MySQL Commands and Queries

```
-- Show databases
SHOW DATABASES;

-- Create a new database
CREATE DATABASE advance_java;

-- Use the new database
USE advance_java;

-- Show tables
SHOW TABLES;

-- Create a new table
CREATE TABLE emp(
    id INT,
    name VARCHAR(50),
    salary INT
);

-- Select all records from the table
SELECT * FROM emp;

-- Insert data into the table
INSERT INTO emp VALUES(1, 'abc', 1000);
INSERT INTO emp(id, name) VALUES(2, 'xyz');

-- Update data in the table
UPDATE emp SET salary = 5500 WHERE id = 1;

-- Delete data from the table
DELETE FROM emp WHERE id = 2;

-- Select specific records
SELECT * FROM emp WHERE id = 1;
SELECT id FROM emp;
SELECT id, name FROM emp;
SELECT name, id FROM emp;

-- Drop the table and database
DROP TABLE emp;
DROP DATABASE advance_java;

-- Alter table to add a new column
ALTER TABLE emp ADD salary INT;

-- Primary Key with table creation
CREATE TABLE emp(
    id INT PRIMARY KEY,
    name VARCHAR(50),
```

```sql
    salary INT
);

-- Primary Key after table creation
CREATE TABLE emp(
    id INT,
    name VARCHAR(50),
    salary INT
);
ALTER TABLE emp ADD PRIMARY KEY(id);

-- Foreign key without ON UPDATE CASCADE & ON DELETE CASCADE
CREATE TABLE emp(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    salary INT
);
CREATE TABLE dept(
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);
ALTER TABLE emp ADD dept_id INT;
ALTER TABLE emp ADD FOREIGN KEY(dept_id) REFERENCES dept(id);

-- Drop tables
DROP TABLE emp;
DROP TABLE dept;

-- Foreign key with ON UPDATE CASCADE & ON DELETE CASCADE
CREATE TABLE emp(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    salary INT
);
CREATE TABLE dept(
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);
ALTER TABLE emp ADD dept_id INT;
ALTER TABLE emp ADD FOREIGN KEY(dept_id) REFERENCES dept(id) ON UPDATE CASCADE ON
DELETE CASCADE;

-- Drop tables
DROP TABLE emp;
DROP TABLE dept;

-- Foreign key in single SQL query with ON UPDATE CASCADE & ON DELETE CASCADE
CREATE TABLE dept(
    id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);
CREATE TABLE emp(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    salary INT,
    dept_id INT,
    FOREIGN KEY(dept_id) REFERENCES dept(id) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Use the advance_java database
USE advance_java;

-- Drop tables
DROP TABLE emp;
DROP TABLE dept;

-- Create the emp table
CREATE TABLE emp(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    salary INT
);

-- Aggregate functions
SELECT MAX(salary) FROM emp;
SELECT MIN(salary) FROM emp;
SELECT SUM(salary) FROM emp;
SELECT COUNT(salary) FROM emp;
SELECT COUNT(*) FROM emp;

-- Order by
SELECT * FROM emp;
SELECT * FROM emp ORDER BY salary;
SELECT * FROM emp ORDER BY salary DESC;
SELECT * FROM emp ORDER BY name;
SELECT * FROM emp ORDER BY name DESC;

-- Select with conditions
SELECT * FROM emp WHERE id = 1;
SELECT * FROM emp WHERE name = 'abc';
SELECT * FROM emp WHERE name = 'a';
SELECT * FROM emp WHERE name LIKE 'a';
SELECT * FROM emp WHERE name LIKE 'a%';

-- Limit results
SELECT * FROM emp LIMIT 0, 2;
SELECT * FROM emp LIMIT 2, 2;

-- Second highest salary
SELECT * FROM emp ORDER BY salary DESC LIMIT 1, 1;
SELECT MAX(salary) FROM emp WHERE salary < (SELECT MAX(salary) FROM emp);
SELECT * FROM emp WHERE salary = (SELECT MAX(salary) FROM emp WHERE salary < (SELECT
MAX(salary) FROM emp));
```

```sql
-- Aliases
SELECT emp.id AS emp_id, name, salary FROM emp;
SELECT emp.id AS emp_id, name AS emp_name, salary AS emp_salary FROM emp;
SELECT e.id AS emp_id, name AS emp_name, salary AS emp_salary FROM emp AS e;
SELECT e.id emp_id, name emp_name, salary emp_salary FROM emp e;
SELECT e.id AS emp_id, name AS emp_name, salary AS emp_salary, d.id AS dept_id,
d.dept_name AS dept_name FROM emp AS e LEFT JOIN dept AS d ON e.id = d.id;

-- Joins
SELECT * FROM emp;
SELECT * FROM dept;
SELECT * FROM emp INNER JOIN dept ON emp.id = dept.id;
SELECT * FROM emp LEFT JOIN dept ON emp.id = dept.id;
SELECT * FROM emp RIGHT JOIN dept ON emp.id = dept.id;
SELECT * FROM emp LEFT JOIN dept ON emp.id = dept.id UNION SELECT * FROM emp RIGHT
JOIN dept ON emp.id = dept.id;
SELECT * FROM emp LEFT JOIN dept ON emp.id = dept.id WHERE dept.id IS NULL UNION
SELECT * FROM emp RIGHT JOIN dept ON emp.id = dept.id WHERE emp.id IS NULL;

-- Testing queries
SELECT * FROM emp LEFT JOIN dept ON emp.id = dept.id;
SELECT * FROM emp LEFT JOIN dept ON emp.id = dept.id WHERE dept.id IS NULL;
SELECT * FROM emp RIGHT JOIN dept ON emp.id = dept.id;
SELECT * FROM emp RIGHT JOIN dept ON emp.id = dept.id WHERE emp.id IS NULL;
SELECT * FROM emp LEFT JOIN dept ON emp.id = dept.id WHERE dept.id IS NULL UNION
SELECT * FROM emp RIGHT JOIN dept ON emp.id = dept.id WHERE emp.id IS NULL;

-- Marks and total calculation
SELECT * FROM marksheet;
SELECT *, (physics + chemistry + maths) AS total FROM marksheet;
SELECT *, (physics + chemistry + maths) AS total FROM marksheet ORDER BY total;
SELECT *, (physics + chemistry + maths) AS total FROM marksheet ORDER BY total DESC;
SELECT *, (physics + chemistry + maths) AS total FROM marksheet ORDER BY total DESC
LIMIT 0, 3;
SELECT *, (physics + chemistry + maths) AS total FROM marksheet WHERE physics >= 33
AND chemistry >= 33 AND maths >= 33 ORDER BY total DESC LIMIT 0, 3;

-- Group by
SELECT salary, COUNT(*) FROM emp GROUP BY salary;
```