# UiPath Connect SQL Server

SQL, **S**tructured **Q**uery **L**anguage, is a programming language designed to manage data stored in relational databases. SQL operates through simple, declarative statements. This keeps data accurate and secure, and helps maintain the integrity of databases, regardless of size.

SQL commands are mainly categorized into four categories as:

**1. DDL – Data Definition Language**

- **Create:** This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- **Drop:** This command is used to delete objects from the database.
- **Alter:** This is used to alter the structure of the database.
- **Truncate:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
- **Comment:** This is used to add comments to the data dictionary.
- **Rename:** This is used to rename an object existing in the database.

**2. DQL - Data Query Language**

- **Select:** It is used to retrieve data from the database.

**3. DML - Data Manipulation Language**

- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete records from a database table.

**4. DCL - Data Control Language**

- **Grant:** This command gives users access privileges to the database.
- **Revoke:** This command withdraws the user's access privileges given by using the GRANT command.

**5.TCL - Transaction Control Language**

- **Commit:** Commits a Transaction.
- **RollBack:** Rollbacks a transaction in case of any error occurs.
- **Savepoint:** Sets a savepoint within a transaction.

## Column Constraints:

Column constraints are the rules applied to the values of individual columns:

- **Primary Key**: Constraint can be used to uniquely identify the row.
- **Unique**: Columns have a different value for every row.
- **Not Null**: Columns must have a value.
- **Default:** Assigns a default value for the column when no value is specified.

There can be only one **Primary Key** column per table and multiple **Unique** columns.

- **Create Database:**
  Create database databse_name;

- **Display Databases's list:**
  Show databases;

- **Connect databse:**
  Use database_name;

- **Display table names:**
  Show table_name;

- **Display rows in table:**
  Select * from table_name;

## Relational Databases

A relational database is a database that organizes information into one or more tables...

All data stored in a relational database is of a certain data type. Some of the most common data types are:

- **Integer:** A positive or negative whole number
- **Text:** A text string
- **Date:** The date formatted as YYYY-MM-DD

# Statements

A statement is text that the database recognizes as a valid command. Statements always end in a semicolon (;).

## Create Statement:

Create Table is a clause. Clauses perform specific tasks in SQL.

**Syntax:**

**Create** Table table_name (

column_1 data_type,

column_2 data_type,

column_3 data_type

);

Table_name refers to the name of the table that the command is applied to. (Column_1 data_type, column_2 data_type, column_3 data_type) is a parameter. A parameter is a list of columns, data types, or values that are passed to a clause as an argument.

## Insert Statement:

The **Insert Into** statement is used to add a new record (row) to a table.

It has two forms as shown:

**Syntax:**

- Insert into columns in order.

**Insert** Into table_name **Values** (value1, value2);

- Insert into columns by name.

**Insert** Into table_name (column1, column2) **Values** (value1, value2);

## Alter Statement:

The **Alter Table** statement is used to modify the columns of an existing table. When combined with the **Add** clause, it is used to add a new column.

**Syntax:**

**Alter** Table table_name **Add** column_name datatype;

## Update Statement:

The **Update** statement is used to edit records (rows) in a table. It includes a **Set** clause that indicates the column to edit and a **Where** clause for specifying the record(s).

**Syntax:**

**Update** table_name

**Set** column1 = value1, column2 = value2

**Where** some_column = some_value;

## Delete Statement:

The **Delete** statement is used to delete records (rows) in a table. The **Where** clause specifies which record or records that should be deleted. If the **Where** clause is omitted, all records will be deleted.

**Syntax:**

**Delete** From table_name

**Where** some_column = some_value;

## Queries:

### And Operator:

The **And** operator allows multiple conditions to be combined. Records must match both conditions that are joined by **And** to be included in the result set.

**Syntax:**

**Select** column1, column2,.
**From** table_name
**Where** condition1 **And** condition2 **And** condition3 ...;

### Or Operator:

The **Or** operator allows multiple conditions to be combined. Records matching either condition joined by the **Or** are included in the result set.

**Syntax:**

**Select** column-name1, column_name2,
**From** table_name
**Where** condition1 **Or** condition2 **Or** condition3;

## Where Clause:

The **Where** clause is used to filter records.
It is used to extract only those records that fulfill a specified condition.
The **Where** clause is not only used in **Select** statements, it is also used in **Update, Delete**, etc.!

**Synatx:**

**Select** column1, column2, ...

**From** table_name

**Where** condition;

## As Clause:

SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable.

An alias is created with the **As** keyword.

**Syntax:**

**Select** column_name **As** alias_name

**From** table_name;

## Distinct Clause:

The **Select Distinct** statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

**Syntax:**

**Select Distinct** column1, column2, ...

**From** table_name;

**Select** Country **From** Customers;

## Between Operator:

The **Between** operator selects values within a given range. The values can be numbers, text, or dates.
The **Between** operator is inclusive: begin and end values are included.

**Syntax:**

**Select** *
**From** table_name
**Where** column_name **Between** value1 **And** value2;

## Order By Keyword:

The **Order By** keyword is used to sort the result-set in ascending or descending order.

The **Order By** keyword sorts the records in ascending order (**ASC**) by default. To sort the records in descending order, use the **DESC** keyword.

**Syntax:**

**Select** * **From** table_name

**Order By** column_name **DESC**;

## Limit Clause:

The **Limit** clause is used to narrow, or limit, a result set to the specified number of rows.

**Syntax:**

**Select** *

**From** table_name

**Limit** condition;

## Like Operator

The **Like** operator is used in a **Where** clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the **Like** operator:

- The percent sign **(%)** represents zero, one, or multiple characters.
- The underscore sign **(_)** represents one, single character.

**Example:**

The SQL statement selects all customers with a CustomerName starting with "a":

**Select** * **From** Customers(table_name)

**Where** CustomerName(column_name) **Like** 'a%';

## Aggregate Functions

### Group By Statement:

The **Group By** statement groups rows that have the same values into summary rows.
The **Group By** statement is often used with aggregate functions **(COUNT (), MAX (), MIN (), SUM (), AVG ())** to group the result-set by one or more columns.

**Syntax:**

**Select** column_name(s)

**From** table_name

**Where** condition

**Group By** column_name(s);

**Max ():**

The **Max () aggregate** function takes the name of a column as an argument and returns the largest value in a column.

**Syntax:**

**Select** Max(column_name)

**From** table_name;

**Min ():**

The **Min ()** aggregate function returns the smallest value in a column.

**Syntax:**

**Select Min**(column_name)

**From** table_name;

**Count ():**

The **Count ()** aggregate function returns the total number of rows that match the specified criteria.

**Syntax:**

**Select Count**(column_name)

**From** table_name;

**Avg():**

The **Avg()** aggregate function returns the average value in a column.

**Syntax:**

**Select Avg**(column_name)

**From** table_name;

**Sum ();**

The **Sum()** function returns the total sum of a numeric column.

**Syntax:**

**Select Sum**(column_name)

**From** table_name;

## Having Clause:

The **Having** clause is used to further filter the result set groups provided by the **Group By** clause. **Having** is often used with aggregate functions to filter the result set groups based on an aggregate property.

**Syntax:**

**Select** column_name(s)

**From** table_name

**Where** condition

**Group By** column_name(s)

**Having** condition;

## Union Operator:

The **Union** operator is used to combine the result-set of two or more **Select** statements.

Every **Select** statement within **Union** must have the same number of columns.

**Syntax:**

**Select** column_name(s) **From** table1

**Union**

**Select** column_name(s) **From** table2;

## SQL Joins:

A **Join** clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of sql join's:

**1. Inner Join:** The **Join** clause allows for the return of results from more than one table by joining them together with other results based on common column values specified using an **On** clause.

**Syntax:**

**Select** column_name(s)

**From** table1

**Inner Join** table2

**On** table1.column_name = table2.column_name;

**2. Left Join:** Returns all records from the left table, and the matched records from the right table.

**Synatx:**

**Select** column_name(s)

**From** table1

**Left Join** table2

**On** table1.column_name = table2.column_name;

**3. Right Join:** Returns all records from the right table, and the matched records from the left table.

**Syntax:**

**Select** column_name(s)

**From** table1

**Right Join** table2

**On** table1.column_name = table2.column_name;

**4. Full Join:** Returns all records when there is a match in either left or right table.

**Syntax:**

**Select** column_name(s)

**From** table1

**Full Join** table2

**On** table1.column_name = table2.column_name

## Sql Primary Key Constraint:

- The **Primary Key** constraint uniquely identifies each record in a table.
- Primary keys must contain **Unique** values, and cannot contain **Null** values.
- A table can have only one primary key; and in the table, this primary key can consist of single or multiple columns (fields).

**Example:**

**Create Table** Persons (

Id int **Not Null,**

LastName varchar (255) **Not Null**,

FirstName varchar (255),

Age int,

**Primary Key** (Id)

);

## Sql Foreign Key Constraint:

- The **ForEign Key** constraint is used to prevent actions that would destroy links between tables.
- A **Foreign Key** is a field (or collection of fields) in one table, that refers to the **Primary Key** in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

**Example:**

**Create Table** Orders (

OrderID int **Not Null**,

OrderNumber int **Not Null**,

PersonID int,

**Primary Key** (OrderID),

, **Foreign Key** (PersonID) **References** Persons (PersonID)

)

## Need to connect UiPath with SQL Server

As You Know about Different Input and Output Sources Like Ques,Excel/Data Table,Email,Folder/Files Etc.

Same Here, Now Database(Table) is the best input/output source too.Once you connect the Database with UiPath Studio,you can easily store/Fetch the configuration/inputdata to/from the Database

Use the SQL Server To Store the configuration data to the Table and read config data from table instead of using Excel

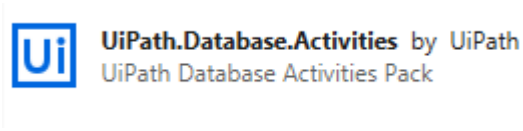
Use the SQL server to store the Input data to the Table and Read From Table Instead of using Excel or Queue.

It is used to keep the config/Input data secure in the Table.

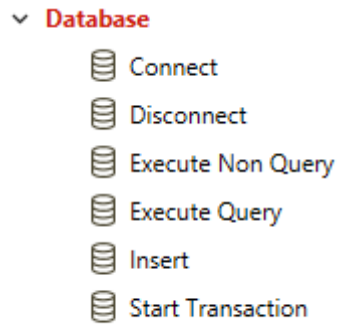
Update the status of the record in Table.

## Get Database Activities in UiPath Studio

To work with the Database Activities,we need to install the Database Package From the Manage Package in UiPath Studio

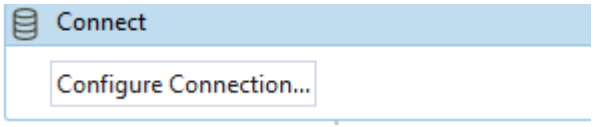



The Package is the Collection of all the necessary Database Activities which is required to establish a connection with UiPath and SQLServer Database,Execute the Query and Non Query and Disconnect the Connection when there is not required.

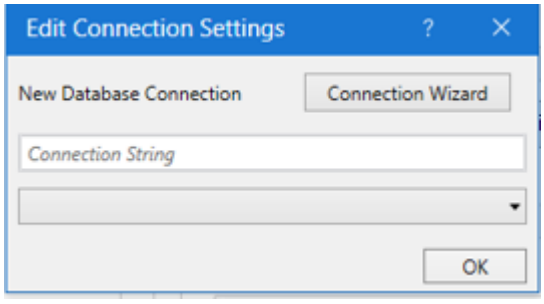



## Connect UiPath with SQL Server DataBase

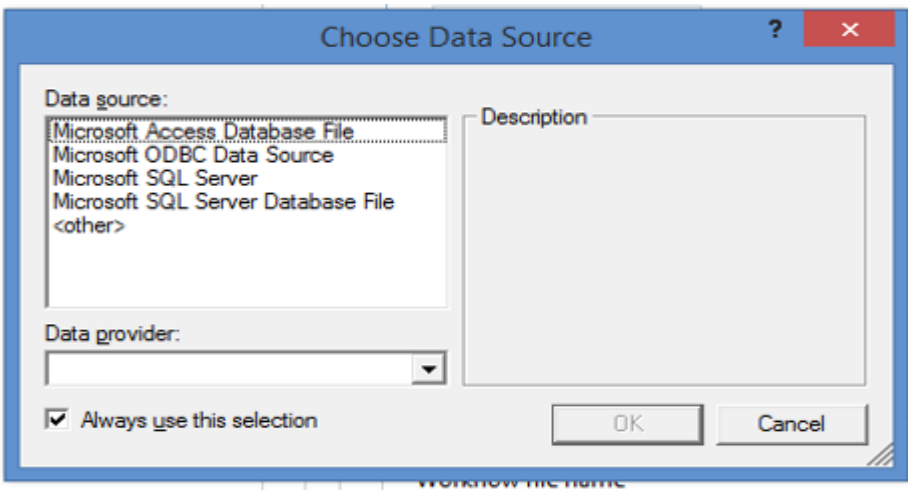
To work with the Database,we first need to establish a connection between UiPath and SQL server



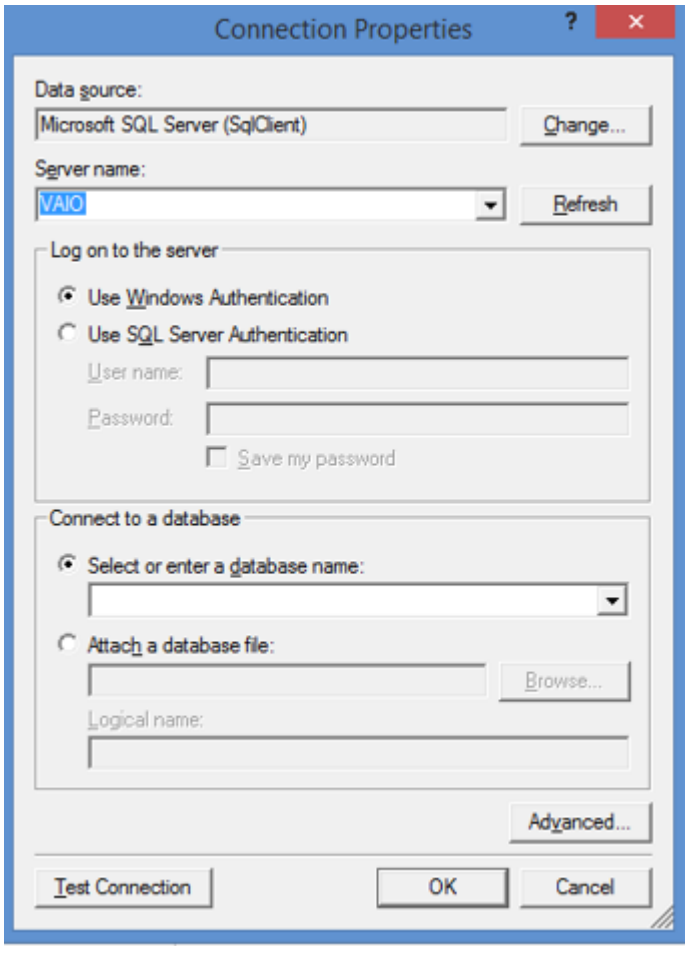

Click on Configure Connection




Click on Connection wizard




Select Data Source Microsoft SQL Server and click on Ok




Here Provide Server name ,Select window Authentication ,Select or enter Database Name and click on Test Connection and Click on Ok

It is used to connect the database with UiPath.It gives us option to write the Connection String and Connnection Provider Name

Once You establish the connection,Connection can be stored in a variable as Database Connection Variable Type which is used Further

## Connection String

Connection String is a collection of Information which is used to connect UiPath(Any Application) with the Database. For Example: ServerName, Database,User Id,Password..etc

**Server Name** :Require Server Name as DataSourse which shows in SSMS while connecting to server.

**Database Name:** Require Database Name as intial Catalog on which we are about to perform Query.

**Windows Authentication:**It is used when you use windows Credential to connect SQL server. Each Data Provider has Different Syntax.

| **Data Provider** | **Syntax** |
|---|---|
| SQL Client: | Integrated Security=True;or Integrated Security=SSPI |
| OLEDB | Integrated Security=SSPI |
| ODBC | Trusted_Connection=Yes |
| OracleClient | Integrated Security=Yes |

**Data Providers Purpose**

System.Data.SQLClient: It Provide Data access For Microsoft SQL server

System.Data.OLED: It Provides data access for Data source exposed using OLE DB

System.Data.ODBC: It Provides data access For Data source exposed using ODBC

System.Data.OracleClient: It Provides data access For Oracle

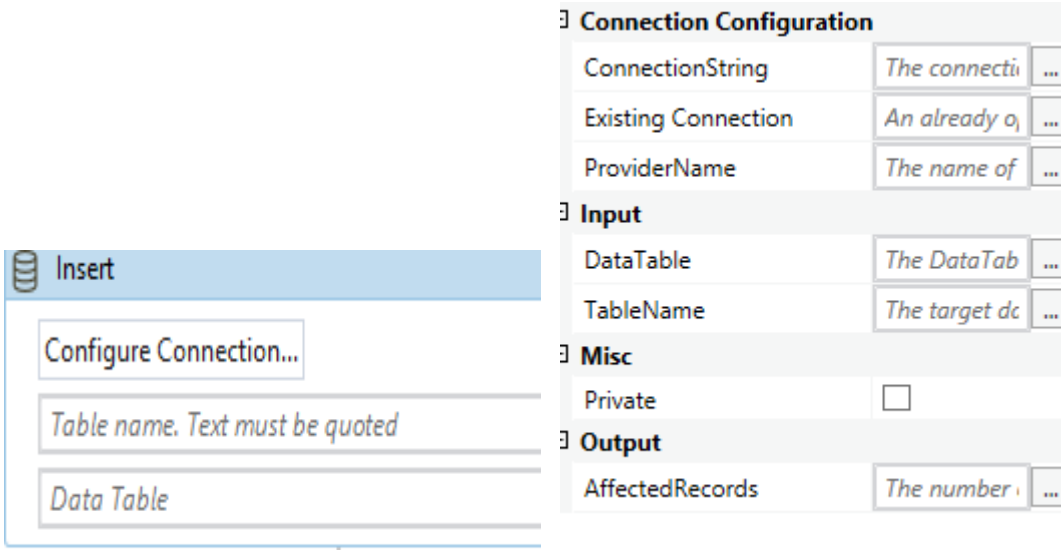
**Insert:**

Here we can insert Excel Data into Database

Inputs are Table Name: which Table we want to insert

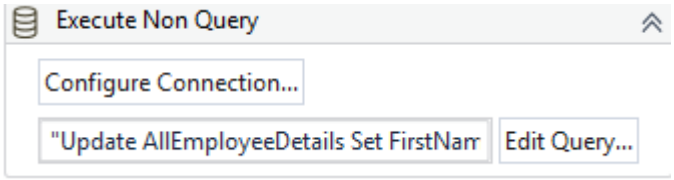
DataTable: which Table we are inserting

In Existing Connection :Provide Connection Variable which is From Connect activity Output




**Execute Non Query in UiPath**

Execute Non Query is used for executing queries that does not return any Data. It is used to execute the SQL statements like Insert, Upadate, Delete etc It returns the number of rows affected



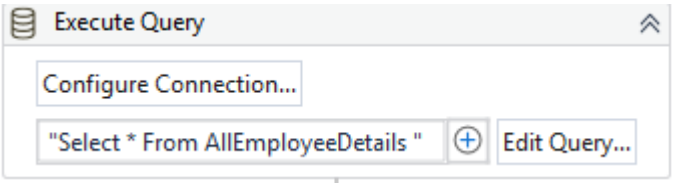

We can write Query in Execute Non Query Activity to insert,Update or delete the Data in SQL Server Database. In the Query, you can pass the variable/parameters to insert Dynamic value to the Database

“insert into Table Name(Name, Value, Description) Values(‘”+Name+”’, ‘”+Value+”’, ‘”+Desc+”’)”

“insert into Table Name(Name, Value, Description) Values(@Name, @Value, @Desc)”

**Execute Query**

It is used For executing queries That returns query results in DataTable. It is used to exxcute the SQL statements like Select Query



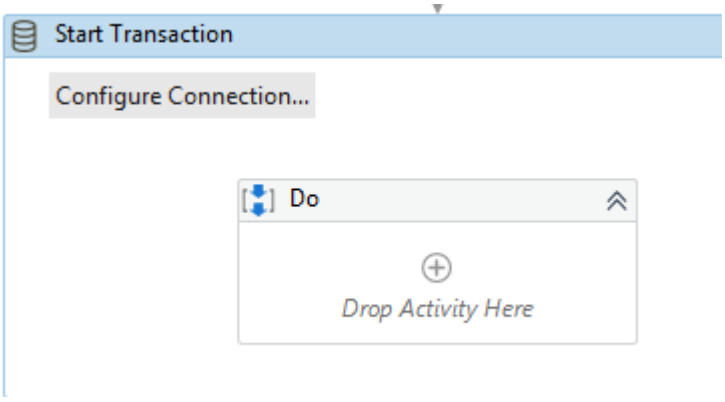

You can write Query in Execute Query Activity to Select the Data From SQL Server DataBase. In the Query, you can pass the variable/Parameters to pass dynamic valu to the Database

“Select * From Table _Name

“Select ColumnName From Table_Name wher Name=’”+Name+””””

**Start Transaction Activity**

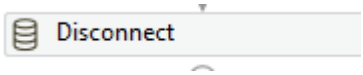
It is a container where we can start our DataBase Transactions.It connects to a Database and perform multiple Transactions with the database.




This activity returns a Database Connection variable. If use Transaction is set true, the Contained operations are executed in a single transaction,  applied at the end if none of them Failed. If use Transaction is set false ,all operations are committed individually. When this activity ends, the Connection to the database is Closed.

**Disconnect**

In Order to disconnect Database connection we can use this activity in UiPath




**For any Documentation, Blog-Writing and RPA Implementation. Please contact rpa@gxplabs.com**