Projektarbete inom affärssystem

PROAFF - 7,5hp HT2023

Stockholms universitet

# Tutorial 3 – UiPath SQL Server Automation

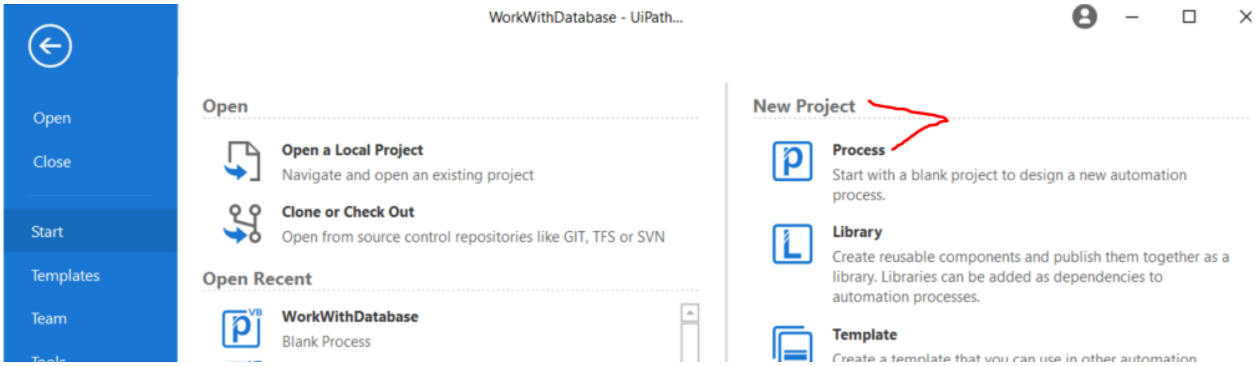
v01

# 1. Introduction

The purpose of this tutorial is to teach you how to automate data manipulation using SQL Server, such as storing and retrieving data to and from Microsoft SQL Server. The inputs and outputs can come from web scraping, emails, files, and Excel.

## 1.1.Create a New Project

- Start UiPath, select Process under **New Project** from the Start tab as illustrated below.



Open

**Open a Local Project**
Navigate and open an existing project

**Clone or Check Out**
Open from source control repositories like GIT, TFS or SVN

Open Recent

**WorkWithDatabase**
Blank Process

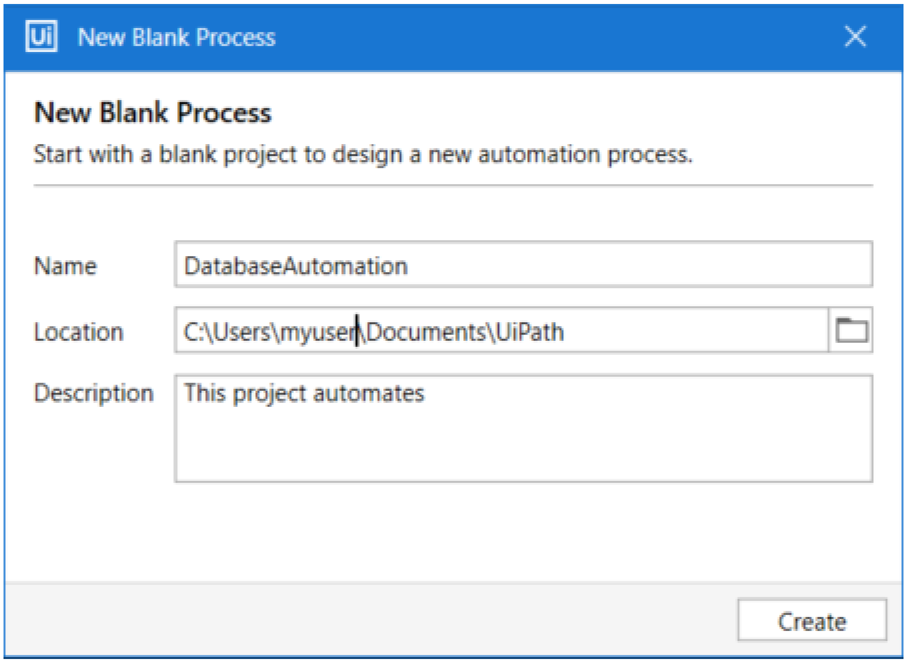
New Project

**Process**
Start with a blank project to design a new automation process.

**Library**
Create reusable components and publish them together as a library. Libraries can be added as dependencies to automation processes.

**Template**
Create a template that you can use in other automation

- In the **New Blank Process** dialog box illustrated below, enter “DatabaseAutomation” under **Name.**



New Blank Process

Start with a blank project to design a new automation process.

Name: DatabaseAutomation

Location: C:\Users\myuser\Documents\UiPath

Description: This project automates

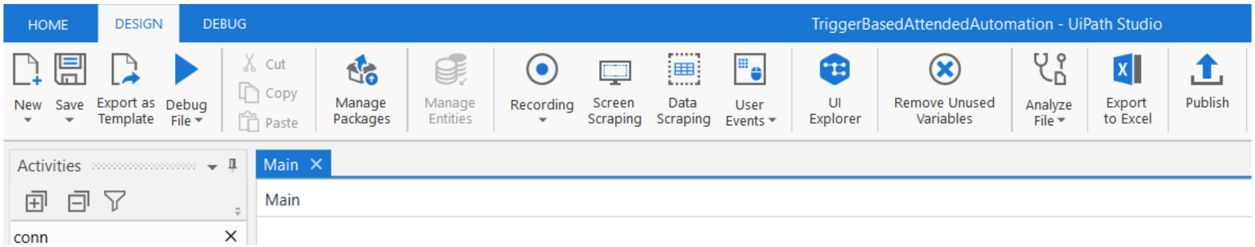
- Specify the location of your file under **Location.**
- Specify what your robot will do under **Description.**
- Click on **Create.**

## 1.2.Installing Database Activities

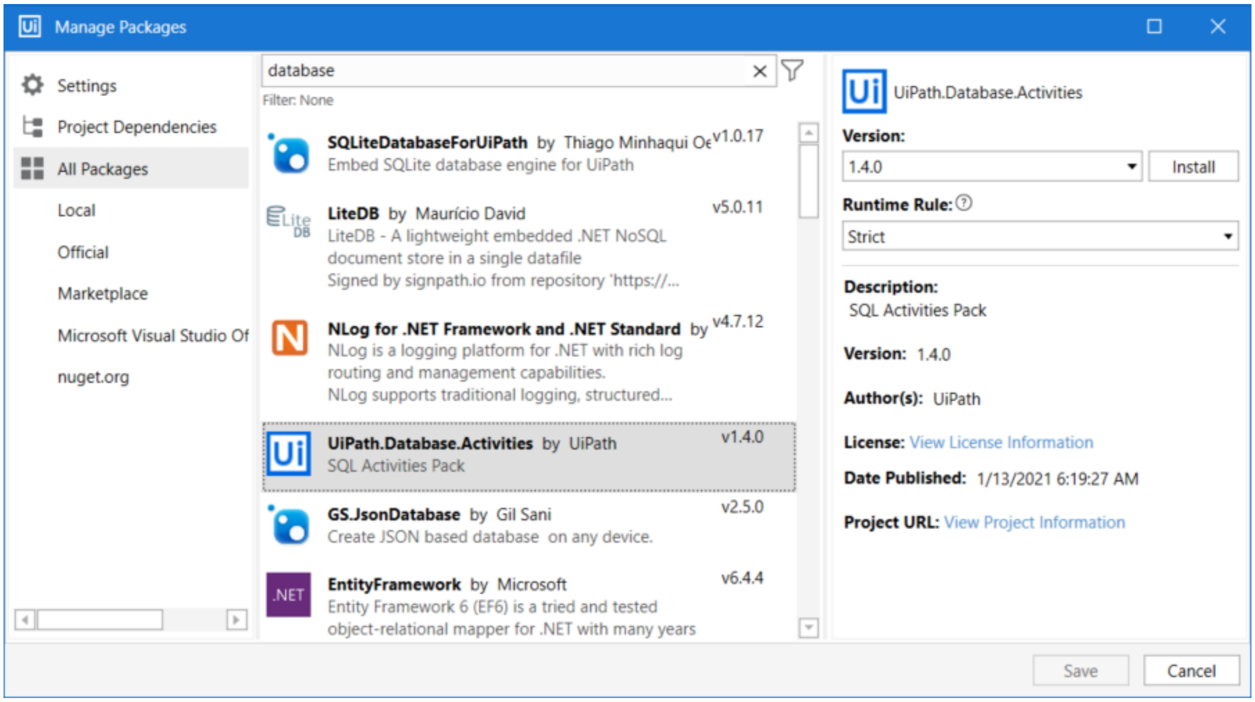
Database activities are not installed and are not available in UiPath studio by default. Follow the following steps to install **Database Activities** in your UiPath studio.

### Steps

- Under **Design** ribbon, select **Manage Packages**



- Select **All Packages**
- Enter “database sql” under the search box in the **Manage Packages** dialog box



database

Filter: None

**SQLiteDatabaseForUiPath** by Thiago Minhaqui Oє v1.0.17
Embed SQLite database engine for UiPath

**LiteDB** by Maurício David v5.0.11
LiteDB - A lightweight embedded .NET NoSQL document store in a single datafile
Signed by signpath.io from repository 'https://...

**NLog for .NET Framework and .NET Standard** by v4.7.12
NLog is a logging platform for .NET with rich log routing and management capabilities.
NLog supports traditional logging, structured...

**UiPath.Database.Activities** by UiPath v1.4.0
SQL Activities Pack

**GS.JsonDatabase** by Gil Sani v2.5.0
Create JSON based database on any device.

**EntityFramework** by Microsoft v6.4.4
Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years

UiPath.Database.Activities

**Version:** 1.4.0

**Runtime Rule:** Strict

**Description:**
SQL Activities Pack

**Version:** 1.4.0

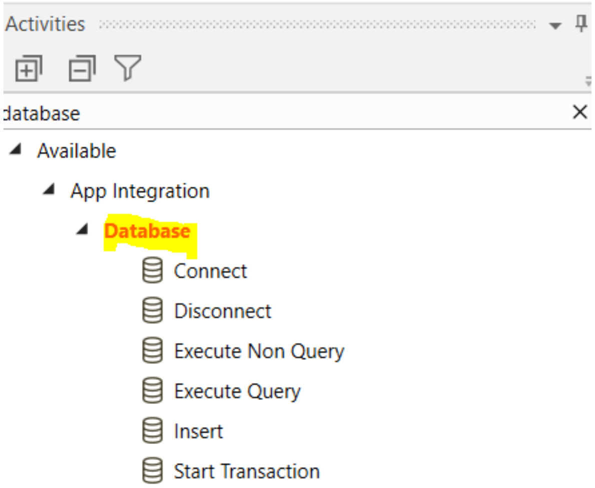
**Author(s):** UiPath

**License:** View License Information

**Date Published:** 1/13/2021 6:19:27 AM

**Project URL:** View Project Information

- Select **UiPath.Database.Activities**
- Click on **Install**
- Click **Save**
- Now you will get Database Activities Under **Activities**, as illustrated below



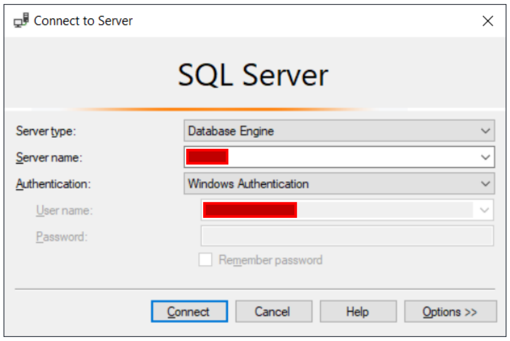
Activities

database

- Available
  - App Integration
    - Database
      - Connect
      - Disconnect
      - Execute Non Query
      - Execute Query
      - Insert
      - Start Transaction

## 1.3.Create Database UiPath and a Table

- From the Start menu, start Microsoft SQL Server Management Studio
- Select **Server type**: Database Engine
- Select **Server name**: in your case – you could enter “localhost” or your machine name
- Select **Authentication**: Windows Authentication
- Click on **Connect**



Server type: Database Engine

Server name: [redacted]

Authentication: Windows Authentication

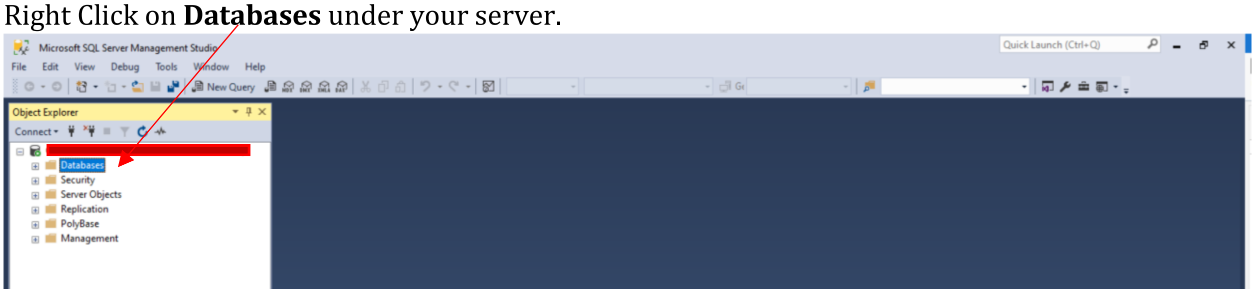
User name: [redacted]

Password:

□ Remember password

### Creating Databases

- Right Click on **Databases** under your server.

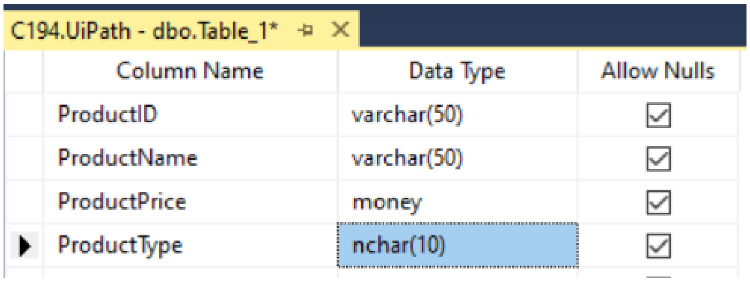


[redacted]

- Select **New Databases…**
- Enter UiPath under **Database name:**
- Click on **OK**

### Creating Table

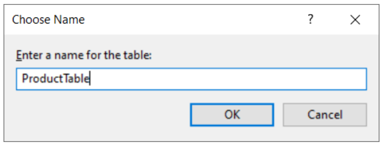
- Rick Click on Tables under UiPath database > Click on **Table…**
- Under **Column Name** enter column names and corresponding **Data Types** as illustrated below.



C194.UiPath - dbo.Table_1*

| | Column Name | Data Type | Allow Nulls |
|---|---|---|---|
| | ProductID | varchar(50) | ☑ |
| | ProductName | varchar(50) | ☑ |
| | ProductPrice | money | ☑ |
| ▶ | ProductType | nchar(10) | ☑ |

- Save the table as – *ProductTable*

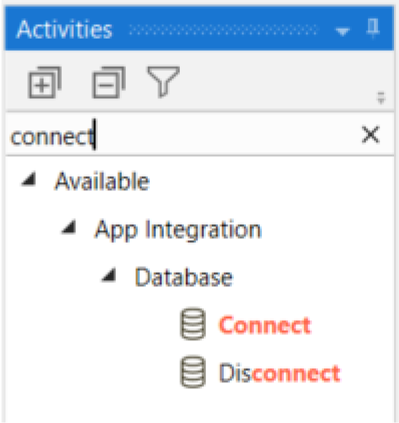


Choose Name

Enter a name for the table: ProductTable

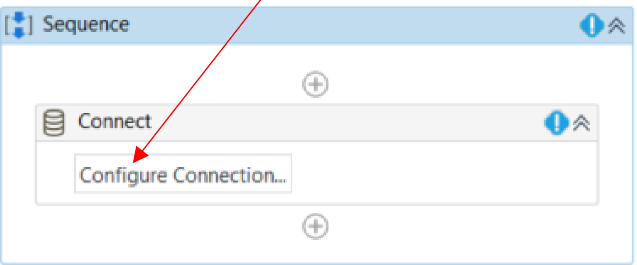
## 1.4.Create Connection to the UiPath Database in UiPath Studio

- Create a connection with the SQL server
  - From the Activities panel, add **Connect** activity to the Designer panel

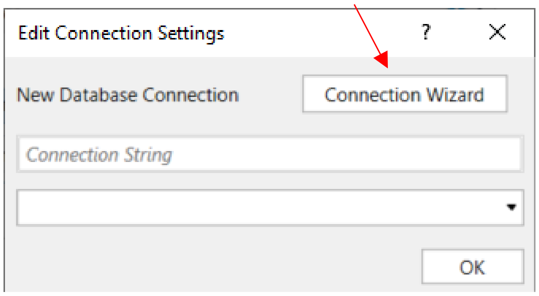


connect

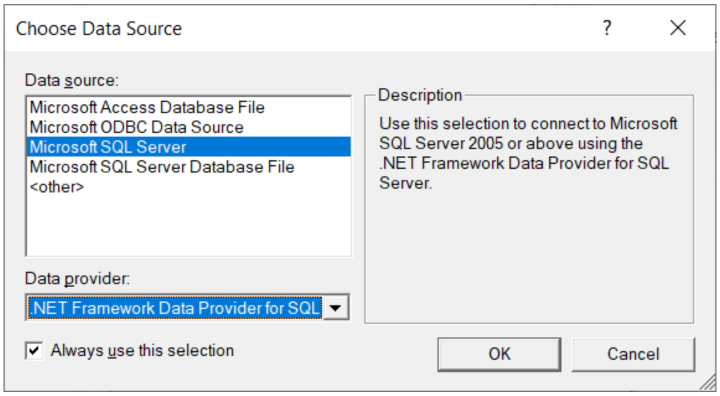
  - Click on **Configure Connection**



- Edit **Connection Settings** dialog box will be displayed, and from this dialog box select  or click on **Connection Wizard**



- Select **Microsoft SQL Server** from the Data Source list as illustrated below
- Select .NET Framework DATA Provider for SQL Server from **Data provider** list box


Choose Data Source
?
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server Database File
<other>
Description
OK
Cancel

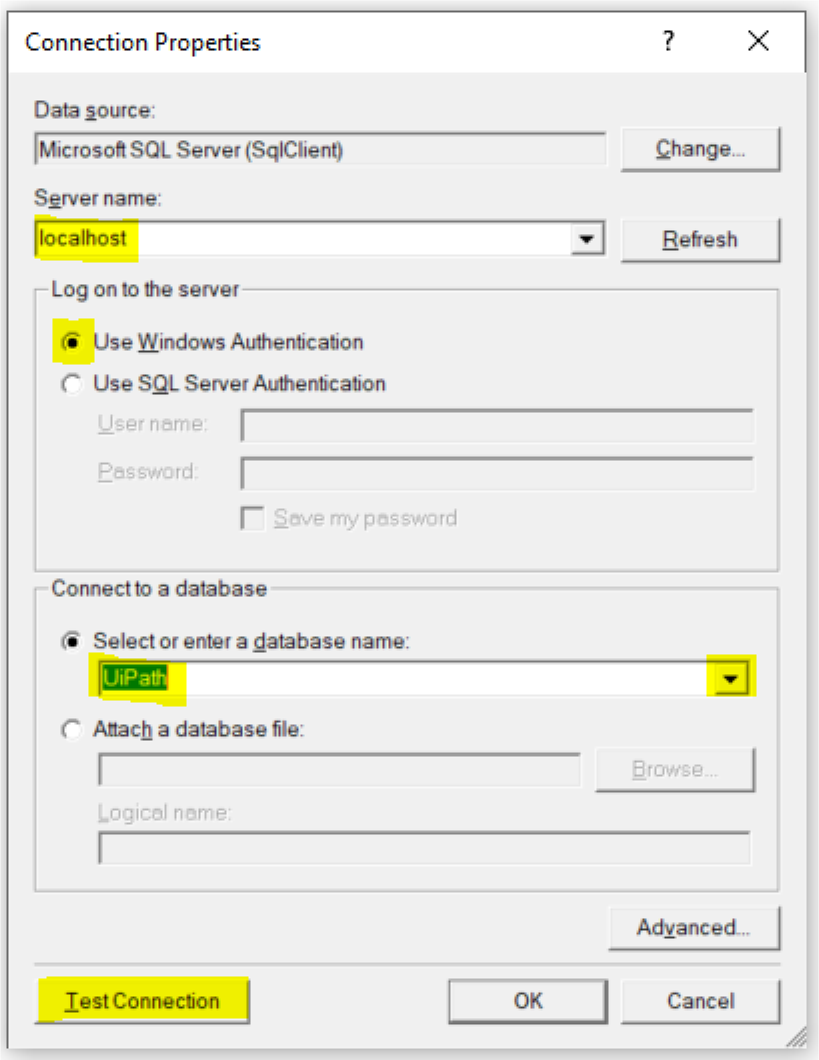
Data source: Microsoft SQL Server

Use this selection to connect to Microsoft SQL Server 2005 or above using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for SQL

☑ Always use this selection

- Click **OK**
- **Connection Properties** dialog box will pop up as illustrated below, enter localhost under **Server name:**
- Under **Log on to the server**, select the **Use Windows Authentication**
- Under **Connect to database**, select the database "UiPath"



Connection Properties

Data source:

Microsoft SQL Server (SqlClient)

Server name:

localhost

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

□ Save my password

Connect to a database

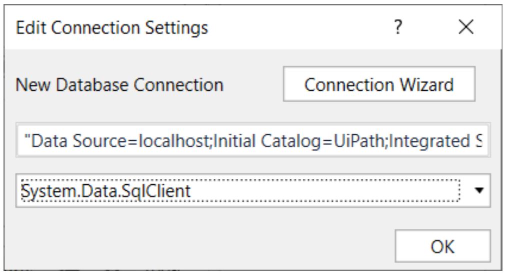
● Select or enter a database name:

UiPath

○ Attach a database file:

Logical name:

- Click on the **Test Connection button**
- If the connection is successful, you will get the notification “Test connection succeeded.”
- Click **OK** two times, then you
- From the **Edit Connection Settings**, select System.Data.SqlClient as illustrated below and click **OK**



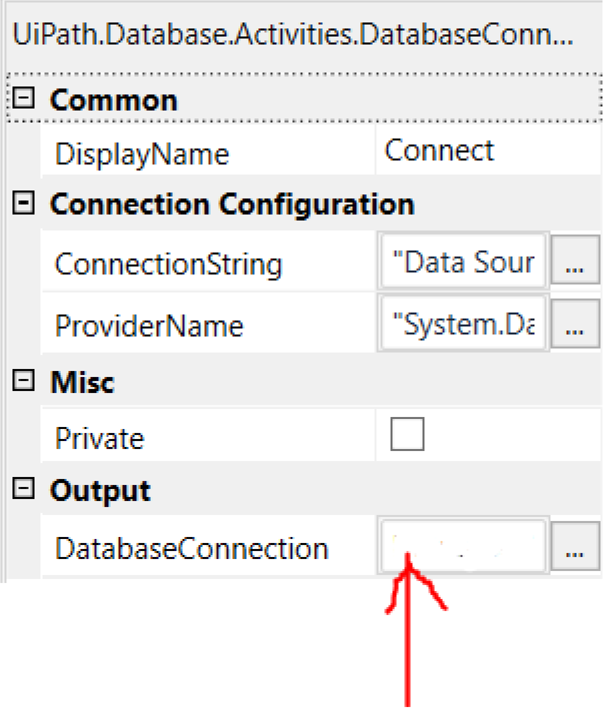
Edit Connection Settings

New Database Connection

"Data Source=localhost;Initial Catalog=UiPath;Integrated S

System.Data.SqlClient

- In the Properties panel, you can see that Connection Configuration information such as Connection String and Provider Name is already specified, as you did in the previous steps. You need to declare a variable and assign it to the **Output** under the **DatabaseConnection** -
    i. Click in **DatabaseConnection** textbox under **Output** > press **Ctrl** + **K** (this will enable you to create a variable from here)



UiPath.Database.Activities.DatabaseConn...

⊟ **Common**

DisplayName: Connect

⊟ **Connection Configuration**

ConnectionString: "Data Sour ...

ProviderName: "System.Da ...

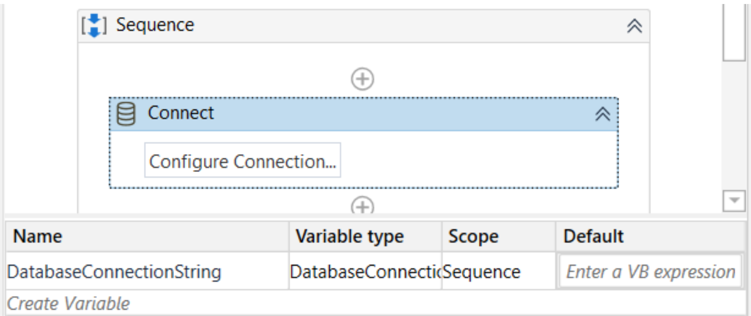
⊟ **Misc**

Private: □

⊟ **Output**

DatabaseConnection: ...

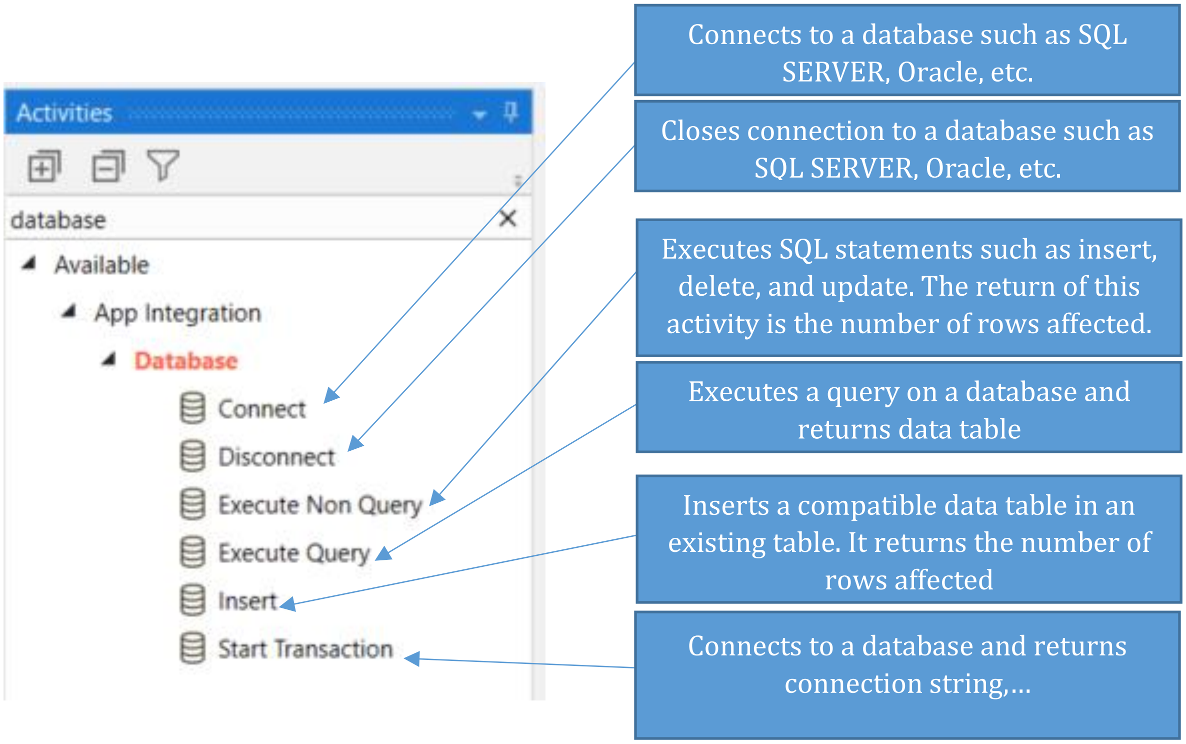
    ii. Enter **DatabaseConnectionString** as your new variable
    iii. If you click on the Variables panel, you will see that the variable created data type is DatabaseConnection. The connection string is stored in this variable.



| Name | Variable type | Scope | Default |
|---|---|---|---|
| DatabaseConnectionString | DatabaseConnecti | Sequence | Enter a VB expression |
| Create Variable | | | |

## 1.5.Add Execute Non Query Activity

Database activities and their use are illustrated below:



Connects to a database such as SQL SERVER, Oracle, etc.

Closes connection to a database such as SQL SERVER, Oracle, etc.

Executes SQL statements such as insert, delete, and update. The return of this activity is the number of rows affected.

Executes a query on a database and returns data table

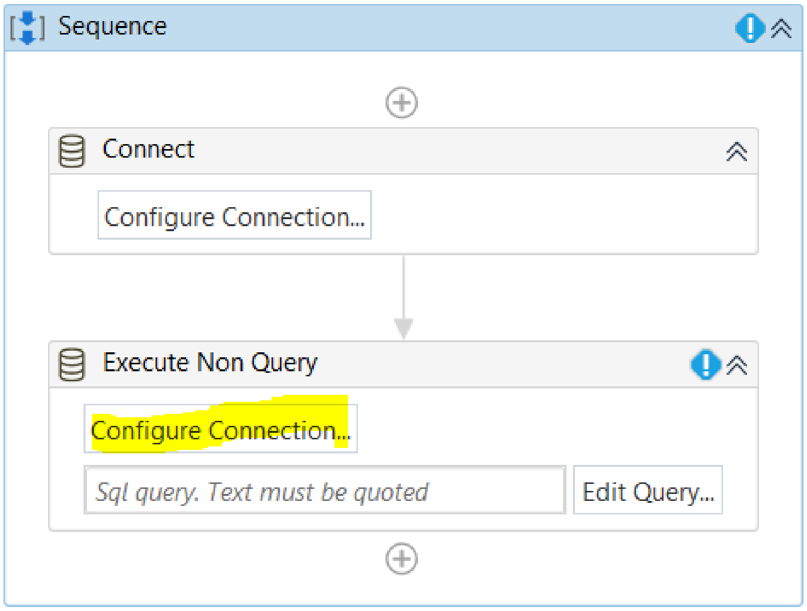
Inserts a compatible data table in an existing table. It returns the number of rows affected

Connects to a database and returns connection string,…

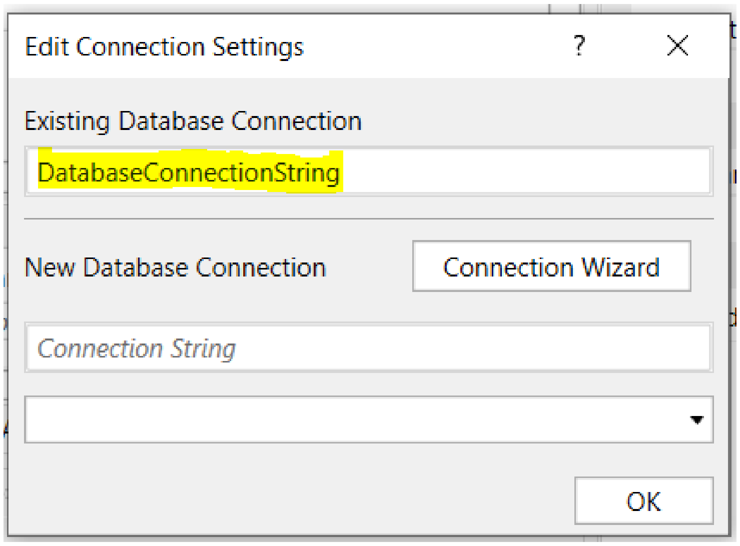
### Add Execute Non Query Steps

Now you can add “**Execute Query**” OR “**Execute Non Query**”. The **Execute Non Query** includes statements that do not return data or rows such as Insert, Delete and Update, but it returns how many rows are affected.

- Add **Execute Non Query** under **Connect** activity as illustrated below.



- Click on **Configure Connection**
- Set Existing Database Connection, by entering - DatabaseConnectionString as illustrated below > click OK



Edit Connection Settings

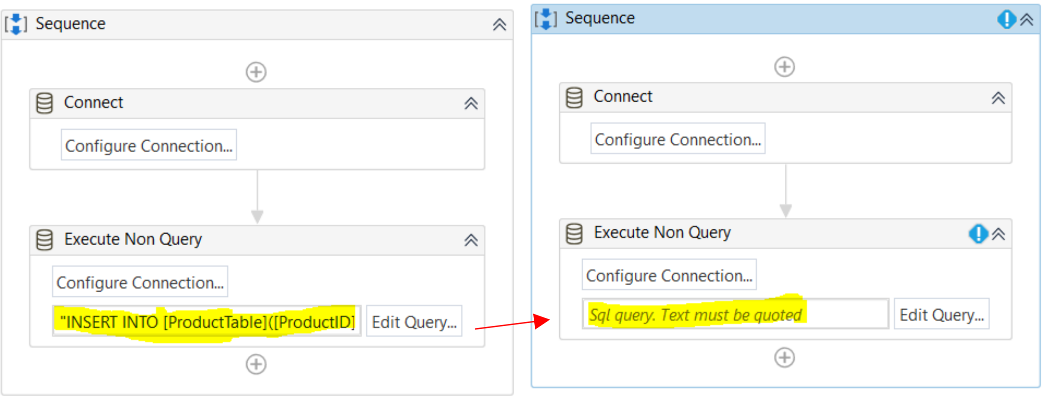
Existing Database Connection

DatabaseConnectionString

New Database Connection

Connection String

- Under SQL query textbox enter the query in double quotes, where it is highlighted in yellow as illustrated below. For the query use - *"INSERT INTO [ProductTable]([ProductID],[ProductName],[ProductPrice],[ProductType]) VALUES ('P001','Printer', 5666, 'Hardware')"*

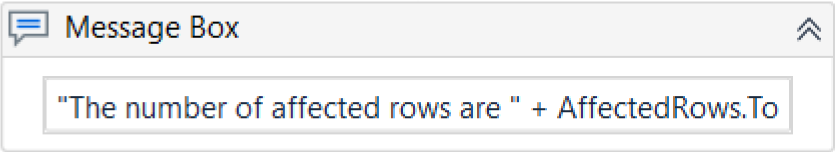


"INSERT INTO [ProductTable]([ProductID]

Sql query. Text must be quoted

## Add a Message Box

- Define a variable "AffectedRows" with a datatype Int32
- Create a message box that shows the number of affected rows.
    - Add a **Message Box** activity below **Execute Non Query** activity to display the number of rows affected to the user.
- Add the text - "The number of affected rows is " + AffectedRows.ToString(), in the text box as illustrated below.



"The number of affected rows are " + AffectedRows.To

**Exercise**

- Add **Execute Query** below the Message Box,
- Add a SQL that selects all records from the *ProductTable.* The SQL query for the **Execute Query** activity is – "Select * from ProductTable"
- Define a *DataTable* variable with the name *ProductT*, use (CTRL+K) after clicking on **DataTable**, which is found under the **Output** of the Properties panel of the **Execute Query** activity.
- Set the database connection (use existing connection)
- Finally, store the result of the **Execute Query**, i.e., the whole data stored in the database, in an excel file with a file name "ImportedFromSQL.xlsx"
- Finally, add **Disconnect** activity to disconnect the database connection.