A quick guide on common

SQL ERRORS

And how to resolve them

Lovee Kumar

In SQL, Errors can often occur due to various reasons

HERE'S A LIST OF SOME COMMON ONES AND HOW TO RESOLVE THEM

Lovee Kumar

Duplicate Values

Happens when you try to insert data with a primary key that already exists.

Error:

```
INSERT INTO table1 (id, name) VALUES (1,
'John');
```

If ID is a primary key and there's already a row with ID 1, this will fail. You need to insert a unique ID.

Solution:

```
INSERT INTO table1 (id, name) VALUES (2,
'John');
```

Null Values

Occurs when you try to perform operations on null values.

Error:

```
SELECT column1 + column2
FROM table1;
```

If column1 or column2 has a null value, it'll fail. You need to handle nulls, perhaps with **COALESCE**.

Solution:

```
SELECT COALESCE(column1, 0) +
COALESCE(column2, 0)
FROM table1;
```

Data Type Mismatch

Occurs when you try to insert or compare data of different data types.

Error:

```
SELECT column1
FROM table1
WHERE
column2 = '5';
```

Here, column2 is presumed to be an integer. So, the '5' should not be in quotes

Solution:

```
SELECT column1
FROM table1
WHERE
column2 = 5;
```

Case Sensitivity

SQL is case sensitive. If you incorrectly use uppercase or lowercase letters, it may lead to errors.

Error:

```
SELECT Column1
FROM table1;
```

If the column name is actually 'column1' and not 'Column1', this will fail. SQL is case sensitive.

Solution:

```
SELECT column1
FROM table1;
```

Incorrect Use Of Joins

Occurs when you use functions incorrectly or with the wrong data types.

Error:

```
SELECT *
FROM table1 JOIN table2;
```

You need to provide the join condition using the **ON** clause.

Solution:

```
SELECT *
FROM table1 JOIN table2 ON
table1.id = table2.id;
```

Inconsistent Naming Conventions

Occurs when you use inconsistent naming conventions for tables, columns, and other database objects.

Error:

```
SELECT columnOne
FROM table1;
SELECT column1
FROM table1;
```

Stick with a single naming convention to avoid confusion. Here, *columnOne* and *column1* are presumably the same, but named differently.

Solution:

```
SELECT column1
FROM table1;
```

Ambiguous Column Name

Happens when you reference a column name that exists in multiple tables.

Error:

```
SELECT id
FROM table1
JOIN table2 ON table1.id = table2.id;
```

If both table1 and table2 have a column named *id*, this will fail. You need to specify which table's *id* to select.

Solution:

```
SELECT table1.id
FROM table1
JOIN table2 ON
table1.id = table2.id;
```

Improper Use Of Function

Occurs when you use functions incorrectly or with the wrong data types.

Error:

```
SELECT SUBSTRING(column1, '1', '5')
FROM table1;
```

The second and third parameters of **SUBSTRING** should be integers.

Solution:

```
SELECT SUBSTRING(column1, 1, 5)
FROM table1;
```

Incorrect Use of Quotes

Occurs when you use the wrong type of quotes or forget to enclose text values in quotes.

Error:

```
SELECT *
FROM table1
WHERE name = "John";
```

In SQL, string literals should be enclosed in single quotes.

Solution:

```
SELECT *
FROM table1
WHERE name = 'John';
```

Incorrect Use Of Group By

Occurs when you use the GROUP BY clause incorrectly or with the wrong syntax

Error:

```
SELECT department_id, COUNT(*)
FROM employees
GROUP BY name;
```

Use the correct columns for the **GROUP BY** clause.

Solution:

```
SELECT department_id, COUNT(*)
FROM employees
GROUP BY department_id;
```

PRACTICE MAKES PERFECT

The exact behavior of these SQL statements may depend on the specific SQL variant and database system you are using

Don’t be afraid to make errors – nothing teaches better than making errors and debugging