Lecture-3

Database Management system:

SQL Mastery

Topics

- Schema creation, insert/delete/update/select.
- Joins: Inner, Left, Right, Full.
- Subqueries, With all The Clauses.
- Views, Indexes, Stored Procedures, Functions.

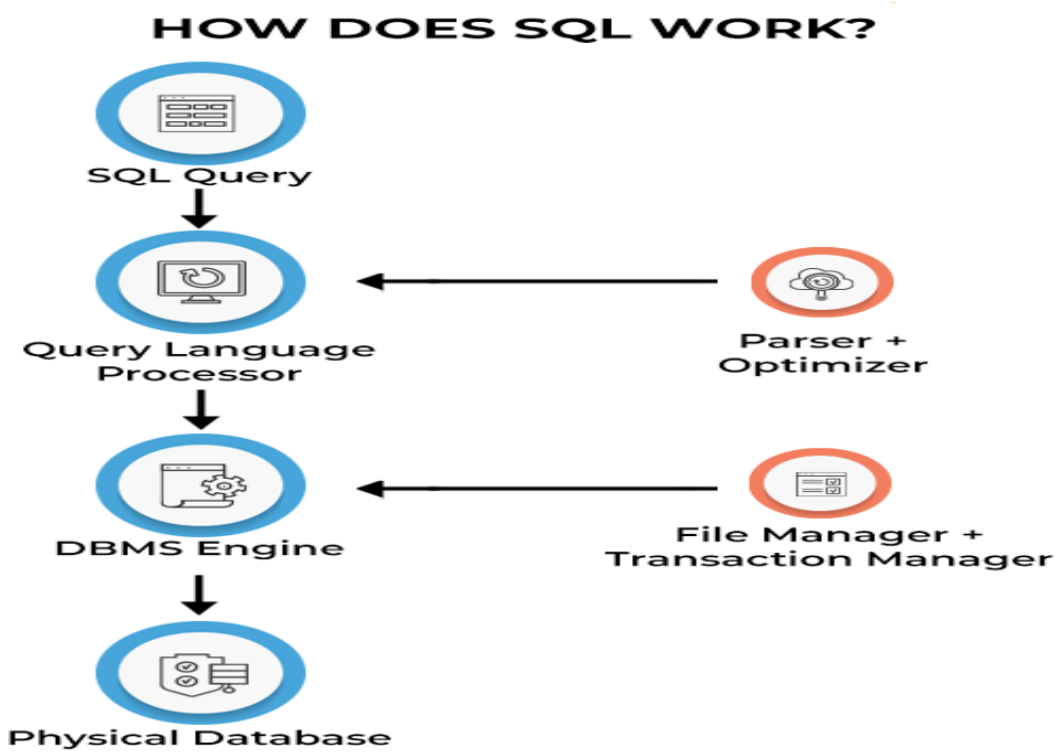
What is SQL?

- SQL stands for **Structured Query Language**.
- Structured query language (SQL) is a programming language for storing and processing information in a relational database.
- A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values.
- You can use SQL statements to store, update, remove, search, and retrieve information from the database. You can also use SQL to maintain and optimize database performance.
- Used to communicate with relational databases

Why is SQL important?

- Structured query language (SQL) is a popular query language that is frequently used in all types of applications.
- Data analysts and developers learn and use SQL because it integrates well with different programming languages.
- For example, they can embed SQL queries with the Java programming language to build high-performing data processing applications with major SQL database systems such as Oracle or MS SQL Server. SQL is also fairly easy to learn as it uses common English keywords in its statements

How does SQL work?



What are SQL commands?

- Structured query language (SQL) commands are specific keywords or SQL statements that developers use to manipulate the data stored in a relational database. You can categorize SQL commands as follows.
- Data definition language
- Data definition language (DDL) refers to SQL commands that design the database structure. Database engineers use DDL to create and modify database objects based on the business requirements. For example, the database engineer uses the CREATE command to create database objects such as tables, views, and indexes.

What are SQL commands? ctd

- Data query language
- Data query language (DQL) consists of instructions for retrieving data stored in relational databases. Software applications use the SELECT command to filter and return specific results from a SQL table.
- Data manipulation language
- Data manipulation language (DML) statements write new information or modify existing records in a relational database. For example, an application uses the INSERT command to store a new record in the database.
- Data control language
- Database administrators use data control language (DCL) to manage or authorize database access for other users. For example, they can use the GRANT command to permit certain applications to manipulate one or more tables.

What are SQL commands? ctd

- Transaction control language
- The relational engine uses transaction control language (TCL) to automatically make database changes. For example, the database uses the ROLLBACK command to undo an erroneous transaction.
- What are SQL standards?
- SQL standards are a set of formally defined guidelines of the structured query language (SQL). The American National Standards Institute (ANSI) and International Organization for Standardization (ISO) adopted the SQL standards in 1986. Software vendors use the ANSI SQL standards to build SQL database software for developers.

SQL vs. MySQL

- Structured query language (SQL) is a standard language for database creation and manipulation.
- MySQL is a relational database program that uses SQL queries.
- While SQL commands are defined by international standards, the MySQL software undergoes continual upgrades and improvements.

What is NoSQL?

- NoSQL refers to non-relational databases that don't use tables to store data. Developers store information in different types of NoSQL databases, including graphs, documents, and key-values.

SQL vs. NoSQL

- Structured query language (SQL) provides a uniform data manipulation language, but NoSQL implementation is dependent on different technologies. Developers use SQL for transactional and analytical applications, whereas NoSQL is suitable for responsive, heavy-usage applications.

What is Schema?

- A **schema** is a collection of database objects like tables, triggers, stored procedures, etc. A schema is connected with a user which is known as the **schema owner**. The database may have one or more schema.
- Syntax
- CREATE SCHEMA syntax is:

CREATE SCHEMA schemaname AUTHORIZATION ownername]GO

DDL – Data Definition Language

- **Purpose:** Define and manage database structures

  Common Commands:

- **CREATE**: Create new tables/databases
- **ALTER**: Modify existing tables
- **DROP**: Delete tables or databases
- **TRUNCATE**: Remove all records from a table

DDL – Data Definition Language ctd

- CREATE TABLE Students (ID INT, Name VARCHAR(50), Age INT);
- ALTER TABLE Students ADD Email VARCHAR(100);
- DROP TABLE Students;

DML – Data Manipulation Language

- **Purpose:** Define and manage database structures

  Common Commands:

- **CREATE**: Create new tables/databases
- **ALTER**: Modify existing tables
- **DROP**: Delete tables or databases
- **TRUNCATE**: Remove all records from a table

DML – Data Manipulation Language ctd

- INSERT INTO Students (ID, Name, Age) VALUES (1, 'Alice', 22);

Or

- INSERT INTO users (username, email) VALUES ('john_doe', 'john@example.com');

- UPDATE Students SET Age = 23 WHERE ID = 1;

- DELETE FROM Students WHERE ID = 1;

- SELECT * FROM Students;

SQL Clauses and Keywords

- `WHERE`: Filters records
- `ORDER BY`: Sorts results
- `GROUP BY`: Groups data for aggregation
- `HAVING`: Filters groups

SQL Clauses and Keywords example

WHERE

SELECT * FROM orders WHERE amount > 100;

ORDER BY

SELECT * FROM customers ORDER BY last_name ASC;

GROUP BY

SELECT department, COUNT(*) FROM employees GROUP BY department;

HAVING

SELECT department, COUNT(*)

FROM employees

GROUP BY department

HAVING COUNT(*) > 5;

SQL Operators and Functions

- **Comparison:** =, !=, <, >, <=, >=
- **Logical:** AND, OR, NOT
- **Aggregate Functions:** COUNT(), SUM(), AVG(), MAX(), MIN()

DML – Data Manipulation Language

- **Purpose:** Define and manage database structures

  Common Commands:

- **CREATE**: Create new tables/databases
- **ALTER**: Modify existing tables
- **DROP**: Delete tables or databases
- **TRUNCATE**: Remove all records from a table

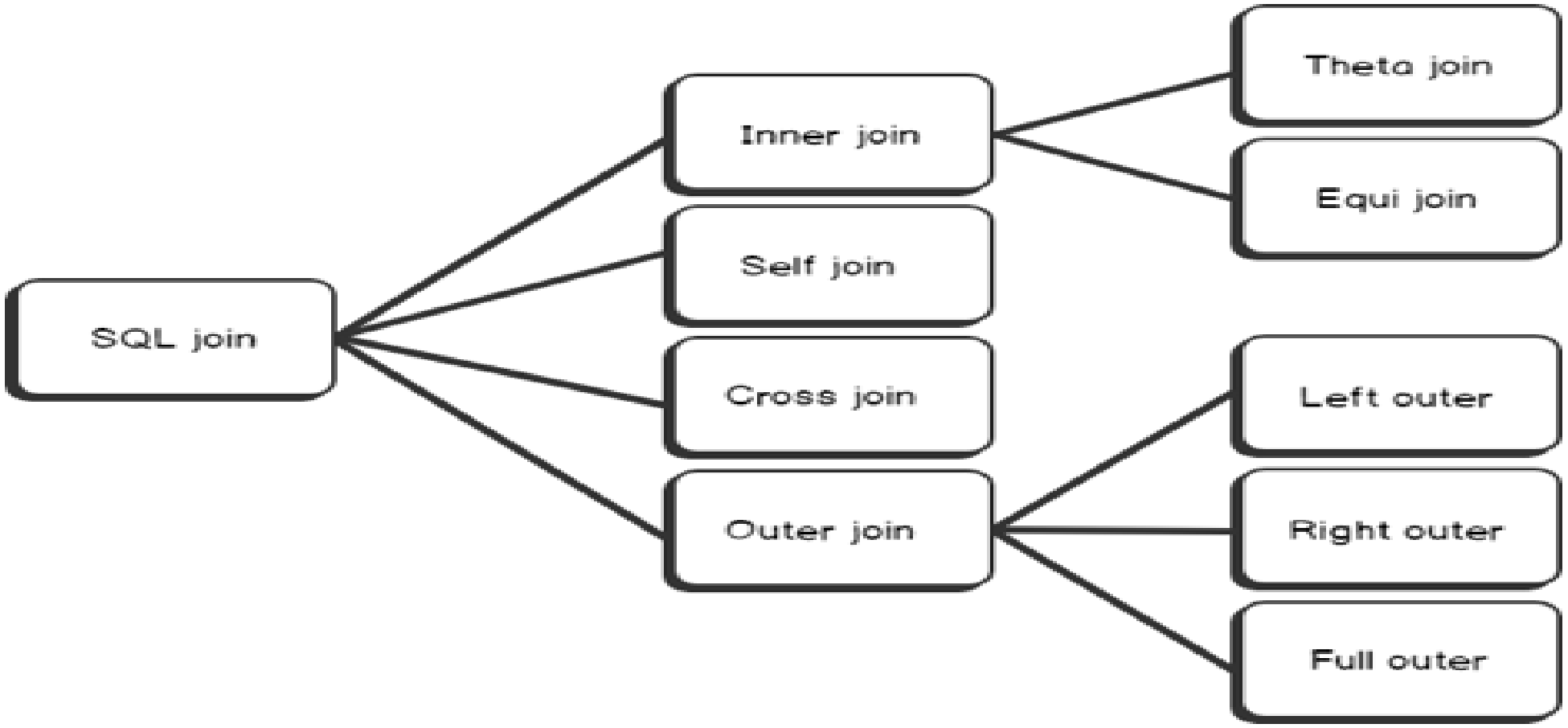
What Are SQL Joins?

- Joins are used to combine rows from two or more tables.

Or

- A SQL Join statement combines data or rows from two or more tables based on a common field between them.
- They are performed based on a related column between them (usually a foreign key).

Types of SQL Joins



Cross Join

- A cross join is a Cartesian Product join – it is every record in Table A combined with every record in Table B.
- It gives the same results as not using a WHERE clause when querying two tables in MySQL
- SELECT * from TableA CROSS JOIN TableB
- SELECT * from TableA, TableB

Cross Join Example

SQL Joins

```
SELECT *
  FROM Students S CROSS JOIN Enrolled E
```

S

S.name	S.sid
Jones	11111
Smith	22222

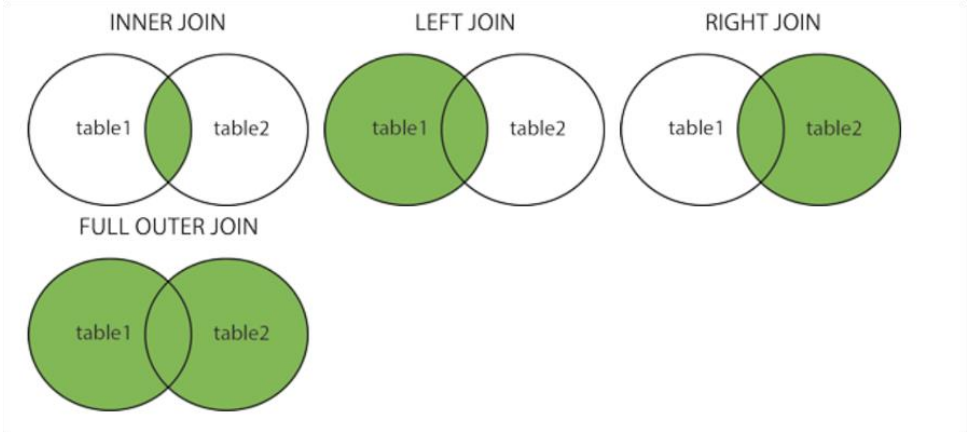
E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150

S.name	S.sid	E.sid	E.classid
Jones	11111	11111	History105
Jones	11111	11111	DataScience194
Jones	11111	22222	French150
Smith	22222	11111	History105
Smith	22222	11111	DataScience194
Smith	22222	22222	French150

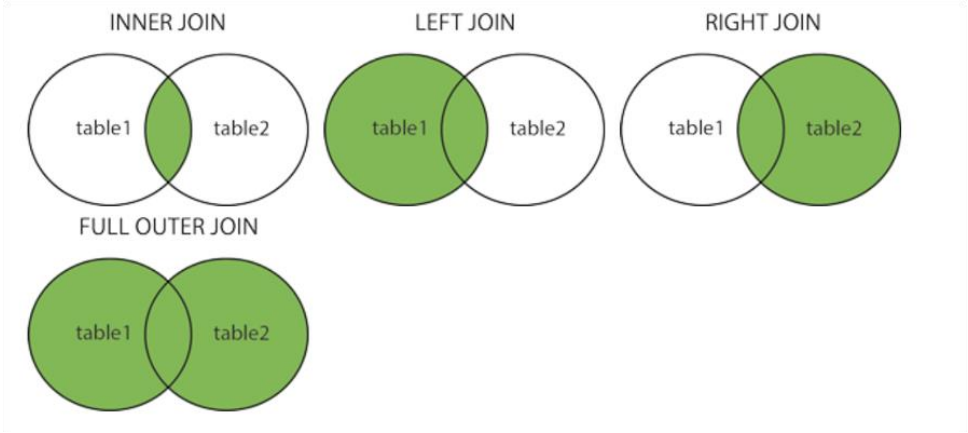
Different Types of SQL JOINS

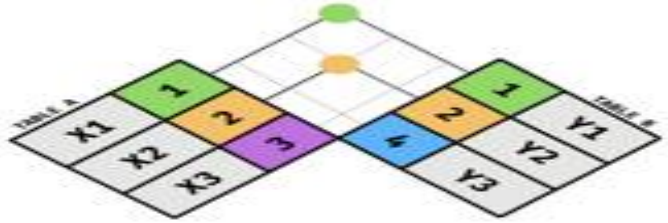
- Here are the different types of the JOINs in SQL:
- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



Different Types of SQL JOINS

- Here are the different types of the JOINs in SQL:
- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

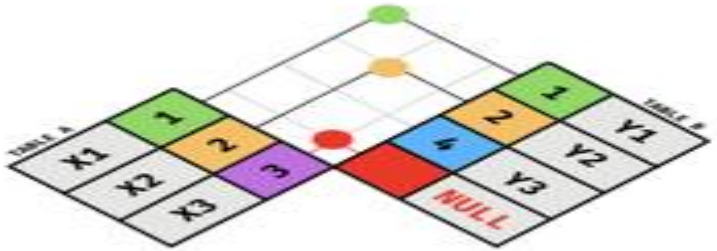




INNER JOIN

```
SELECT
      <SELECT LIST>
FROM        TABLE_A A
INNER JOIN TABLE_B B
      ON A.KEY = B.KEY
```

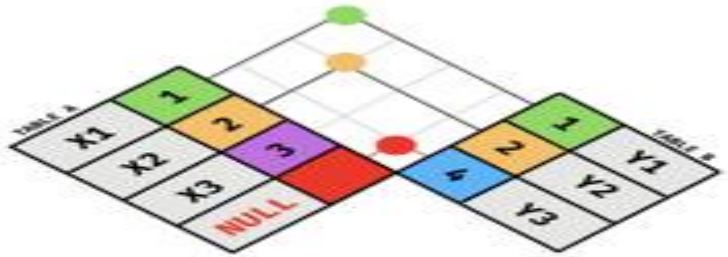
KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2



LEFT JOIN

```
SELECT
      <SELECT LIST>
FROM        TABLE_A A
LEFT JOIN TABLE_B B
      ON A.KEY = B.KEY
```

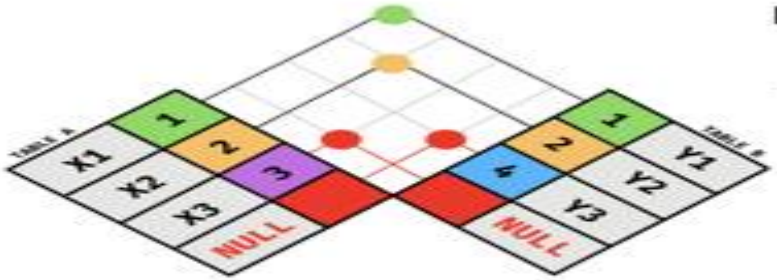
KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2
3	X3	NULL



RIGHT JOIN

```
SELECT
      <SELECT LIST>
FROM        TABLE_A A
RIGHT JOIN TABLE_B B
      ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2
4	NULL	Y3



FULL OUTER JOIN

```
SELECT
      <SELECT LIST>
FROM        TABLE_A A
FULL OUTER JOIN TABLE_B B
      ON A.KEY = B.KEY
```

KEY	VAL_X	VAL_Y
1	X1	Y1
2	X2	Y2
3	X3	NULL
4	NULL	Y3

What is a Subquery?

- A **subquery** is a query nested inside another query.
- Also called **inner query** or **nested query**.
- Enclosed in **parentheses** ( ).
- The result is used by the **outer query**.

Subquery Types

- **Single-row Subquery** – returns only one row.
- **Multiple-row Subquery** – returns multiple rows.
- **Multiple-column Subquery** – returns multiple columns.
- **Correlated Subquery** – depends on the outer query.

Subqueries in SELECT Clause

- SELECT name, (SELECT MAX(salary) FROM employees) AS MaxSalary FROM employees;

Subqueries in FROM Clause

- SELECT avg_salary FROM (SELECT department_id, AVG(salary) AS avg_salary FROM employees GROUP BY department_id) dept_avg;

Subqueries in HAVING Clause

- SELECT department_id, COUNT(*) FROM employees

GROUP BY department_id HAVING COUNT(*) > (SELECT AVG(emp_count) FROM (SELECT COUNT(*) AS emp_count FROM employees GROUP BY department_id));

- SELECT name FROM employees

  WHERE salary > (SELECT

AVG(salary) FROM employees);

Subqueries in Insert Statement

- INSERT INTO backup_employees (id, name, salary) SELECT id, name, salary FROM employees WHERE department_id = 10;

Subqueries in UPDATE Statement

- UPDATE employees SET salary = salary * 1.1

WHERE department_id = (SELECT department_id FROM departments WHERE name = 'Sales');

Subqueries in DELETE Statement

- DELETE FROM employees WHERE department_id = (SELECT department_id FROM departments WHERE location_id = 1700);

Correlated Subqueries

- SELECT name FROM employees WHERE salary > (SELECT AVG(salary) FROM employees WHERE department_id = e.department_id);

Subqueries Summary

- Subqueries provide flexibility and power.
- Used in many SQL clauses: SELECT, FROM, WHERE, HAVING, etc.
- Be cautious of performance impacts.
- Prefer joins for better performance in some cases.

Introduction to Views

- **Definition:**
  A View is a virtual table based on the result of an SQL query.
- **Features:**
- Simplifies complex queries
- Enhances security (can restrict access to certain columns)
- Can be used like a regular table

Views Example

- CREATE VIEW EmployeeView AS SELECT Name, Department FROM Employees WHERE Status = 'Active';

Introduction to Indexes

Definition:

An Index is a database object that improves the speed of data retrieval.

Features:

- Acts like a table of contents
- Faster SELECTs, slower INSERT/UPDATE/DELETE

Example

```
CREATE INDEX idx_name ON Employees(Name);
```

Types of Indexes

- **Clustered Index**: Sorts data rows in the table (one per table)
- **Non-clustered Index**: Separate from data rows (multiple allowed)
- **Unique Index**: Ensures all values are unique

Use Case: Speeding up search queries

Introduction to Stored Procedures

Definition:

A Stored Procedure is a group of SQL statements saved in the database and executed as a unit.

Features:

- Reduces code duplication
- Enhances performance and security
- Supports parameters

Example

```
CREATE PROCEDURE GetActiveEmployees
AS BEGIN
SELECT * FROM Employees WHERE Status = 'Active';
END;
```

Parameters in Stored Procedures

Types:

- IN (Input)
- OUT (Output)
- INOUT (Both)

Example with Parameters:

```
CREATE PROCEDURE GetEmployeeByID @EmpID INT
AS
BEGIN
  SELECT * FROM Employees WHERE ID = @EmpID;
END;
```

Introduction to Functions

Definition:

A Function is a stored program that returns a single value or table.

Differences from Procedures:

- Must return a value
- Cannot perform modifications (usually)

Example:

```
CREATE FUNCTION GetFullName(@FirstName NVARCHAR(50), @LastName NVARCHAR(50))
RETURNS NVARCHAR(101)
AS
BEGIN
  RETURN @FirstName + ' ' + @LastName;
END;
```

Scalar vs Table-Valued Functions

- **Scalar Functions:** Return a single value
- **Table-Valued Functions (TVF):** Return a table

Use Cases:

- Reusable calculations
- Encapsulate business logic

Differences Summary Table

Feature	View	Index	Stored Procedure	Function
Virtual Table	✅	❌	❌	❌
Improves Query	✅	✅	✅	✅
Returns Value	❌	❌	❌	✅
Accepts Params	❌	❌	✅	✅
Modifies Data	❌ (usually)	❌	✅	❌ (usually)

FAQ

1. What is the difference between **`CREATE TABLE`** and **`CREATE TABLE AS`**?
2. How do you create a table with a primary key and a foreign key?
3. How do you define default values for columns during table creation?
4. How do you create a table with constraints (e.g., **`CHECK`**, **`UNIQUE`**)?
5. Write a SQL query to create a table **`employees`** with the following fields:

`emp_id` (Primary Key), `name` (VARCHAR), `salary` (FLOAT) `dept_id` (Foreign Key referencing `departments` table)

FAQ

1. How can you insert multiple rows in a single **`INSERT`** statement?
2. How do you insert data into a table from another table (using **`INSERT INTO ... SELECT`**)?
3. How do you select all columns from a table?
4. What is the difference between **`WHERE`** and **`HAVING`**?
5. How do you use **`ORDER BY`** and **`LIMIT`**?
6. How do you use aggregate functions like **`COUNT()`**, **`AVG()`**, **`MAX()`**?
7. What is the difference between **`INNER JOIN`**, **`LEFT JOIN`**, **`RIGHT JOIN`**, and **`FULL JOIN`**?
8. How do you write a correlated subquery in a **`SELECT`** statement?
9. How would you retrieve the second-highest salary from an **`employees`** table?

FAQ

1. How do you delete rows based on a subquery?
2. What is the difference between **DELETE** and **TRUNCATE**?
3. xplain different types of SQL JOINs (INNER, LEFT, RIGHT, FULL).
4. What's the difference between INNER JOIN and LEFT JOIN?
5. How would you retrieve records that exist in one table but not in another?
6. What is a CROSS JOIN? When would you use it?
7. Can you join more than two tables in a single SQL query? Provide an example.
8. Write a query to fetch customers and their orders, including those with no orders.
9. Explain how NATURAL JOIN works and when to avoid it.

FAQ

1. Can you update data through a view? What are the restrictions?
2. What is the difference between a simple view and a complex view?
3. What is an index in SQL? How does it improve performance?
4. What are the different types of indexes (e.g., clustered vs. non-clustered)?
5. Write a SQL query using GROUP BY and HAVING clauses.
6. Can user-defined functions (UDFs) be used in SQL? How do you create one?
7. How do you pass parameters to a stored procedure?
8. Can a stored procedure return a value? If so, how?

Queries?

Thanks