SQL Coding Conventions & Best Practices

Following proper SQL conventions improves **readability, maintainability, and performance** of your database queries. Here are key conventions:

---

1. Naming Conventions

Proper naming ensures clarity and consistency.

Table Naming

- Use **plural** form for tables (indicates a collection of records).
- Use **snake_case** or **camelCase** consistently.

Example:

```
CREATE TABLE employees; -- Good ✅
CREATE TABLE EmployeeData; -- Avoid ❌
```

Column Naming

- Use **descriptive names**.
- Avoid spaces; use **snake_case** or **camelCase**.

Example:

```
CREATE TABLE employees (
   emp_id INT PRIMARY KEY,  -- Good ✅
   first_name VARCHAR(50),  -- Good ✅
   salary_amount DECIMAL(10,2)  -- Good ✅
);
```

Primary Key Naming

- Use {table_name}_id for primary keys.

Example:

```
CREATE TABLE customers (
   customer_id INT PRIMARY KEY  -- Good ✅
);
```

Foreign Key Naming

- Use {table1}_{table2}_id format.

Example:

```
CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   customer_id INT,
   FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

---

2. SQL Formatting

Well-formatted SQL improves readability.

Keywords in Uppercase

- SQL keywords should be **UPPERCASE**.

Example:

```
SELECT first_name, last_name FROM employees WHERE salary > 50000;
```

Use Indentation for Readability

- Break long queries into multiple lines.

Example:

```
SELECT first_name, last_name, salary
FROM employees
WHERE department = 'IT'
  AND salary > 50000
ORDER BY salary DESC;
```

---

3. Query Optimization

Efficient queries improve performance.

Use `LIMIT` When Fetching a Subset of Data

- Avoid fetching unnecessary data.

Example:

```
SELECT * FROM employees LIMIT 10;
```

Avoid `*`, Specify Columns

- Fetch only the required columns.

Example:

```
SELECT first_name, last_name, salary FROM employees;
```

Use Indexing

- Index frequently searched columns for faster retrieval.

Example:

```
CREATE INDEX idx_employees_lastname ON employees(last_name);
```

---

4. Joins & Relationships

Maintain **clear relationships** between tables.

Use Aliases in Joins

- Improves readability.

Example:

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

---

5. Use Constraints for Data Integrity

Constraints prevent incorrect data entry.

Example:

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    email VARCHAR(100) UNIQUE,  -- Prevent duplicate emails
    salary DECIMAL(10,2) CHECK (salary > 0)  -- Prevent negative salaries
);
```

---

6. Use Comments for Clarity

Comments help others understand your queries.

Example:

```
-- Fetch top 5 highest-paid employees
SELECT first_name, salary
FROM employees
ORDER BY salary DESC
LIMIT 5;
```

---

7. Avoid Hardcoding Values

Use **parameters** or **dynamic values**.

Example:

```
PREPARE stmt FROM 'SELECT * FROM employees WHERE department = ?';
```

🚀 Summary of SQL Best Practices

✅ Best Practice	❌ Avoid
Use **meaningful names** for tables/columns	table1, data1, valueX
Use **UPPERCASE for SQL keywords**	lowercase keywords
Use **snake_case** or **camelCase**	Spaces or mixed styles
Indent queries properly	Writing everything in one line
Use LIMIT to fetch fewer rows	Fetching all rows (SELECT *)
Use indexes for fast queries	Searching without indexing
**Specify column names instead of **``	Always using SELECT *
Use constraints (UNIQUE, CHECK, FOREIGN KEY)	Allowing bad data entry
Comment your SQL code	Unclear or undocumented queries

Would you like **practice exercises** or **real-world examples**? 😊