Starting date SQL;07-01-2024 class -01

**What is sql(structured query(inquiry, sawal) language) ??** is a software/standard/language which is used to communicate with database. We use it to Maintain, manipulate, manage the data base.

**MySQL:**

* MySQL is a specific relational database management system (RDBMS) that implements the SQL language.
* MySQL was initially developed by a Swedish company, MySQL AB, and later acquired by Sun Microsystems (now part of Oracle Corporation).
* It is an open-source RDBMS and is one of the most widely used databases in the world, known for its speed, reliability, and ease of use.
* MySQL supports SQL as its query language, allowing users to interact with the database using standard SQL commands.

**What is data ??** data is information, word, number, description, observation which describes that thing.

**What is data base?**

Data base is a structured collection of data, which provide a systematic and organized approach to storing, managing, and retrieving data.

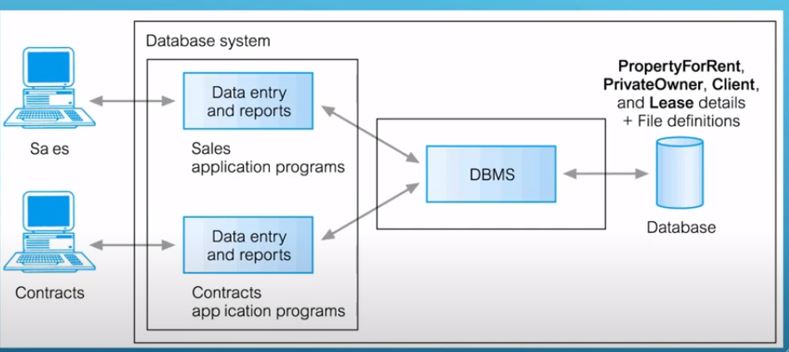
**What is data base management system DBMS?**

Is a software which is used to communicate with data base?

**Types of DBMS?**

**1. relational data base**

**2. non relational data base**



**What is relational data base???**

Data base which store the data in the form of tables containing rows and column is called relational data base.

Examples of software.

Mysql , postgresql, oracle, Microsoft sql server

**Non relational data base???**

Data base which do not store data in the form of table, it store the data in the form of **Document-Oriented**.

**Key-Value Stores**

**Graph Databases**

**What is schema?**

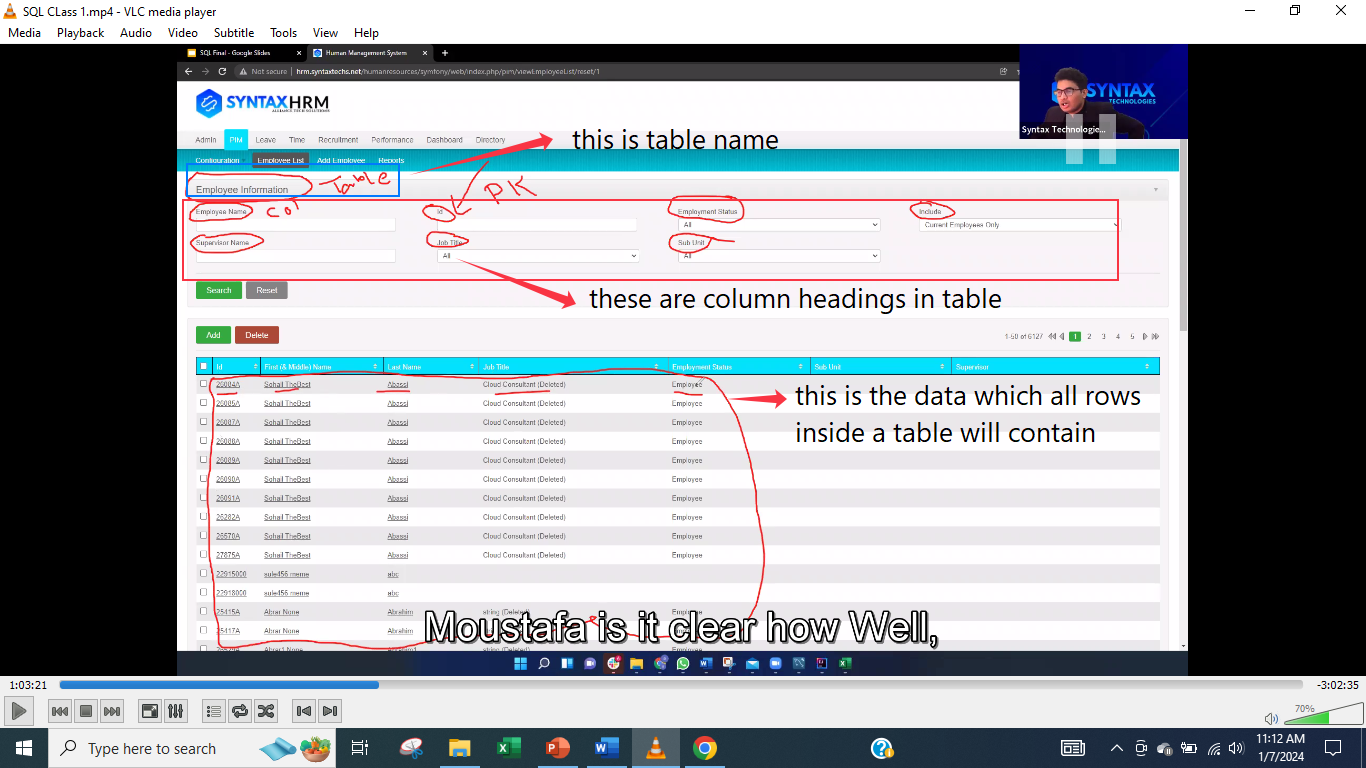
Schema is a visual information about the table and the relation of tables with each other.

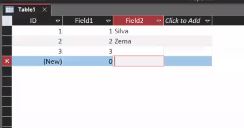
This is visual / graphical representation of tables and their relationship with each other.

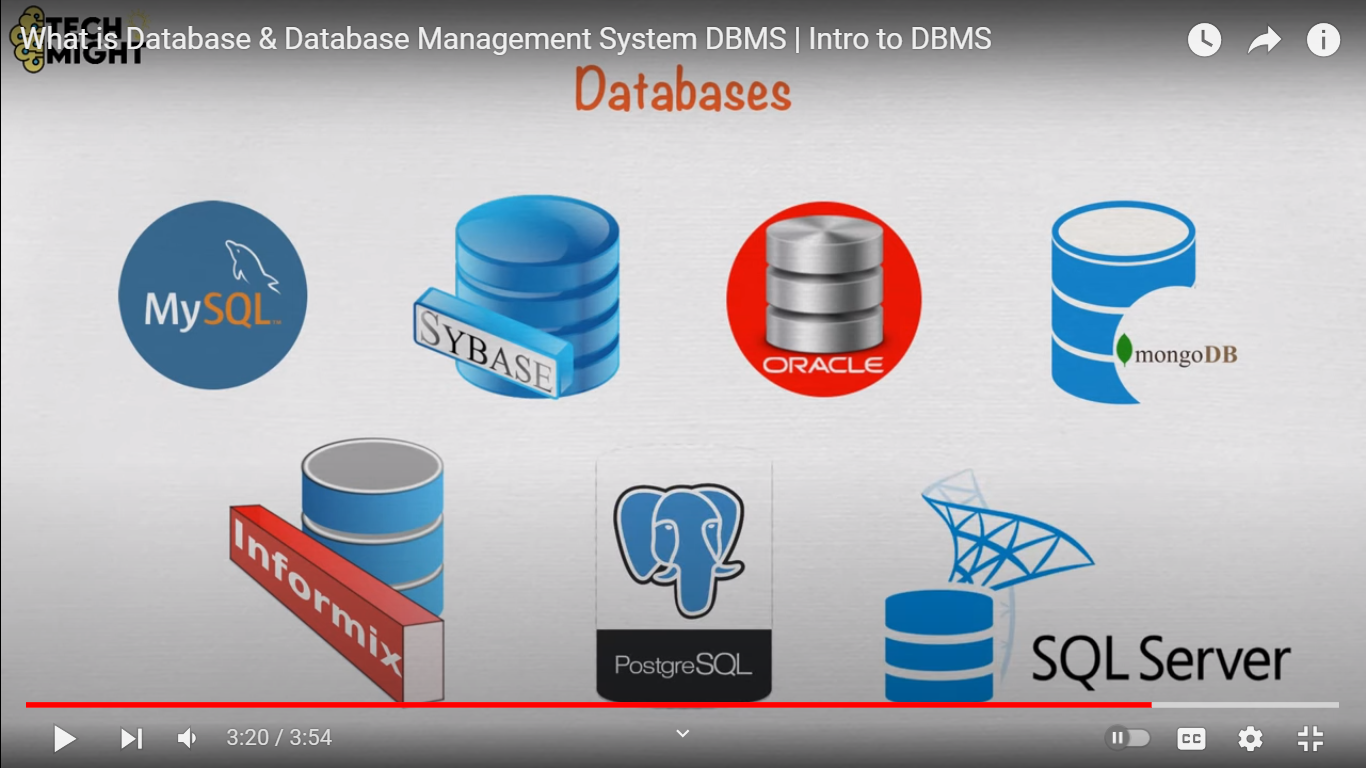
How table ins data base is represented bye

**Why we use data base instead of storing data in excel file???**

Reason is that excel file can not handle large data they become , and other reason is security issue , every one can access every data.







Select \* from employe ------ star mean all column in table

Sql commands

**Data Definition Language (DDL):**

* **CREATE DATABASE**: Creates a new database.
* **DROP DATABASE**: Deletes an existing database.
* **CREATE TABLE**: Creates a new table.
* **ALTER TABLE**: Modifies an existing table structure.
* **DROP TABLE**: Deletes an existing table.

**Data Manipulation Language (DML):**

* **SELECT**: Retrieves data from one or more tables.
* **INSERT INTO**: Inserts new records into a table.
* **UPDATE**: Modifies existing records in a table.
* **DELETE FROM**: Deletes records from a table.

**Data Query Language (DQL):**

* **SELECT**: Used to query the database and retrieve data.
* **SHOW DATABASES**: Lists all databases.
* **SHOW TABLES**: Lists all tables in the current database.

---------------------------------------------------------------------------------------------------------

SELECT \* FROM employees;

Select; is used to print all the columns in employee table.

SELECT distinct last name

Distinct---- is used to remove duplicate values from the table.

SELECT lastname from classicmodels.employees limit 2;

Is used to print just two rows of lastname column

Where---- key word is used to filter the rows.

Relational operator

----- >

---- <

---- <=

----- >=

------ !

------ <> this is also for ! operator.

not is actually logical operator, it invert the true value to false.

Basically it is used to negate the result.

SELECT\*from classicmodels.employees where reportsto>1002 ;

Convert (‘id’, decimal) ---- this convert the data type from varchar to decimal.

Order by ---- is used to print the ascending order column data.

-------------------------------------------------------------

When you want the result should not null

SELECT \* FROM classicmodels.employees WHERE lastName is NOT null;

----------------------------------------------------------

SELECT\*from classicmodels.orders where comments is null;

When row is null inside column print all.

---------------------------------------------------------

When the element is not present in table, empty entry. Print all

SELECT\*from classicmodels.orders where comments=’ ’;

Live oracle link

[https://livesql.oracle.com/apex/f?p=590:1:113715287212769:::RP::#](https://livesql.oracle.com/apex/f?p=590:1:113715287212769:::RP::)

what is difference in where clause and group by clause;

* Use the **WHERE** clause to filter rows before any grouping.
* Use the **HAVING** clause to filter groups after a **GROUP BY** clause based on aggregate functions.

Sub queries, this queries is inside another query which is used to give intermediate result which is difficult by other queries.

**What is foreign key?**

Whenever a primary key of one table is present inside other table then this primary key inside another table is called foreign key.

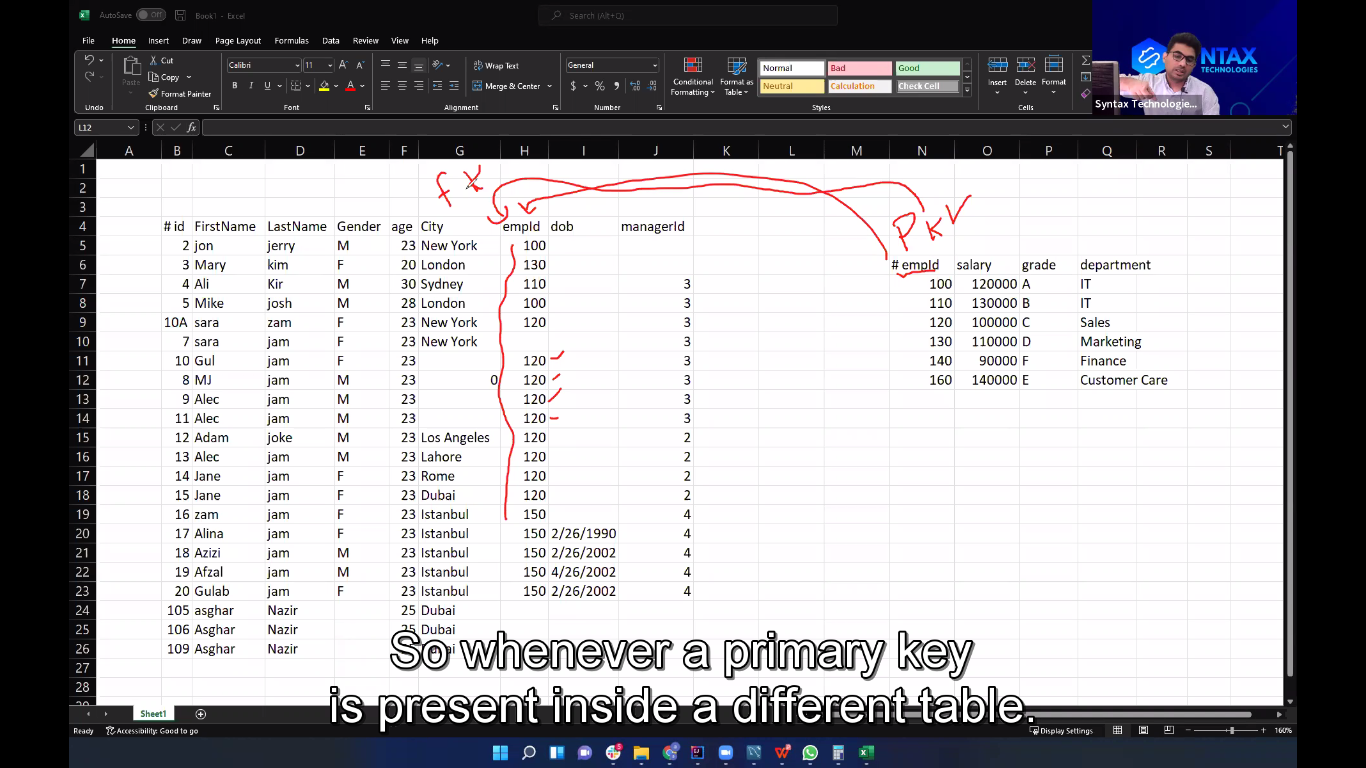
Foreign keys used to create link between two tables.

**Primary key cannot duplicate and cannot null**

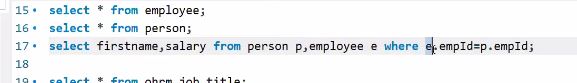
**Foreign key can be null and can be duplicate.**

The answer is in below photo. Multiple employes can have same department, multiply employee can have same salary.

New employe has not yet assign department and salary.



Date; 13-01-2024 class 03 sql



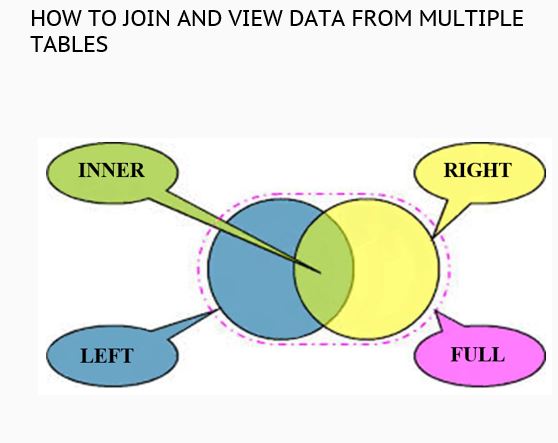
Alias is you can assign the table name shorter by your own choice.

Foreign key is the column name in other table , mean the id name in one table can be id-job in another table as a foreign key. Name can be change.

If the column in primary key and foreign key are same like empid=empid then you must add the table name also other wise sql will confuse about same name.

---------------------------------------------------------------------------------

In SQL, a JOIN is a clause used to combine rows from two or more tables based on a related column between them. The purpose of a JOIN is to retrieve data from multiple tables in a single query, allowing you to correlate information from different sources. There are several types of JOIN operations, including INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.



SELECT

columns

FROM

table1

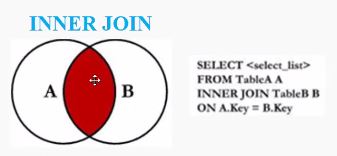
JOIN

table2 ON table1.column\_name = table2.column\_name;

------------------------------------------------------------------------------------

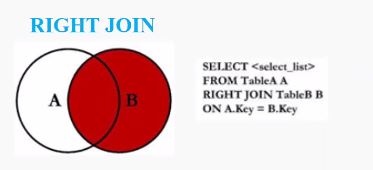
Inner join ? will bring only those rows for which values are present in both the table,

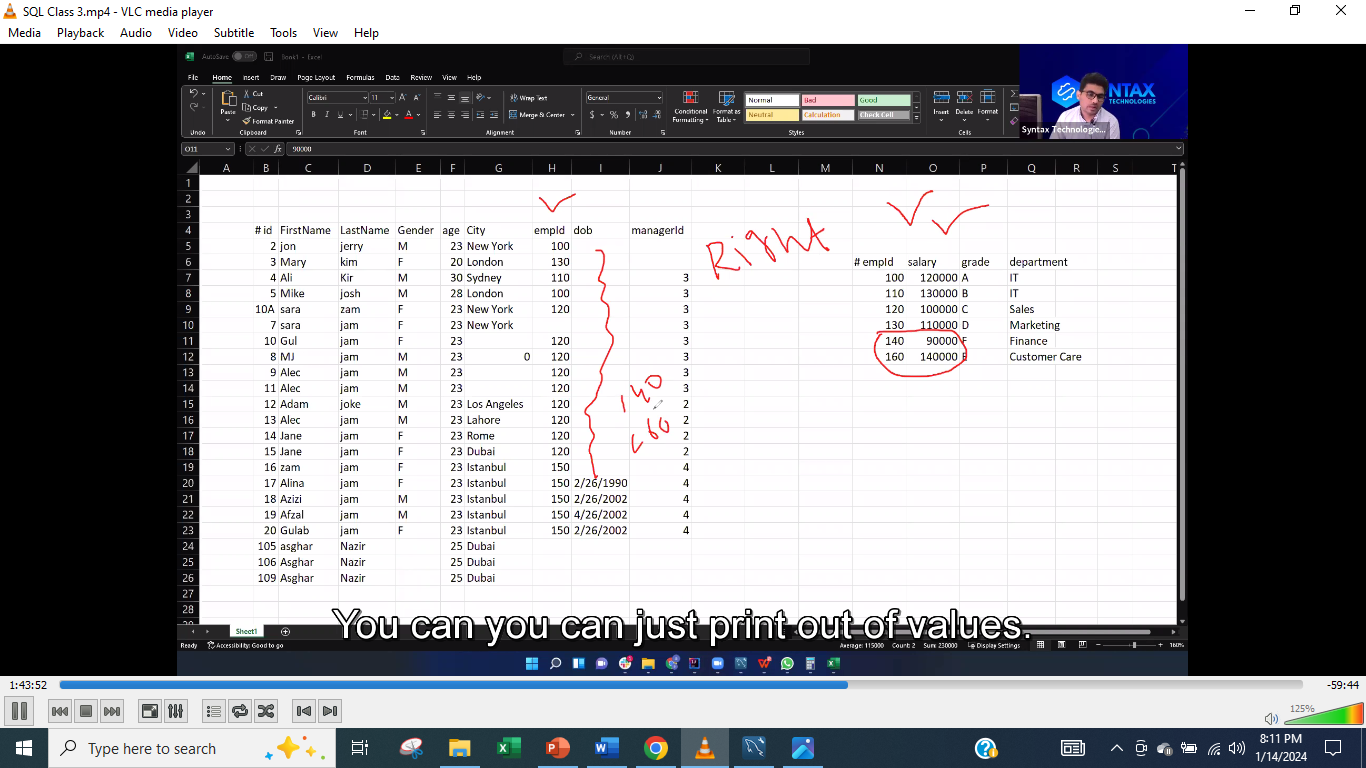
In SQL, an INNER JOIN is a type of JOIN operation used to combine rows from two or more tables based on a related column between them. The result includes only the rows that have matching values in the specified column from both tables. If there are no matches, those rows are excluded from the result.



-----------------------------------------------------------------------------------

In SQL, a RIGHT JOIN is one of the types of JOIN operations used to combine rows from two or more tables based on a related column between them. The result includes unmatched rows from the right (second) table and matched rows from the left (first) table. If there are no matches, NULL values are returned for columns from the left table.

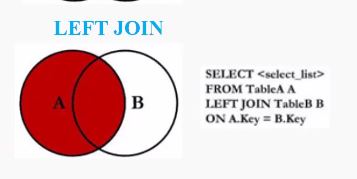




------------------------------------------------------------------------------------------

In SQL, a LEFT JOIN is one of the types of JOIN operations used to combine rows from two or more tables based on a related column between them. The result includes all rows from the left (first) table and the matching rows from the right (second) table. If there are no matches, NULL values are returned for columns from the right table.

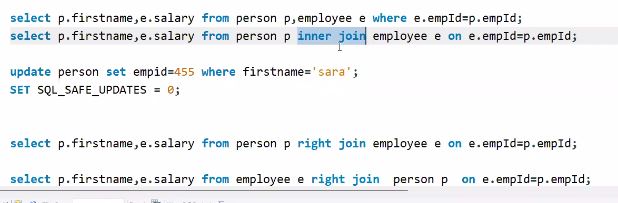
Here is the basic syntax for a LEFT JOIN:



-----------------------------------------------------------------------------------------

In SQL, a FULL JOIN (or FULL OUTER JOIN) is a type of JOIN operation used to combine rows from two or more tables based on a related column between them. The result includes all rows from both tables, with matching rows from the left (first) and right (second) tables. If there are no matches, NULL values are returned for columns from the table that does not have a match.





----------------------------------------------------------------------------------------

**A self join** in SQL is a specific case of a join operation where a table is joined with itself. This can be useful when you have a table that contains hierarchical data or when you need to compare rows within the same table. To perform a self join, you use an alias to differentiate between the two instances of the same table.

SELECT

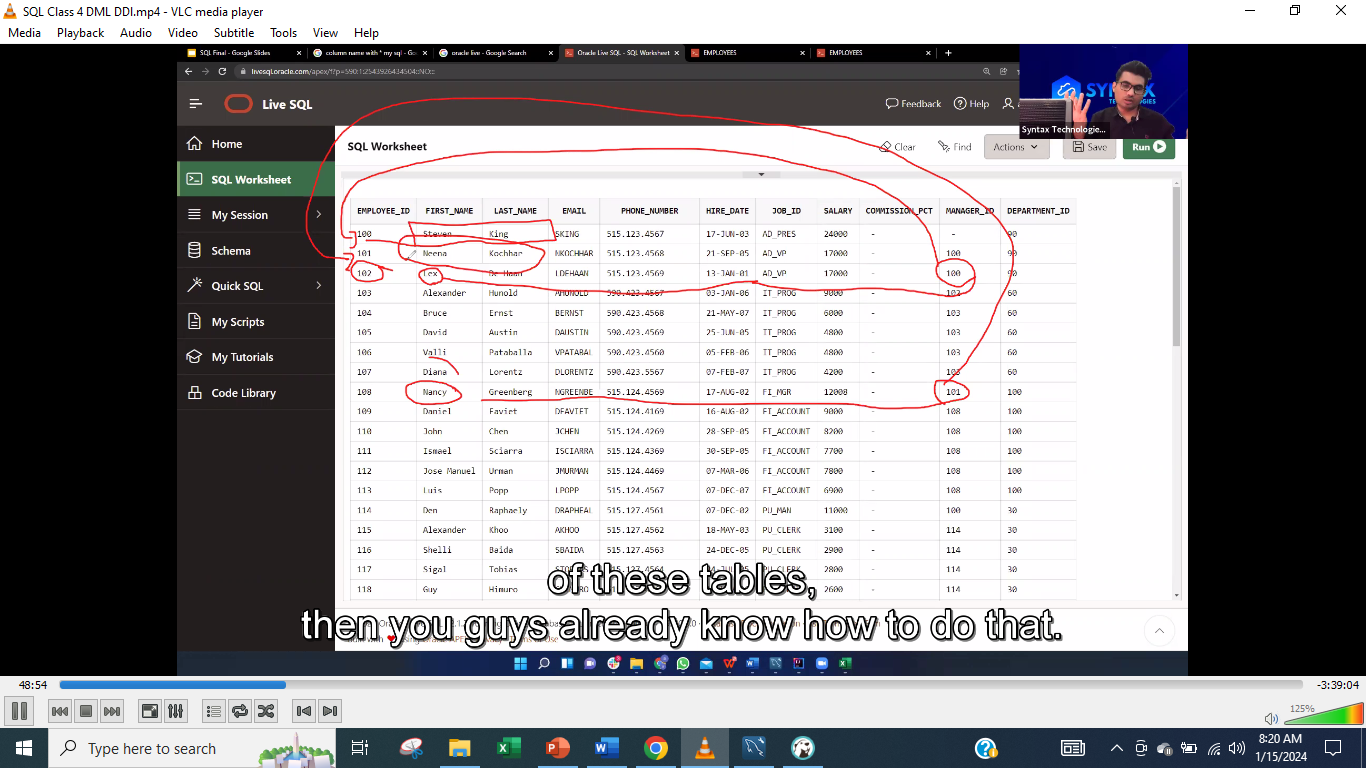
t1.column1, t1.column2, ..., t2.column1, t2.column2, ...

FROM

table\_name t1

JOIN

table\_name t2 ON t1.common\_column = t2.common\_column;



free sql live hosting data base link

<https://www.freemysqlhosting.net/>

**Numeric Types:**

**INT or INTEGER:** -2,147,483,648 to 2,147,483,647 signed, 0 to 4,294,967,295 unsigned

* Example: **INT(10)**

DECIMAL or NUMERIC: Fixed-point numbers

Example: DECIMAL(10, 2) (10 digits with 2 after the decimal point)

1. **FLOAT:** Example: **FLOAT(8, 2)**
2. **DOUBLE:** Example: **DOUBLE(15, 4)**

**String Types:**

1. **CHAR:** Fixed-length character strings.
   * Example: **CHAR(10)**
   * For example, if you have a **CHAR(10)** column and you insert the string 'abc', which is only 3 characters long, the remaining 7 characters will be filled with padding characters, typically spaces.
2. **VARCHAR:** Variable-length character strings.
   * Example: **VARCHAR(255)**
   * VARCHAR: Stores variable-length strings. It only uses as much storage as needed to store the actual data. If you define a VARCHAR(10), it will only occupy the necessary amount of storage for the specific data.
   * Example: VARCHAR(10) with the value 'abc' will be stored as 'abc' (no padding).

**Date and Time Types:**

DATE: Date values in 'YYYY-MM-DD' format.

TIME: Time values in 'HH:MM:SS' format.

DATETIME: Combined date and time values in 'YYYY-MM-DD HH:MM:SS' format.

TIMESTAMP: Similar to DATETIME, but with automatic update on record modification.

YEAR: Year in 2-digit or 4-digit format.

**Set Type:**

1. **SET:** A string object where the value can be chosen from a set of predefined values.
2. **SET:** Example: **SET('option1', 'option2', 'option3')**

Your user name is your data base name = sql12676917

**What is the difference in delete and drop ???**

Drop remove the complete table and data present inside it

Delete remove the complete rows or specified row from the table , and can be retrieves data.

Truncate ; is use to delete rows

Like **DELETE**, **TRUNCATE (TRASHANA, SHAAKHY TARANSHA, CHOTA KRNA, SMALL KRNA)** removes all rows from a table, but it is typically faster and uses fewer system resources. It can also retrieved data.

**Summary (DML): SUDI**

* DML commands (SELECT, UPDATE, DELETE, INSERT) primarily deal with manipulating or interacting with the data stored in tables.

**Summary (DDL): DR CAT**

DDL commands (DROP, RENAME, CREATE, ALTER, TRUNCATE) deal with the structure and definition of the database itself.

CREATE TABLE example\_table (

id INT PRIMARY KEY,

name VARCHAR(255)

);

**Explanation:** This **CREATE TABLE** statement creates a new table named **example\_table** with columns **id** and **name**. Once executed, this change is permanent; you can't roll back the creation of the table.

---------------------------------------------------------------------------------------------------

**DML is Roll back**/ undo/restore/go previous state.

Roll back mean is that if there is error in completing the transaction any transaction then go back to the previous step . don’t move the transaction forward.

* + decide to transfer money from Account A to Account B.

1. **Operations:**
   * This transfer involves two operations: deducting money from Account A and adding the same amount to Account B.

Rollback (Failure):

If there's a problem (e.g., insufficient funds), you "rollback" the transaction. This means undoing any changes made during the transaction. The money stays in Account A, and nothing is added to Account B.

---------------------------------------------------------------------------------------------------

Seamlessly ----- بغیر کسی رکاوٹ کے

HOSTNAME; 3.237.189.167

**Connection** is an interface in the java.sql package that represents a connection to a database.

**Statement** is an interface in the java.sql package that represents a SQL statement that can be executed against a database.

**ResultSet** is an interface in the java.sql package that represents the result of a database query, usually a table of data representing a database result set.

**Certainly! ResultSetMetaData** is an interface in the java.sql package that provides methods to get information about the columns in a ResultSet. It allows you to obtain metadata about the structure of the result set, such as the number of columns, column names, data types, and other properties. This information can be useful when you need to dynamically work with the columns **of a result set without having prior knowledge of the database schema.**

In the context of a ResultSet in Java, the ResultSetMetaData interface provides metadata about the result set obtained from a database query. It includes information about the columns in the result set, such as names, data types, and other properties.

**To obtain data from rows we use**

The resultSet.next() method is used in Java to move the cursor in a ResultSet object to the next row. It returns true if there is a next row, and false if there are no more rows in the result set.

Here's how it's typically used in a loop to iterate through the rows of a result set:

ResultSet resultSet = statement.executeQuery("SELECT \* FROM your\_table");

try {

// Iterate over rows in the result set

while (resultSet.next()) {

// Retrieve data from the result set for the current row

int id = resultSet.getInt("ID");

String description = resultSet.getString("Description");

// Process the data as needed

System.out.println("ID: " + id + ", Description: " + description);

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

// It's good practice to close the ResultSet, Statement, and Connection when done

try {

resultSet.close();

statement.close();

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class ResultSetMethodsExample {

public static void main(String[] args) {

// Replace these with your database credentials and connection details

String jdbcUrl = "jdbc:mysql://localhost:3306/your\_database";

String username = "your\_username";

String password = "your\_password";

try (Connection connection = DriverManager.getConnection(jdbcUrl, username, password);

Statement statement = connection.createStatement()) {

// Assuming you have a table named "example\_table" with columns "id" and "value"

ResultSet resultSet = statement.executeQuery("SELECT \* FROM example\_table");

// Using next() to move to the first row

if (resultSet.next()) {

// Using getInt and getString with column index

int idAtIndex = resultSet.getInt(1);

String valueAtIndex = resultSet.getString(2);

// Displaying values

System.out.println("First Row - ID: " + idAtIndex + ", Value: " + valueAtIndex);

}

// Using next() to move to the second row

if (resultSet.next()) {

// Using getInt and getString with column name

int idAtName = resultSet.getInt("id");

String valueAtName = resultSet.getString("value");

// Displaying values

System.out.println("Second Row - ID: " + idAtName + ", Value: " + valueAtName);

}

// Using previous() to move to the previous row (if any)

if (resultSet.previous()) {

int idPrev = resultSet.getInt("id");

String valuePrev = resultSet.getString("value");

System.out.println("Previous Row - ID: " + idPrev + ", Value: " + valuePrev);

}

// Using last() to move to the last row (if any)

if (resultSet.last()) {

int idLast = resultSet.getInt("id");

String valueLast = resultSet.getString("value");

System.out.println("Last Row - ID: " + idLast + ", Value: " + valueLast);

}

// Using absolute(int row) to move to an absolute row number (if within bounds)

if (resultSet.absolute(3)) {

int idAbsolute = resultSet.getInt("id");

String valueAbsolute = resultSet.getString("value");

System.out.println("Absolute Row 3 - ID: " + idAbsolute + ", Value: " + valueAbsolute);

}

// Close the ResultSet when done (statement and connection will be closed automatically)

resultSet.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

https://github.com/SyntaxTechnologies/CucumberFrameworkBatch11.git