Anshum Banga
www.linkedin.com/in/anshumbanga

SQL Notes

Contents

Anshum Banga
www.linkedin.com/in/anshumbanga

What is MySQL?

MySQL is an open-source Relational Database Management System (RDBMS).

Stores data in tables (rows and columns).

Uses SQL (Structured Query Language) for querying and managing data.

My SQL Command Categories

1. DDL (Data Definition Language)

Command	**Purpose**	**Example**
CREATE	Creates a new database or table	CREATE TABLE students (id INT, name VARCHAR(100));
ALTER	Modifies structure of a table	ALTER TABLE students ADD age INT;
DROP	Deletes an existing table or database	DROP TABLE students;
TRUNCATE	Deletes all data but keeps table	TRUNCATE TABLE students;

2. DML (Data Manipulation Language)

Command	**Purpose**	**Example**
INSERT	Inserts data into a table	INSERT INTO students (id, name) VALUES (1, 'Anshum');
UPDATE	Updates existing data in a table	UPDATE students SET name = 'Raj' WHERE id = 1;
DELETE	Deletes data from a table	DELETE FROM students WHERE id = 1;

3. DQL (Data Query Language)

Command	Purpose	Example
SELECT	Fetches data from a table	SELECT * FROM students;
		SELECT name FROM students WHERE age > 20;

4. DCL (Data Control Language)

Command	Purpose	Example
GRANT	Gives privileges to a user	GRANT SELECT ON students TO 'user1'@'localhost';
REVOKE	Removes privileges from a user	REVOKE SELECT ON students FROM 'user1'@'localhost';

5. TCL (Transaction Control Language)

Command	Purpose	Example
COMMIT	Saves changes made during the transaction	COMMIT;
ROLLBACK	Reverts changes made during the transaction	ROLLBACK;
SAVEPOINT	Creates a point to rollback to within a transaction	SAVEPOINT save1;
		ROLLBACK TO save1;

SQL Data Types

Numerical Data Types

Data Type	Category	Description	Range / Limit
INT / INTEGER	Numeric	Whole numbers	-2,147,483,648 to 2,147,483,647 (4 bytes)
SMALLINT	Numeric	Smaller whole numbers	-32,768 to 32,767 (2 bytes)
BIGINT	Numeric	Large whole numbers	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
DECIMAL(p,s)	Numeric	Fixed precision numbers	User-defined (p = total digits, s = digits after decimal)
NUMERIC(p,s)	Numeric	Synonym for DECIMAL	Same as above
FLOAT(p)	Numeric	Approximate floating-point	4 or 8 bytes (precision varies by DBMS)
REAL	Numeric	Approximate float (single precision)	~6 decimal digits precision
DOUBLE PRECISION	Numeric	Approximate float (double precision)	~15 decimal digits precision

Character Data types

Data Type	Category	Description	Range / Limit
CHAR(n)	String	Fixed-length string	0 to 255 characters (MySQL), up to 8000 (SQL Server)
VARCHAR(n)	String	Variable-length string	0 to 65,535 (MySQL), up to 8000 or MAX (SQL Server)
TEXT	String	Long text	Up to 65,535 (MySQL TEXT), 2 GB in others

Date Time Data Types

Data Type	Category	Description	Range / Limit
DATE	Date/Time	Date only (YYYY-MM-DD)	1000-01-01 to 9999-12-31 (MySQL)
TIME	Date/Time	Time only (HH:MM:SS)	'-838:59:59' to '838:59:59' (MySQL)
DATETIME	Date/Time	Date and time	1000-01-01 00:00:00 to 9999-12-31 23:59:59
TIMESTAMP	Date/Time	Date and time with timezone	1970-01-01 to 2038-01-19 (MySQL)
YEAR	Date/Time	Year (2 or 4 digits)	1901 to 2155 (MySQL)

Boolean Data Types

Data Type	Category	Description	Range / Limit
BOOLEAN	Boolean	True/False	0 (false), 1 (true); often stored as TINYINT

Binary Data Types

Data Type	Category	Description	Range / Limit
BINARY(n)	Binary	Fixed-length binary data	Up to 255 bytes
VARBINARY(n)	Binary	Variable-length binary	Up to 65,535 bytes
BLOB	Binary	Binary large object	TinyBlob (255), Blob (64 KB), MediumBlob (16 MB), LongBlob (4 GB)

Other Data Types

Data Type	Category	Description	Range / Limit
ENUM	Custom/String	Predefined string list	Up to 65,535 distinct values (MySQL only)
JSON	Semi-structured	Stores JSON data	Up to 1 GB in PostgreSQL, 64 KB in MySQL
UUID	Identifier	Universal unique identifier	128-bit, e.g., 550e8400-e29b-41d4-a716-446655440000

SIGNED and UNSIGNED

Type	Meaning	Range Example (INT)
SIGNED	Allows **both negative and positive values** (default behavior)	-2,147,483,648 to +2,147,483,647
UNSIGNED	Allows **only positive values** (doubles upper limit)	0 to 4,294,967,295

SQL constraints

Constraint	Description	Syntax Example
NOT NULL	Ensures a column **cannot have NULL values**	name VARCHAR(100) NOT NULL
UNIQUE	Ensures all values in a column are **unique/different**	email VARCHAR(100) UNIQUE
		ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE(email);
PRIMARY KEY	Uniquely identifies each row (**NOT NULL + UNIQUE**)	id INT PRIMARY KEY
		PRIMARY KEY (id, roll_number) *(composite key)*
FOREIGN KEY	Creates a link between two tables; maintains **referential integrity**	FOREIGN KEY (student_id) REFERENCES students(id)
		ALTER TABLE orders ADD CONSTRAINT fk_student FOREIGN KEY(student_id) REFERENCES students(id);
CHECK	Ensures data in a column **meets a condition**	salary INT CHECK (salary > 0)
		ALTER TABLE employees ADD CONSTRAINT chk_salary CHECK (salary > 0);
DEFAULT	Sets a **default value** if none is provided	status VARCHAR(20) DEFAULT 'pending'
AUTO_INCREMENT	Automatically increases numeric value by 1 (mostly for primary key)	id INT AUTO_INCREMENT PRIMARY KEY *(MySQL specific)*

MySQL JOINs

🔸 **What is a JOIN?**

- A JOIN is used to combine rows from **two or more tables** based on a **related column** between them.
- Usually, JOINs are used with **primary key–foreign key** relationships.

🧩 **Tables for Example**

📋 **students Table**

student_id	name	city
1	Anshum	Delhi
2	Raj	Mumbai
3	Priya	Jaipur

📋 **marks Table**

student_id	subject	marks
1	Math	90
1	Science	85
2	Math	70
4	Science	60

1. INNER JOIN

Feature	Details
Purpose	Returns records **with matching values in both tables**
Non-matching?	Rows without a match are **excluded**
Syntax	SELECT * FROM A INNER JOIN B ON A.id = B.id;

```sql
SELECT students.name, marks.subject, marks.marks
FROM students
INNER JOIN marks
ON students.student_id = marks.student_id;
```

📊 **Output:**

name	subject	marks
Anshum	Math	90
Anshum	Science	85
Raj	Math	70

2. LEFT JOIN (or LEFT OUTER JOIN)

Feature	Details
Purpose	Returns **all rows from the left table**, and matched rows from right
Non-matching?	If no match, returns **NULL** for right table's columns
Syntax	SELECT * FROM A LEFT JOIN B ON A.id = B.id;

```sql
SELECT students.name, marks.subject, marks.marks
FROM students
LEFT JOIN marks
ON students.student_id = marks.student_id;
```

📊 **Output:**

name	subject	marks
Anshum	Math	90
Anshum	Science	85
Raj	Math	70
Priya	NULL	NULL

3. RIGHT JOIN (or RIGHT OUTER JOIN)

Feature	Details
Purpose	Returns **all rows from the right table**, and matched from the left
Non-matching?	If no match, returns **NULL** for left table's columns
Syntax	SELECT * FROM A RIGHT JOIN B ON A.id = B.id;

```sql
SELECT students.name, marks.subject, marks.marks
FROM students
RIGHT JOIN marks
ON students.student_id = marks.student_id;
```

📊 **Output:**

name	subject	marks
Anshum	Math	90
Anshum	Science	85
Raj	Math	70
NULL	Science	60

4. FULL JOIN (or FULL OUTER JOIN) ⚠ *(Not natively supported in MySQL)*

Feature	Details
Purpose	Returns **all rows from both tables**, and matches where possible
Non-matching?	Returns **NULL** where there is no match in either left or right
Workaround	Simulate using UNION of LEFT JOIN and RIGHT JOIN

sql Copy Edit

```sql
SELECT students.name, marks.subject, marks.marks
FROM students
LEFT JOIN marks ON students.student_id = marks.student_id

UNION

SELECT students.name, marks.subject, marks.marks
FROM students
RIGHT JOIN marks ON students.student_id = marks.student_id;
```

Output:

name	subject	marks
Anshum	Math	90
Anshum	Science	85
Raj	Math	70
Priya	NULL	NULL
NULL	Science	60

Summary Table

JOIN Type	Returns...	Missing Matches Shown as NULL in...
INNER JOIN	Matching rows from both tables	None
LEFT JOIN	All from left + matching from right	Right table
RIGHT JOIN	All from right + matching from left	Left table
FULL JOIN	All from both tables	Both tables