**1. Database Operations**

# a) Create a Database

To create a new database:

CREATE DATABASE my\_database;

**b) Use a Database**

To select a database to work with:

USE my\_database;

**c) Show Databases**

To list all databases:

SHOW DATABASES;

**2. Table Operations**

**a) Create a Table**

To create a table with specific columns and data types:

CREATE TABLE customer (

id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

### **b) Show Tables**

To display all tables in the current database:

SHOW TABLES;

**c) Describe Table Structure**

To see the structure of a table:

DESCRIBE customer;

**d) Drop a Table**

To delete a table:

DROP TABLE customer;

**3. Insert Data**

**a) Insert Single Row**

To insert one row of data:

INSERT INTO customer (first\_name, last\_name, email)

VALUES ('John', 'Doe', 'john.doe@example.com');

**b) Insert Multiple Rows**

To insert multiple rows:

INSERT INTO customer (first\_name, last\_name, email)

VALUES

('Jane', 'Smith', 'jane.smith@example.com'),

('Alice', 'Brown', 'alice.brown@example.com');

**4. Retrieve Data (SELECT Queries)**

**a) Select All Columns**

To retrieve all columns from a table:

SELECT \* FROM customer;

**b) Select Specific Columns**

To retrieve specific columns:

SELECT first\_name, email FROM customer;

**c) Filter Data with WHERE**

To retrieve data based on a conditon:

SELECT \* FROM customer

WHERE first\_name = 'John';

**d) Use Logical Operators**

Combine conditions using AND or OR:

SELECT \* FROM customer

WHERE first\_name = 'John' AND email LIKE '%example.com';

**e) Order Data**

To sort results by a column:

SELECT \* FROM customer

ORDER BY first\_name ASC; -- Sort in ascending order

**5. Update Data**

**a) Update Specific Rows**

To update data in a table:

UPDATE customer

SET email = 'john.new@example.com'

WHERE first\_name = 'John';

**b) Update All Rows**

To update all rows (use with caution!):

UPDATE customer

SET last\_name = 'Updated';

**6. Delete Data**

**a) Delete Specific Rows**

To delete rows based on a condition:

DELETE FROM customer

WHERE first\_name = 'John';

**b) Delete All Rows**

To delete all data in a table (but keep the table):

DELETE FROM customer;

**7. Table Modifications**

**a) Add a Column**

To add a new column to an existing table:

ALTER TABLE customer

ADD phone\_number VARCHAR(15);

**b) Modify a Column**

To change the data type or constraints of a column:

ALTER TABLE customer

MODIFY phone\_number BIGINT;

**c) Drop a Column**

To delete a column:

ALTER TABLE customer

DROP COLUMN phone\_number;

**8. Constraints**

**a) Add a Primary Key**

To add a primary key to an existing table:

ALTER TABLE customer

ADD CONSTRAINT pk\_customer\_id PRIMARY KEY (id);

**b) Add a Unique Key**

To add a unique constraint:

ALTER TABLE customer

ADD CONSTRAINT unique\_email UNIQUE (email);

**9. Joins**

**a) Inner Join**

Retrieve matching rows from two tables:

SELECT orders.order\_id, customer.first\_name, customer.last\_name

FROM orders

INNER JOIN customer ON orders.customer\_id = customer.id;

**b) Left Join**

Retrieve all rows from the left table, even if there is no match in the right table:

SELECT customer.first\_name, orders.order\_id

FROM customer

LEFT JOIN orders ON customer.id = orders.customer\_id;

**c) Right Join**

Retrieve all rows from the right table, even if there is no match in the left table:

SELECT customer.first\_name, orders.order\_id

FROM customer

RIGHT JOIN orders ON customer.id = orders.customer\_id;

**10. Aggregate Functions**

**a) Count**

Count the number of rows:

SELECT COUNT(\*) AS total\_customers FROM customer;

**b) Sum**

Calculate the total of a column:

SELECT SUM(order\_total) AS total\_sales FROM orders;

**c) Group By**

Group rows by a column and perform aggregation:

SELECT first\_name, COUNT(\*) AS order\_count

FROM customer

JOIN orders ON customer.id = orders.customer\_id

GROUP BY first\_name;

**d) Having Clause**

Filter grouped data:

SELECT first\_name, COUNT(\*) AS order\_count

FROM customer

JOIN orders ON customer.id = orders.customer\_id

GROUP BY first\_name

HAVING order\_count > 1;

**11. Advanced Features**

**a) Subqueries**

Use a query inside another query:

sql

Copy code

SELECT first\_name, last\_name

FROM customer

WHERE id IN (

SELECT customer\_id FROM orders WHERE order\_total > 500

);

**b) Create Views**

Create a virtual table:

CREATE VIEW customer\_orders AS

SELECT customer.first\_name, orders.order\_total

FROM customer

JOIN orders ON customer.id = orders.customer\_id;

**c) Indexes**

Improve query performance:

sql

Copy code

CREATE INDEX idx\_customer\_email ON customer (email);