MySQL Relational Database Cheatsheet

Author: Placements Simplified Editorial Team

Introduction: Structured Query Language (SQL)

MySQL is an open-source relational database management system (RDBMS) that stores data in structured tables linked by relationship keys. SQL is case-insensitive, but writing keywords in UPPERCASE is the industry standard for query readability.

1. Accessing Databases and Tables

List all databases in active server:

```
SHOW DATABASES;
```

Switch to a database:

```
USE database_name;
```

List all tables in active database:

```
SHOW TABLES;
```

2. Creating Tables & Constraints

Create a table with structural field types:

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT DEFAULT 18,
    email VARCHAR(100) UNIQUE,
    mentor_id INT,
    FOREIGN KEY (mentor_id) REFERENCES teachers(id)
);
```

3. CRUD - Reading and Writing Data

Inserting values into tables:

```
INSERT INTO students (name, age) VALUES ('Harry', 21);
```

Read rows (select queries):

```
SELECT * FROM students;
SELECT DISTINCT name FROM students;
SELECT name, age FROM students WHERE age > 18;
SELECT * FROM students WHERE age BETWEEN 18 AND 24;
SELECT * FROM students WHERE name LIKE 'Ha%';
```

Update field values:

```
UPDATE students SET age = 22 WHERE id = 1;
```

Delete rows:

```
DELETE FROM students WHERE id = 1;
```

4. Grouping & Ordering results

Sort records (ascending/descending):

```
SELECT * FROM students ORDER BY age DESC, name ASC;
```

Group records by attribute value with conditions:

```
SELECT age, COUNT(*) FROM students GROUP BY age HAVING COUNT(*) > 5;
```

5. SQL Joins

Joins match records across foreign keys:

```
-- Inner Join (Intersection matches)
SELECT s.name, t.name
FROM students s
INNER JOIN teachers t ON s.mentor_id = t.id;

-- Left Join (All left students + matched teachers)
SELECT s.name, t.name
FROM students s
LEFT JOIN teachers t ON s.mentor_id = t.id;

-- Full Outer Join (MySQL Workaround using UNION)
SELECT s.name, t.name FROM students s LEFT JOIN teachers t ON s.mentor_id = t.id
UNION
SELECT s.name, t.name FROM students s RIGHT JOIN teachers t ON s.mentor_id = t.id;
```

6. Indexes & Transactions

Create query indexes:

```
CREATE INDEX idx_name ON students(name);
DROP INDEX idx_name ON students;
```

Atomicity Transactions (ACID compliance):

```
START TRANSACTION;
UPDATE accounts SET balance = balance - 500 WHERE id = 1;
UPDATE accounts SET balance = balance + 500 WHERE id = 2;
COMMIT; -- Permanently save changes
-- Or on failure:
ROLLBACK; -- Undo all changes
```