Day27_SQL_Introduction

June 24, 2025

Today, we are going to explore the basics of **SQL (Structured Query Language)** — the standard language used to interact with relational databases.

This notebook will give you a simple yet technical overview of key SQL concepts, commands, and examples.

In the upcoming sessions, we will dive deeper into each topic with hands-on practice and more real-world examples.
This is just the beginning — a foundation for what's to come!

---

1 What is SQL?

SQL (Structured Query Language) is a language used to communicate with **relational databases**.

It allows us to: - Create and manage databases - Insert, update, and delete records - Query data using simple statements - Control access and security

> Think of SQL as a way to "talk" to databases to get or modify information.

Why Learn SQL?

- Easy to understand and use
- Used in data analysis, web apps, software backends, business dashboards
- Works with popular databases like MySQL, PostgreSQL, Oracle, and SQL Server
- Must-have for data scientists, analysts, and backend developers

---

SQL History

Year	Event
1970	Dr. E.F. Codd proposed the relational database model
1974	SQL language was developed at IBM
1986	SQL became an ANSI standard
Today	Most RDBMS like MySQL, Oracle, PostgreSQL support SQL

How SQL Works (Behind the Scenes)

When you write an SQL command:

1. **Parser** checks if it's valid
2. **Query Optimizer** finds the fastest way to run it
3. **Execution Engine** fetches or modifies the data

---

2 SQL Command Categories

SQL commands are classified into **5 major types**:

Category	Full Form	Examples	Purpose
DDL	Data Definition Language	CREATE, DROP, ALTER	Define and modify database structure
DML	Data Manipulation Language	INSERT, UPDATE, DELETE	Add or change data
DQL	Data Query Language	SELECT	Read data from tables
DCL	Data Control Language	GRANT, REVOKE	Set permissions
TCL	Transaction Control Language	COMMIT, ROLLBACK	Handle transactions

---

3 Basic SQL Terminology

- **Database**: A collection of organized data.
- **Table**: Like an Excel sheet; stores rows and columns.
- **Row / Record**: A single entry in a table.
- **Column / Field**: A specific attribute of data.
- **Primary Key**: A unique ID for each row.
- **Foreign Key**: A reference to a primary key in another table.

Sample Table: CUSTOMERS

Let's work with this table for our examples:

```
+----+----------+-----+-----------+----------+
\| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
\|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
\|  2 | Khilan   |  25 | Delhi     |  1500.00 |
\|  3 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

4 Basic SQL Statements

4.1 SELECT – View Data

```
SELECT * FROM CUSTOMERS;
```

Output:

```
Returns all rows and columns from the table.
```

4.2 CREATE – Create a Table

```
CREATE TABLE CUSTOMERS (
    ID INT PRIMARY KEY,
    NAME VARCHAR(50),
    AGE INT,
    ADDRESS VARCHAR(100),
    SALARY DECIMAL(10,2)
);
```

4.3 INSERT – Add Data

```
INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
```

4.4 UPDATE – Modify Data

```
UPDATE CUSTOMERS
SET SALARY = 3000.00
WHERE ID = 1;
```

4.5 DELETE – Remove Data

```
DELETE FROM CUSTOMERS
WHERE ID = 1;
```

5 SQL Clauses (with Examples)

5.1 WHERE – Filter Rows

```sql
SELECT * FROM CUSTOMERS
WHERE AGE > 25;
```

5.2 ORDER BY – Sort Results

```sql
SELECT * FROM CUSTOMERS
ORDER BY SALARY DESC;
```

5.3 GROUP BY – Group Data

```sql
SELECT AGE, COUNT(*) FROM CUSTOMERS
GROUP BY AGE;
```

5.4 HAVING – Filter Groups

```sql
SELECT AGE, COUNT(*) FROM CUSTOMERS
GROUP BY AGE
HAVING COUNT(*) > 1;
```

5.5 LIKE – Pattern Matching

```sql
SELECT * FROM CUSTOMERS
WHERE NAME LIKE 'K%';
```

5.6 BETWEEN – Range Filter

```sql
SELECT * FROM CUSTOMERS
WHERE SALARY BETWEEN 2000 AND 5000;
```

5.7 IN – Match Multiple Values

```sql
SELECT * FROM CUSTOMERS
WHERE AGE IN (22, 25, 30);
```

---

6 SQL Joins (Simple Overview)

Type	Description
INNER JOIN	Returns records with matching values in both tables
LEFT JOIN	All records from left table, and matching ones from right
RIGHT JOIN	All records from right table, and matching ones from left
FULL JOIN	All records from both tables

Example:

```
SELECT A.ID, A.NAME, B.AMOUNT
FROM CUSTOMERS A
INNER JOIN ORDERS B ON A.ID = B.CUSTOMER_ID;
```

---

7 Constraints

Constraint	Description
NOT NULL	Column cannot be empty
UNIQUE	All values in column must be different
PRIMARY KEY	Uniquely identifies each row
FOREIGN KEY	Refers to primary key in another table
CHECK	Ensures values meet a condition
DEFAULT	Sets default value if none provided

---

8 Data Types (SQL Server Example)

Type	Example	Description
INT	25	Whole numbers
DECIMAL(p,s)	123.45	Fixed-point numbers
VARCHAR(n)	‘Hello’	Variable-length text
DATE	‘2024-01-01’	Date
DATETIME	‘2024-01-01 12:00:00’	Date and time

Practice Tips

- Start with `SELECT`, `WHERE`, `ORDER BY`
- Create small tables and try `INSERT`, `UPDATE`, `DELETE`
- Practice JOINS with 2 simple tables
- Write queries to filter, group, and sort data

- Always test and read your output carefully

---

9 Summary

- SQL helps manage and query data stored in relational databases
- Core commands: `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- Clauses like `WHERE`, `GROUP BY`, and `JOIN` add power
- Understanding table structure is key to writing effective queries

  Keep practicing small queries and build your confidence!