

DBMS

Database Management System with MySQL

Yash Lathiya

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No. | Topic | Date | Page No. |
|  |  |  |  |
| 1 | Overview of DBMS |  |  |
| 2 | MYSQL |  |  |
| 3 | Overview of Workbench |  |  |
| 4 | Database Design |  |  |
| 5 | SQL Basics |  |  |
| 6 | Data Sorting |  |  |
| 7 | Null Value & Keyword |  |  |
| 8 | Auto Increment |  |  |
| 9 | Data Languages |  |  |
| 10 | Limit |  |  |
| 11 | Aggregate Functions |  |  |
| 12 | Sub Queries |  |  |
| 13 | Joins |  |  |
| 14 | Unions |  |  |
| 15 | Index |  |  |
| 16 | View |  |  |
| 17 | Backup, Restore, Explain |  |  |

Table of Contents

Overview of DBMS

Database

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

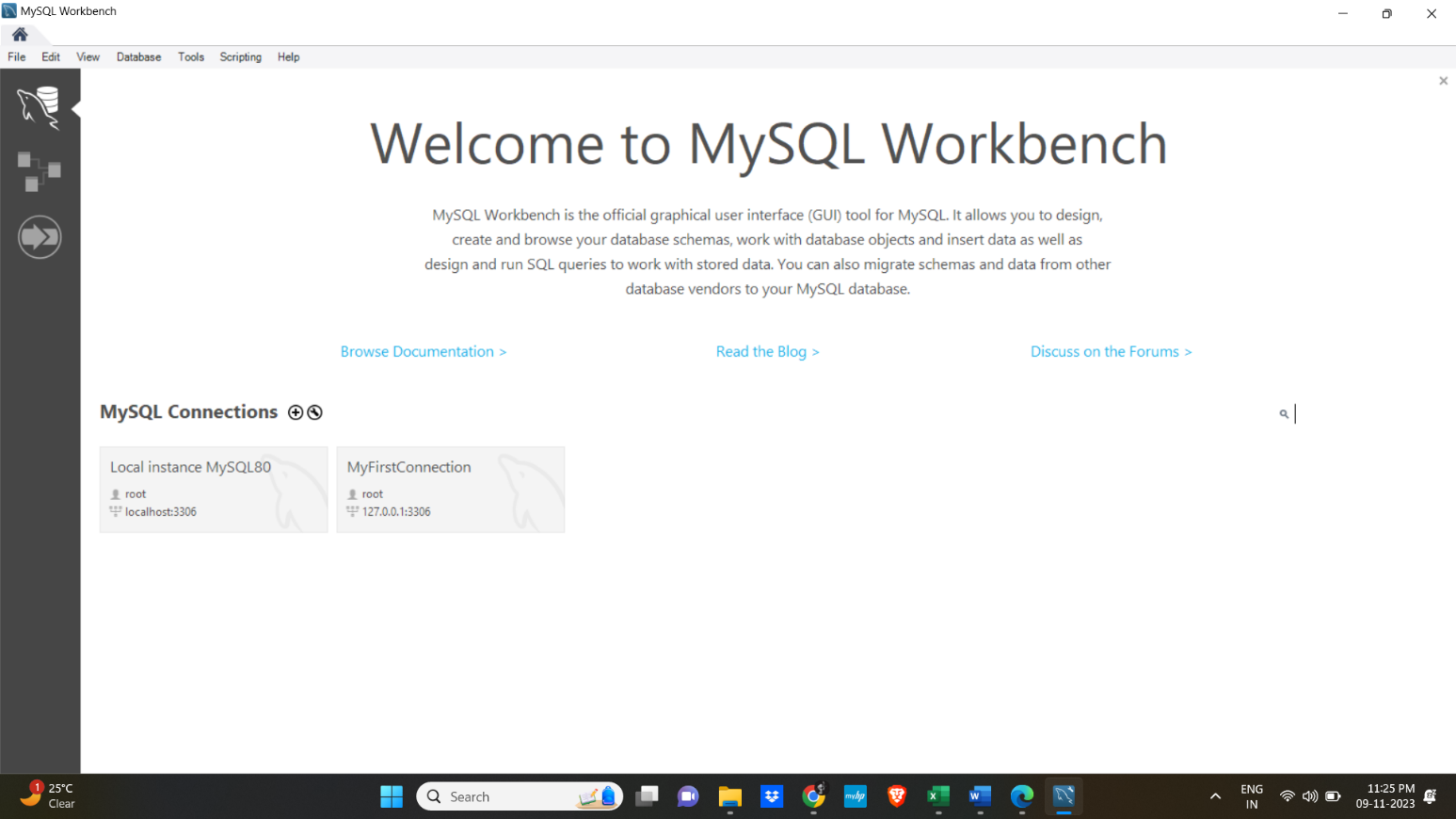
Database Management System

DBMS is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data.

MYSQL

MySQL is structured query language which is also relational database management system. By using different types of data languages, we can insert, remove and retrieve data in efficient manner with help of MySQL.

Overview of Workbench



Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

Workbench features

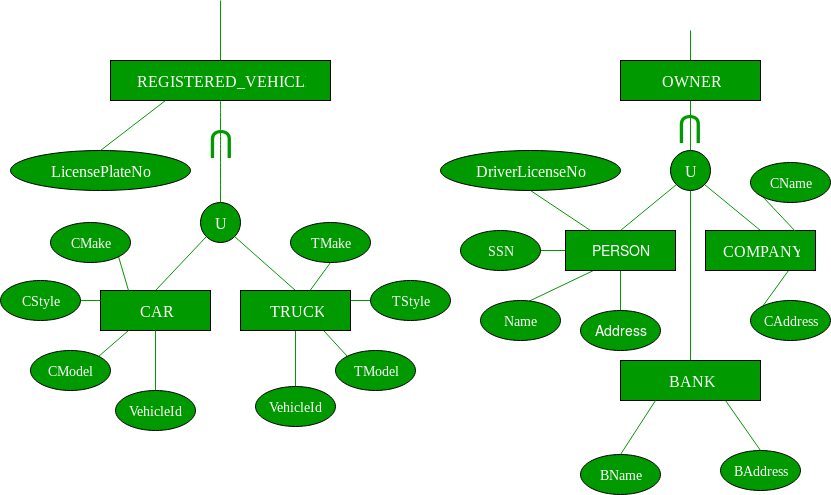
Connection

Model

ScriptingDatabase Design

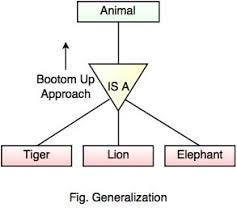
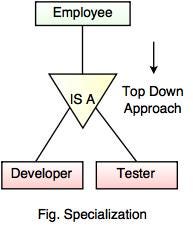
EER Diagram

Database design can be done through applying significant constrains on database like applying primary key, foreign key, data length constraints. All these types of constrains can be visibly represented by EER diagram which is enhanced entity relation diagram.



EER diagram also includes concepts of

Generalization & specialization..



SQL Basics

Data Sorting

-- Sorting of Data

-- Here data is sorted by ID then FirstName then LastName in ascending order

SELECT

Id,

FirstName,

LastName

FROM

MyFirstTable

ORDER BY

Id, FirstName, LastName ASC;

-- Here data is sorted by ID then FirstName then LastName in decending order

SELECT

Id,

FirstName,

LastName

FROM

MyFirstTable

ORDER BY

Id, FirstName, LastName DESC;

Null Value & Keyword

There is three type of values null values, non null values & empty values.

Null values are values which are not provided to database, It will not displayed when select query is fired.

Empty values are values which is empty string, It represents that value is not there or not provided, It will displayed when select query is fired.

Not null is keyword which implements a functionality in which specification and insertion of values are compulsory otherwise it will generate errors, It does not support null values.

By default columns are treated in such a way that it allows null values.

ALTER TABLE

MyFirstTable

ADD

Address varchar(255) NOT NULL;

Auto Increment

-- Create Table which contains one auto increment field

-- Auto Increment strats from 1001 with increment of 1

-- while both contains 1 as default values.

CREATE TABLE

Employee (

Id int PRIMARY KEY AUTO\_INCREMENT,

FirstName varchar(255),

LastName varchar(255),

Organization varchar(255),

Salary int

) AUTO\_INCREMENT = 1001;

Data Languages

DDL

* Data Definition Language is a set of SQL commands used to define, modify, and manage the structure of a database.
* It includes commands like
* **CREATE** (used to create objects like tables, indexes, etc.)
* **ALTER** (used to modify existing objects)
* **DROP** (used to delete objects)
* **TRUNCATE** (used to remove all records from a table), and
* **RENAME** (used to rename objects).

DML

* Data Manipulation Language is a set of SQL commands used to manage data within a database.
* It includes commands like
* **SELECT** (used to retrieve data)

**INSERT** (used to add new records),

**UPDATE** (used to modify existing records),

**DELETE** (used to remove records).

DCL

* Data Control Language is a set of SQL commands used to control access to data within a database.
* It includes commands like
* **GRANT** (used to provide specific privileges to users or roles),
* **REVOKE** (used to revoke privileges), and
* **DENY** (used to deny access).

TCL

* Transaction Control Language is a set of SQL commands used to manage transactions within a database.
* It includes commands like
* **COMMIT** (used to save changes made during the current transaction)
* **ROLLBACK** (used to undo changes made during the current transaction), and
* **SAVEPOINT** (used to set points within a transaction to which you can later roll back).

DQL

* Data Query Language is a subset of SQL that is used to query and retrieve data from a database.
* The primary command in DQL is **SELECT**, which is used to fetch data from one or more tables based on specified criteria.
* Example: **SELECT \* FROM table\_name WHERE condition;**

Limit

SELECT

ProductID,

ProductName,

Price

FROM

Products

LIMIT -- Selecting first 50 records

50

OFFSET -- Neglecting first 10 records

10;

Aggregate Functions

-- Count

SELECT

COUNT(OrderId)

FROM

Orders;

-- Sum

SELECT

SUM(Price)

FROM

Products

WHERE

Price < 100;

-- Average

SELECT

AVG(Price)

FROM

Products;

-- Minimum

SELECT

MIN(Price)

FROM

Products

WHERE

PRICE > 100;

-- Maximum

SELECT

MAX(Price)

FROM

Products

WHERE

PRICE < 200;

Sub Queries

-- Find details of product whose price is more than avg. price

SELECT

ProductID,

ProductName,

Price

FROM

Products

WHERE

ProductID

IN

(SELECT

ProductID

FROM

products

WHERE

price > (SELECT AVG(Price) FROM Products)

);

-- Corelated Subqueries

SELECT

ProductID,

ProductName,

Price

FROM

Products v\_product

WHERE

Price > (SELECT

AVG(Price)

FROM

Products

);

There are operators like IN, NOT IN, EXIST which is used while implementing subqueries.

Joins



SELECT

Orders.OrderID,

Customers.CustomerID,

OrderDate

FROM

Customers

INNER JOIN

Orders

ON

Orders.CustomerID = Customers.CustomerID;



SELECT

Orders.OrderID,

Customers.CustomerID,

OrderDate

FROM

Customers

LEFT JOIN

Orders

ON

Orders.CustomerID = Customers.CustomerID;



SELECT

Orders.OrderID,

Customers.CustomerID,

OrderDate

FROM

Customers

RIGHT JOIN

Orders

ON

Orders.CustomerID = Customers.CustomerID;



SELECT

Orders.OrderID,

Customers.CustomerID,

OrderDate

FROM

Customers

CROSS JOIN

Orders

ON

Orders.CustomerID = Customers.CustomerID;

Unions

-- Union

It displays all unique values of dataset after performing union operation.

In simple words, It returns a set.

No duplication of values in union.

SELECT

'Shipper' AS Type,

ShipperID,

ShipperName

FROM

Shippers

UNION SELECT

'Supplier' AS Type,

SupplierID,

SupplierName

FROM

Suppliers;

-- Union All

It displays all values (with duplication) which represents data combination of both queries.

As it allows duplicate values, It may or may not be a set.

SELECT

'Shipper' AS Type,

ShipperID,

ShipperName

FROM

Shippers

UNION SELECT

'Supplier' AS Type,

SupplierID,

SupplierName

FROM

Suppliers;

Index

An index is a database object that improves the speed of data retrieval operations on a table. It serves as a mechanism to optimize query performance by allowing the database engine to quickly locate and access the rows of a table.

Purpose of Indexing:

* Faster Data Retrieval:

Indexes provide a faster way to look up and retrieve specific rows from a table, especially when dealing with large datasets.

* Query Optimization:

Indexes optimize the execution of SELECT, JOIN, and WHERE clauses in SQL queries.

CREATE INDEX

Index\_ID

ON

Customers(CustomerID);

-- Unique Index

-- Doesn’t allow duplicate values

CREATE UNIQUE INDEX

Index\_Contact

ON

Customers(CustomerID, ContactName);

View

-- view

-- view is a virtual table which is based on sql statements and conditions.

-- view has rows and columns as real tables which can be initialized or created as well as updated and dropped

-- create view

-- France Customers

CREATE VIEW

v\_franceCustomers

AS

SELECT

CustomerID,

CustomerName,

City

FROM

Customers

WHERE

Country = 'France';

-- How to use view

SELECT

CustomerID,

CustomerName,

City

FROM

v\_franceCustomers;

-- UPDATE VIEW

CREATE OR REPLACE VIEW

v\_franceCustomers

AS

SELECT

CustomerID,

CustomerName,

City,

Address

FROM

Customers

WHERE

Country = 'France';

-- Drop(Discard) View

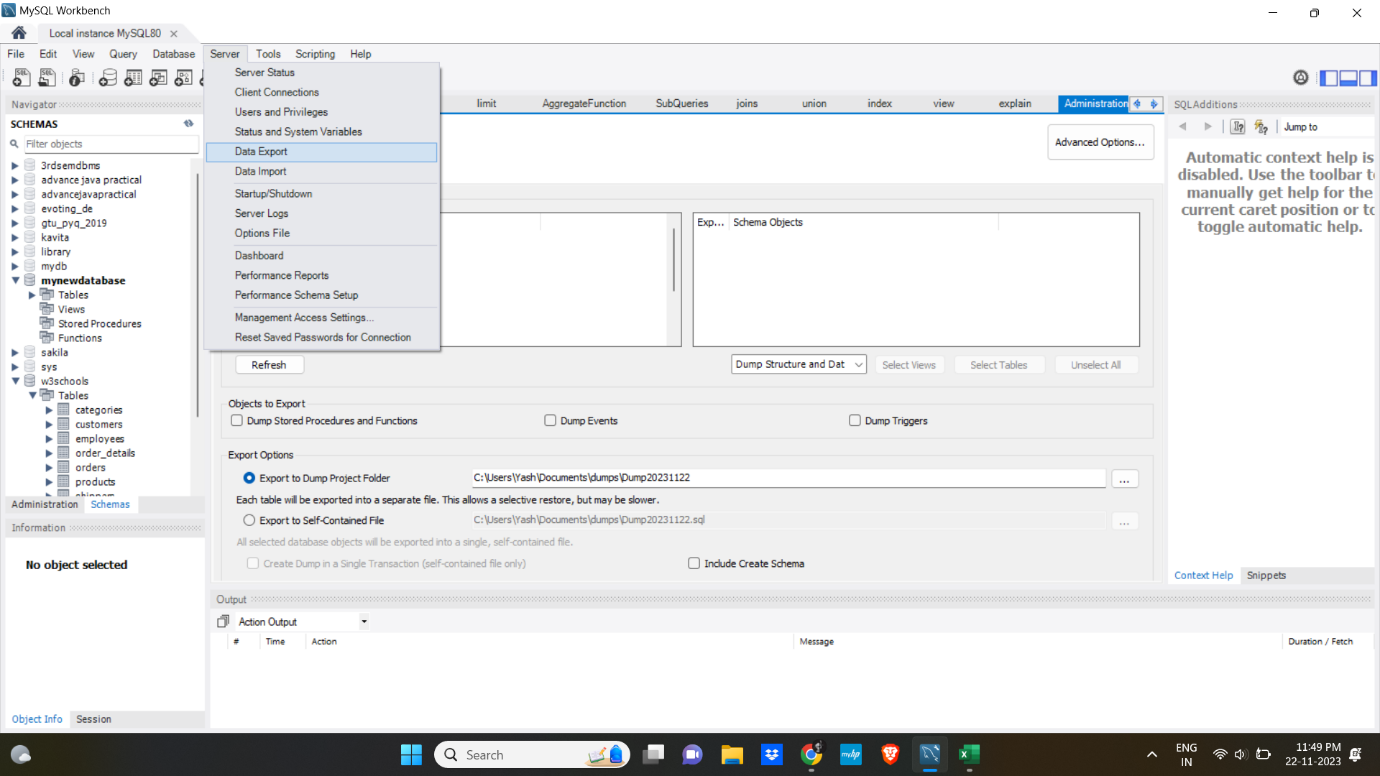
DROP VIEW

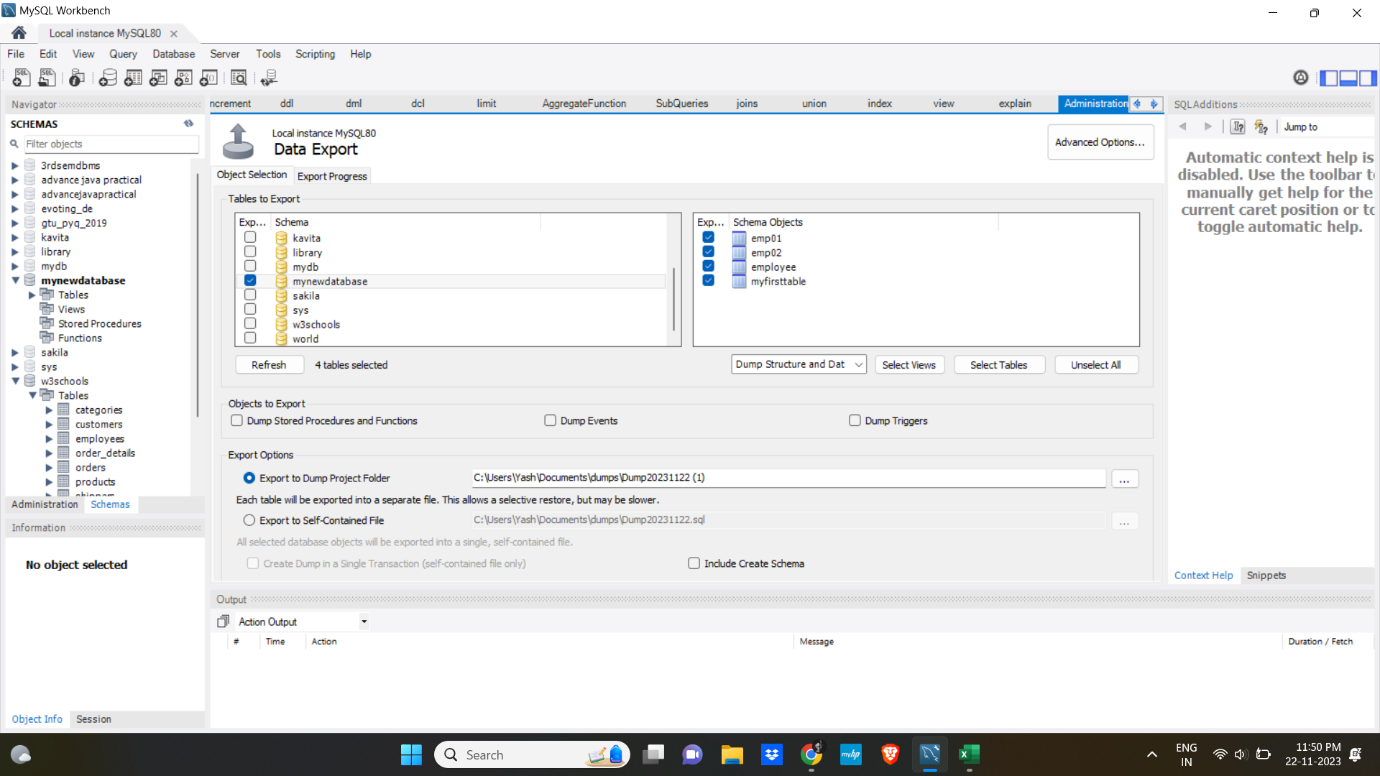
v\_franceCustomers;

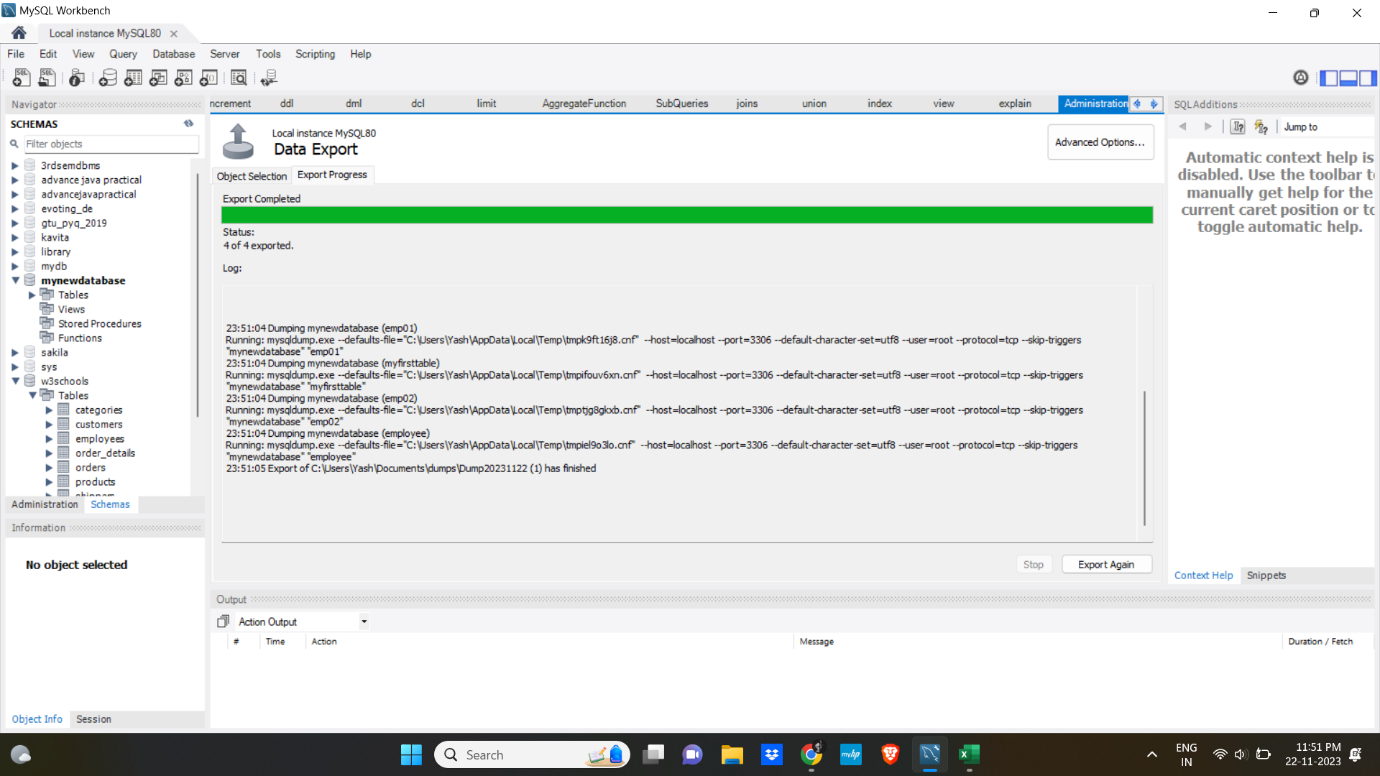
Backup, Restore, Explain

Backup

Backing up a MySQL database is crucial for data protection and disaster recovery. The mysqldump utility is commonly used to create backups in MySQL.

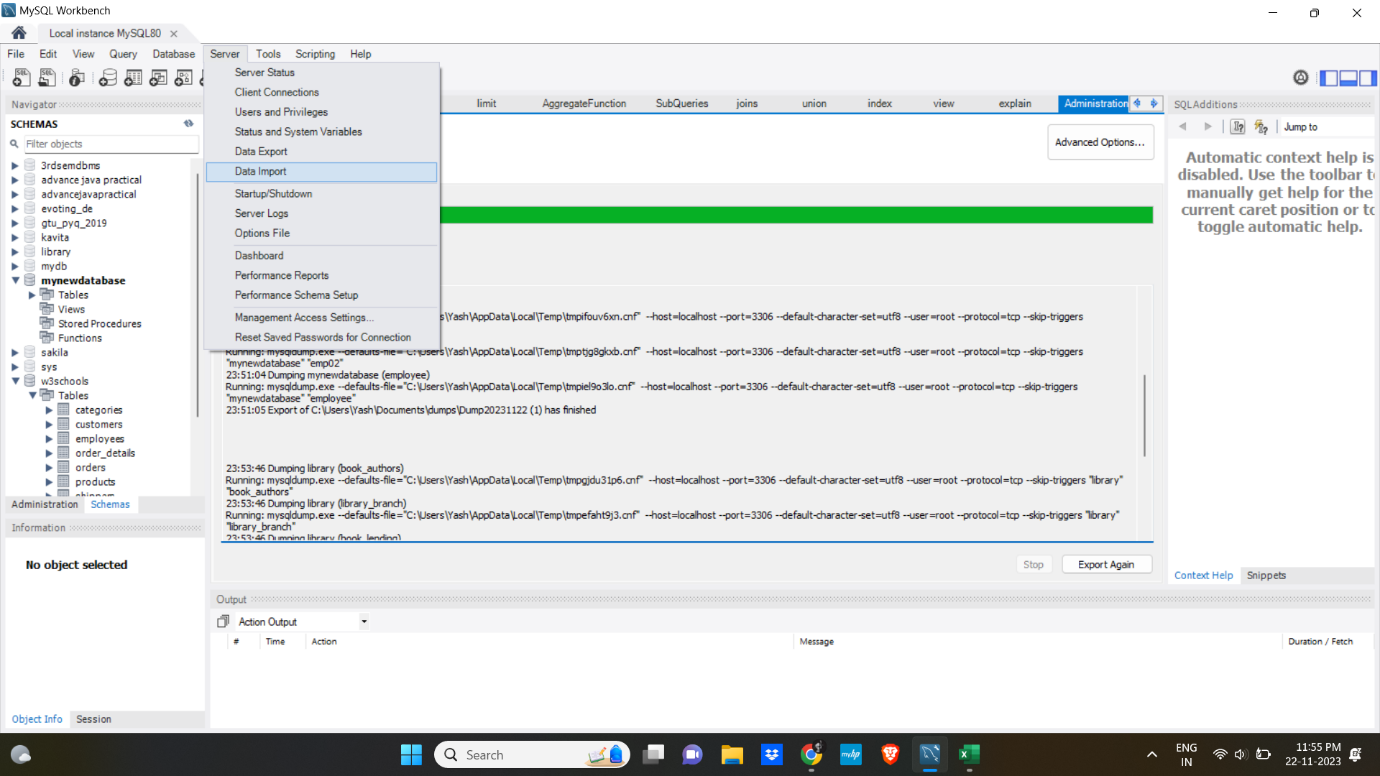


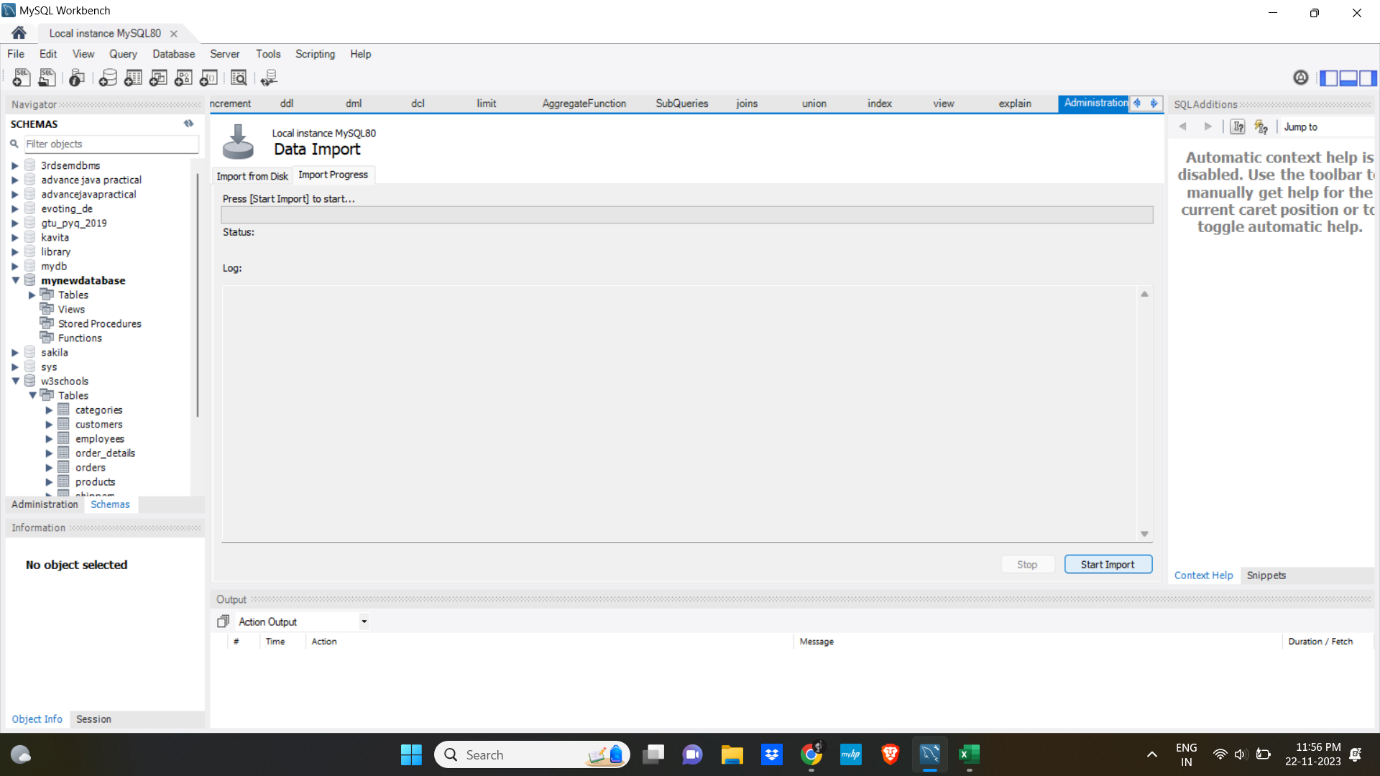




Restore

Restoring a MySQL database involves recreating the database from a backup.





Explain

The EXPLAIN keyword is used to obtain information about how the MySQL optimizer executes a SQL query. It provides insights into the execution plan that MySQL has chosen for a specific query, including details about the indexes used, the order of table access, and optimization strategies.

* **id**:

An identifier for the query within the execution plan.

* **select\_type**:

The type of SELECT query (e.g., SIMPLE, PRIMARY, SUBQUERY).

* **table**:

The table referenced in the output row.

* **type**:

The type of join that MySQL has chosen for the query (e.g., index scan, full table scan).

* **possible\_keys**:

The indexes that MySQL could potentially use.

* **key**:

The index that MySQL has chosen to use.

* **rows**:

The estimated number of rows MySQL expects to examine.

EXPLAIN SELECT

CustomerID,

CustomerName,

City

FROM

Customers

WHERE

CustomerID > 85;

