



Ossynini Investment Nig. Ltd RC 629398

7 Awolowo Way Ikeja. Tel: 7300280, 08080264777, 08023128954



HEAD OFFICE:

St. Michael's Plaza, Opposite Oando Filling Station, Badore Road, Ajah-Lekki, Lagos.

BIN BINARY TECHNOLOGIES

SQL SERVER Master Plus

SQL SERVER

What is SQL?

SQL stands for structured query language. SQL is the standardized language used to access the database. SQL SERVER is a relational database management system that allows you to manage relational databases. It is an application software licensed by Microsoft Corporation.

SQL SERVER can run on various platforms UNIX, Linux, Windows, etc. You can install it on a server or even in a desktop. Besides, SQL SERVER is reliable, scalable, and fast.

What is DBMS?



DATABASE MANAGEMENT SYSTEM [DBMS]

Is an application used to manage databases. DBMS typically serves as an interface between the users or their applications and databases allowing users to insert, retrieve, update and manage how the information is structured and optimized.

SQL ENVIRONMENT

- Catalog: A set of schemas that constitute the description of a database
- Schema: The structure that contains descriptions of objects created by a user (base tables, views, constraints)
- Data Definition Language (DDL): Commands that define a database, including creating, altering, and dropping tables and establishing constraints
- Data Manipulation Language (DML): Commands that maintain and query a database
- Data Control Language (DCL): Commands that control a database, including administering privileges and committing data

INSTALLATION OF MYSQL SERVER

Microsoft Corporation produces SQL SERVER Developer & Express editions and the developer edition is a full-featured free edition, licensed for use as a development and test database in a non-production environment. We are going to install SQL SERVER 2019 Developer Edition on Windows Server 2019. It is provided for macOS, Ubuntu, SLES, RHEL and Windows operating systems.

You can download a SQL SERVER Setup for installation here:

<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

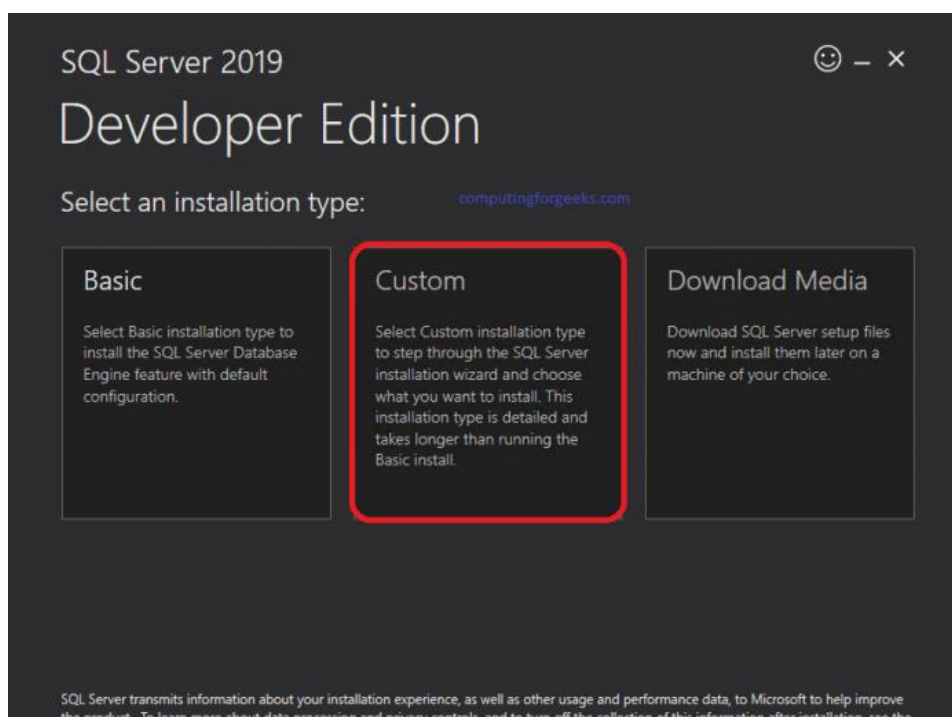
Step 1: Run Installed Application

Navigate to the directory/folder where you downloaded the application then double-click it to start the installation process.

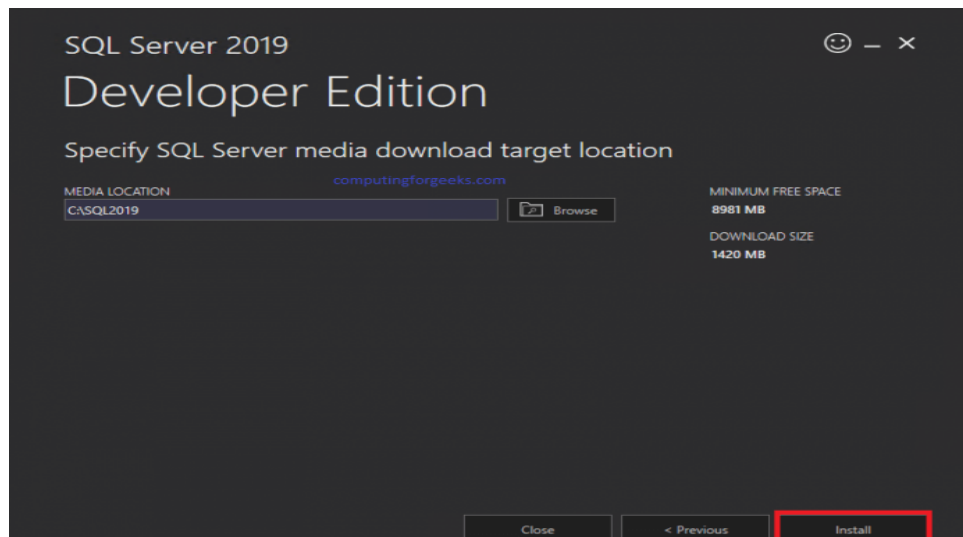
Step 2: Select Custom Installation

After you have chosen the custom installation, click on **"Install"**. Allow the application to download and install packages.

You have the *"Basic"*, *"Custom"* and *"Download Medium"* installation types. I recommend you choose *"Custom"* to tweak your installation of SQL Server 2019 Developer Edition on Windows Server 2019.

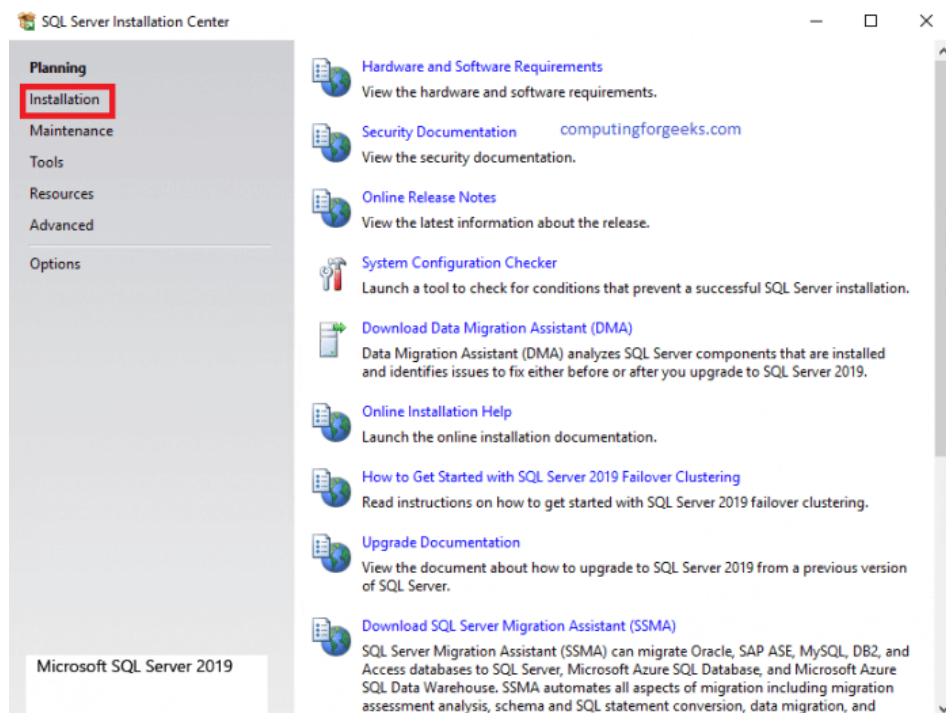


Specify a directory where SQL Server will store files downloaded for installation.



Step 3: Choose Installation

After all of the packages have downloaded, a new page as illustrated below will follow. Click on **“Installation”**

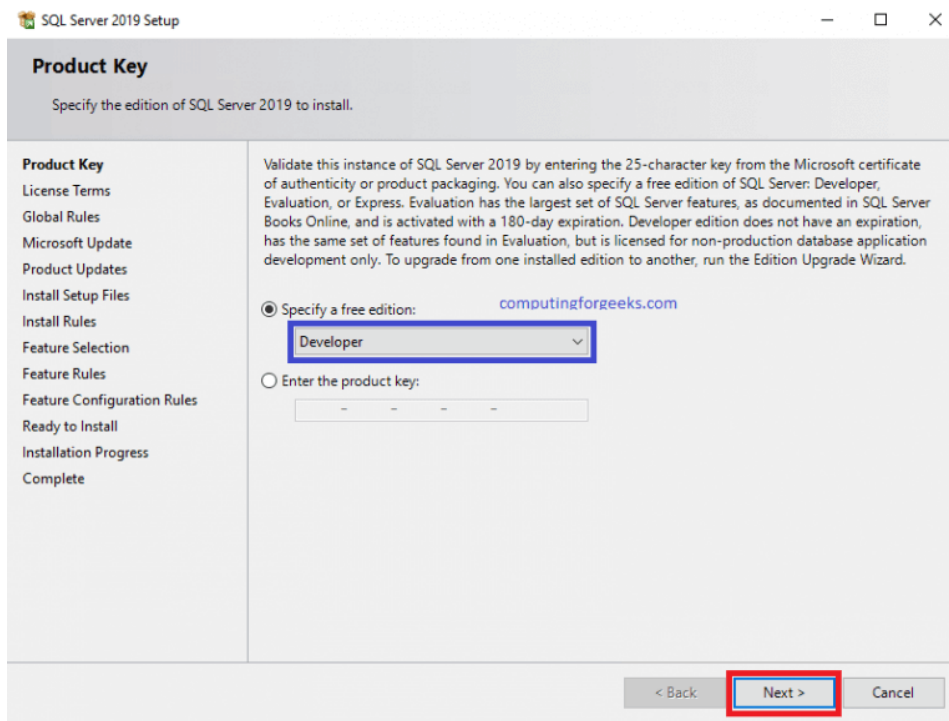


Step 4: Choose New SQL Server standalone

After Step 3, choose the first option in the list and proceed to the next step. This is illustrated below.

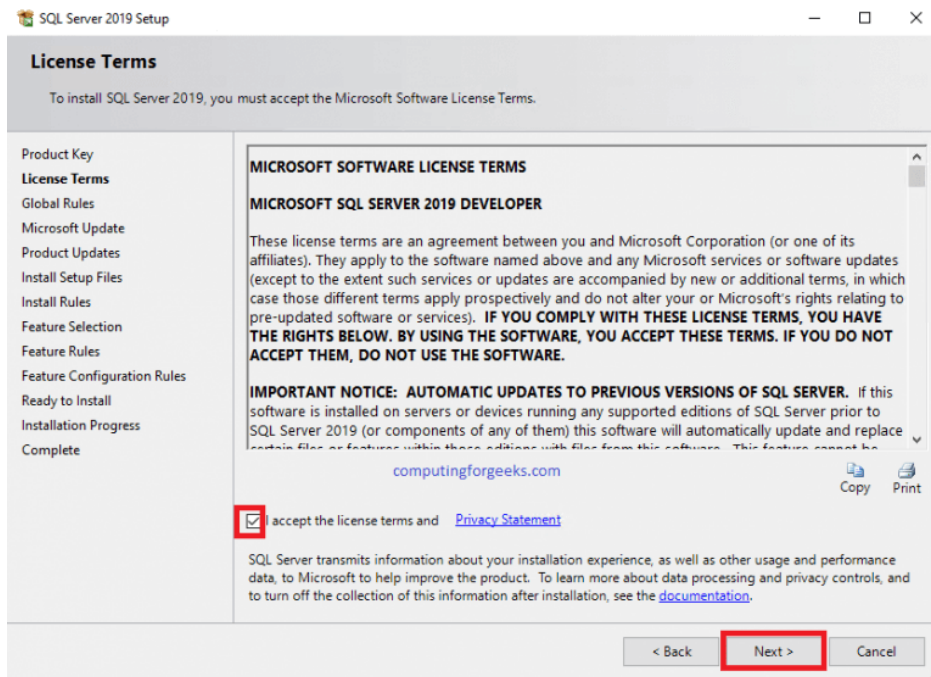
Step 5: Specify Edition

This step depends on whether you need **“Developer”**, **“Evaluation”** or **“Express”**. Their differences are described in the window. Since we are installing the **“Developer”** edition, let us proceed with that option. Click **“Next”**



Step 6: Accept License Terms

Select the **“I accept the license terms and Privacy Statement”** Radio and click on **“Next”**.



Step 7: Microsoft Update

You can either select the **“Microsoft update”** radio or not in this step then click **“Next”**

Step 8: Install Rules

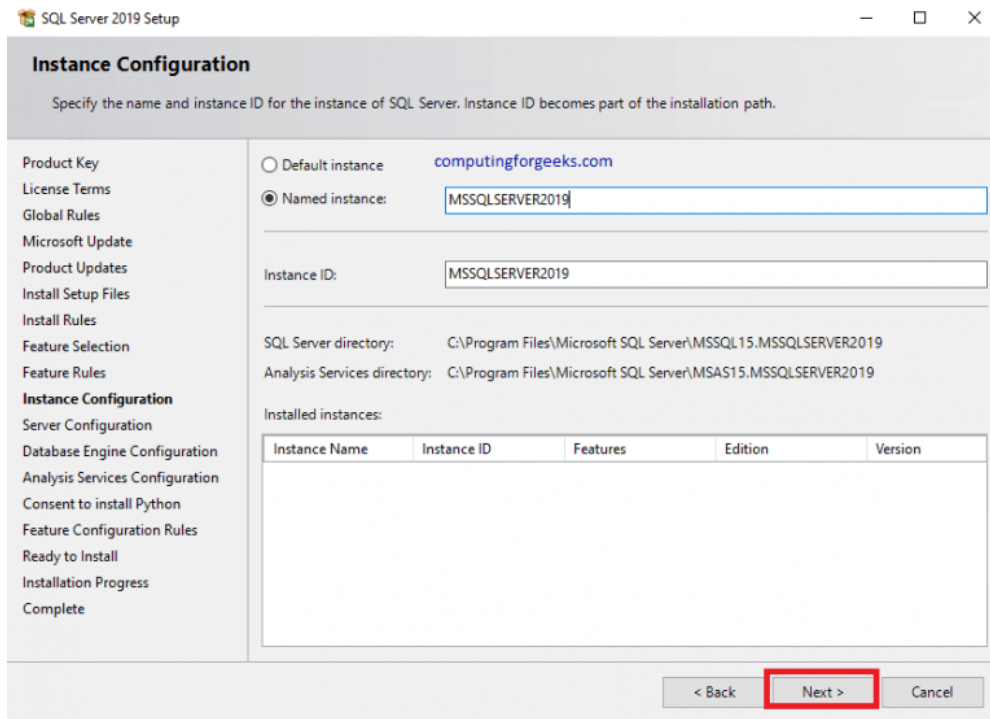
Windows will check the system to find out any potential problems that can occur while the setup is running. If your firewall is running, you might see a warning sign for **“Windows Firewall”** which reminds you to open appropriate ports after installation is complete. Click **“Next”** after you are satisfied with the results of that page.

Step 9: Feature Selection

In this step, you are presented with many features that SQL Server comes with. In case you would wish a certain feature to be included in your SQL instance, then this is the part that you choose their radio buttons. Go ahead and choose what you prefer to have. Click **“Next”** after you are done.

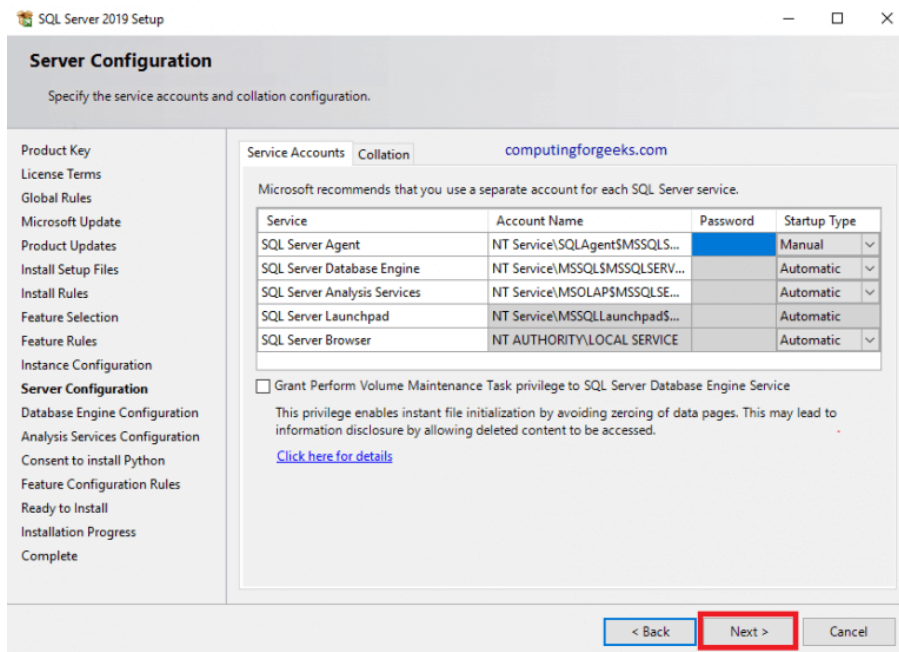
Step 10: Instance Configuration

Name your instance or leave the default instance in this step and click on **“Next”**.



Step 11: Server Configuration

Specify the service accounts and collation configuration here. Click "**Next**" after that.



Step 12: Configure Database Engine Parameters

This step gives you the opportunity to choose authentication mechanisms you will use, setting a password for the administrator account and choosing other admin accounts.

I prefer **mixed-mode authentication** where a user can either be from a domain or added manually in the SQL instance. Choose the one that suits you best, enter the password and any other setting that you would wish to configure.

If you are keen, you will notice that there are other Tabs on this page. They are "**Data Directories**" where you can specify where you would wish your logs to reside, where your root directory will be, where you would wish to place your backup directory and database directory.

If you have different partitions in your server, you can place these directories in a smart way. Other tabs include "**Memory**", "**TempDB**", "**MaxDOP**", and "**FILESTREAM**". Look into them and set your customized settings therein.

Click on "**Next**" when you are done. Do not forget to add an admin user.

Step 13: Analysis Services Configuration

Since I enabled analysis functionality, this is the step where I configure them. Choose an administrator and set the server mode that you prefer. Click on "**Next**".

Step 14: Accept Installation of Services

In case you had chosen specific packages to suit your needs in Step 9, you will be presented with installation guide and settings tailored for that specific service. For example, I had chosen Python and now I am being asked to consent its installation. After you are done with all of the prompts, click "**Next**" to proceed.

Step 15: Ready to Install

This step gives a summary of what we have been doing so far. The packages we have chosen to install, the settings we have configured and many more. Look at the list carefully. After you are satisfied with the summary, then gladly click on "**Install**"

Step 16: Finish up

After everything has installed. Click on "**Close**". Your SQL instance is installed and ready to be used.

Conclusion

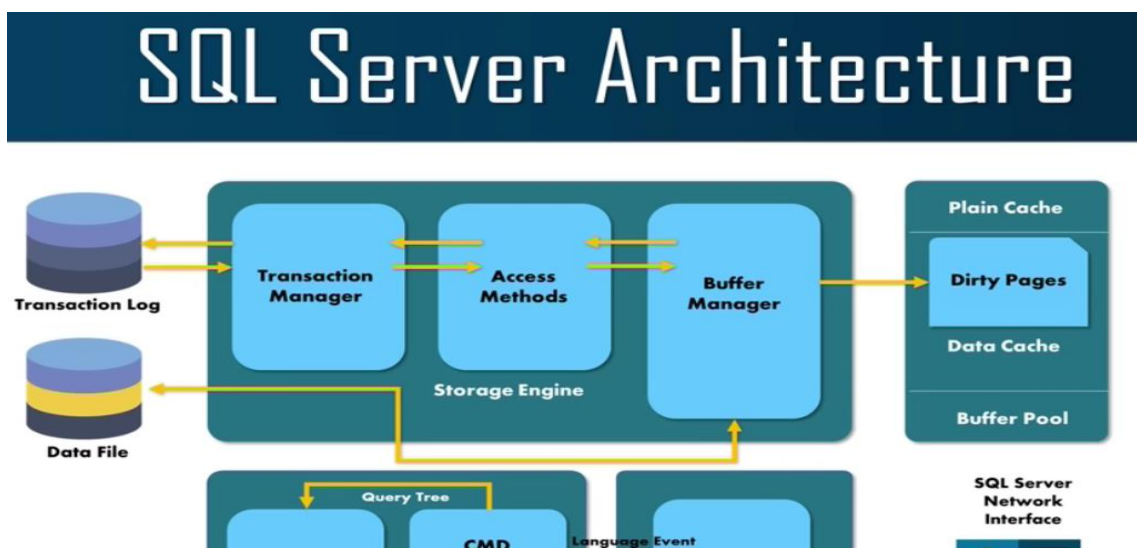
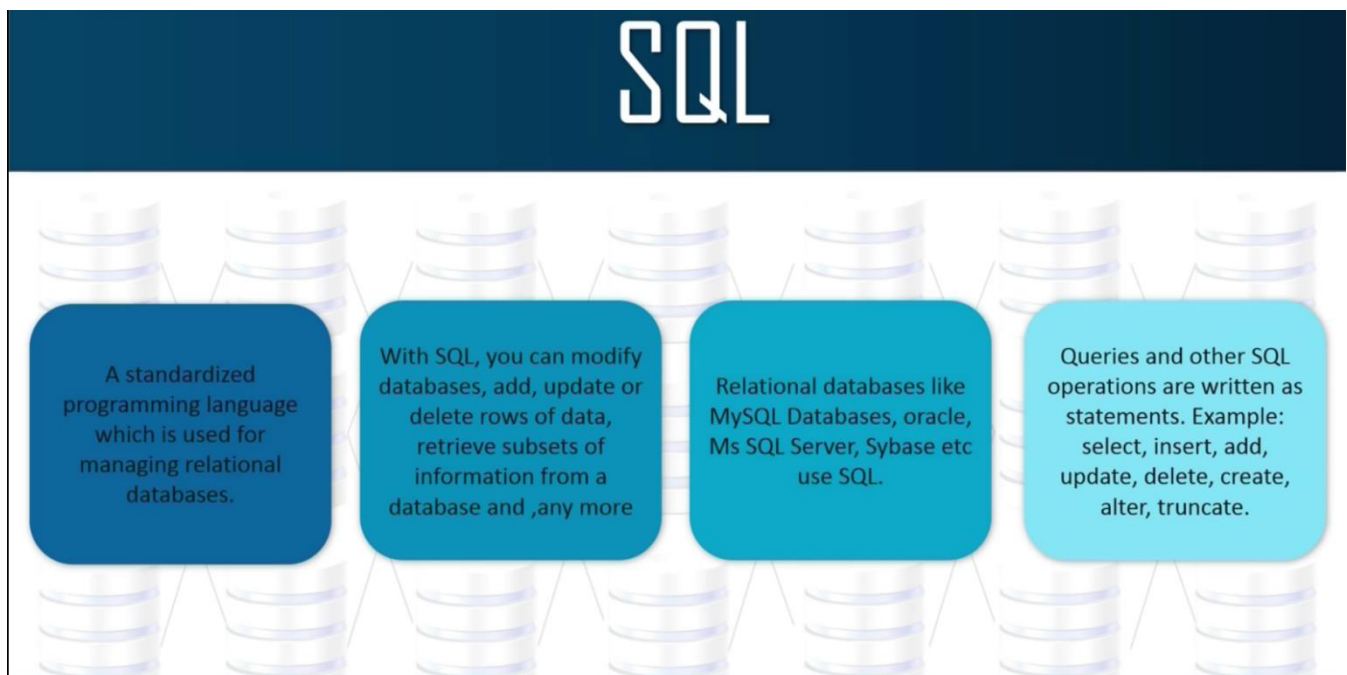
Our SQL Server 2019 instance has been finally installed. What remains is to connect to it, create databases and take advantage of its resources.

Next, we are going to look at how to connect to our SQL instance using SQL Server Management Studio. We shall install it and kick-off using our SQL resources.

WHAT IS A DATA DICTIONARY?

In SQL SERVER the data dictionary is a set of database tables used to store information about a database's definition. The dictionary contains information about database objects such as tables, indexes, columns, datatypes, and views.

The data dictionary is used by SQL SERVER to execute queries and is automatically updated whenever objects are added, removed, or changed within the database.



SQL SERVER DATABASE

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management

system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database Data

within the most common types of databases in operation today is typically modeled in rows

and columns in a series of tables to make processing and data querying efficient. The data can

then be easily accessed, managed, modified, updated, controlled, and organized. Most databases

use structured query language (SQL) for writing and querying data.

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Databases' folder expanded under the 'BCHIMSELF2FA6' server. The right pane shows a table of database information for the 'BCHIMSELF2FA6' database. The table has columns: name, database_id, source_database_id, owner_sid, create_date, compatibility_level, collation_name, user_access, and use. The 'master' database is highlighted in the table.

name	database_id	source_database_id	owner_sid	create_date	compatibility_level	collation_name	user_access	use
master	1	NULL	0xd01	2003-04-08 09:13:36.390	150	SQL_Latin1_General_CP1_CI_AS	0	ML
tempdb	2	NULL	0xd01	2022-04-17 13:33:40.047	150	SQL_Latin1_General_CP1_CI_AS	0	ML
model	3	NULL	0xd01	2003-04-08 09:13:36.390	150	SQL_Latin1_General_CP1_CI_AS	0	ML
msdb	4	NULL	0xd01	2019-09-24 14:21:42.270	150	SQL_Latin1_General_CP1_CI_AS	0	ML
employee	5	NULL	0xd01050000000000000051500000061B3CA24583928ADA8838B...	2022-04-08 16:30:31.503	150	SQL_Latin1_General_CP1_CI_AS	0	ML
Spring2_WEB_2022	6	NULL	0xd01050000000000000051500000061B3CA24583928ADA8838B...	2022-04-09 02:55:51.443	150	SQL_Latin1_General_CP1_CI_AS	0	ML
Accounts	7	NULL	0xd01050000000000000051500000061B3CA24583928ADA8838B...	2022-04-17 03:56:26.270	150	SQL_Latin1_General_CP1_CI_AS	0	ML

To see all the existing Databases in your SQL SERVER Execute the following query:

SELECT * FROM SYS.DATABASES;

CREATE SQL SERVER DATABASE

- To create database in SQL SERVER, Use the below command:

Create database database_name;

Eg Create database students;// Here the database name is students

- CREATE DATABASE is the command, and next one is your database name.
- And at the end of statement, add a semicolon.

First, you specify the database_name following the CREATE DATABASE clause. The database name must be unique within the SQL SERVER instance. If you try to create a database with a name that already exists, SQL issues an error.

DATABASE TABLE

A database table is composed of records and fields that hold data. Tables are also called datasheets. Each table in a database holds data about a different, but related, subject.

SQL SERVER CREATE TABLE SYNTAX

To create a new table within a database, you use the SQL SERVER CREATE TABLE statement.

The CREATE TABLE statement is one of the most complex statements in SQL SERVER.

First, you specify the name of the table that you want to create after the CREATE TABLE clause. The table name must be unique within a database.

RECORDS

Data is stored in records. A record is composed of fields and contains all the data about one particular person, company, or item in a database. In this database, a record contains the data for one customer support incident report. Records appear as rows in the database table.

FIELDS

A field is part of a record and contains a single piece of data for the subject of the record.

The format is:

```
CREATE TABLE Table_name(  
    column_name data_type UNIQUE);
```

Or you can define the UNIQUE constraint as the table constraint as follows:

```
CREATE TABLE table_name (column_name data_type UNIQUE);
```

Eg

```
Create table [dbo].[Loan](  
[Account_No] [varchar] (250)  
, [Account_Name] [varchar] (255)  
, [Loan_Amount] [float]  
, [Loan_Date] [date]  
, [Loan_Interest] [float]  
, [Spread] [varchar]  
, [Termination_Date] [date]  
, [Account_Manager] [varchar] (255)  
, UNIQUE (Account_No));
```

Go

SQL SERVER SHOW HOST DATABASES

To list all databases on a **SQL SERVER** server host, you use the **SELECT * FROM SYS.DATABASES;** command as follows:

SQL SERVER DATA TYPES

String Data Types

Data type	Description	Max size	Storage
Char(n)	Fixed width character string	8,000 characters	Defined width
Varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
Varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
Text	Variable width character string	2GB of text data	4 bytes + number of chars
Nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
Nvarchar	Variable width Unicode string	4,000 characters	
Nvarchar(max)	Variable width Unicode string	536,870,912 characters	
Ntext	Variable width Unicode string	2GB of text data	
Binary(n)	Fixed width binary string	8,000 bytes	
Varbinary	Variable width binary string	8,000 bytes	
Varbinary(max)	Variable width binary string	2GB	
Image	Variable width binary string	2GB	

SQL SERVER NUMERIC DATA TYPES

Data type	Description	Storage
Tinyint	Allows whole numbers from 0 to 255	1 byte
Smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
Int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
Bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
Decimal(p,s)	Fixed precision and scale numbers. Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
Numeric(p,s)	Fixed precision and scale numbers. Allows numbers from $-10^{38} + 1$ to $10^{38} - 1$. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0	5-17 bytes
Smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
Money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
Float(n)	Floating precision number data from $-1.79E + 308$ to	4 or 8 bytes

	1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.
Real	Floating precision number data from -3.40E + 38 to 3.40E + 38 4 bytes

SQL SERVER DATE AND TIME DATA TYPES

MySQL provides types for date and time as well as the combination of date and time. In addition, MySQL supports timestamp data type for tracking the changes in a row of a table. If you just want to store the year without date and month, you can use the YEAR data type.

The following table illustrates the MySQL date and time data types:

Data type	Description
DATE	A date Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(<i>fsp</i>)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(<i>fsp</i>)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(<i>fsp</i>)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'

YEAR

A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000.

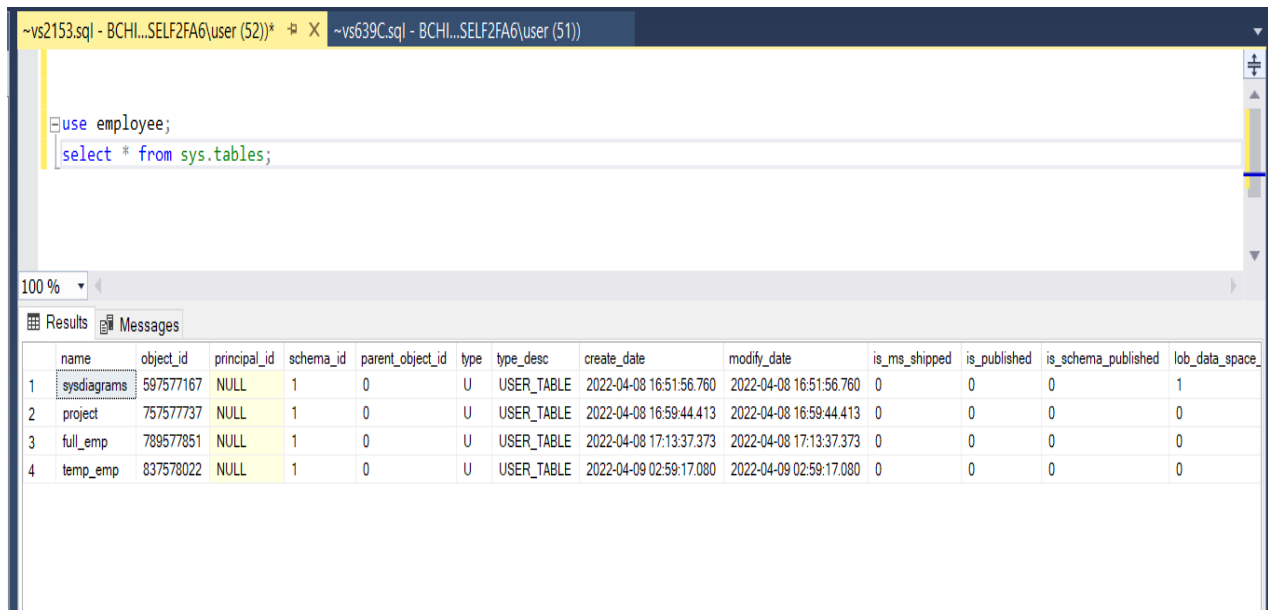
MySQL 8.0 does not support year in two-digit format.

SQL SERVER SHOW TABLES

To list tables in a SQL SERVER database, you follow these steps:

1. Login to the SQL SERVER database using a MSSMS
2. Switch to a specific database using the USE statement.
3. Use the SELECT * FROM SYS.TABLES; command.

The following illustrates the syntax of the SQL SERVER SELECT * FROM SYS.TABLES; command:



	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published	is_schema_published	lob_data_space
1	sysdiagrams	597577167	NULL	1	0	U	USER_TABLE	2022-04-08 16:51:56.760	2022-04-08 16:51:56.760	0	0	0	1
2	project	757577737	NULL	1	0	U	USER_TABLE	2022-04-08 16:59:44.413	2022-04-08 16:59:44.413	0	0	0	0
3	full_emp	789577851	NULL	1	0	U	USER_TABLE	2022-04-08 17:13:37.373	2022-04-08 17:13:37.373	0	0	0	0
4	temp_emp	837578022	NULL	1	0	U	USER_TABLE	2022-04-09 02:59:17.080	2022-04-09 02:59:17.080	0	0	0	0

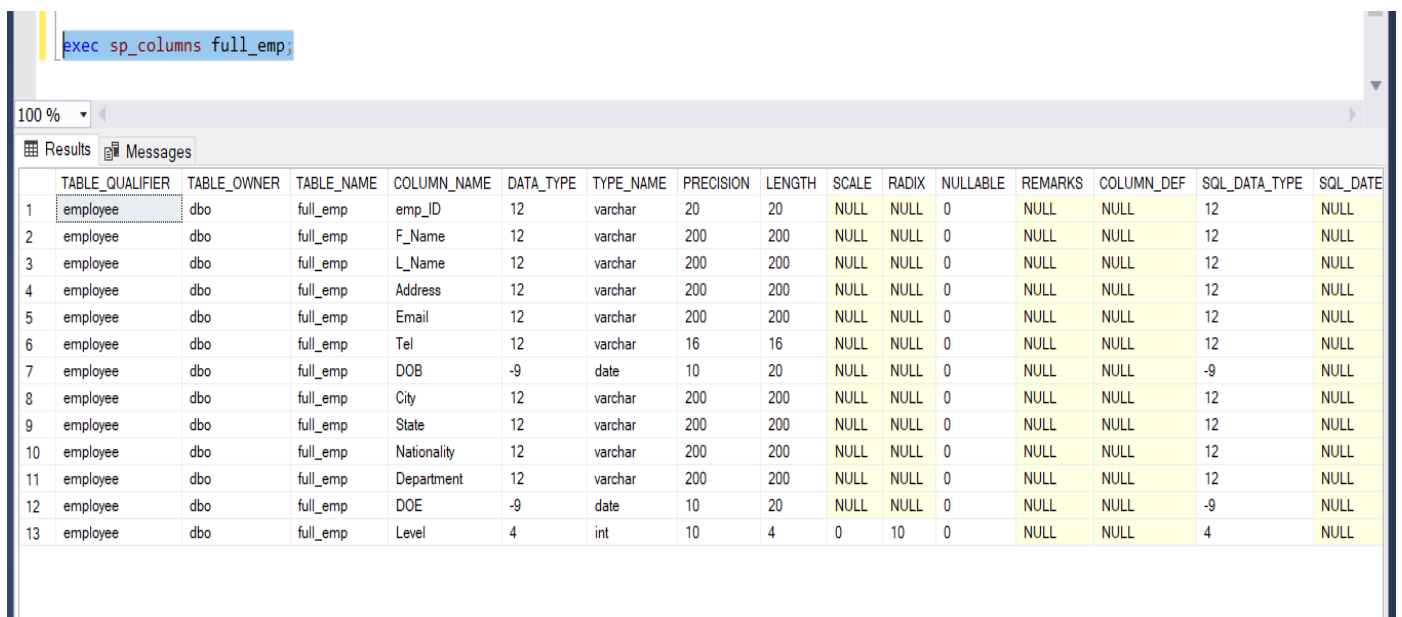
DESCRIBE STATEMENT

To show all columns of a table, you use the following steps:

1. Login to the SQL SERVER database Using MSSMS.
2. Switch to a specific database.
3. Use the *exec sp_columns Table_Name* statement.

e.g

exec sp_columns full_emp;



	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE	RADIX	NULLABLE	REMARKS	COLUMN_DEF	SQL_DATA_TYPE	SQL_DATE
1	employee	dbo	full_emp	emp_ID	12	varchar	20	20	NULL	NULL	0	NULL	NULL	12	NULL
2	employee	dbo	full_emp	F_Name	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
3	employee	dbo	full_emp	L_Name	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
4	employee	dbo	full_emp	Address	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
5	employee	dbo	full_emp	Email	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
6	employee	dbo	full_emp	Tel	12	varchar	16	16	NULL	NULL	0	NULL	NULL	12	NULL
7	employee	dbo	full_emp	DOB	-9	date	10	20	NULL	NULL	0	NULL	NULL	-9	NULL
8	employee	dbo	full_emp	City	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
9	employee	dbo	full_emp	State	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
10	employee	dbo	full_emp	Nationality	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
11	employee	dbo	full_emp	Department	12	varchar	200	200	NULL	NULL	0	NULL	NULL	12	NULL
12	employee	dbo	full_emp	DOE	-9	date	10	20	NULL	NULL	0	NULL	NULL	-9	NULL
13	employee	dbo	full_emp	Level	4	int	10	4	0	10	0	NULL	NULL	4	NULL

SQL SERVER INSERT STATEMENT

The INSERT statement allows you to insert one or more rows into a table. The following illustrates the syntax of the INSERT statement:

INSERT INTO table_name (c1, c2,...)

VALUES (v1, v2,...);

- First, specify the table name and a list of comma-separated columns inside parentheses after the INSERT INTO clause.
- Then, put a comma-separated list of values of the corresponding columns inside the parentheses following the VALUES keyword.

Example of Insert Statement:

```
insert into [dbo].[full_emp] ([emp_ID], [F_Name], [L_Name], [Address], [Email], [Tel],  
[DOB], [City], [State], [Nationality], [Department], [DOE], [Level])  
values ('BBT/STF/0022', 'Basil', 'Nzewure', 'SSS Quarters, Life Camp, Abuja',  
'sales@binbinarytechnology.com', '+2348038071202', '05-05-1990', 'Abuja', 'Abuja',  
'Nigeria', 'Software Engineering', '05-05-2000', 08);
```

INSERTING MULTIPLE COLUMNS AT ONCE

Example

```
Insert into [dbo].[full_emp] ([emp_ID], [F_Name], [L_Name], [Address], [email], [Tel],  
[DOB], [City], [Nationality], [Department], [Level], [Photo], [Gender])  
Values ('BBT/STF/0131', 'Denen', 'Kaki', 'Azare Bauchi State', 'dencity@yahoo.com',  
' +234809755553', '1983-06-6', 'Azare', 'Nigeria', 'Business Admin', 300, NULL, 'Male'),  
( 'BBT/STF/0232', 'Prince', 'Okere', 'Potiskun, Yobe State', 'prince@yahoo.com',  
' +234809755876', '1984-02-3', 'Potiskun', 'Nigeria', 'Political Science', 300, NULL, 'Male'),  
( 'BBT/STF/0326', 'Yerima', 'Mshellia', 'Borno, Borno State', 'mshellia@yahoo.com',  
' +234809755864', '1980-01-01', 'Chibok', 'Nigeria', 'Political Law', 400, NULL, 'Male'),  
( 'BBT/STF/0427', 'Comfort', 'Ude', 'Coal City Camp, Enugu State', 'comfort@yahoo.com',  
' +234809755543', '1990-01-03', 'Enugu', 'Nigeria', 'Economics', 200, NULL, 'Female'),  
( 'BBT/STF/0528', 'Prince', 'Okere', 'Potiskun Yobe State', 'prince@yahoo.com',  
' +234809755876', '1984-02-3', 'Potiskun', 'Nigeria', 'Political Science', 300, NULL, 'Male'),  
( 'BBT/STF/0629', 'Ibrahim', 'Abdulahi', 'Azare, Bauchi State', 'ibrahim@yahoo.com',  
' +234809753649', '1994-01-05', 'Azare', 'Nigeria', 'Business Law', 300, NULL, 'Male'),  
( 'BBT/STF/0730', 'Yelkon', 'Elisha', 'Darazo, Bauchi State', 'yekon@yahoo.com',  
' +234809755876', '1984-02-3', 'Potiskun', 'Nigeria', 'Philosophy', 300, NULL, 'Female');
```

NOTE: The type, date is also wrapped with quotes unlike in MySQL, date type is never treated as a string

INSERTING TABLE VALUES WITHOUT SPECIFYING THE COLUMN NAMES

Unlike the example above, you can insert rows into your table without explicitly spelling out the column names:

Example:

```
Insert into full_emp Values('BBT/STF/0031', 'Denen', 'Kaki', 'Azare Bauchi State',  
'dencity@yahoo.com', '+234809755553', '1983-06-6', 'Azare', 'Nigeria', 'Business Admin',  
300, NULL, 'Male');
```

```

Insert into full_emp Values('BBT/STF/0032', 'Prince', 'Okere', 'Potiskun, Yobe State',
'prince@yahoo.com', '+234809755876', '1984-02-3', 'Potiskun', 'Nigeria', 'Political
Science', 300, NULL, 'Male');
Insert into full_emp Values('BBT/STF/0026', 'Yerima', 'Mshellia', 'Borno, Borno State',
'mshellia@yahoo.com', '+234809755864', '1980-01-01', 'Chibok', 'Nigeria', 'Political Law',
400, NULL, 'Male');
Insert into full_emp Values('BBT/STF/0027', 'Comfort', 'Ude', 'Coal City Camp, Enugu
State', 'comfort@yahoo.com', '+234809755543', '1990-01-03', 'Enugu', 'Nigeria',
'Economics', 200, NULL, 'Female');
Insert into full_emp Values('BBT/STF/0028', 'Prince', 'Okere', 'Potiskun Yobe State',
'prince@yahoo.com', '+234809755876', '1984-02-3', 'Potiskun', 'Nigeria', 'Political
Science', 300, NULL, 'Male');
Insert into full_emp Values('BBT/STF/0029', 'Ibrahim', 'Abdulahi', 'Azare, Bauchi State',
'ibrahim@yahoo.com', '+234809753649', '1994-01-05', 'Azare', 'Nigeria', 'Business Law',
300, NULL, 'Male');
Insert into full_emp Values('BBT/STF/0030', 'Yelkon', 'Elisha', 'Darazo, Bauchi State',
'yekon@yahoo.com', '+234809755876', '1984-02-3', 'Potiskun', 'Nigeria', 'Philosophy',
300, NULL, 'Female');

```

SQL SERVER SELECT STATEMENT

The SELECT statement allows you to get the data from tables or views. A table consists of rows and columns like a spreadsheet. Often, you want to see a subset rows, a subset of columns, or a combination of two. The result of the SELECT statement is called a result set that is a list of rows, each consisting of the same number of columns. Select * from table_name;

To select all data from the full_emp table, below is the select statement:

```
SELECT * FROM full_emp;
```

To query from selected columns:

```
SELECT column_1, column_2, ...FROM table_name;
```

Example:

```
select emp_ID, F_Name, L_Name, Email, Tel, Department from full_emp;
```

SELECTING A CERTAIN NUMBER OF ROWS

To select specific number of rows from your database table you specify the select statement with the top keyword alongside the wild character asterisk *.

Example:

Select TOP 4 * from full_emp;

SELECT DISTINCT

Select DISTINCT queries your SQL SERVER Database table and returns a result set of the field only.

Example:

Select Distinct Email from full_emp;

SQL SERVER ORDER BY, ASC & DESC QUERY

You can query your database and alphabetically arrange them in Ascending Order or Descending Order .

To return a result set in Ascending Order, you use Order By Column_Name ASC

Example:

Select * from full_emp ORDER BY L_Name ASC;

Your result set returns a default in Ascending Order if you did not specify the order in which you want your result set to return.

Example:

Select * from full_emp ORDER BY L_Name;

This will return a sorted result set but in Ascending Order.

Select * from full_emp ORDER BY F_Name, L_Name;

To return a result set in Descending Order, you use Order By Column_Name DESC

Example:

Select * from full_emp ORDER BY L_Name DESC;

SELECTING CERTAIN COLUMNS AND ORDERING THEM IN ASCENDING OR DESCENDING ORDER

You can achieve this with comma separated list of table columns together with the Order By Clause.

Examples:

Select L_Name, F_Name, Email from full_emp Order By F_Name;

SORTING WITH MORE THAN ONE COLUMN

Sorting your table with more than one column returns you a result set according to your criteria.

Example 1:

```
Select * from full_emp ORDER BY F_Name, L_Name;
```

Example 2:

```
Select * from full_emp ORDER BY L_Name, F_Name;
```

This will give a different result since Last Name comes first before First Name. Meaning that the Last Name column will be sorted first before the First Name column.

You can decide to sort the different columns in Ascending or Descending manner

Example 3:

```
Select * from full_emp ORDER BY L_Name ASC, F_Name DESC;
```

DEPLOYING AGGREGATE FUNCTIONS

You can make use of Aggregate functions in SQL SERVER database.

For instance you want to count the total number of accounts (Account numbers) in your database or the total number of customers on Nsukka branch, or the total number of employees in your branch from the employee database.

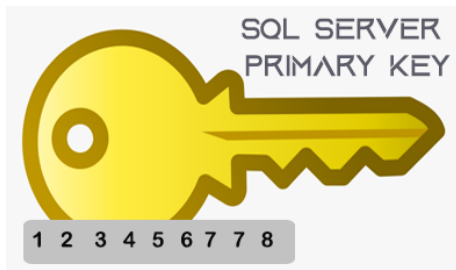
Example:

```
Select Count(email), City from full_emp Group by City;
```

	(No column name)	City
1	2	Abuja
2	5	Azare
3	2	Chibok
4	2	Enugu
5	1	Lagos
6	2	Owerri
7	7	Potiskun

The above table shows us 2 Abuja Accounts, 5 Azare Accounts, 2 Chibok Accounts, 2 Enugu Accounts, 1 Lagos Account, 2 Owerri Accounts and 7 Potiskun Accounts.

SQL SERVER PRIMARY KEY



A primary key is a column that uniquely identifies each row in the table. You must follow the rules below when you define a primary key for a table:

- A primary key must contain unique values.
- A primary key column cannot contain NULL values. It means that you have to declare the primary key column with the NOT NULL attribute. If you don't, SQL SERVER will force the primary key column as NOT NULL implicitly.
- A table has only one primary key.

```
CREATE TABLE users (  
  user_id INT PRIMARY KEY,  
  username VARCHAR(40),  
  password VARCHAR(255),  
  email VARCHAR(255)  
);
```

UPDATING TABLE TO SET PRIMARY KEY

You can update your table and set Primary of your ID or any field of your choice. Execute the following statement:

```
alter table Department_task  
add Primary Key(id)
```

SQL SERVER UNIQUE INDEX

To enforce the uniqueness value of one or more columns, you often use the PRIMARY KEY constraint. However, each table can have only one primary key. Hence, if you want to have

a more than one column or a set of columns with unique values, you cannot use the primary key

constraint.

Luckily, SQL SERVER provides another kind of index called UNIQUE index that allows you to enforce the uniqueness of values in one or more columns. Unlike the PRIMARY KEY index, you can have more than one UNIQUE index per table.

Example:

```
create table Python_Students(  
std_ID varchar(200) PRIMARY KEY,  
F_Name varchar(250),  
L_Name varchar(250),  
Address varchar(8000),  
Email varchar(250) UNIQUE,  
Tel bigint UNIQUE,  
DOB date,  
DOA date,  
Study_Center varchar(200),  
Study_Status varchar(100),  
City varchar(200),  
State varchar(200),  
Nationality varchar(200));
```

SQL SERVER NOT NULL CONSTRAINT

The NOT NULL constraint is a column constraint that forces the values of a column to non-NULL values only.

The syntax of the NOT NULL constraint is as follows:

```
Column_name data_type NOT Null;
```

A column may contain one NOT NULL constraint only, which specifies a rule that the column must not contain any NULL value.

```
CREATE TABLE tasks (  
id varchar(200) PRIMARY KEY,
```



```
title VARCHAR(255) NOT NULL,  
start_date DATE NOT NULL,  
end_date DATE);
```

SQL SERVER AUTO INCREMENT SEQUENCE

In SQL SERVER, a sequence is a list of integers generated in the ascending order i.e., 1,2,3... Many applications need sequences to generate unique numbers mainly for identification e.g., customer ID in CRM, employee numbers in HR, and equipment numbers in the services management system.

To create a sequence in SQL SERVER automatically, you set the AUTO_INCREMENT attribute to a column, which typically is a primary key column.

The following rules are applied when you use the AUTO_INCREMENT attribute:

- Each table has only one AUTO_INCREMENT column whose data type is typically the integer.
- The AUTO_INCREMENT column must be indexed, which means it can be either PRIMARY KEY or UNIQUE index.
- The AUTO_INCREMENT column must have a NOT NULL constraint. When you set the AUTO_INCREMENT attribute to a column, SQL SERVER automatically adds the NOT NULL constraint to the column implicitly.

The following statement creates a table named employees that has the emp_ID column as an AUTO_INCREMENT column:

```
CREATE TABLE Faculties(  
emp_ID int IDENTITY (1,1) NOT NULL, F_Name VARCHAR(50), L_Name VARCHAR(50),  
Address varchar(200),  
Department varchar(200),  
Level int);
```

In the above example, the **starting value for IDENTITY is 1** and it should **increment by 1** for every new record added. You can mention these values, according to your wish. Also, to insert values in the above table, you have to use the INSERT query in the following way:

```
INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Cynthia', 'Nzewure', '5 Okigwe Road', 'BlockchainInnovation', 13);
```

Here, if you observe, I have omitted the emp_ID column, as the ID will be automatically generated. So, if you insert 5 more values using the below queries:

```
INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('John','Doe','5 New York City USA', 'Ethical Hacking', 12);
INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Amaka','Njoku','5, St Marys, Ubomiri, Owerri, Imo State', 'Cloud Computing',
16);
INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Anayo','Uba','6, Umuiwuogu, Ubomiri, Owerri, 'Mobile Development', 15);
INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Jesse','Nzewure','4, Olawale Sikiru Str, Allied Garden Estate, Ajah Lekki - Lagos
State', 'Full Stack Development', 16);
INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Chioma','Nzewure','10, Kakuri, Kaduna State', 'Data Processing', 15);
```

the emp_ID will automatically increment itself. You don't need to control it anymore by any means. SQL SERVER will always increment each row inserted into the table by 1. The

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure for 'BCHIMSELF2FA6 (SQL Server 15.0.2000.5)', including databases, system databases, database snapshots, accounts, employee, database diagrams, tables, system tables, file tables, external tables, graph tables, dbo.full_emp, dbo.project, dbo.Python_Students, dbo.temp_emp, views, external resources, synonyms, programmability, service broker, storage, security, Spring2_WEB_2022, security, server objects, replication, PolyBase, always on high availability, management, integration services catalogs, SQL Server Agent (Agent XPs disabled), and XEvent Profiler.

The central query window shows the following SQL code:

```
select * from employee.INFORMATION_SCHEMA.TABLES;

INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('John','Doe','5 New York City USA', 'Ethical Hacking', 12);

INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Amaka','Njoku','5, St Marys, Ubomiri, Owerri, Imo State', 'Cloud Computing', 16);

INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Anayo','Uba','6, Umuiwuogu, Ubomiri, Owerri, Imo State', 'Mobile Development', 15);

INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Jesse','Nzewure','4, Olawale Sikiru Str, Allied Garden Estate, Ajah Lekki - Lagos State', 'Full Stack Development', 16);

INSERT INTO Faculties(F_Name, L_Name, Address, Department, Level)
VALUES ('Chioma','Nzewure','10, Kakuri, Kaduna State', 'Data Processing', 15);
```

The bottom results window shows the data inserted into the Faculties table:

emp_ID	F_Name	L_Name	Address	Department	Level
1	Cynthia	Nzewure	5 Okigwe Road	Blockchain Innovation	13
2	John	Doe	5 New York City USA	Ethical Hacking	12
3	Amaka	Njoku	5, St Marys, Ubomiri, Owerri, Imo State	Cloud Computing	16
4	Anayo	Uba	6, Umuiwuogu, Ubomiri, Owerri, Imo State	Mobile Development	15
5	Jesse	Nzewure	4, Olawale Sikiru Str, Allied Garden Estate, Aj...	Full Stack Development	16
6	Chioma	Nzewure	10, Kakuri, Kaduna State	Data Processing	15

The status bar at the bottom indicates: Query executed successfully. | BCHIMSELF2FA6 (15.0 RTM) | BCHIMSELF2FA6(user (52)) | employee | 00:00:00 | 6 rows

RESETTING YOUR AUTOINCREMENT

To reset your auto increment to restart from the beginning, you just execute the code:

```
DBCC CHECKIDENT (contract_employees, RESEED, 0)
```

Each time you reset your auto increment, your ID will begin from the next nth number above the initial starting value. For example, if your auto increment were running from 1 when you reset, it will restart from 2.

SQL DROP DATABASE STATEMENT

Drop database statement is used to drop an existing database. A database can be dropped regardless of its state: offline, read-only, suspect, and so on. To display the current state of a database, use the **sys.databases** catalog view.

Dropping a database deletes the database from an instance of SQL Server and deletes the physical disk files used by the database.

```
DROP DATABASE faculty;
```

ARGUMENTS

IF EXISTS

Applies to: SQL Server (SQL Server 2016 (13.x) through current version).

Conditionally drops the database only if it already exists.

database_name Specifies the name of the database to be removed. The query to check if a database exists before dropping is as follows:

Example 1:

```
DROP DATABASE IF EXISTS Transfer;
```

Example 2:

The following example first checks to see if a database named Category exists. If so, the example changes the database named Sales to single-user mode to force disconnect of all other sessions, then drops the database.

```
USE employee;
```

```
GO
```

```
DECLARE @SQL varchar(1000);
```

```
IF EXISTS (SELECT 1 FROM sys.databases WHERE [name] = N'Transfer')
```

```
BEGIN
    SET @SQL = N'USE [Transfer];

    ALTER DATABASE Transfers SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    USE [employee];

    DROP DATABASE Transfer;';
EXEC (@SQL);
END;
```

DROPPING MULTIPLE DATABASES

Applies to: SQL Server 2008 and later.

The following code deletes all the listed databases.

```
DROP DATABASE Sales, Purchases;
```

SQL DROP TABLE COMMAND

You may be deploying a table to a database and need to be sure a table with the same name doesn't still exist or attempting to do a CREATE TABLE will fail. If you try to do a DROP TABLE first and the table does not exist, it will also fail with the error "Msg 3701 Level 11 State 5, 25 Cannot drop the table because it does not exist or you do not have permission". How do you solve these problems?

The way out is to add an additional logic to the query. This additional logic will check whether the table exists or not. If it exists, Transact-SQL [T-SQL] will return an error message such as this:

Msg 3701, Level 11, State 5, Line 19

Cannot drop the table 'dbo.customers', because it does not exist or you do not have permission.

Example:

```
Drop table [dbo].[customers]
```

If the above table does not exist, T-SQL will return an error message notifying you of the non existence of the table.

DROP Table IF EXISTS (SQL Server 2016 and up)

You can by-pass such delay and errors by adding more logic argument such as IF EXIST.

Example:

-----Drop table if it exists

```
drop table IF EXISTS [dbo].[customers] ;  
GO
```

-----Run drop table if there is a row in sys.tables

```
if exists (Select * from Sys.tables WHERE SCHEMA_NAME(schema_id) Like 'dbo' and name  
like 'Customers')
```

```
Drop table [dbo].[Customers]; Go
```

SQL SERVER ALTER COMMAND

SQL SERVER ALTER command is very useful when you want to change a name of your table, any table field or if you want to add or delete an existing column in a table.

Dropping, Adding a Column

If you want to drop an existing column from an existing SQL SERVER table, then you will use the **DROP** clause along with the **ALTER** command as shown below –

Alter table table_name drop column column_name

```
alter table full_emp drop column State;
```

A DROP clause will not work if the column is the only one left in the table.

SQL SERVER ADD COLUMN STATEMENT

The add column command is executed to add a certain column in a SQL SERVER Database table. It is executed alongside alter statement with the following syntax:

Alter table table_name add column_name data_type

```
alter table full_emp add Gender varchar(7);
```

SQL SERVER ALTER DATA TYPE QUERY

Every data inserted into SQL SERVER Database is of a certain type. This is what is called Data type. SQL SERVER has several data types such as varchar, date and time, float, double etc.

To alter or change a column's data type, you must use the alter statement.

Example:

Syntax:

Alter table t

able_name alter column column_name data_type

SQL SERVER ADD COLUMN STATEMENT

To add a new column to an existing table, you use the:

ALTER TABLE ADD COLUMN statement as follows:

Alter table table_name add column_name data_type

Example:

Alter table students add Std_City varchar(200);

Let's examine the statement in more detail.

- First, you specify the table name after the ALTER TABLE clause.
- Second, you put the new column and its definition after the ADD COLUMN clause.
- Finally, you specify the data type of the field.

ADDING MULTIPLE COLUMNS

Add Multiple columns to a SQL SERVER table

The following example adds two columns to the table Department_task

Example 1:

Alter table Department_task add Duration varchar(200), Cost float;

Example 2:

Alter table Department_task add Task_ID int Identity(1,1), No_Of_Staff int, Location varchar(200);

SQL SERVER RENAME TABLE STATEMENT:

To rename a table

1. In **Object Explorer**, connect to an instance of Database Engine.
2. On the Standard bar, select **New Query**.
3. The following example renames the Tasks table to Department_task in the employee schema.

Execute the following:

```
exec sp_rename 'Tasks', 'Department_task';
```

SQL SERVER RENAME COLUMN STATEMENT

The following example renames the column Title in the table Department_task to Name in the employee database.

```
EXEC sp_rename 'dbo.Department_task.title', 'Task_Title', 'COLUMN';
```

SQL SERVER CHANGE COLUMN ORDER (NOT AVAILABLE)

To change the column order

This task is not supported using Transact-SQL statements. The table must be dropped and recreated in order to change column order.

Delete columns using Transact-SQL

You can delete columns using Transact-SQL in SSMS, or command-line tools such as the sqlcmd utility.

The following example shows you how to delete a column.

```
ALTER TABLE dbo.table_name DROP COLUMN column_name;
```

Example:

```
alter table Department_task drop column Duration;
```


SQL SERVER SERVER - UPDATE QUERY

Modification of data in a database is paramount, there are always requirement where the existing data in a SQL SERVER table needs to be modified. You can do so by using the SQL **UPDATE** command and the WHERE clause. This will modify any field value of any SQL SERVER database table.

This code block has the SQL SERVER syntax of the Update Statement to update or modify data in SQL SERVER database table.

```
update full_emp set F_Name = 'Ron', L_Name = 'Weasley' WHERE emp_ID = 'BBT/STD/001';
```

SQL SERVER UPDATE to modify values in multiple columns. To update values in multiple columns, you need to specify the assignments in the SET clause. For example, the following statement updates both last name and email columns of employee number 1056:

```
UPDATE employees SET lastname = 'Jude Nzewure', email = 'info@binbinarytechnology.com' WHERE employeeNumber = 1056;
```

MERGING TABLES IN SQL SERVER DATABASE.

SQL SERVER Provides opportunity for merging tables which is critical in any environment.

The SQL Server MERGE command is the combination of INSERT, UPDATE and DELETE commands consolidated into a single statement. Here is how to get started with the SQL Server MERGE command:

1. Start off by identifying the target table name which will be used in the logic.
2. Next identify the source table name which will be used in the logic.
3. Determine the appropriate search conditions in the ON clause in order to match rows.
4. Specify logic when records are matched or not matched between the target and source i.e. comparison conditions.
5. For each of these comparison conditions code the logic. When matched, generally an UPDATE condition is used. When not matched, generally an INSERT or DELETE condition is used.

First, Create The Two Tables

```
Create table Savings_acc(Acc_No varchar(200), Acc_Name varchar(200), Acc_bal  
varchar(200), branch varchar(200), );
```

Go

```
Create table Current_acc(Acc_No varchar(200), Acc_Name varchar(200), Acc_bal  
varchar(200), branch varchar(200));
```

Secondly, Insert Into The Tables

```
insert into Savings_acc values('101936458761', 'N200000', 'John Doe', 'Ajah');  
insert into Savings_acc values('101936458762', 'N250000', 'John Waxwel', 'New York');  
insert into Savings_acc values('101936458763', 'N100000', 'John Nzewure', 'Owerri');  
insert into Savings_acc values('101936458764', 'N50000', 'Okoye Nicholas', 'Abuja');
```

```
insert into Current_acc values('101936458765', 'N500000', 'Abayomi Mike', 'Marina');  
insert into Current_acc values('101936458766', 'N2000000', 'Abuka Isreal', 'Lokoja');  
insert into Current_acc values('101936458767', 'N300000', 'Ugochukwu Oparaocha',  
'Mbaise');  
insert into Current_acc values('101936458768', 'N320000', 'Onyeka Sylvester', 'Asaba');
```

```
select * from Savings_acc;
```

Go

```
Select * from Current_acc;
```

Then Run Your Merge Query

```
Merge Current_acc TARGET USING Savings_acc SOURCE ON (Target.Acc_No =  
Source.Acc_No)  
WHEN MATCHED AND Target.Acc_Name <> Source.Acc_Name Or Target.Acc_bal <>  
Source.Acc_bal or Target.branch <> Source.branch  
THEN UPDATE SET Target.Acc_Name = Source.Acc_Name, Target.Acc_bal =  
Source.Acc_bal, Target.branch = Source.branch  
When NOT Matched By Target THEN Insert (Acc_No, Acc_Name, Acc_bal, branch) Values  
(Source.Acc_No, Source.Acc_Name, Source.Acc_bal, Source.branch)  
WHEN NOT MATCHED By Source THEN  
Delete;
```

SQL SERVER WHERE CLAUSE

We can use a conditional clause called the **WHERE Clause** to filter out the results. Using this WHERE clause, we can specify a selection criterion to select the required records from a table.

This is the code block for the generic SQL SERVER syntax of select statement with the WHERE clause to fetch data from the SQL SERVER table:

```
Select field1, field2, field3....fieldN from Table_name [Where Condition];
```

e.g

```
Select Std_Name, Std_Department, Std_Faculty from student where Std_Name  
= "Basil Nzewure";
```

The **WHERE** clause works like an **if condition** in any programming language. This clause is used to compare the given value with the field value available in a SQL SERVER table. If the given value from outside is equal to the available field value in the SQL SERVER table, then it returns that row.

Here is the list of operators, which can be used with the **WHERE** clause.

SQL SERVER Operators

Operator	Description	Example
	then the condition becomes true.	true.
!=	Checks if the values of the two operands are equal or not, if the values are not equal then the condition becomes true.	(A != B) is true.
>	Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of the left operand is less than the value of the right operand, if yes then the condition becomes true.	(A < B) is true.
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.	(A <= B) is true.
=	Checks if the values of the two operands are equal or not, if yes,	(A=B) is not

DATABASE TRACKING

First and foremost, you have to enable tracking on your database and your database tables:

To alter your database and set tracking on it you execute the following code:

```
alter database employee
Set Change_Tracking = ON
(Change_Retention = 14 Days, Auto_Cleanup = ON);
```

Change Retention specifies the number of days the tracking will exist in the database before clean up.

ENABLE CHANGE TRACKING FOR A TABLE

Change tracking must be enabled for each table that you want tracked. When change tracking is enabled, change tracking information is maintained for all rows in the table that are affected by a DML operation.

To alter your table and set tracking on it you execute the following code:

```
Alter table Department_task
Enable Change_Tracking
WITH (TRACK_COLUMNS_UPDATED = ON)
```

Disable Change Tracking for a Database or Table

Change tracking must first be disabled for all change-tracked tables before change tracking can be set to OFF for the database.

To disable change tracking on your table, you execute the following code:

```
Alter table Department_task
Disable Change_Tracking
```

Get All The Rows With Associated Version

```
-- Get all current rows with associated version
SELECT t.[id], t.Task_Title, t.start_date, t.end_date, t.Team_Leader, t.Duration, t.Cost,
t.Task_ID, t.No_Of_Staff, t.Location,
c.SYS_CHANGE_VERSION, c.SYS_CHANGE_CONTEXT
FROM Department_task AS t
CROSS APPLY CHANGETABLE
(VERSION Department_task, ([id]), (t.[id])) AS c;
```

-----Listing all changes that were made since a specific version

```
DECLARE @last_sync_version bigint;
SET @last_sync_version = 1;
SELECT [id], id,
SYS_CHANGE_VERSION, SYS_CHANGE_OPERATION,
SYS_CHANGE_COLUMNS, SYS_CHANGE_CONTEXT
FROM CHANGETABLE (CHANGES Department_task, @last_sync_version) AS C;
```

Listing Changes Made Since Specific Version

The following example lists all changes that were made in a table since the specified version (@last_sync_version). [id] is a column in a composite primary key.

-----Listing all changes that were made since a specific version

```
DECLARE @last_sync_version bigint;
SET @last_sync_version = 1;
SELECT [id], id,
SYS_CHANGE_VERSION, SYS_CHANGE_OPERATION,
SYS_CHANGE_COLUMNS, SYS_CHANGE_CONTEXT
FROM CHANGETABLE (CHANGES Department_task, @last_sync_version) AS C;
```

Obtaining all changed data for a synchronization

The following example shows how you can obtain all data that has changed. This query joins the change tracking information with the user table so that user table information is returned. A LEFT OUTER JOIN is used so that a row is returned for deleted rows.

-- Get all changes (inserts, updates, deletes)

```
DECLARE @last_sync_version bigint;
SET @last_sync_version = 1;
SELECT t.id, t.Task_Title, t.start_date, t.end_date, t.Team_Leader, t.Duration, t.Cost,
t.Task_ID, t.No_Of_Staff, t.Location, t.SSN,
c.SYS_CHANGE_VERSION, c.SYS_CHANGE_OPERATION,
c.SYS_CHANGE_COLUMNS, c.SYS_CHANGE_CONTEXT
FROM CHANGETABLE (CHANGES Department_task, @last_sync_version) AS c
LEFT OUTER JOIN Department_task AS t
ON t.[id] = c.[id] AND t.id = c.id;
```

DATABASE BACKUP

Create a folder where you would want to backup your database.

For instance, inside your Local Disk. Then execute your backup statement

Example:

```
BACKUP DATABASE employee  
    TO DISK = 'C:\SQLServerBackups\full_emp.bak';  
GO
```

TO RESTORE YOUR DATABASE

When you are specifying a backup file and restore, you should enter its full path and file name. If you specify only the file name or a relative path when you are backing up to a file, the backup file is put in the default backup directory.

Example:

```
Restore Database employee  
    from Disk = 'C:\SQLServerBackups\full_emp.bak';  
GO
```

Here is your default backup path:

C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup

Specify a Universal Naming Convention (UNC) name

To specify a network share in a backup or restore command, use the fully qualified universal naming convention (UNC) name of the file for the backup device. A UNC name has the form `\\Systemname\ShareName\Path\FileName`.

Example:

```
BACKUP DATABASE AdventureWorks2012  
    TO DISK = '\\BackupSystem\BackupDisk1\AW_backups\AdventureWorksData.Bak';  
GO
```

So you specify the path of your network to backup your database.

SQL SERVER – BACKUP QUERY

You can take a backup of a database lets say employee_DB by:

1. Right clicking on the database
 2. Click On Tasks
 3. Click on Backup
 4. Delete the path already there by clicking delete
 5. Click on Add
 6. On the destination on disk browse the location where you want to store the backup
 7. Locate your C:/
 8. On the file name, type the name you want to store it with e.g employee04172022.bak
 9. Click OK
 10. Click on Backup option on the left
 11. On set Backup Compression, select compress backup
 12. On top, you will see script combobox, select Script Action To New Query Window
 12. Close the dialog box. SQL will generate backup script.
- E,g

```
BACKUP DATABASE [employee] TO DISK = N'C:\employee04172022.bak'
WITH NOFORMAT, NOINIT,
NAME = N'employee-Full Database Backup',
SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10
```

We will now take the sql query with the sys.database to a build backup statement for all the databases.

Lets Start That right away:

1. Make the code a select statement and treat them as string as follows:

```
Select 'BACKUP DATABASE [employee] TO DISK = N" C:\employee04172022.bak' '
WITH NOFORMAT, NOINIT,
NAME = N";employee-Full Database Backup",
SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10'
```

NOTE: My string was added in 6 places, it was added before Backup immediately after Select keyword, another one was added to N" before C:\ and another was added to bak', another to N" and another to Backup" and finally to 10'

Finally, Select it and execute the query.

```
select 'BACKUP DATABASE [employee] TO DISK = N" C:\employee04172022.bak' '
WITH NOFORMAT, NOINIT,
NAME = N";employee-Full Database Backup",
SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10'
```

from sys.databases

REPLACING THE STATIC QUERY VALUE WITH REAL DATABASE NAMES

To replace the value of the databases, you have to concatenate the object variable and replace the [employee] name above with it.

Example: ['+ name +'].

You will now have the following code to display all the backed-up databases.

```
select 'BACKUP DATABASE [' + name +'] TO DISK = N" C:\'+name+ '04172022.bak' '  
WITH NOFORMAT, NOINIT, NAME = N" +name+ '-Full Database Backup',  
SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10'  
from sys.databases
```

DATABASE PRIVILEGES

System **privileges**. A system **privilege** is the right to perform a particular action or to perform an

action on any object of a particular type. Objects include tables, views, materialized views,

synonyms, indexes, sequences, cache groups, replication schemes and PL/**SQL** functions,

procedures and packages.

