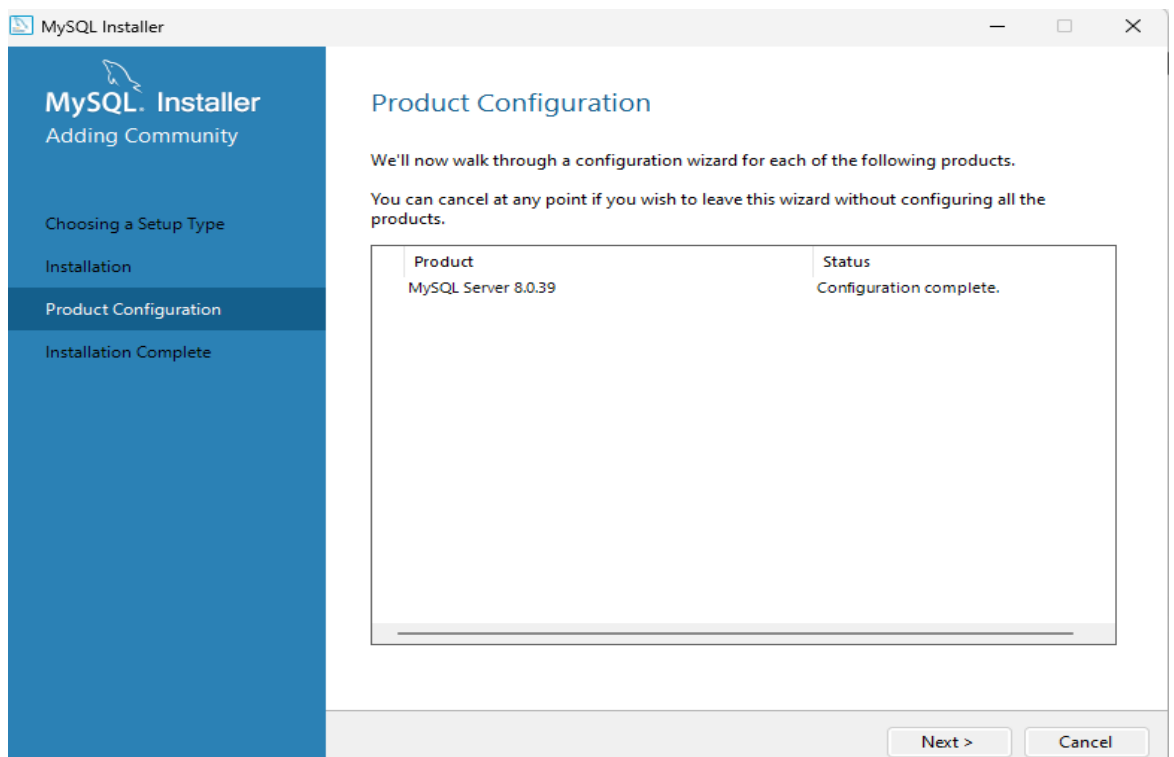
14. Apply Configuration:

- Click **"Execute"** to apply the server configuration. After it completes, click **"Finish"**.



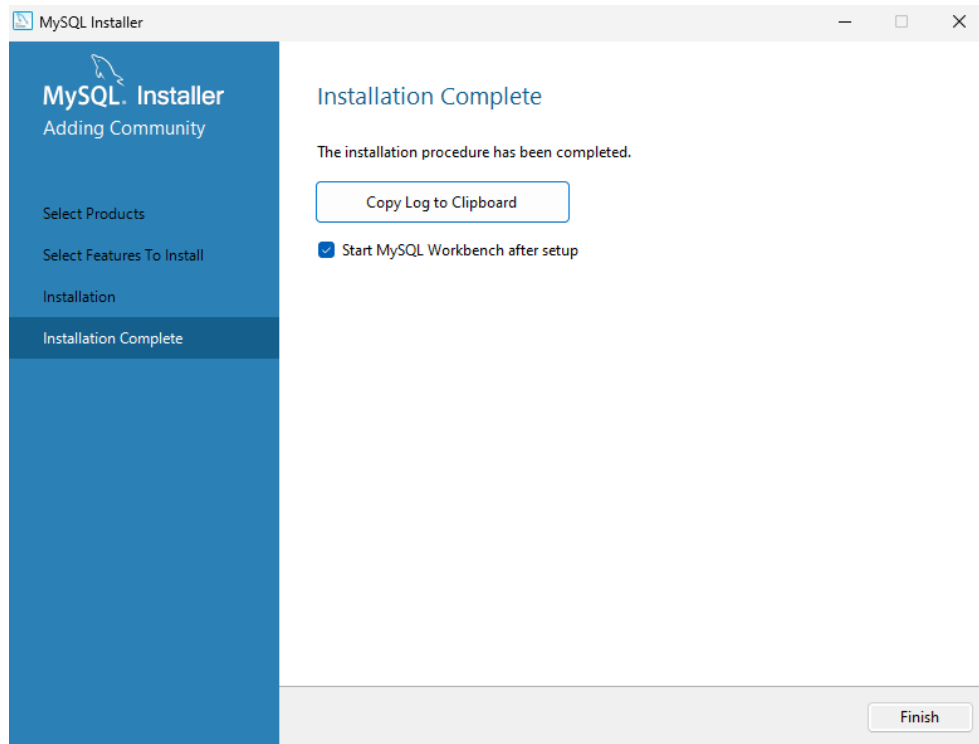
15. Complete Installation:

- Click **"Next"** and then **"Finish"** to complete the installation process.



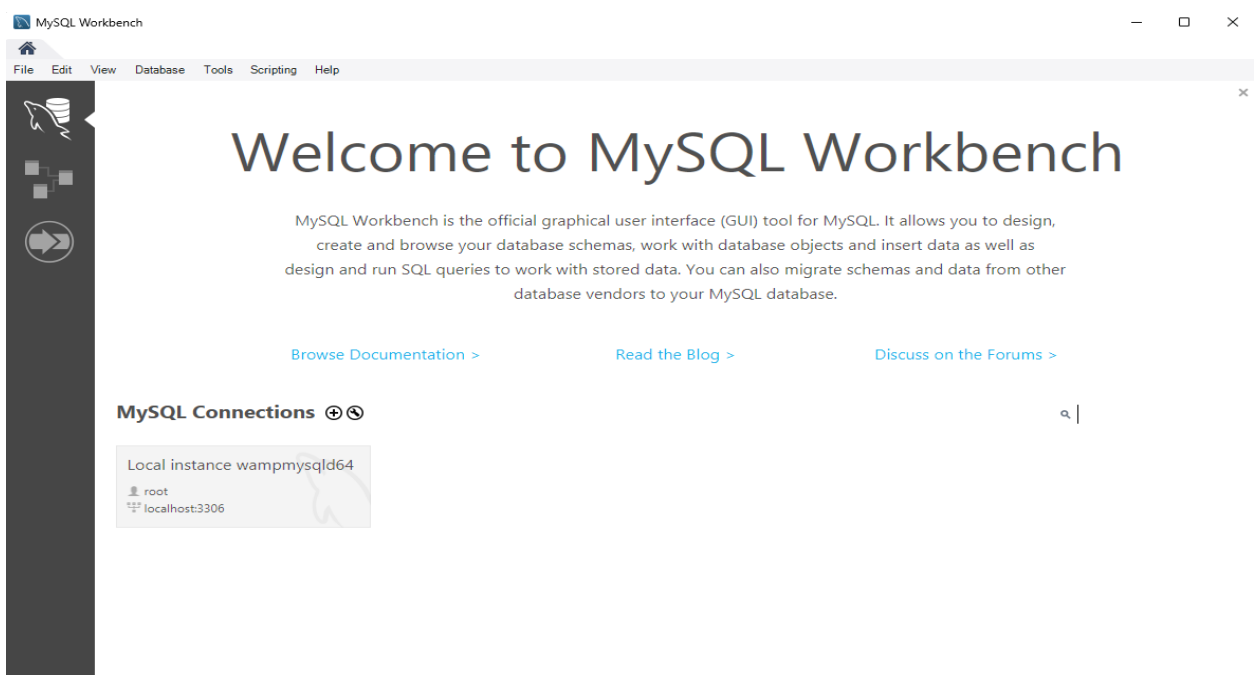
16. Launch MySQL Workbench:

- You can choose to launch MySQL Workbench immediately or do it later from the Start menu.



17. Connect to MySQL Server:

- Open MySQL Workbench, select "**Local instance**", and enter the root password to connect.



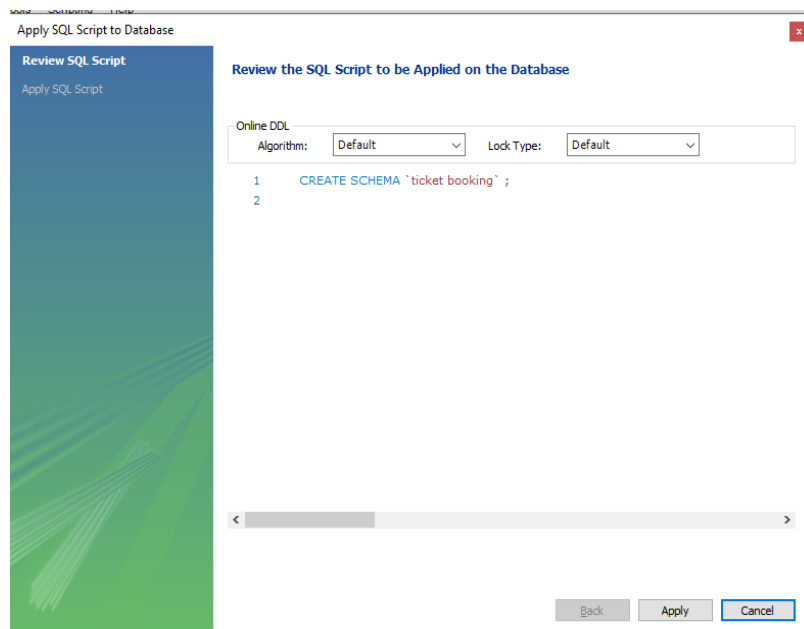
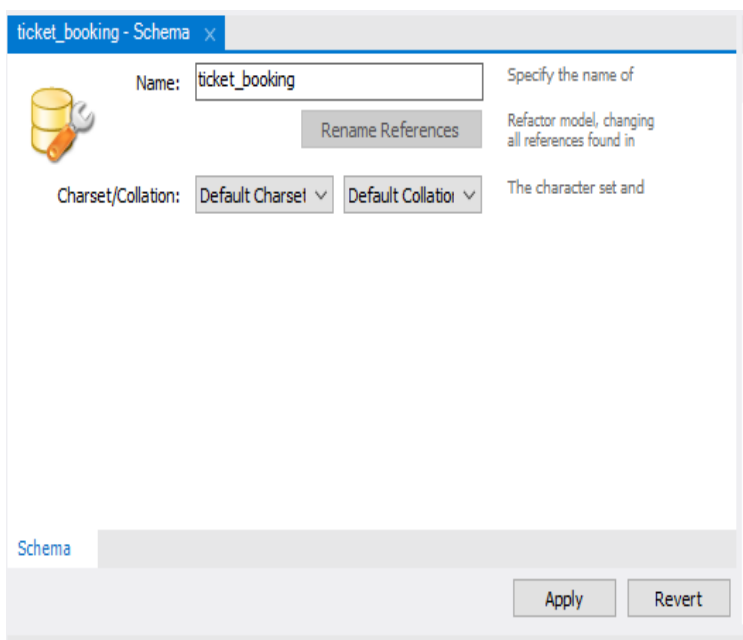
Steps to Create a Database

Create a New Database (Schema)

You can create a new database using either the graphical interface or SQL commands.

Using the Graphical Interface

1. **Click on the "Create Schema" Icon:**
 - In the toolbar, click on the **Create Schema** icon (it looks like a database with a plus sign) or go to **Database > Create Schema** from the menu.
2. **Enter Schema Name:**
 - In the dialog box that appears, enter `ticket_booking` as the schema name.
3. **Set Collation (Optional):**
 - You can choose a collation from the dropdown if needed, but the default is usually sufficient.
4. **Apply Changes:**
 - Click the **Apply** button. You will see a preview of the SQL statement that will be executed.
 - Click **Apply** again to execute the statement.
5. **Close the Dialog:**
 - After the database is created, click **Finish** to close the dialog.



Steps to Create Tables in Database

Select Your Database

- In the left pane, locate the **SCHEMAS** section.
- If you haven't created a database yet, you can create one by right-clicking in the **SCHEMAS** section and selecting **Create Schema**. Name it (e.g., ticket_booking).
- Once your database is ready, right-click on the database and select **Set as Default Schema**.

Create the Movies Table

1. **Right-click on the Tables section** under your selected database and choose **Create Table**.
2. In the table editor, enter the following details:
 - **Table Name:** Movies
 - **Columns:**
 - MovieID (INT, Not Null)
 - Title (VARCHAR(45), Primary Key, Not Null)
 - Genre (VARCHAR(100) ,Not Null)
 - Director (VARCHAR(100) ,Not Null)
 - Duration (INT) - Duration in minutes
 - Rating (DECIMAL(2, 1) ,Not Null)
 - Price (INT)- Price of the movie ticket
3. Click **Apply** to generate the SQL query and then click **Apply** again to create the table.

Table Name: movie Schema: ticket_booking Engine: InnoDB

Charset/Collation: Default Charset Default Collation

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
MOVIE_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TITLE	VARCHAR(45)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
GENRE	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
DIRECTOR	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
DURATION	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: PRICE Data Type: INT

Charset/Collation: Default Charset Default Collation

Comments:

Storage: ☐ Virtual ☐ Stored
☐ Primary Key ☐ Not Null ☐ Unique
☐ Binary ☐ Unsigned ☐ Zero Fill
☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 CREATE TABLE `ticket_booking`.`movie` (  
2   `movie_id` INT NOT NULL,  
3   `Title` VARCHAR(45) NOT NULL,  
4   `Genre` VARCHAR(45) NOT NULL,  
5   `Director` VARCHAR(45) NOT NULL,  
6   `Rating` INT NOT NULL,  
7   `Price` INT NOT NULL,  
8   PRIMARY KEY (`Title`));  
9
```

Create the Theaters Table

1. Right-click on the Tables section and select Create Table.
2. Enter the following details:
 - Table Name: Theaters
 - Columns:
 - TheaterID (INT)
 - Name (VARCHAR(45) , Primary Key, Not Null)
 - Location (VARCHAR(45) , Not Null)
 - Capacity (INT, Not Null)
3. Click Apply to generate and execute the SQL query.

theater - Table

Table Name: theater Schema: ticket_booking

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
THEATER_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TH_NAME	VARCHAR(45)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
LOCATION	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
CAPACITY	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type: Charset/Collation: Default: Comments: Storage: ☐ Virtual ☐ Stored ☐ Primary Key ☐ Not Null ☐ Unique ☐ Binary ☐ Unsigned ☐ Zero Fill ☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Apply SQL Script to Database



Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 CREATE TABLE `ticket_booking`.`theater` (  
2   `THEATER_ID` INT NOT NULL,  
3   `NAME` VARCHAR(45) NOT NULL,  
4   `LOCATION` VARCHAR(45) NOT NULL,  
5   `CAPACITY` INT NOT NULL,  
6   PRIMARY KEY (`THEATER_ID`));  
7
```

< >

Back Apply Cancel

Create the Showtimes Table

1. Right-click on the Tables section and select Create Table.
2. Enter the following details:
 - Table Name: Showtimes
 - Columns:
 - ShowtimeID (INT, Primary Key, NOT NULL)
 - Title (VARCHAR(45), NOT NULL)
 - TH_NAME(VARCHAR(45) , NOT NULL)
 - ShowDateTime (TIMESTAMP, NOT NULL)
3. Click Apply to generate and execute the SQL query.

Table Name: Schema: **ticket_booking**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
SHOWTIME_ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TITLE	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TH_NAME	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
SHOW_DATETIME	TIMESTAMP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation: Default:

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```
1 CREATE TABLE `ticket_booking`.`showtime` (  
2   `SHOWTIME_ID` INT NOT NULL,  
3   `TITLE` VARCHAR(45) NOT NULL,  
4   `TH_NAME` VARCHAR(45) NOT NULL,  
5   `SHOW_DATETIME` TIMESTAMP NOT NULL,  
6   PRIMARY KEY (`SHOWTIME_ID`));  
7
```


Create the Customers Table

1. Right-click on the Tables section and select Create Table.
2. Enter the following details:
 - Table Name: Customers
 - Columns:
 - CustomerID (INT, Primary Key)
 - Name (VARCHAR(255), Not Null)
 - Email (VARCHAR(255), Not Null)
 - Phone (INT(10))
3. Click Apply to generate and execute the SQL query.

Table Name: Schema: **ticket_booking**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CUSTOMER_ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
C_NAME	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
EMAIL	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
PHONE_NO	INT(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation:

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☒ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```
1 CREATE TABLE `ticket_booking`.`customer` (  
2   `CUSTOMER_ID` INT NOT NULL,  
3   `C_NAME` VARCHAR(45) NOT NULL,  
4   `EMAIL` VARCHAR(45) NULL,  
5   `PHONE_NO` INT NULL,  
6   PRIMARY KEY (`CUSTOMER_ID`));  
7
```

Back Apply Cancel

Create the Bookings Table

1. **Right-click on the Tables section** and select **Create Table**.
2. Enter the following details:
 - **Table Name:** Bookings
 - **Columns:**
 - BookingID (INT, Primary Key, Auto Increment)
 - CustomerID (INT, Foreign Key referencing Customers(CustomerID))
 - Showtime_ID (INT, Foreign Key referencing Showtimes>ShowtimeID))
 - Seats_Booked (INT)
3. Click **Apply** to generate and execute the SQL query.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
BOOKING_ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
CUSTOMER_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
SHOWTIME_ID	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
SEATS_BOOKED	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

```
1 CREATE TABLE `ticket_booking`.`booking` (  
2   `BOOKING_ID` INT NOT NULL,  
3   `CUSTOMER_ID` INT NOT NULL,  
4   `SHOWTIME_ID` INT NOT NULL,  
5   `SEATS_BOOKED` INT NOT NULL,  
6   PRIMARY KEY (`BOOKING_ID`));  
7
```

Preparing Report

Movie Table

MOVIE ID	TITLE	GENRN	DIRECTOR	DURATION (Min)	RATING (out of 10)	PRICE (IND)
001	Sooryavanshi	Action	Rohit Shetty	145	6.2	150
002	Pagal Panti	Comedy	Anees Bazmee	149	3.2	120
003	Your Name	Drama	Makoto Shinkai	106	8.4	200
004	Dark Knight	Action	Christopher Nolan	152	9.0	250
005	A Silent Voice	Drama	Naoko Yamada	130	8.2	200
006	Weathering With You	Fantasy	Makoto Shinkai	112	7.5	200
007	Bhool Bhulaiyaa	Comedy	Anees Bazmee	159	7.4	150
008	Inception	Action	Christopher Nolan	148	8.8	250
009	Cirkus	Comedy	Rohit Shetty	138	3.5	150
010	Liz And The Blue Bird	Music	Naoko Yamada	90	7.2	200

Theater Table

Theater ID	Name	Location	Capacity
10001	City Mall	Kota	700
10002	Ambience Mall	Delhi	400
10003	World Trade Park	Jaipur	500
10004	Select Citywalk	Delhi	800
10005	INOX	Kota	600

Showtime Table

Showtime ID	TiTle	Theater ID	ShowDateTime
6201	Sooryavanshi	INOX	2024-07-01 19:00:00
6202	Pagal Panti	THE CIYT MALL	2024-07-25 20:00:00
6203	Your Name	WORLD TRADE PARK	2024-08-01 16:00:00
6204	Dark Knight	THE CIYT MALL	2024-08-27 23:00:00
6205	A Silent Voice	SELECT CITYWALK	2024-07- 24 15:00:00
6206	Weathering With You	AMBIENCE MALL	2024-09-10 23:30:00
6207	Bhool Bhulaiyaa	INOX	2024 -09-30 10:00:00
6208	Inception	SELECT CITYWALK	2024-08-19 10:35:00
6209	Cirkus	WORLD TRADE PARK	2024-09-08 20:15:00
6210	Liz And The Blue Bird	INOX	2024-10-01 19:25:00

Booking Table

Booking ID	Customer ID	ShowTime ID	Seats Booked
2002	101	6201	5
2003	102	6202	6
2004	103	6203	2
2005	104	6204	9
2006	105	6205	3
2007	106	6206	1
2008	107	6207	9
2009	108	6208	2
2010	109	6209	4
2011	110	6210	8

Customer Table

Customer ID	C_Name	Email	Phone No.
101	Esha Mehta	EshaMehta@gmail.com	1234567891
102	Karan Patel	KaranPatel@gmail.com	1593574561
103	Emily White	EmilyWhite@gmail.com	9874562492
104	Jessiaca Jones	JessiacaJones@gmail.com	2326548163
105	Saniya Sharma	SaniyaSharma@gmail.com	2582583949
106	Vikram Joshi	VikramJoshi@gmail.com	2295566488
107	Jane Smith	JaneSmith@gmail.com	7894758964
108	Neha Kapoor	NehaKapoor@gmail.com	6635977458
109	Ananya Joshi	AnanyaJoshi@gmail.com	2335943269
110	James wilson	Jameswilson@gmail.com	4582692347

Steps to record data in the table

Insert Data into the Movies Table


1. Right-click on the Movies table in the left pane and select Select Rows - Limit 1000.
2. In the results grid that appears, you will see an empty row at the bottom.
3. Enter the data for the columns:
 - MovieID: (Auto-generated if set as Auto Increment, leave blank)
 - Title: e.g., Inception
 - Genre: e.g., Sci-Fi
 - Director: e.g., Christopher Nolan
 - Duration: e.g., 148
 - Rating: e.g., 8.8
 - Price: e.g., 10.00
4. After entering the data, click the Apply button.
5. Review the generated SQL statement and click Apply again to execute it.

	MOVIE_ID	TITLE	GENRE	DIRECTOR	DURATION	RATING	PRICE
▶	1	Sooryavanshi	Action	Rohit Shetty	145	6.2	150
	2	Pagal Panti	Comedy	Anees Bazmee	149	3.2	120
	3	Your Name	Drama	Makoto Shinkai	106	8.4	200
	4	Dark Knight	Action	Christopher Nolan	152	9.0	250
	5	A Silent Voice	Drama	Naoko Yamada	130	8.2	200
	6	Weathering With You	Fantasy	Makoto Shinkai	112	7.5	200
	7	Bhool Bhulaiyaa	Comedy	Anees Bazmee	159	7.4	150
	8	Inception	Action	Christopher Nolan	148	8.8	250
	9	Cirkus	Comedy	Rohit Shetty	138	3.5	150
	10	Liz And The Blue Bird	Music	Naoko Yamada	90	7.2	200
	NULL	NULL	NULL	NULL	NULL	NULL	NULL



Insert Data into the Theaters Table

1. **Right-click on the Theaters table** and select **Select Rows - Limit 1000**.
2. In the results grid, enter the data:
 - **TheaterID:** (Auto-generated if set as Auto Increment, leave blank)
 - **Name:** e.g., Cineplex
 - **Location:** e.g., Downtown
 - **Capacity:** e.g., 200
3. Click **Apply** and then **Apply** again to execute.

Result Grid				
Filter Rows: <input type="text"/>				
Edit: 				
	THEATER_ID	TH_NAME	LOCATION	CAPACITY
▶	10003	AMBIENCE MALL	DELHI	600
	10004	INOX	KOTA	500
	10005	SELECT CITYWALK	DELHI	800
	10001	THE CIYT MALL	KOTA	600
	10002	WORLD TRADE PARK	JAIPUR	700
✱	NULL	NULL	NULL	NULL

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

```
1  INSERT INTO `ticket_booking`.`theater` (`THEATER_ID`, `TH_NAME`, `LOCATION`
2  INSERT INTO `ticket_booking`.`theater` (`THEATER_ID`, `TH_NAME`, `LOCATION`
3  INSERT INTO `ticket_booking`.`theater` (`THEATER_ID`, `TH_NAME`, `LOCATION`
4  INSERT INTO `ticket_booking`.`theater` (`THEATER_ID`, `TH_NAME`, `LOCATION`
5  INSERT INTO `ticket_booking`.`theater` (`THEATER_ID`, `TH_NAME`, `LOCATION`
6  |
```

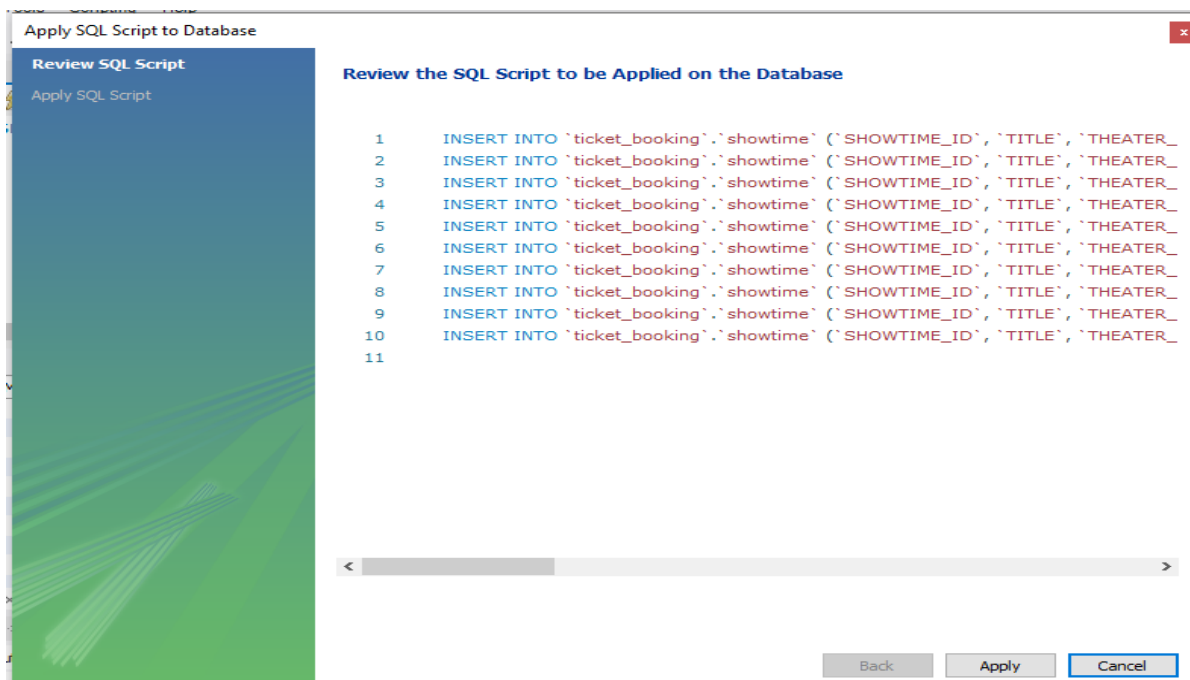
< >

BackApplyCancel

Insert Data into the Showtimes Table

1. **Right-click on the Showtimes table** and select **Select Rows - Limit 1000**.
2. Enter the data:
 - **ShowtimeID:** (Auto-generated if set as Auto Increment, leave blank)
 - **MovieID:** e.g., 1 (the ID of the movie you entered)
 - **TheaterID:** e.g., 1 (the ID of the theater you entered)
 - **ShowDateTime:** e.g., 2024-08-10 19:00:00
3. Click **Apply** and then **Apply** again to execute.

	SHOWTIME_ID	TITLE	TH_NAME	SHOW_DATETIME
▶	6201	Sooryavanshi	INOX	2024-07-01 19:00:00
	6202	Pagal Panti	THE CIYT MALL	2024-07-25 20:00:00
	6203	Your Name	WORLD TRADE PARK	2024-08-01 16:00:00
	6204	Dark Knight	THE CIYT MALL	2024-08-27 23:00:00
	6205	A Silent Voice	SELECT CITYWALK	2024-07-24 15:00:00
	6206	Weathering With You	AMBIENCE MALL	2024-09-10 23:30:00
	6207	Bhool Bhulaiyaa	INOX	2024-09-30 10:00:00
	6208	Inception	SELECT CITYWALK	2024-08-19 10:35:00
	6209	Cirkus	WORLD TRADE PARK	2024-09-08 20:15:00
	6210	Liz And The Blue Bird	INOX	2024-10-01 19:25:00
	NULL	NULL	NULL	NULL



Insert Data into the Customers Table

1. **Right-click on the Customers table** and select **Select Rows - Limit 1000**.
2. Enter the data:
 - **CustomerID:** (Auto-generated if set as Auto Increment, leave blank)
 - **Name:** e.g., John Doe
 - **Email:** e.g., john@example.com
 - **Phone:** e.g., 1234567890
3. Click **Apply** and then **Apply** again to execute.

	CUSTOMER_ID	C_NAME	EMAIL	PHONE_NO
▶	101	Esha Mehta	EshaMehta@gmail.com	1234567891
	102	Karan Patel	KaranPatel@gmail.com	1593574561
	103	Emily White	EmilyWhite@gmail.com	9874562492
	104	Jessica Jones	JessicaJones@gmail.c...	2326548163
	105	Saniya Sharma	SaniyaSharma@gmail.c...	2582583949
	106	Vikram Joshi	VikramJoshi@gmail.com	2295566488
	107	Jane Smith	JaneSmith@gmail.com	7894758964
	108	Neha Kapoor	NehaKapoor@gmail.com	6635977458
	109	Ananya Joshi	AnanyaJoshi@gmail.com	2335943269
	110	James wilson	Jameswilson@gmail.com	4582692347



Insert Data into the Bookings Table

2. **Right-click on the Bookings table** and select **Select Rows - Limit 1000**.

3. Enter the data:

- **BookingID:** (Auto-generated if set as Auto Increment, leave blank)
- **CustomerID:** e.g., 1 (the ID of the customer you entered)
- **ShowtimeID:** e.g., 1 (the ID of the showtime you entered)
- **SeatsBooked:** e.g., 2

4. Click **Apply** and then **Apply** again to execute.

	BOOKING_ID	CUSTOMER_ID	SHOWTIME_ID	SEATS_BOOKED
▶	1001	101	6201	5
	1002	102	6202	6
	1003	103	6203	2
	1004	104	6204	9
	1005	105	6205	3
	1006	106	6206	1
	1007	107	6207	9
	1008	108	6208	2
	1009	109	6209	4
	1010	110	6210	8
✱	NULL	NULL	NULL	NULL

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

```
1  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2002', '101', '6201', '4');
2  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2003', '102', '6202', '6');
3  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2004', '103', '6203', '2');
4  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2005', '104', '6204', '9');
5  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2006', '105', '6205', '3');
6  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2007', '106', '6206', '1');
7  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2008', '107', '6207', '9');
8  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2009', '108', '6208', '7');
9  ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2010', '109', '6209', '4');
10 ER_ID`, `SHOWTIME_ID`, `SEATS_BOOKED`) VALUES ('2011', '110', '6210', '8');
11
```

< >

Back

Apply

Cancel

Foreign Key

A **foreign key** is a field (or a collection of fields) in one table that uniquely identifies a row of another table or the same table. It establishes a relationship between the two tables, enforcing referential integrity in the database. Key points about foreign keys:

- **Relationship:** A foreign key in the child table points to a primary key in the parent table, creating a link between the two tables.
- **Referential Integrity:** Foreign keys ensure that relationships between tables remain consistent. For instance, a foreign key value must match a primary key value in the referenced table or be null.

Steps to Insert Foreign Key

1. **Open MySQL Workbench** and connect to your MySQL server.
2. **Select the database** where you want to create the foreign key.
3. **Right-click on the table** where you want to add the foreign key constraint and select "Alter Table".
4. **In the Table Editor**, go to the "Foreign Keys" tab.
5. **Click on the "+" icon** to add a new foreign key.
6. **In the Foreign Key panel:**
 - Enter a name for the foreign key constraint.
 - Select the column(s) that will be the foreign key.
 - Choose the referenced table.
 - Select the referenced column(s) in the referenced table.
7. **Optionally, you can specify additional options** for the foreign key, such as:
 - **ON DELETE:** Specify the action to take when a referenced row in the parent table is deleted (e.g., CASCADE, SET NULL, NO ACTION).

- **ON UPDATE:** Specify the action to take when a referenced row in the parent table is updated (e.g., CASCADE, SET NULL, NO ACTION).
8. Click **"Apply"** to create the foreign key constraint.
 9. If prompted, review the generated SQL statement and click **"Apply"** again to execute it.

- **Add foreign key in table Showtime**
 - **Title (Movie Table)**

showtime - Table

Table Name: Schema: **ticket_booking**

Charset/Collation: Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
movie	'ticket_booking', 'movie'	<input checked="" type="checkbox"/> TITLE	TITLE
TH_NAME	'ticket_booking', 'theater'	<input type="checkbox"/> TH_NAME	
		<input type="checkbox"/> SHOW_DATETIME	

Foreign Key Options

On Update:

On Delete:

☐ Skip in SQL generation

Foreign Key Comment:

Columns Indexes **Foreign Keys** Triggers Partitioning Options

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```

1  ALTER TABLE `ticket_booking`.`showtime`
2  ADD INDEX `movie_idx` (`TITLE` ASC) VISIBLE;
3  ;
4  ALTER TABLE `ticket_booking`.`showtime`
5  ADD CONSTRAINT `movie`
6  FOREIGN KEY (`TITLE`)
7  REFERENCES `ticket_booking`.`movie` (`TITLE`)
8  ON DELETE NO ACTION
9  ON UPDATE NO ACTION;
10

```

Back Apply Cancel

○ Theater ID (Theater Table)

showtime - Table

Table Name: Schema: **ticket_booking**

Charset/Collation: Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
movie	'ticket_booking'. 'movie'	<input type="checkbox"/> SHOWTIME_ID	
TH_NAME	'ticket_booking'. 'theater'	<input type="checkbox"/> TITLE	
		<input checked="" type="checkbox"/> TH_NAME	TH_NAME
		<input type="checkbox"/> SHOW_DATETIME	

Foreign Key Options

On Update:

On Delete:

☐ Skip in SQL generation

Foreign Key Comment:

Columns Indexes **Foreign Keys** Triggers Partitioning Options

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```
1 ALTER TABLE `ticket_booking`.`showtime`
2 ADD INDEX `TH_NAME_idx` (`TH_NAME` ASC) VISIBLE;
3 ;
4 ALTER TABLE `ticket_booking`.`showtime`
5 ADD CONSTRAINT `TH_NAME`
6 FOREIGN KEY (`TH_NAME`)
7 REFERENCES `ticket_booking`.`theater` (`TH_NAME`)
8 ON DELETE NO ACTION
9 ON UPDATE NO ACTION;
10
```

Back Apply Cancel

- Add foreign key in table Showtime
 - Customer ID (Customer Table)

booking - Table

Table Name: Schema: **ticket_booking**

Charset/Collation: Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
FK_CUSTOMER_ID	ticket_booking`.`customer`	<input checked="" type="checkbox"/> BOOKING_ID	
FK_SHOWTIME_ID	ticket_booking`.`showtime`	<input checked="" type="checkbox"/> CUSTOMER_ID	CUSTOMER_ID
		<input type="checkbox"/> SHOWTIME_ID	
		<input type="checkbox"/> SEATS_BOOKED	

Foreign Key Options

On Update:

On Delete:

☐ Skip in SQL generation

Foreign Key Comment:

Columns Indexes **Foreign Keys** Triggers Partitioning Options

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```

1  ALTER TABLE `ticket_booking`.`booking`
2  ADD INDEX `FK_CUSTOMER_ID_idx` (`CUSTOMER_ID` ASC) VISIBLE;
3  ;
4  ALTER TABLE `ticket_booking`.`booking`
5  ADD CONSTRAINT `FK_CUSTOMER_ID`
6  FOREIGN KEY (`CUSTOMER_ID`)
7  REFERENCES `ticket_booking`.`customer` (`CUSTOMER_ID`)
8  ON DELETE NO ACTION
9  ON UPDATE NO ACTION;
10

```

Back Apply Cancel

○ ShowTime ID (ShowTime Table)

booking - Table

Table Name: Schema: **ticket_booking**

Charset/Collation: Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
FK_CUSTOMER_ID	'ticket_booking'. 'customer'	<input type="checkbox"/> BOOKING_ID	
FK_SHOWTIME_ID	'ticket_booking'. 'showtime'	<input type="checkbox"/> CUSTOMER_ID	
		<input checked="" type="checkbox"/> SHOWTIME_ID	SHOWTIME_ID
		<input type="checkbox"/> SEATS_BOOKED	

Foreign Key Options

On Update:

On Delete:

☐ Skip in SQL generation

Foreign Key Comment:

Columns Indexes **Foreign Keys** Triggers Partitioning Options

Apply Revert

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Lock Type:

```

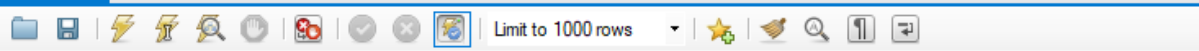
1  ALTER TABLE `ticket_booking`.`booking`
2  DROP FOREIGN KEY `FK_SHOWTIME_ID`;
3  ALTER TABLE `ticket_booking`.`booking`
4  ;
5  ALTER TABLE `ticket_booking`.`booking` ALTER INDEX `FK_SHOWTIME_ID_idx` VI
6  ALTER TABLE `ticket_booking`.`booking`
7  ADD CONSTRAINT `FK_SHOWTIME_ID`
8  FOREIGN KEY (`SHOWTIME_ID`)
9  REFERENCES `ticket_booking`.`showtime` (`SHOWTIME_ID`)
10 ON DELETE NO ACTION
11 ON UPDATE NO ACTION;
12

```

Back Apply Cancel

Query

1. Retrieve all movies:



Limit to 1000 rows


```
1 • select TITLE from ticket_booking.movie;
```

```
2
```

Result Grid

TITLE
Your Name
Weathering With You
Sooryavanshi
Pagal Panti
Liz And The Blue Bird
Inception
Dark Knight
Cirkus
Bhool Bhulaiyaa
A Silent Voice
NULL

2. Retrieve all customers:



Query 1

Limit to 1000 rows

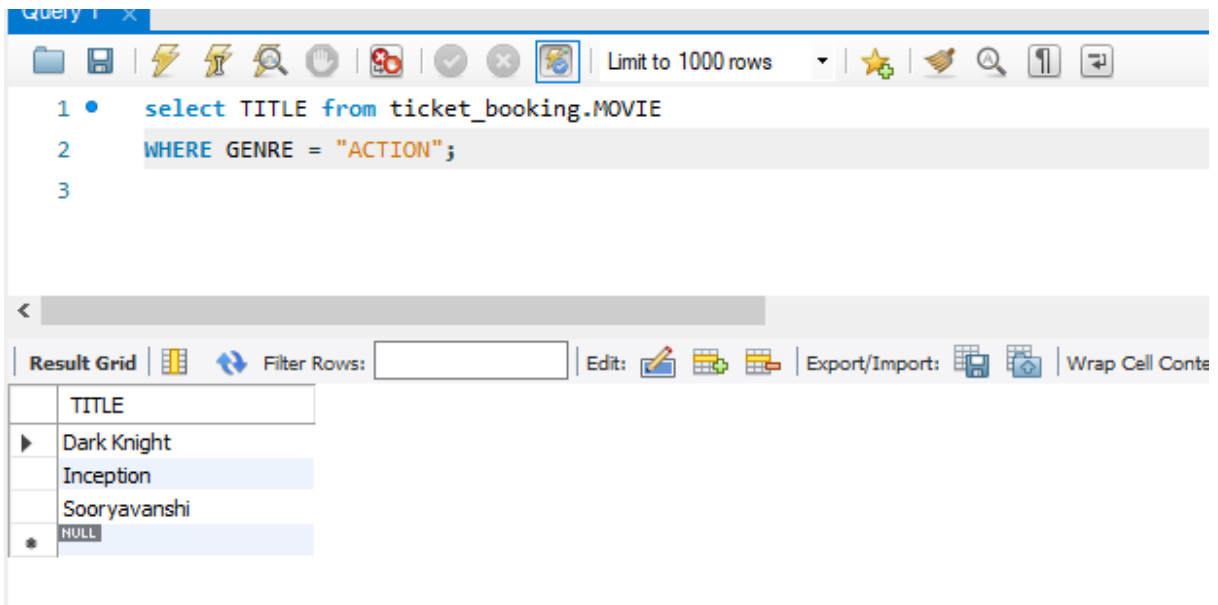
```
1 • select C_NAME from ticket_booking.CUSTOMER;
```

```
2
```

Result Grid

C_NAME
Esha Mehta
Karan Patel
Emily White
Jessiaca Jones
Saniya Sharma
Vikram Joshi
Jane Smith
Neha Kapoor
Ananya Joshi
James wilson

3. Retrieve all movies of a specific genre (ACTION):



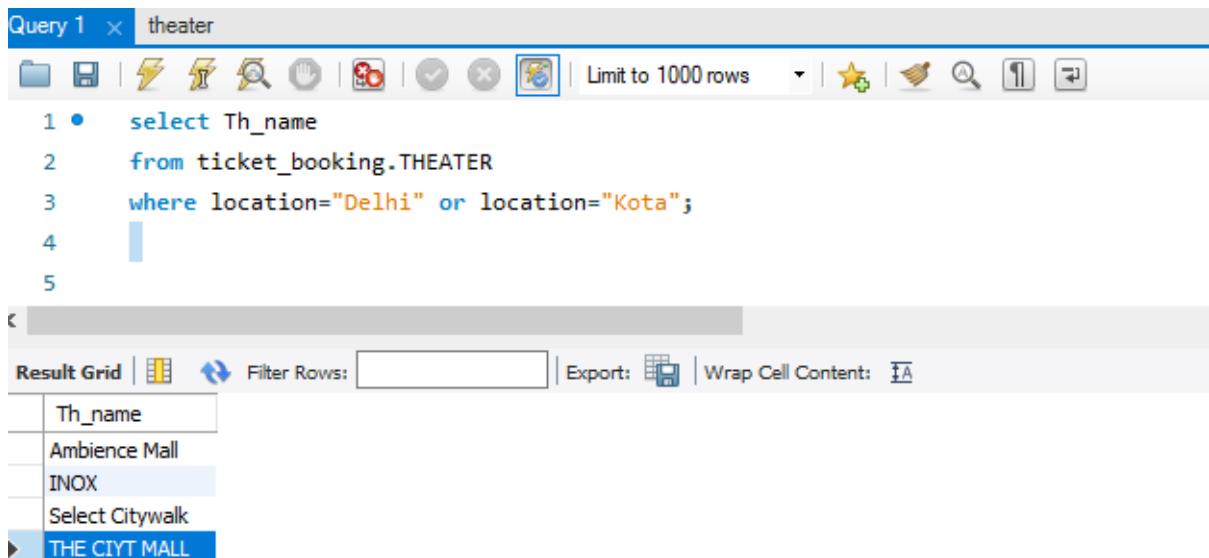
The screenshot shows a SQL query editor window titled "Query 1". The query is as follows:

```
1 • select TITLE from ticket_booking.MOVIE
2   WHERE GENRE = "ACTION";
3
```

Below the query editor, the "Result Grid" is displayed. It shows a table with one column, "TITLE", and four rows of data:

TITLE
Dark Knight
Inception
Sooryavanshi
NULL

4. Retrieve all theaters in a specific location (DELHI,Kota):



The screenshot shows a SQL query editor window titled "Query 1" with the tab name "theater". The query is as follows:

```
1 • select Th_name
2   from ticket_booking.THEATER
3   where location="Delhi" or location="Kota";
4
5
```

Below the query editor, the "Result Grid" is displayed. It shows a table with one column, "Th_name", and five rows of data:

Th_name
Ambience Mall
INOX
Select Citywalk
THE CIYT MALL

5. Find the total capacity of all theaters:

Query 1 theater theater

Limit to 1000 rows

```
1 • select sum(capacity) as total_capacity
2   from ticket_booking.THEATER;
3
4
```

Result Grid

total_capacity
3000

6. Get the total number of seats available in each theater:

Query 1 showtime

Limit to 1000 rows

```
1 • SELECT theater.TH_NAME,theater.CAPACITY
2   FROM theater;
3
4
5
```

Result Grid

TH_NAME	CAPACITY
Ambience Mall	400
INOX	600
Select Citywalk	800
THE CIYT MALL	700
World Trade Park	500
NULL	NULL

7. Retrieve all record with director name Makoto Shinkai:

Query 1 x movie

```
1 • SELECT * from movie
2   where movie.DIRECTOR="Makoto Shinkai";
3
```

Result Grid

	MOVIE_ID	TITLE	GENRE	DIRECTOR	DURATION	RATING	PRICE
▶	6	Weathering With You	Fantasy	Makoto Shinkai	112	7.5	200
	3	Your Name	Drama	Makoto Shinkai	106	8.4	200
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

8. Retrieve the names and emails of customers who booked seats for a specific movie (e.g., 'Inception'):

Query 1 x booking

```
1 • SELECT c_name ,email
2   FROM ticket_booking.customer
3   inner join Booking ON Customer.Customer_ID = Booking.Customer_ID
4   inner JOIN Showtime ON Booking.Showtime_ID = Showtime.Showtime_ID
5   WHERE showtime.SHOWTIME_ID = 6208;
6
7
```

Result Grid

	c_name	email
▶	Jane Smith	JaneSmith@gmail.com
	Neha Kapoor	NehaKapoor@gmail.com

9. Retrieve all showtimes along with movie titles and theater names:

Query 1

```
1 • SELECT movie.TITLE ,theater.TH_NAME,showtime.SHOW_DATETIME
2 FROM showtime
3 inner join movie on showtime.TITLE = movie.TITLE
4 inner join theater on showtime.TH_NAME = theater.TH_NAME;
```

Result Grid

	TITLE	TH_NAME	SHOW_DATETIME
▶	Weathering With You	Ambience Mall	2024-09-10 23:30:00
	Sooryavanshi	INOX	2024-07-01 19:00:00
	Bhool Bhulaiyaa	INOX	2024-09-30 10:00:00
	Liz And The Blue Bird	INOX	2024-10-01 19:25:00
	A Silent Voice	Select Citywalk	2024-07-24 15:00:00
	Inception	Select Citywalk	2024-08-19 10:35:00
	Pagal Panti	THE CIYT MALL	2024-07-25 20:00:00
	Dark Knight	THE CIYT MALL	2024-08-27 23:00:00
	Your Name	World Trade Park	2024-08-01 16:00:00
	Cirkus	World Trade Park	2024-09-08 20:15:00

10. Find the average duration of movies by genre:

Query 1

```
1 • SELECT avg(duration) from movie;
2
3
4
5
```

Result Grid

	avg(duration)
▶	132.9000

11. Find movies with a duration longer than the average duration of all movies:

Query 1 x

Limit to 1000 rows

```
1 • SELECT Title, Duration
2 FROM Movie
3 WHERE Duration > (SELECT AVG(Duration) FROM Movie);
4
```

Result Grid

	Title	Duration
▶	Bhool Bhulaiyaa	159
	Cirkus	138
	Dark Knight	152
	Inception	148
	Pagal Pantu	149
	Sooryavanshi	145

12. List the names of all customers along with the movies they have booked

Query 1 x booking theater showtime showtime - Table

Limit to 1000 rows

```
1 • SELECT Customer.C_Name AS Customer_Name, Movie.Title AS Movie_Name
2 FROM Customer
3 INNER JOIN Booking ON Customer.Customer_ID = Booking.Customer_ID
4 INNER JOIN Showtime ON Booking.Showtime_ID = Showtime.Showtime_ID
5 INNER JOIN Movie ON Showtime.N_MOVIE = Movie.Title;
```

Result Grid

	Customer_Name	Movie_Name
▶	Esha Mehta	Sooryavanshi
	Karan Patel	Pagal Pantu
	Emily White	Your Name
	Jessica Jones	Dark Knight
	Saniya Sharma	A Silent Voice
	Vikram Joshi	Weathering With You
	Jane Smith	Inception
	Neha Kapoor	Inception
	Ananya Joshi	Cirkus
	James Wilson	Liz And The Blue Bird

13. List the names of customers who have booked more than 5 seats in total

Query 1 x booking theater showtime showtime - Table

Limit to 1000 rows

```
1 • SELECT Customer.C_Name AS Customer_Name, booking.seats_booked
2 FROM Customer
3 INNER JOIN Booking ON Customer.Customer_ID = Booking.Customer_ID
4 where booking.SEATS_BOOKED >= 5;
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Customer_Name	seats_booked
▶	Karan Patel	6
	Jessiaca Jones	9
	Jane Smith	9
	Neha Kapoor	7
	James Wilson	8

14. Find the total number of showtimes scheduled for each theater:

Query 1 x movie

Limit to 1000 rows

```
1 • SELECT Th_Name AS Theater_Name, COUNT(Showtime_ID) AS Total_Showtimes
2 FROM Showtime
3 GROUP BY Th_Name;
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Theater_Name	Total_Showtimes
▶	AMBIENCE MALL	1
	INOX	3
	SELECT CITYWALK	2
	THE CIYT MALL	2
	WORLD TRADE PARK	2

15. List all theaters that have hosted showings of movies rated above 8:

Query 1 x movie

Limit to 1000 rows

```
1 • SELECT DISTINCT theater.TH_Name, movie.RATING
2   FROM theater
3   inner join showtime ON theater.TH_NAME = showtime.TH_NAME
4   inner join movie ON showtime.TITLE = movie.TITLE
5   WHERE movie.RATING > 8 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	TH_Name	RATING
▶	Select Citywalk	8.2
	THE CIYT MALL	9.0
	Select Citywalk	8.8
	World Trade Park	8.4

16. Count the number of movies directed by each director:

Query 1 x movie

Limit to 1000 rows

```
1 • SELECT movie.Director, COUNT(*) AS Number_Of_Movies
2   FROM movie
3   GROUP BY Director;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Director	Number_Of_Movies
▶	Naoko Yamada	2
	Anees Bazmee	2
	Rohit Shetty	2
	Christopher Nolan	2
	Makoto Shinkai	2

17. Find the showtimes for a movie in a specific theater:

Query 1 x showtime

Limit to 1000 rows

```
1 • SELECT showtime.TITLE as movie,theater.TH_NAME as Theater_Name,showtime.Show_DateTime
2 FROM showtime
3 inner join theater ON showtime.TH_NAME = theater.TH_NAME
4 WHERE showtime.TITLE = 'A Silent Voice'
5 AND theater.TH_NAME = 'SELECT CITYWALK';
6
```

Result Grid

movie	Theater_Name	Show_DateTime
A Silent Voice	Select Citywalk	2024-07-24 15:00:00

18. List all movies that have not been booked yet:

Query 1 x showtime

Limit to 1000 rows

```
1 • SELECT movie.TITLE,movie.GENRE,movie.DIRECTOR,movie.DURATION,movie.PRICE,movie.RATING
2 FROM movie
3 LEFT JOIN showtime ON movie.TITLE= showtime.TITLE
4 LEFT JOIN booking ON showtime.SHOWTIME_ID = booking.SHOWTIME_ID
5 WHERE booking.BOOKING_ID IS NULL;
```

Result Grid

TITLE	GENRE	DIRECTOR	DURATION	PRICE	RATING
Bhool Bhulaiyaa	Comedy	Anees Bazmee	159	150	7.4

19. List all theaters with their showtimes sorted by capacity:

```
1 • SELECT theater.TH_NAME, theater.CAPACITY, showtime.TITLE, showtime.SHOW_DATETIME
2 FROM theater
3 inner JOIN showtime ON theater.TH_NAME = showtime.TH_NAME
4 ORDER BY theater.CAPACITY DESC;
5
```

TH_NAME	CAPACITY	TITLE	SHOW_DATETIME
Select Citywalk	800	A Silent Voice	2024-07-24 15:00:00
Select Citywalk	800	Inception	2024-08-19 10:35:00
THE CIYT MALL	700	Pagal Panti	2024-07-25 20:00:00
THE CIYT MALL	700	Dark Knight	2024-08-27 23:00:00
INOX	600	Sooryavanshi	2024-07-01 19:00:00
INOX	600	Bhool Bhulaiyaa	2024-09-30 10:00:00
INOX	600	Liz And The Blue Bird	2024-10-01 19:25:00
World Trade Park	500	Your Name	2024-08-01 16:00:00
World Trade Park	500	Cirkus	2024-09-08 20:15:00
Ambience Mall	400	Weathering With You	2024-09-10 23:30:00

20. List the top 3 customers who have booked the most seats overall:

```
1 • SELECT customer.C_NAME, SUM(booking.SEATS_BOOKED) AS TotalSeatsBooked
2 FROM customer
3 inner JOIN booking ON customer.CUSTOMER_ID = booking.CUSTOMER_ID
4 GROUP BY customer.C_NAME
5 ORDER BY TotalSeatsBooked DESC
6 LIMIT 3;
```

C_NAME	TotalSeatsBooked
Jane Smith	9
Jessiaca Jones	9
James wilson	8

User Interface

Report

Creating a report in Microsoft Access using external data from a text file involves several steps.

Here's a step-by-step guide:

Step 1: Import Data from a Text File

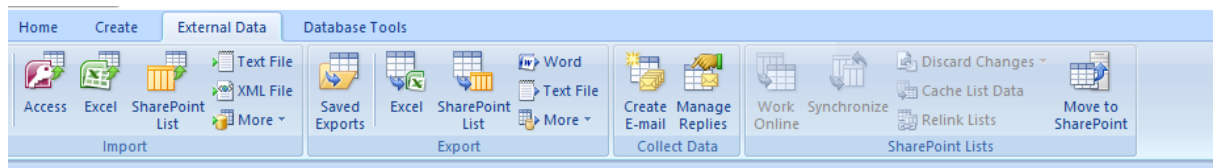
1. Open Microsoft Access:

- Launch MS Access and open the database where you want to create the report.



2. Import the Text File:

- Go to the **External Data** tab on the Ribbon.



- Click on **Text File** in the Import & Link group.
- Browse and select the text file you want to import.
-
- Choose the appropriate option (e.g., Import the source data into a new table, Append a copy, or Link to the data source by creating a linked table).

Get External Data - Text File

Select the source and destination of the data

Specify the source of the data.

File name: C:\Users\user\Desktop\MOVIE.csv Browse...

Specify how and where you want to store the data in the current database.

☒ **Import the source data into a new table in the current database.**
If the specified table does not exist, Access will create it. If the specified table already exists, Access might overwrite its contents with the imported data. Changes made to the source data will not be reflected in the database.

☐ **Append a copy of the records to the table:** Table1
If the specified table exists, Access will add the records to the table. If the table does not exist, Access will create it. Changes made to the source data will not be reflected in the database.

☐ **Link to the data source by creating a linked table.**
Access will create a table that will maintain a link to the source data. You cannot change or delete data that is linked to a text file. However, you can add new records.

OK Cancel

- Follow the Import Text Wizard, specifying the delimiter (comma, tab, etc.), and configure the field types if needed.

Import Text Wizard

Your data seems to be in a 'Delimited' format. If it isn't, choose the format that more correctly describes your data.

☒ Delimited - Characters such as comma or tab separate each field
☐ Fixed Width - Fields are aligned in columns with spaces between each field

Sample data from file: C:\USERS\USER\DESKTOP\MOVIE.CSV.

```

1 MOVIE_ID,TITLE,GENRE,DIRECTOR,DURATION,RATING,PRICE
2 5,"A Silent Voice",Drama,"Naoko Yamada",130,8.2,200
3 7,"Bhool Bhulaiyaa",Comedy,"Anees Bazmee",159,7.4,150
4 9,Cirkus,Comedy,"Rohit Shetty",138,3.5,150
5 4,"Dark Knight",Action,"Christopher Nolan",152,9.0,250
6 8,Inception,Action,"Christopher Nolan",148,8.8,250
7 10,"Liz And The Blue Bird",Music,"Naoko Yamada",90,7.2,200
8 2,"Pagal Panti",Comedy,"Anees Bazmee",149,3.2,120
9 1,Sooryavanshi,Action,"Rohit Shetty",145,6.2,150
10 6,"Weathering With You",Fantasy,"Makoto Shinkai",112,7.5,200
11 3,"Your Name",Drama,"Makoto Shinkai",106,8.4,200
  
```

Advanced... Cancel < Back Next > Finish

Import Text Wizard

What delimiter separates your fields? Select the appropriate delimiter and see how your text is affected in the preview below.

Choose the delimiter that separates your fields:

☐ Tab ☐ Semicolon ☒ Comma ☐ Space ☐ Other:

☒ First Row Contains Field Names Text Qualifier:

MOVIE_ID	TITLE	GENRE	DIRECTOR	DURATION	RATING	PRICE
5	A Silent Voice	Drama	Naoko Yamada	130	8.2	200
7	Bhool Bhulaiyaa	Comedy	Anees Bazmee	159	7.4	150
9	Cirkus	Comedy	Rohit Shetty	138	3.5	150
4	Dark Knight	Action	Christopher Nolan	152	9.0	250
8	Inception	Action	Christopher Nolan	148	8.8	250
10	Liz And The Blue Bird	Music	Naoko Yamada	90	7.2	200
2	Pagal Panti	Comedy	Anees Bazmee	149	3.2	120
1	Sooryavanshi	Action	Rohit Shetty	145	6.2	150
6	Weathering With You	Fantasy	Makoto Shinkai	112	7.5	200
3	Your Name	Drama	Makoto Shinkai	106	8.4	200

Advanced... Cancel < Back Next > Finish

Import Text Wizard

Microsoft Access recommends that you define a primary key for your new table. A primary key is used to uniquely identify each record in your table. It allows you to retrieve data more quickly.

☐ Let Access add primary key.
☒ Choose my own primary key.
☐ No primary key.


MOVIE ID	TITLE	GENRE	DIRECTOR	DURATION	RATING	PRICE
5	A Silent Voice	Drama	Naoko Yamada	130	8.2	200
7	Bhool Bhulaiyaa	Comedy	Anees Bazmee	159	7.4	150
9	Cirkus	Comedy	Rohit Shetty	138	3.5	150
4	Dark Knight	Action	Christopher Nolan	152	9.0	250
8	Inception	Action	Christopher Nolan	148	8.8	250
10	Liz And The Blue Bird	Music	Naoko Yamada	90	7.2	200
2	Pagal Panti	Comedy	Anees Bazmee	149	3.2	120
1	Sooryavanshi	Action	Rohit Shetty	145	6.2	150
6	Weathering With You	Fantasy	Makoto Shinkai	112	7.5	200
3	Your Name	Drama	Makoto Shinkai	106	8.4	200

Advanced... Cancel < Back Next > Finish

- o Finish the import process, and the data will appear in a new table in your database.

Import Text Wizard

That's all the information the wizard needs to import your data.



Import to Table:

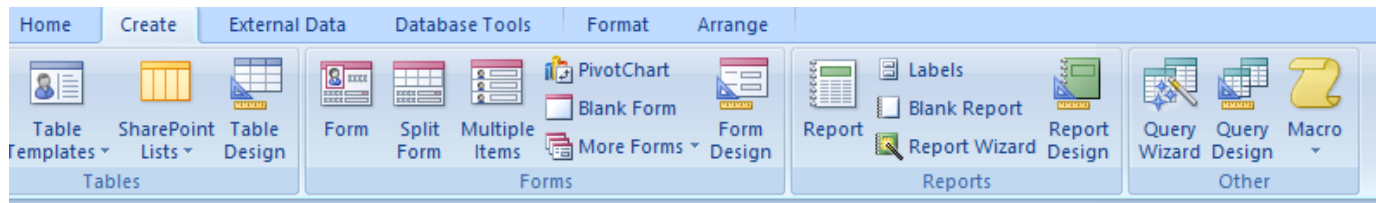
☐ I would like a wizard to analyze my table after importing the data.

Advanced... Cancel < Back Next > Finish

Form

Create a Form

- **Select the Table:**
 - In the Navigation Pane, select the table you just imported.
- **Go to the "Create" tab:**
 - Click on "Form" in the "Forms" group to automatically create a basic form based on the selected table.




MOVIE

Report

 MOVIE		Wednesday, September 4, 2024 11:54:49 AM					
MOVIE_ID	TITLE	GENRE	DIRECTOR		DURATION	RATING	PRICE
1	Sooryavanshi	Action	Rohit Shetty		145	6.2	150
2	Pagal Panti	Comedy	Anees Bazmee		149	3.2	120
3	Your Name	Drama	Makoto Shinkai		106	8.4	200
4	Dark Knight	Action	Christopher Nolan		152	9	250
5	A Silent Voice	Drama	Naoko Yamada		130	8.2	200
6	Weathering With You	Fantasy	Makoto Shinkai		112	7.5	200
7	Bhool Bhulaiyaa	Comedy	Anees Bazmee		159	7.4	150
8	Inception	Action	Christopher Nolan		148	8.8	250
9	Cirkus	Comedy	Rohit Shetty		138	3.5	150
10	Liz And The Blue Bird	Music	Naoko Yamada		90	7.2	200

Form

 MOVIE

MOVIE_ID:	1
TITLE:	Sooryavanshi
GENRE:	Action
DIRECTOR:	Rohit Shetty
DURATION:	145
RATING:	6.2
PRICE:	150

Theater

Report

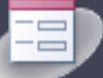


THEATER

Wednesday, September 4, 2024
11:50:04 AM

THEATER_ID	TH_NAME	LOCATION	CAPACITY
10002	Ambience Mall	Delhi	400
10005	INOX	Kota	600
10004	Select Citywalk	Delhi	800
10001	THE CIYT MALL	Kota	700
10003	World Trade Park	Jaipur	500

Form




THEATER


THEATER_ID:	10002
TH_NAME:	Ambience Mall
LOCATION:	Delhi
CAPACITY:	400

Showtime

Report

		Wednesday, September 4, 2024 11:36:57 AM	
SHOWTIME_ID	TITLE	TH_NAME	SHOW_DATETIME
6201	Sooryavanshi	INOX	2024-07-01 19:00:00
6202	Pagal Panti	THE CIYT MALL	2024-07-25 20:00:00
6203	Your Name	WORLD TRADE PARK	2024-08-01 16:00:00
6204	Dark Knight	THE CIYT MALL	2024-08-27 23:00:00
6205	A Silent Voice	SELECT CITYWALK	2024-07-24 15:00:00
6206	Weathering With You	AMBIENCE MALL	2024-09-10 23:30:00
6207	Bhool Bhulaiyaa	INOX	2024-09-30 10:00:00
6208	Inception	SELECT CITYWALK	2024-08-19 10:35:00
6209	Cirkus	WORLD TRADE PARK	2024-09-08 20:15:00
6210	Liz And The Blue Bird	INOX	2024-10-01 19:25:00
10			


Form

 Showtime

SHOWTIME_ID:	6201
TITLE:	Sooryavanshi
TH_NAME:	INOX
SHOW_DATETIME:	2024-07-01 19:00:00

Customer

Report


		CUSTOMER		Wednesday, September 4, 2024	
				11:46:01 AM	
CUSTOMER_ID	C_NAME	EMAIL	PHONE_NO		
101	Esha Mehta	EshaMehta@gmail.com	1234567891		
102	Karan Patel	KaranPatel@gmail.com	1593574561		
103	Emily White	EmilyWhite@gmail.com	9874562492		
104	Jessiaca Jones	JessiacaJones@gmail.com	2326548163		
105	Saniya Sharma	SaniyaSharma@gmail.com	2582583949		
106	Vikram Joshi	VikramJoshi@gmail.com	2295566488		
107	Jane Smith	JaneSmith@gmail.com	7894758964		
108	Neha Kapoor	NehaKapoor@gmail.com	6635977458		
109	Ananya Joshi	AnanyaJoshi@gmail.com	2335943269		
110	James wilson	Jameswilson@gmail.com	4582692347		

Form


		CUSTOMER	
CUSTOMER_ID:	101		
C_NAME:	Esha Mehta		
EMAIL:	EshaMehta@gmail.com		
PHONE_NO:	1234567891		

Booking

Report

 Booking				Tuesday, September 3, 2024
				11:16:13 PM
BOOKING_ID	CUSTOMER_ID	SHOWTIME_ID	SEATS_BOOKED	
2002	101	6201	4	
2003	102	6202	6	
2004	103	6203	2	
2005	104	6204	9	
2006	105	6205	3	
2007	106	6206	1	
2008	107	6208	9	
2009	108	6208	7	
2010	109	6209	4	
2011	110	6210	8	
10				

Form

 Booking

BOOKING_ID:	2002
CUSTOMER_ID:	101
SHOWTIME_ID:	6201
SEATS_BOOKED:	4

Future Enhancements for Movie Booking Database

1. Enhanced User Authentication and Role Management:

- **Description:** Implement a user authentication system with different roles (e.g., admin, customer, theater manager) to manage access and permissions.
- **Benefit:** Enhance security and better manage user actions.

2. Advanced Search and Filtering Options:

- **Description:** Add search capabilities to filter movies by genre, rating, director, and showtimes.
- **Benefit:** Improve user experience by allowing customers to find preferred movies quickly.

3. Integration with Payment Gateways:

- **Description:** Integrate secure payment processing for online booking and ticket payment.
- **Benefit:** Streamline the booking process and enhance customer convenience.

4. Dynamic Pricing Model:

- **Description:** Adjust ticket prices based on demand, time of day, or special events.
- **Benefit:** Maximize revenue and provide discounts during off-peak times.

5. Customer Feedback and Ratings System:

- **Description:** Allow customers to leave feedback and rate movies after watching.
- **Benefit:** Provide valuable insights to improve future offerings.

6. Mobile Application Development:

- **Description:** Develop a mobile app for browsing movies, checking showtime, and booking tickets.

- **Benefit:** Enhance accessibility and cater to the growing trend of mobile usage.

7. **Data Analytics and Reporting Tools:**

- **Description:** Implement tools to analyze booking trends, customer preferences, and theater performance.
- **Benefit:** help Theater managers make informed decisions based on data insights.

8. **Integration with Social Media:**

- **Description:** Allow users to share bookings and movie experiences on social media.
- **Benefit:** Increase visibility and attract more customers through social sharing.

9. **Loyalty and Rewards Program:**

- **Description:** Introduce a loyalty program that rewards frequent bookings with discounts or free tickets.
- **Benefit:** Enhance customer retention and encourage repeat business.

10. **Multi-Language Support:**

- **Description:** Provide multi-language support for the database interface.
- **Benefit:** Improve accessibility and user experience for non-native speakers.

Conclusion

This database design effectively manages movies, theaters, showtime, customers, and bookings in a movie theater booking system. It is structured across five key tables—**Movies**, **Theaters**, **Showtime**, **Customers**, and **Bookings**—each capturing essential details relevant to its entity.

The **Movies** table tracks movie details like title, genre, and price. The **Theaters** table manages theater-specific information, including name and location. **Showtime** link movie to theater, ensuring proper scheduling. **Customers** stores customer data, while **Bookings** records transactions, linking customers to showtime and seats booked.

Primary and foreign keys ensure data integrity and prevent redundancy, enabling efficient queries and smooth operations. This SQL database design provides a solid foundation for a scalable, efficient booking system.

References

- [geeksforgeeks.org](https://www.geeksforgeeks.org)
- [simplilearn.com](https://www.simplilearn.com)
- Chat-GPT
- YouTube
- Wikipedia
- Google
- Smart Draw
- A- level Introduction to database management system