SQL & PL/SQL Style Guide

SQL & PL/SQL Style Guide

This guide provides best practices for writing readable and maintainable SQL and PL/SQL code.

1. SQL Case Conventions

- SQL keywords (SELECT, FROM, WHERE) should be in lowercase (modern best practice).
- Table and column names should be in lowercase (unless using camelCase or snake_case for readability).
- Use uppercase only for readability in legacy Oracle SQL scripts.

Example (Recommended Modern Style):

```
select first_name, last_name
from employees
where department_id = 10;
```

Example (Traditional Uppercase):

```
SELECT FIRST_NAME, LAST_NAME
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 10;
```

2. PL/SQL Case Conventions

- PL/SQL keywords (DECLARE, BEGIN, END) should be in uppercase for readability.
- SQL statements inside PL/SQL should follow SQL case conventions (preferably lowercase).
- Use camelCase or snake_case for variables.

Example:

```
DECLARE
    v_name VARCHAR2(50);
```

```
BEGIN
    SELECT first_name INTO v_name
    FROM employees
    WHERE employee_id = 101;

    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_name);
END;
/
```

3. Indentation & Formatting

- Indent SQL commands inside PL/SQL blocks for clarity.
- Align SQL clauses properly to improve readability.

Example (Good Formatting):

```
SELECT first_name, last_name, salary
FROM employees
WHERE department_id = 10
AND salary > 50000;
```

Example (Bad Formatting):

```
SELECT first_name,last_name,salary FROM employees WHERE department_id=10 AND salary>50000;
```

4. Naming Conventions

- Use singular names for table names (e.g., "Employee" instead of "Employees").
- Use snake_case or camelCase for column and variable names.
- Avoid reserved keywords for table and column names.

Example:

```
CREATE TABLE employee (
    employee_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50),
```

```
    last_name VARCHAR2(50)
);
```

5. Avoid Using Double Quotes for Identifiers

- Column and table names are case-insensitive by default.
- Using double quotes makes them case-sensitive and should be avoided.

Example (Not Recommended):

```
SELECT "FirstName" FROM "Employees";
```

6. Use Meaningful Aliases in Queries

- Use descriptive aliases instead of single letters.

Example (Recommended):

```
SELECT first_name AS employee_first_name, last_name AS employee_last_name
FROM employees;
```

Example (Not Recommended):

```
SELECT first_name AS f, last_name AS l
FROM employees;
```

7. Comment Your Code

- Use -- for single-line comments.
- Use /* ... */ for multi-line comments.

Example:

```
-- Get employees with high salaries
SELECT first_name, last_name, salary
FROM employees
WHERE salary > 100000;
```

8. Use MERGE Instead of Manual Insert/Update

- When inserting or updating, use MERGE for better efficiency.

Example:

```
MERGE INTO employees e
USING (SELECT 101 AS employee_id, 'John' AS first_name FROM dual) new_data
ON (e.employee_id = new_data.employee_id)
WHEN MATCHED THEN
    UPDATE SET e.first_name = new_data.first_name
WHEN NOT MATCHED THEN
    INSERT (employee_id, first_name) VALUES (new_data.employee_id, new_data.first_name);
```

Summary

- SQL: Prefer lowercase for readability.
- PL/SQL: Use uppercase for keywords, lowercase for SQL inside.
- Use proper indentation to improve readability.
- Follow naming conventions (no spaces, no reserved words).
- Comment your code for clarity.