**Commonly used DDL commands:**

| **Command** | **Action** | **Basic Syntax** |
|---|---|---|
| CREATE | Defines new database objects. | CREATE TABLE table_Name(Column1 datatype, column2 datatype ); |
| ALTER | Modifies existing database objects. | ALTER TABLE table_Name ADD column_Name datatype; |
| DROP | Deletes existing database objects. | DROP TABLE table_Name; |
| TRUNCATE | Remove all rows from a table | TRUNCATE TABLE table_Name; |
| RENAME | Changes the name of a database object | RENAME TABLE old_Table_Name TO newTableName |
| CREATE INDEX | Creates an index on one or more columns in a table to improve performance. | CREATE INDEX indexName ON tableName; |
| DROP INDEX | Deletes an existing index from a table | DROP INDEX indexName ON table_Name; |
| CREATE VIEW | Creates a virtual table based on the result of a SQL query | CREATE VIEW viewName AS SELECT columns FROM tableName WHERE condition; |
| DROP VIEW | Deletes an existing view from the database | DROP VIEW viewName; |
| CREATE DATABASE | Creates a new database | CREATE DATABASE database_name; |
| DROP DATABASE | Deletes an existing Database along with all its table and data | DROP DATABASE database_name; |
| ALTER DATABASE | Modifies the properties of the database | ALTER DATABASE database_name MODIFY….; |
| COMMENT ON | Adds a comment to a database object | COMMENT ON TABLE table)name IS ‘description’;<br>COMMENT ON COLUMN table_name.column_name IS ‘description;’ |
| ALTER VIEW | Modifies the structure of an existing view | ALTER VIEW view_name AS SELECT….; |

**Common SQL Data Types:**

| **Category** | **Data type** | **Description** | **Example** |
|---|---|---|---|
| **Numeric** | INT | Integer Number (Whole number) | 123, -123 |
| | DECIMAL(p,s) | Fixed-point decimal with precision ‘p’ and scale ‘s’ | 123.12 |
| | FLOAT | Approximate floating point number | 123.456,-0.001 |
| | DOUBLE | Larger floating-point numbers | 123456.789 |
| | BIGINT | Large integer values | 16448949846466 |
| | SMALLINT | Small integer value | -12345, 12345 |
| **String** | VARCHAR(n) | Variable-length string up to ‘n’ characters | ‘Hello world’, ‘SQL’ |
| | CHAR(n) | Fixed-length string with exactly ‘n’ characters | ‘Hello’ |
| | TEXT | Large text data | ‘This is a long description’ |
| | BLOB | The binary large object for storing binary data(e.g images, files) | binary data |
| **Date/Time** | DATE | Stores date(yyyy-mm-dd) | 2024-10-8 |
| | TIME | Stores time (HH: MM) | 12:30 |
| | DATETIME | Store date and time (YYYY-MM-DD HH: MM) | 2024-10-8 12:30 |
| | TIMESTAMP | Stores timestamp; auto-updated when rows are modified | 2024-10-8 12:30 |
| **Boolean** | Boolean | Stores TRUE or FALSE values | TRUE, FALSE |
| **Miscellaneous** | ENUM | stores one value from a predefined list of values | (‘small’,’ medium’, ‘large’) |
| | JSON | stores JSON-formatted data | {“key”:” value”} |
| | UUID | Universal Unique Identifier | “550e8400-e29b-41d4-a716-446655440000’” |

**Common SQL Constraints**

| Constraint | Description | Example |
|---|---|---|
| **PRIMARY KEY** | Uniquely identifies each record in a table. Ensures uniqueness and not null | id INT PRIMARY KEY |
| **FOREIGN KEY** | Establishes a relationship between two tables by linking one table’s column to another table's primary key | FOREIGN KEY (dept_id) REFERENCES departments(id) |
| **NOT NULL** | Ensures that a column cannot have NULL values | name VARCHAR(50) NOT NULL |
| **UNIQUE** | Ensures that all values in a column are different(no duplicates) | email VARCHAR(100) UNIQUE |
| **CHECK** | Ensures that all values in a column satisfy a specific condition | salary DECIMAL(10,2) CHECK (salary >0) |
| **DEFAULT** | Sets a default value for a column if no value is specified | status VARCHAR(10) DEFAULT ‘pending’ |
| **AUTO_INCREAMENT** | Automatically generates a unique number for each new row(used with PRIMARY KEY) | id INT AUTO_INCREAMENT PRIMARY KEY |

**JavaScript mode command(\js):**

| **Commands** | **Description** |
|---|---|
| session = mysql.getSession() | Create a session for running the javascript commands |
| session.getSchemas() | List all databases(schemas) |
| db = session.getSchema(“db_name”) | Select specific database. |
| table = db.getTable(“table_name”) | Select a table |
| table.select().execute() | Query and return data from a table |
| table.insert([columns]).values([values]) | Insert values into a table |
| table.delete().where(‘condition’).execute() | Delete rows that match a condition |

| | |
|---|---|
| session.runSql(“SQL_query”) | Run a raw SQL query using Javascript |

**Python Mode commands(\py):**

| **Commands** | **Description** |
|---|---|
| session = mysql.get_Session() | Create a session for running the Python commands |
| session.get_Schemas() | List all databases(schemas) |
| db = session.get_Schema(“db_name”) | Select specific database. |
| table = db.get_Table(“table_name”) | Select a table |
| table.select().execute() | Query and return data from a table |
| table.insert([columns]).values([values]) | Insert values into a table |
| table.delete().where(‘condition’).execute() | Delete rows that match a condition |
| session.run_sql(“SQL_query”) | Run a raw SQL query using Python |

**Aggregation Functions:**.

| Function | Description |
|---|---|
| COUNT() | Returns the number of rows that match a specified condition |
| SUM() | Returns the total sum of numeric column |
| AVG() | Returns the average value of a numeric column |
| MIN() | Returns the smallest value of a column |
| MAX() | Returns the largest value of a column |
| GROUP_CONCAT() | Concatenates values from multiple rows into a single string |

**Functions:**

| **Function Type** | **Function Name** | **Description** | **Example** |
|---|---|---|---|
| String Function | CONCAT | Combines two or more strings into one | SELECT CONCAT(‘Hello, ‘World!”); |
| | SUBSTRING | Extracts a part of a string | SELECT SUBSTRING(‘Hello, world’, 1,5); |

| | | | |
|---|---|---|---|
| | LENGTH | Returns the length of a string | SELECT LENGTH(‘WORD’); |
| | TRIM | Removes leading/trailing spaces from a string | SELECT TRIM(‘WORD’); |
| | UPPER | Converts a string to a uppercase. | SELECT UPPER (‘word’); |
| | LOWER | Converts a string to a lowercase. | SELECT LOWER (‘WORD’); |
| | REPLACE | Replaces all occurrences of a substring with another substring. | SELECT REPLACE(‘abc def’, ‘abc’, ‘xyz’); |
| Numeric Functions | ROUND | Rounds a number to the specified number of decimal places. | SELECT ROUND (123.456,2); |
| | ABS | Returns the absolute value of a number | SELECT ABS(-45.67); |
| | MOD | Returns the remainder of a division. | SELECT MOD(10,3); |
| | CEIL | Returns the smallest integer greater than or equal to a number | SELECT CEIL(10.2); |
| | FLOOR | Returns the largest integer less than or equal to a number. | SELECT FLOOR(10.7); |
| | POWER | Raises a number to the power of another number. | SELECT POWER(2,3); |
| Date Functions | NOW | Returns the current date and time. | SELECT NOW(); |
| | CURDATE | Returns the current date. | SELECT CURDATE(); |
| | DATE_ADD | Adds a specified time interval to a date | SELECT DATE_ADD(‘2023-01-1’, INTERVAL 5 DAY); |
| | DATEDIFF | Returns the difference between two dates. | SELECT DATEDIFF('2023-02-01', '2023-01-01'); |

| | | | |
|---|---|---|---|
| | YEAR | Extracts the year from a date | SELECT YEAR('2023-10-10'); |
| | MONTH | Extracts the month from a date. | SELECT MONTH('2023-10-10'); |
| | DAY | Extracts the day from a date. | SELECT DAY('2023-10-10'); |
| Aggregate Functions | COUNT | Returns the number of a row in a result set | SELECT COUNT(*)FROM employees; |
| | SUM | Returns the sum of a numeric column | SELECT SUM(salary) FROM employees; |
| | AVG | Returns the average value of a numeric column | SELECT AVG(salary) FROM employees; |
| | MIN | Returns the minimum value in a column | SELECT MIN(salary) FROM employees; |
| | MAX | Returns the maximum value in a column | SELECT MAX(salary) FROM employees; |
| Control Flow Functions | COALESCE | Returns the first non-null value from a list of expressions | SELECT COALESCE(NULL, 'No Data', 'Available'); |
| | CASE | Performs conditional logic (if-else) in SQL | SELECT CASE WHEN age > 18 THEN 'Adult' ELSE 'Minor' END; |
| | IFNULL | Returns the first argument if its not NULL, otherwise returns the second argument | SELECT IFNULL(salary, 0); |
| Mathematical Functions | SQRT | Returns the square root of a number | SELECT SQRT(16); |
| | LOG | Returns the natural logarithm of a number | SELECT LOG(10); |
| | EXP | Returns the exponential value of a number. | SELECT EXP(2); |

|  |  |  |  |
|---|---|---|---|
| Conversion Functions | CAST | Converts a value from one data type to another | SELECT CAST(123 AS CHAR); |
|  | CONVERT | Converts a value from one data type to another (similar to CAST) | SELECT CONVERT(123, CHAR); |
| JSON Functions | JSON_OBJECT | Creates a JSON object key-value pairs. | SELECT JSON_OBJECT('name', 'John', 'age', 30); |
|  | JSON_ARRAY | Creates a JSON array from a list of values. | SELECT JSON_ARRAY('apple', 'banana', 'cherry'); |
|  | JSON_EXTRACT | Extracts data from a JSON document | SELECT JSON_EXTRACT('{"name": "John"}', '$.name'); |
|  | JSON_UNQUOTE | Unquotes a JSON value | SELECT JSON_UNQUOTE('{"name": "John"}'); |
| User-Defined Functions(UDF) | CREATE FUNCTION | Allows creating custom functions for specific use cases. | CREATE FUNCTION AddNumbers(x INT, y INT) RETURNS INT RETURN x + y; |

**Key TCL commands:**

| TCL Command | Description | Use |
|---|---|---|
| COMMIT | This saves the changes made by the transaction permanently to the database | COMMIT; |
| ROLLBACK | Undoes the changes made by the transaction, reverting the database to its previous state | ROLLBACK; |
| SAVEPOINT | Sets a point within a transaction to which you can later roll back | SAVEPOINT savepoint_name; |
| SET TRANSACTION | Defines the properties of a transaction | SET TRANSACTION [READ ONLY / READ WRITE / ISOLATION LEVEL]; |