**Coding Standards (MySQL)**  
**v1.0**  
**October 14, 2024**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Version Description

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 10/06/24 | RM and LA | Document Created |
| V1.0 | 10/14/24 | RM and LA | MySQL  Documentation Included |
| v2.0 |  |  |  |
| v3.0 |  |  |  |

# Contents of this Document

* General
* Naming Conventions
* Commenting Code
* Parenthesis
* Constants
* Line Spacing
* Braces
* Declarations

# General

MySQL will be used as the primary database management system for this project. Queries and database design should prioritize maintainability, readability, and efficiency. All database objects should follow naming conventions and be well-structured for scalability. This document outlines the basic standards for writing readable and maintainable MySQL queries. It covers conventions, structure, and practices that should be followed throughout the project to ensure consistency and clarity.  
  
Goals for code guidelines:  
1. Queries should be easy to read and understand.

2. Database structure should be easy to maintain.

3. Changes to the database scheme or queries should not require complete refactoring.

# Naming Conventions

**Table Names:** Use snake\_case for table names. Table names should be plural to indicate they hold multiple records.

**Example:**

CREATE TABLE users {

user\_id INT PRIMARY KEY,

username VARCHAR(50)

};

**Column Names:** Use snake\_case for columns names. Column names should be descriptive and easy to understand.

**Example:**

user\_id INT,

first\_name VARCHAR(50),

email\_address VARCHAR(100)

# Commenting Code

Comments should be used to explain the purpose of complex quieries, stored procedures or database triggers. Use – for single line comments and /\* \*/ for multi-line comments.

**Example:**

**-- Get the list of active users**

SELECT username

FROM users

/\* This query retrieves all orders placed in the last 30 days \*/

SELECT order\_id, order\_date

FROM orders

WHERE order\_date >= CURDATE() - INTERVAL 30 DAY;

# Parenthesis

Parenthesis should always be used in complex queries to ensure the intended order of operations.

**Example:**  
SELECT \*

FROM orders

WHERE (total\_price > 1000 AND order\_date > '2024-01-01') OR (customer\_id IN (SELECT customer\_id FROM vip\_customers));

NOTE: This example is more complex, which is why we need the use of parenthesis to make the order of statements more obvious.

# Constants

Constants should be used for values that do not change, such as default values, statuses, or fixed numerical values.

**Example:**

-- Default constant for maximum allowed price

SET @MAX\_PRICE = 1000;

SELECT \* FROM products WHERE price <= @MAX\_PRICE;

# Line Spacing

Queries should be formatted with proper line spacing to ensure readability. Use blank lines between different logical blocks such as SELECT, FROM, WHERE, and JOIN clauses.

**Example:**

SELECT u.username, o.order\_id, o.total\_price

FROM users u

JOIN orders o ON u.user\_id = o.user\_id

WHERE o.total\_price > 100;

-- Another example for complex query

SELECT product\_id, product\_name

FROM products

WHERE category\_id = 5

ORDER BY product\_name ASC;

# Braces

For conditions in queries, always use parentheses when needed to group logical conditions. This avoids confusion with operator precedence.

**Example:**

SELECT \*

FROM products

WHERE (category\_id = 1 AND price > 50)

OR (category\_id = 2 AND stock > 100);

# Declarations

All variables, constants, or default values should be declared with SET before using them in queries.

**Example:**

SET @MAX\_ITEMS = 100;

SELECT \*

FROM inventory

WHERE item\_count < @MAX\_ITEMS;

# Error Handling

When writing stored procedures or functions, use appropriate error handling mechanisms like declaring and handling exceptions with DECLARE and HANDLER.

DELIMITER //

CREATE PROCEDURE update\_price(IN product\_id INT, IN new\_price DECIMAL(10, 2))

BEGIN

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

SELECT 'Error: Could not update price';

UPDATE products

SET price = new\_price

WHERE id = product\_id;

END //

DELIMITER ;