SQL SERVER – USEFUL METADATA QUERIES

Metadata queries are really helpful in discovering information for a given database schema. Database information including the tables, views, columns names, data types, indexes, and table constraints are all available using queries such as these.

During this tutorial, I want to explore some useful metadata queries.

Let us start by finding the list of tables created in the given database.

```
select *
from information_schema.tables
where table_type='base table';
```

Now let us list the views created in the given database.

```
select *
from information_schema.tables
where table_type='view';
```

Let us create a query that lists the column names, data types, whether the column allows null or not, and the maximum allowed characters in the row.

```
select column_name, data_type, is_nullable,
character_maximum_length
from information_schema.columns
where table_name='emp';
```

This **query** shows the table name, object id, table creation date, and the last table modified time.

```
select name, object_id, create_date, modify_date
from sys.tables;
```

Listing the created indexes for a table with the column names is frequently required. In this query **a.name** is the table name for which you are listing the indexes. By removing the **a.name** condition, you can see all the created indexes in your database.

```
SELECT a.name table_name,
b.name index_name,
```

```
d.name column_name
FROM sys.tables a,
sys.indexes b,
sys.index_columns c,
sys.columns d
WHERE a.object_id = b.object_id
AND b.object_id = c.object_id
AND b.index_id = c.index_id
AND c.object_id = d.object_id
AND c.column_id = d.column_id
AND a.name = 'emp';
```

This query will list the defined constraints on tables with the column names. In thie example, we can see the emp table's unique, primary or foreign key constraints.

```
SELECT a.table_name,
a.constraint_name,
b.column_name,
a.constraint_type
FROM information_schema.table_constraints a,
information_schema.key_column_usage b
WHERE a.table_name = 'EMP'
AND a.table_name = b.table_name
AND a.table_schema = b.table_schema
AND a.constraint_name = b.constraint_name;
```

Suppose you want to write a ***'select count(1) from table_name'*** query for each table in your database, but you have more than 100 tables in your database. Instead of writing a separate query for each table, you can generate those queries using **SQL**. Therefore, you can write **SQL** code to generate **SQL**.

```
SELECT 'select count(1) from [' + table_name + '];'
FROM information_schema.tables;
```