Deployment Instructions
SIX hotel calculator

- Obtaining the files for the Calculator and putting them in the right place

To download the compressed archive, please access it at the following link; "http://dev.welikepie.com/hotelCalculator/files.zip" and unpack it to any empty folder.

Once the folder has been unpacked there should be three folders; a 'scripts' folder, an 'images' folder and a 'styles' folder. Each folder should contain between 2 and 4 files, each prefixed with "WLPCalc" to differentiate them from pre-existing site files.

Now, we copy the scripts, styles and images folders to the same level as the page you're installing the calculator element on. This means that if you can see the file for your page in File Explorer, you should also be able to see the images, styles and scripts folders as being in the same folder. It is imperative that all three folders remain in the same location relative to each other.

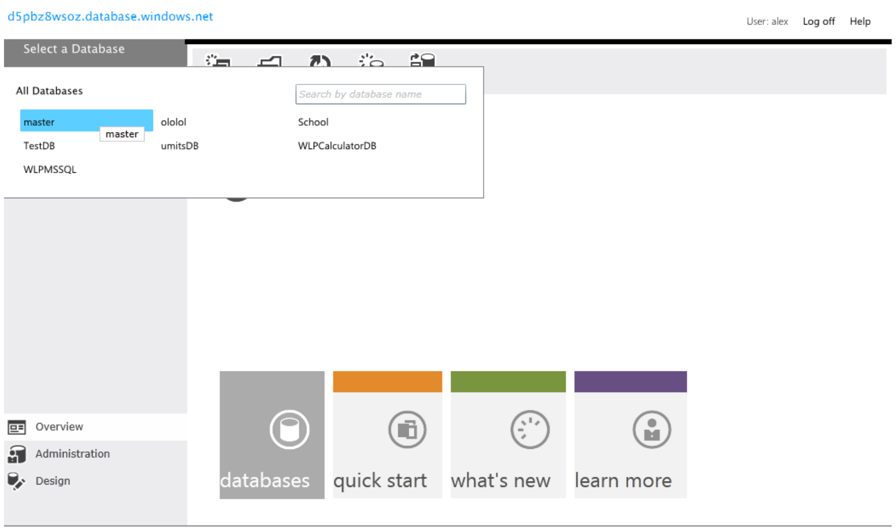
- Notes on deploying the database

Depending on which database type you wish to deploy to depends on which of the following instructions to follow, owing to Azure and server based databases having their own quirks and methods. If you are using the Windows Azure cloud service for your database needs, please reference the **Deploy Database to Microsoft Azure SQL database** section. If, however, your database is being kept on your company's servers, please reference the **Deploy Database to None-Azure Microsoft SQL database** section.

- Deploy database to Microsoft Azure SQL database

N.B for deploying to Azure; Only the server-level principal login (created during the server creation process) or members of the dbmanager database role can create databases. Deploying to non-azure database is much easier.
Warning; Azure is not a free service. Please make sure that, before deploying, you have corporate permission to create a new database and table for this application, as it will cost money.

The first thing to do is make sure to create a new database. To do this, we will need to connect to the instance of MSSQL that we wish to create a new database on.



If you already know what the URL is to your Azure SQL instance, then access that through a web browser (will take format xxxxxxxxxx.database.windows.net). You will also need a username and password to access this. If you do not know what these are, please reference the "**Retrieving User Credentials**" section.

After having successfully logged in to the database server we wish to deploy on, we must click the ‘select database’ option in the top left corner, and select ‘master’, as in the above image.

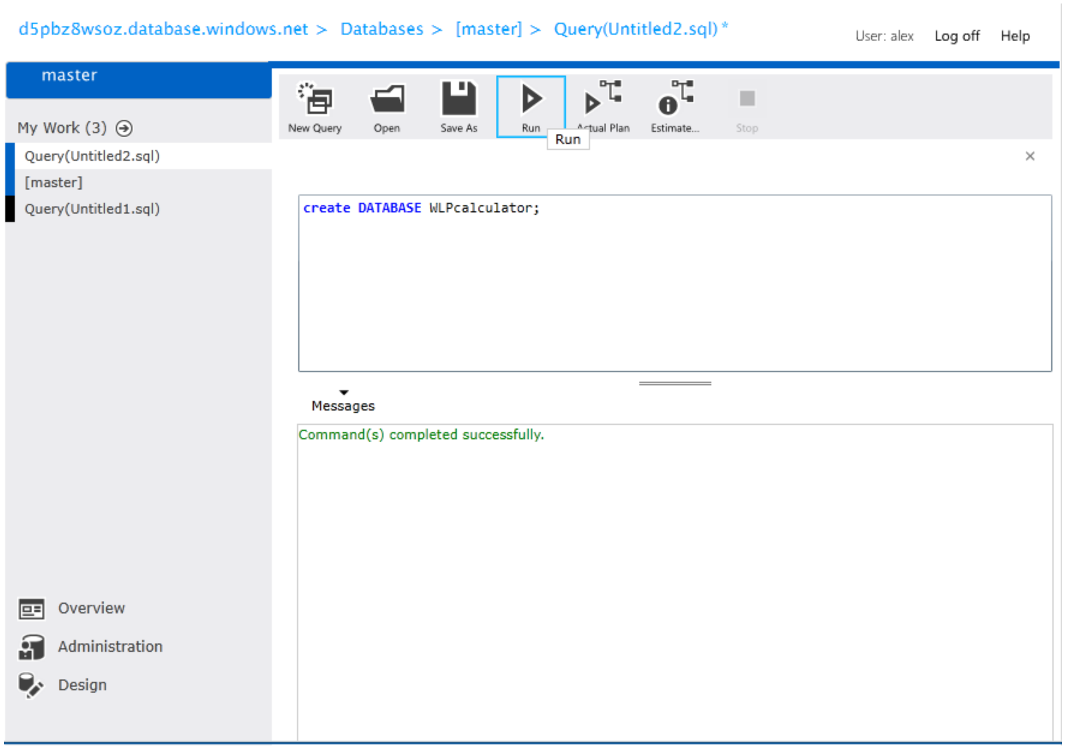
Once we have selected this, we click the ‘new query’ button and use the following code, typing it into the text input field and then clicking “run”.

Code;

“

```
        create DATABASE WLPcalculator;
```

“



```
create DATABASE WLPcalculator;
```

```
Command(s) completed successfully.
```

The result should look as follows, which means our new database has been created.

This database will now show up when we go to select it in the same menu as we selected the master database from before. Once we have selected our new database, we carry out another query with the following code;

Code;
“
```
		if not exists (select * from INFORMATION_SCHEMA.TABLES where TABLE_NAME =
		'WLPCalcTable')
		CREATE TABLE [WLPCalcTable](
		 [ID] [int] IDENTITY(1,1) NOT NULL,
		 [numberofrooms] [int] NOT NULL,
		 [currency] [nvarchar](15) NOT NULL,
		 [averageroomrate] [float] NOT NULL,
		 [contactInput] [nvarchar](50) NULL,
		 [emailInput] [nvarchar](50) NULL,
		 [phoneInput] [nvarchar](50) NULL,
		 [companyInput] [nvarchar](50) NULL,
		 [transactionCountOutput] [float] NULL,
		 [ATVOutput] [float] NULL,
		 [turnoverOutput] [float] NULL,
		 [valueMinOutput] [float] NULL,
		 [upsellOutput] [float] NULL,
		 PRIMARY KEY CLUSTERED
		(
		 [ID] ASC
		)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
		ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
		)else print 'Table already exists in this Database!'
```
“

Once this code has been run, the output will either read “Command(s) completed successfully.” to indicate the table has been created, or “Table already exists in this Database”, to indicate that the table is already present.

If the commands have completed successfully, we’re finished here and have created the table and the database which we will be writing to with the calculator element.

If, however, the table already exists, we will need to make sure that the already present table adheres to the specifications of the table that we wish to create.

To this end, we need to access the table’s column names to make sure it’s the correct table. To do this, we can run the command

Code;
“
```
		select * from WLPCalcTable;
```
”

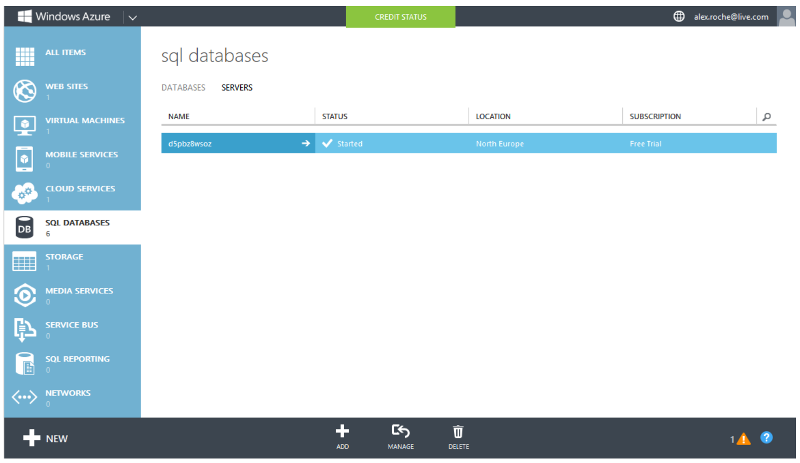
This will return a list of column names, along with any information stored in the database. Additionally, data being present in the table indicates that the deployment has already been successfully carried out and there is at least one instance of the Calculator running and the database deployment process does not need to be carried out. The column names should match the following list and, if there is data present, match the types specified;

Field Name	**Field Type**
ID	Integer
numberofrooms	Integer
currency	Character String
averageroomrate	Floating Point number
contactInput	Character String
emailInput	Character String (Email address)
phoneInput	Character String (Telephone Number)
companyInput	Character String
transationCountOutput	Floating Point Number
ATVOutput	Floating Point Number
turnoverOutput	Floating Point Number
valueMinOutput	Floating Point Number
upsellOutput	Floating Point Number

If, however, this is not the case, then it is indicative that there has been an error in setting up the database, or a table has been misplaced. If this is the case, it is possible that someone else has been working in the code to modify it. If you are collaborating with someone, please get in touch with them.

- Retrieving user credentials for Microsoft Azure

If you know neither SQL URL, username, or password, then you will need to log in to the main Azure instance by accessing https://manage.windowsazure.com . Now you will need to log in using the Microsoft Account associated with Azure. Once you have successfully logged in, you will be at the admin panel for the Azure instance.



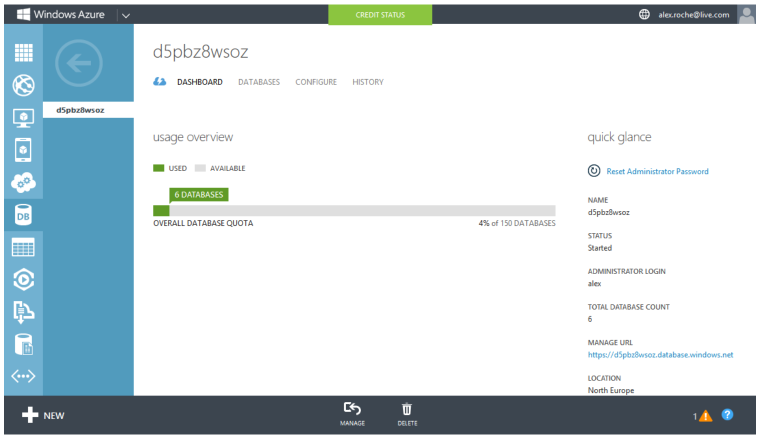
sql databases

DATABASES SERVERS

NAME	STATUS	LOCATION	SUBSCRIPTION
d5pbz8wsoz	✓ Started	North Europe	Free Trial

Once the dashboard is open, we click “SQL Databases” in the left hand column, and then select the database we wish to deploy to by clicking the arrow next to the name of the database we wish to use. If there are no databases present, please reference **How to create a Database Server with Azure**.

Clicking the arrow associated with the name will then bring up a page specific for the database;



d5pbz8wsoz

usage overview

🟩 USED ▨ AVAILABLE

6 DATABASES

OVERALL DATABASE QUOTA — 4% of 150 DATABASES

quick glance

Reset Administrator Password

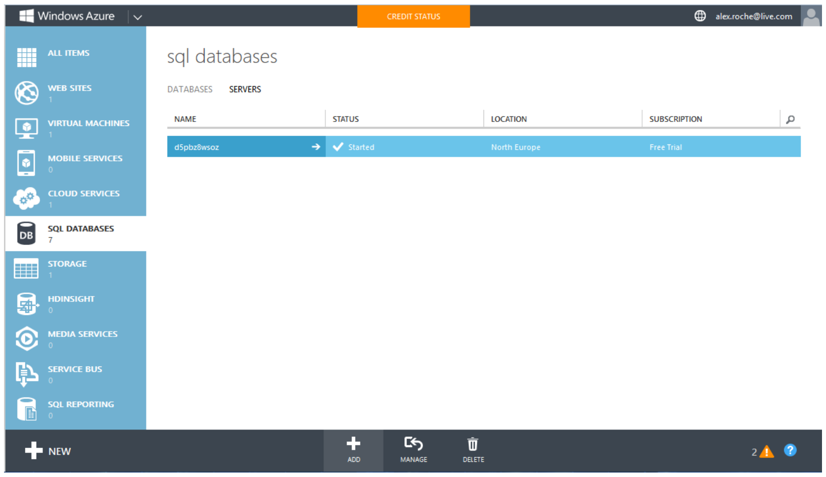
NAME	d5pbz8wsoz
STATUS	Started
ADMINISTRATOR LOGIN	alex
TOTAL DATABASE COUNT	6
MANAGE URL	https://d5pbz8wsoz.database.windows.net
LOCATION	North Europe

From here, we are now able to get to the Database Server we want to set up our Database on by using the Manage URL, and the Administrator login. If nobody knows the password, this can be reset using the 'Reset Administrator Password' option, which, when clicked, summons a prompt to set the password to something new.

Now that we have reset the password, we can log in to the server as we previously intended.

- How to create a database server with Microsoft Azure

To create a new Database Server, you will need to log in to the main Azure account by accessing https://manage.windowsazure.com. Now you will need to log in using the Microsoft Account associated with Azure. Once you have successfully logged in, you will be at the admin panel for the Azure instance.

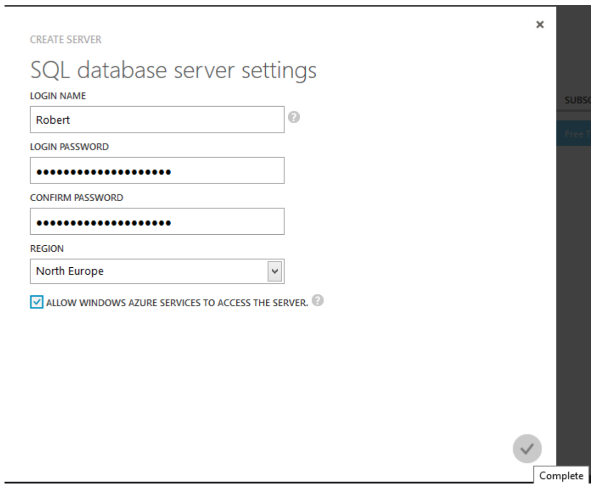


sql databases

NAME	STATUS	LOCATION	SUBSCRIPTION
d5pbz8wsoz	✓ Started	North Europe	Free Trial

Once we have selected the "SQL Databases" tab option on the right hand side, we then need to click on the "Add" option in the bottom bar, highlighted in the above image. Clicking this will result in a modal dialogue popping up requiring information about the new type of server you wish to create.

You will need to specify login information to use when connecting to the server, which is important to keep on hand. In addition, the server region dictates which geographical location the server will be located in. Once the "Complete" button is clicked, the server will be created and will appear in the SQL databases overview once the database server has been created, as shown in the below image.

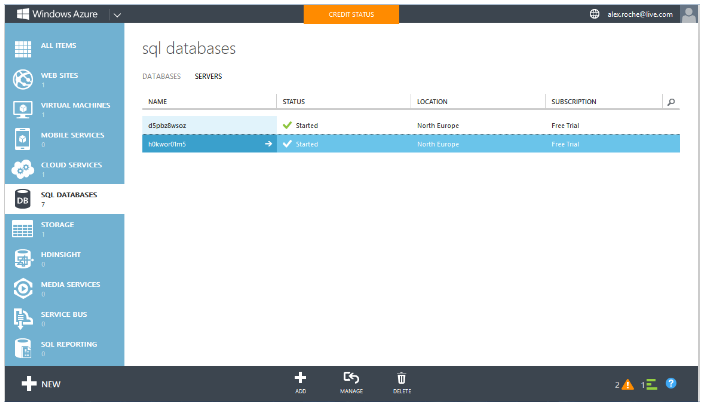


CREATE SERVER

SQL database server settings

LOGIN NAME	Robert
LOGIN PASSWORD	••••••••••••••••••••
CONFIRM PASSWORD	••••••••••••••••••••
REGION	North Europe

☑ ALLOW WINDOWS AZURE SERVICES TO ACCESS THE SERVER.



sql databases

DATABASES SERVERS

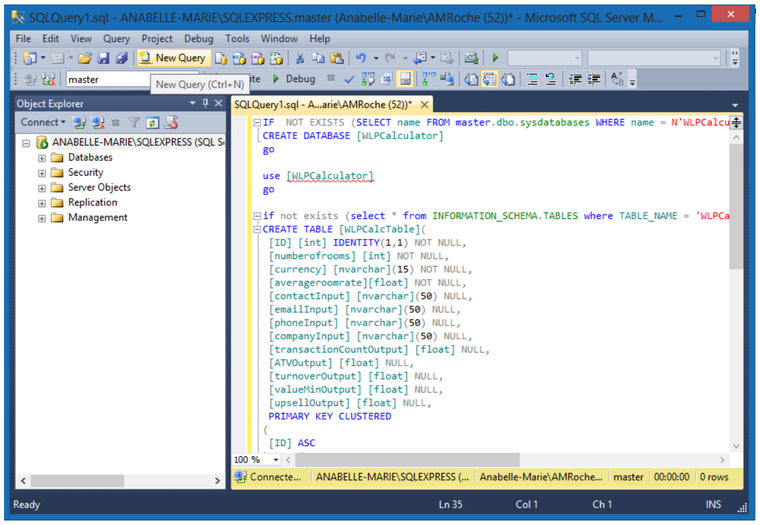
NAME	STATUS	LOCATION	SUBSCRIPTION
d5pbz8wsoz	✓ Started	North Europe	Free Trial
h0kwor01m5	✓ Started	North Europe	Free Trial

At this point, we can return to previous instructions having just created a database server that we can manage and log in to with the generated login credentials in order to create our database.

- Deploy Database to non-azure Microsoft SQL database

If you are not using the Azure database service and are in fact running a microsoft SQL database locally or on a server that you can connect to with a Microsoft SQL client, then the process of deploying the database is greatly simplified; we only need run one script and everything we need is created.

For further instructions connecting to the database, please reference the **Connecting to a non-Azure MSSQL database** section.


SQLQuery1.sql - ANABELLE-MARIE\SQLEXPRESS.master (Anabelle-Marie\AMRoche (52))* - Microsoft SQL Server M...
File Edit View Query Project Debug Tools Window Help
New Query
master
New Query (Ctrl+N)
Debug
Object Explorer
Connect
ANABELLE-MARIE\SQLEXPRESS (SQL S
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - A...arie\AMRoche (52))*
100 %
Connecte...
ANABELLE-MARIE\SQLEXPRESS (...
Anabelle-Marie\AMRoche...
master
00:00:00
0 rows
Ready
Ln 35
Col 1
Ch 1
INS

```
IF  NOT EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'WLPCalcu
CREATE DATABASE [WLPCalculator]
go

use [WLPCalculator]
go

if not exists (select * from INFORMATION_SCHEMA.TABLES where TABLE_NAME = 'WLPCa
CREATE TABLE [WLPCalcTable](
 [ID] [int] IDENTITY(1,1) NOT NULL,
 [numberofrooms] [int] NOT NULL,
 [currency] [nvarchar](15) NOT NULL,
 [averageroomrate][float] NOT NULL,
 [contactInput] [nvarchar](50) NULL,
 [emailInput] [nvarchar](50) NULL,
 [phoneInput] [nvarchar](50) NULL,
 [companyInput] [nvarchar](50) NULL,
 [transactionCountOutput] [float] NULL,
 [ATVOutput] [float] NULL,
 [turnoverOutput] [float] NULL,
 [valueMinOutput] [float] NULL,
 [upsellOutput] [float] NULL,
 PRIMARY KEY CLUSTERED
(
 [ID] ASC
```

Once we are connected to the SQL server, we need only create a new query and run the code to create the database and table that we require. This is a case of clicking the "New Query" button, and copying the next code sample into the query window.

Code;

"

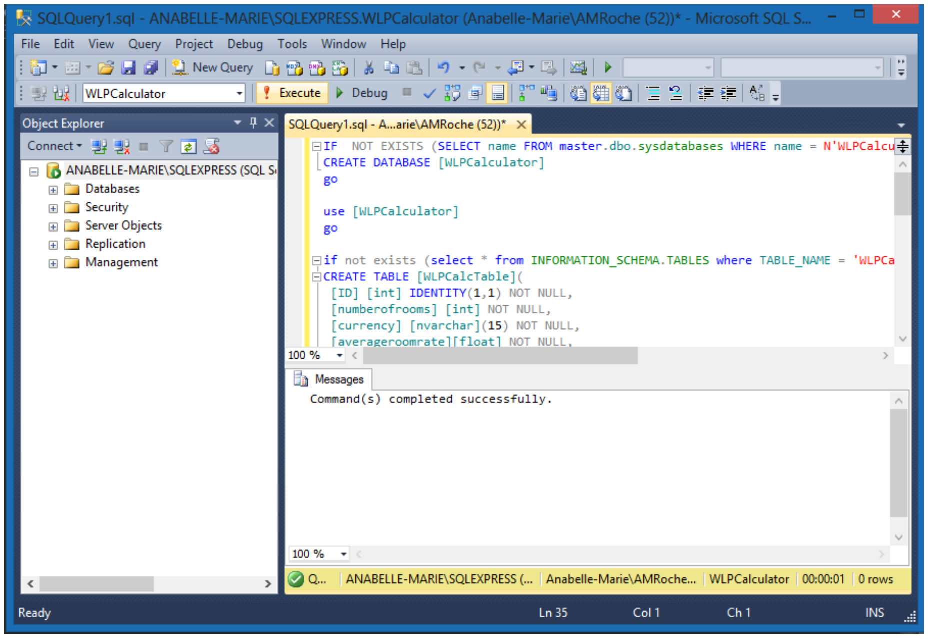
```
IF  NOT EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name =
N'WLPCalculator')
CREATE DATABASE [WLPCalculator]
go

use [WLPCalculator]
go
```

```
if not exists (select * from INFORMATION_SCHEMA.TABLES where TABLE_NAME =
'WLPCalcTable')
CREATE TABLE [WLPCalcTable](
 [ID] [int] IDENTITY(1,1) NOT NULL,
 [numberofrooms] [int] NOT NULL,
 [currency] [nvarchar](15) NOT NULL,
 [averageroomrate][float] NOT NULL,
 [contactInput] [nvarchar](50) NULL,
 [emailInput] [nvarchar](50) NULL,
 [phoneInput] [nvarchar](50) NULL,
 [companyInput] [nvarchar](50) NULL,
 [transactionCountOutput] [float] NULL,
 [ATVOutput] [float] NULL,
 [turnoverOutput] [float] NULL,
 [valueMinOutput] [float] NULL,
 [upsellOutput] [float] NULL,
 PRIMARY KEY CLUSTERED
(
 [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
)else print 'Table already exists in this Database!'
GO

SET ANSI_NULLS OFF
GO

SET QUOTED_IDENTIFIER ON
GO
```
"



```
IF  NOT EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'WLPCalcu
CREATE DATABASE [WLPCalculator]
go

use [WLPCalculator]
go

if not exists (select * from INFORMATION_SCHEMA.TABLES where TABLE_NAME = 'WLPCa
CREATE TABLE [WLPCalcTable](
 [ID] [int] IDENTITY(1,1) NOT NULL,
 [numberofrooms] [int] NOT NULL,
 [currency] [nvarchar](15) NOT NULL,
 [averageroomrate][float] NOT NULL,
```

Messages

```
Command(s) completed successfully.
```

SQLQuery1.sql - ANABELLE-MARIE\SQLEXPRESS.WLPCalculator (Anabelle-Marie\AMRoche (52))* - Microsoft SQL S...
File Edit View Query Project Debug Tools Window Help
New Query
WLPCalculator
Execute
Debug
Object Explorer
Connect
ANABELLE-MARIE\SQLEXPRESS (SQL S
Databases
Security
Server Objects
Replication
Management
SQLQuery1.sql - A...arie\AMRoche (52))*
100 %
Q... ANABELLE-MARIE\SQLEXPRESS (... Anabelle-Marie\AMRoche... WLPCalculator 00:00:01 0 rows
Ready Ln 35 Col 1 Ch 1 INS

With that code inserted into the SQL Query area and having been executed, the database and table contained therein should now be ready to use and “Command(s) completed successfully” should be displayed in the Messages tab, as in the image.

To verify that the database and table have been created correctly (or if the message box reads ‘Table already exists in this Database’), we will now run an additional command;

Code;

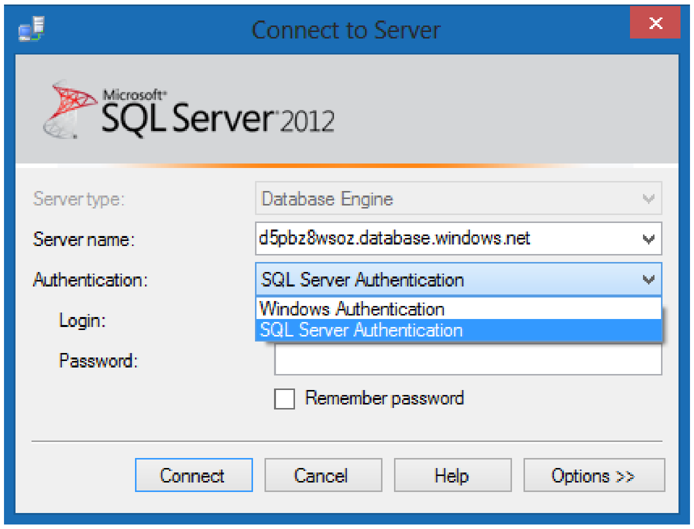
“
```
        use WLPCalculator;
        SELECT * From WLPCalcTable;
        go
```
“

This command, upon successful completion, should display table headings and, if present, data that has been submitted to the database. If there is data, then the data types and column headings should match the following table;

Field Name	**Field Type**
ID	Integer
numberofrooms	Integer
currency	Character String
averageroomrate	Floating Point number
contactInput	Character String
emailInput	Character String (Email address)
phoneInput	Character String (Telephone Number)
companyInput	Character String
transationCountOutput	Floating Point Number
ATVOutput	Floating Point Number
turnoverOutput	Floating Point Number
valueMinOutput	Floating Point Number
upsellOutput	Floating Point Number

Additionally, data being present in the table indicates that the deployment has already been successfully carried out and there is at least one instance of the Calculator running and the database deployment process does not need to be carried out. If, however, this is not the case, then it is indicative that there has been an error in setting up the database, or a table has been misplaced. If this is the case, it is possible that someone else has been working in the code to modify it. If you are collaborating with someone, please get in touch with them.

- Connecting to a non-Azure MSSQL database



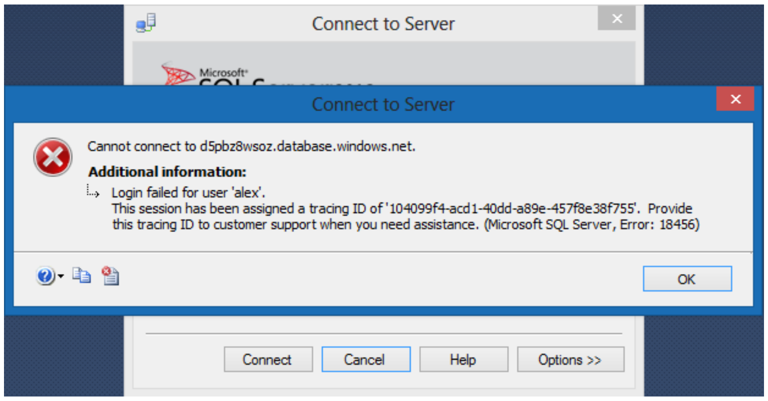
Server type: Database Engine
Server name: d5pbz8wsoz.database.windows.net
Authentication: SQL Server Authentication
Windows Authentication
SQL Server Authentication
Login:
Password:
Remember password

To connect to a non-Azure MSSQL database, we will be using Microsoft's SQL Server Management. The process itself is reproducible, and therefore is independent of the software being used to connect to the database.

On loading of the Microsoft SQL Server Manager, you will be prompted to log in to a database.

For this, you will need to know the name of the server you wish to access, as well as login credentials. To find out the server name, the best approach is to contact the I.T. Department for advice.

Depending on the authentication you are using, depends on which Authentication menu option to select. If you are logging in to the server using your current windows username and password, then use Windows Authentication. If, however, you have been assigned a specific login for the database, then you want to use SQL Server Authentication.



Cannot connect to d5pbz8wsoz.database.windows.net.

Additional information:

Login failed for user 'alex'.
This session has been assigned a tracing ID of '104099f4-acd1-40dd-a89e-457f8e38f755'. Provide this tracing ID to customer support when you need assistance. (Microsoft SQL Server, Error: 18456)

Now it's a case of filling in the username in "Login", and the password in the "Password" field, and clicking connect. If there is an error while connecting, it will be flagged and the relevant errors are displayed in a message box. Once the errors have been solved, you are now logged in to the MSSQL server and can run scripts.

- Adding front-end form handling code to your ASPX file

Copy the following code into the partial class of the c# file associated with the ASPX file of the page you wish to add the calculator to. The partial class should be in a separate file with the same name as the page and a .cs appended. Example; If you have "mouse.aspx", then the file with the content in will be called "mouse.aspx.cs".

Code

"

```
private static String username = "<User>";
private static String password = "<Password>";
private static String serverName = "<ServerName>";
private static String dbName = "WLPCalculator";
private static String tableName = "WLPCalcTable";
```

```
    public class ToDatabase
    {
        public string companyInput { get; set; }
        public int numberofrooms { get; set; }
        public float averageroomrate { get; set; }
        public string currency { get; set; }
        public string contactInput { get; set; }
        public string emailInput { get; set; }
        public string phoneInput { get; set; }

        public float numTransactionOutput { get; set; }
        public float ATVOutput { get; set; }
        public float TurnoverOutput { get; set;}
        public float valueOutput { get; set; }
        public int upsellOutput { get; set; }
    }



    [System.Web.Services.WebMethod]
    public static void DoWork(String input)
    {
        string connectionString = "Persist Security Info=False;";
        connectionString += "User ID=" + username+";" ;
        connectionString += "Password="+password+";";
        connectionString+= "Initial Catalog="+dbName+";";
        connectionString += "Server=" + serverName + ";";
        JavaScriptSerializer parser = new JavaScriptSerializer();
        var info = parser.Deserialize<ToDatabase>(input);

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("INSERT INTO " + tableName + " (companyInput,
numberofrooms,
averageroomrate,currency,contactInput,emailInput,phoneInput,transactionCountOutput,ATVOutput,turnoverO
utput,valueMinOutput,upsellOutput) VALUES (@company, @roomCount,
@roomRate,@currency,@contactInput,@emailInput,@phoneInput,@transCountOut,@ATVOut,@TurnoverO
ut,@valueMinOut,@upsellOut)");
            cmd.CommandType = CommandType.Text;
            cmd.Connection = connection;
            if (info.companyInput != null)
            {               cmd.Parameters.AddWithValue("@company", info.companyInput);}
            else {              cmd.Parameters.AddWithValue("@company", "");}
            if (info.numberofrooms != null)
            {               cmd.Parameters.AddWithValue("@roomCount", info.numberofrooms);}
            else {              cmd.Parameters.AddWithValue("@roomCount", -1);}
            if (info.averageroomrate != null)
            {               cmd.Parameters.AddWithValue("@roomRate", info.averageroomrate);}
            else{               cmd.Parameters.AddWithValue("@roomRate", -1);}
            if (info.currency != null)
            {               cmd.Parameters.AddWithValue("@currency", info.currency);}
            else{               cmd.Parameters.AddWithValue("@currency", "");}
            if (info.contactInput != null)
            {               cmd.Parameters.AddWithValue("@contactInput", info.contactInput);}
            else{               cmd.Parameters.AddWithValue("@contactInput", "");}
            if (info.emailInput != null)
            {               cmd.Parameters.AddWithValue("@emailInput", info.emailInput);}
            else{               cmd.Parameters.AddWithValue("@emailInput", "");}
            if (info.phoneInput != null)
            {               cmd.Parameters.AddWithValue("@phoneInput", info.phoneInput);}
            else{              cmd.Parameters.AddWithValue("@phoneInput", "");}
            if (info.upsellOutput != null)
            {               cmd.Parameters.AddWithValue("@upsellOut", info.upsellOutput);}
            else{               cmd.Parameters.AddWithValue("@upsellOut", "");}
            if (info.valueOutput != null)
            {               cmd.Parameters.AddWithValue("@valueMinOut", info.valueOutput);}
            else{               cmd.Parameters.AddWithValue("@valueMinOut", "");}
            if (info.TurnoverOutput != null)
            {               cmd.Parameters.AddWithValue("@TurnoverOut", info.TurnoverOutput);}
            else{               cmd.Parameters.AddWithValue("@TurnoverOut", "");}
            if (info.ATVOutput != null)
            {               cmd.Parameters.AddWithValue("@ATVOut", info.ATVOutput);}
            else{               cmd.Parameters.AddWithValue("@ATVOut", "");}
            if (info.numTransactionOutput != null)
            {               cmd.Parameters.AddWithValue("@transCountOut",info.numTransactionOutput);}
            else{               cmd.Parameters.AddWithValue("@transCountOut", "");}

            connection.Open();
            cmd.ExecuteNonQuery();
        }
    }
“
```

After having inserted it in to the relevant place in the file, you should have noticed that there are some variables that are not defined properly. These are the database credentials that we will be writing to. At this point, then, replace “<User>" with the username you will be using to access the database, "<Password>" with the password and "<ServerName>" with the web address of the server you are connecting to, surrounded with quotation marks. Alternatively. the <ServerName> field can be augmented with a comma and a digit number to specify a port to connect to.

Code;
“
```
            private static String username = “Robert";
            private static String password = ”little_bobby_tables";
            private static String serverName = "server.myowncompany.com";
            //alternate form : “server.myowncompany.com,1443” , where 1443 represents the port we are
            connecting to.
```
“

In addition, make sure the aspx.cs file also imports System.Web.Script.Serialization, System.Collections.Generic, System.Data.SqlClient, System.Data with the using keyword at the top of the .aspx.cs document;

Code;
“
```
            using System.Web.Script.Serialization;
            using System.Collections.Generic;
            using System.Data.SqlClient;
            using System.Data;
```
“

- Adding HTML to your front-end page.

So that the end user can see the fruits of our efforts, we now need to add a block of code to the HTML of the page to contain the calculator, such that it can be seen and interacted with by the end user.

The following code is to be inserted between whichever elements the calculator should appear. The size of the element can be dictated by changing the <width> and <height> variables in the code and will set the width and height to the pixel widths and heights of the numbers specified.
At the time of writing, values of 340 for width and 740 for height produce an element of appropriate size.
If the resulting element is too small, the overflow will be hidden from the end users sight.

Additionally, the page_name variable in the code needs to be changed to the name of the .aspx file that we are inserting in to; if the name of your page is “hotel.aspx”, then <page_name> needs to be changed to hotel, such that the line reads ‘var postOffTo = “hotel.aspx/DoWork”;’

Code;
“
```
            <div class="calculator" width:<width>px; height:<height>px; overflow:hidden;”>
                        <script type="text/javascript">
                                    (function() {
                                                var postOffTo = "<page_name>.aspx/DoWork";
                                                var link = document.createElement('link');
                                                link.href = 'styles/WLPCalccalculator.css';
                                                link.rel = 'stylesheet';
                                                document.getElementsByTagName('head')[0].appendChild(link);
                                                var scr = document.createElement("script");
                                                scr.type = "text/javascript";
                                                scr.innerHTML = "var toPostTo = "+ '"'+postOffTo+'";';
```

```
                    document.getElementsByTagName('head')[0].appendChild(scr);
                    var link = document.createElement('script');
                    link.src = 'scripts/' + 'WLPCalcnoConsole' + '.js';
                    link.type = 'text/javascript';
                    document.getElementsByTagName('head')[0].appendChild(link);
                })();
            </script>
        </div>
“
```

- Final Notes

With adding the frontend elements, the deployment of the calculator is finished. To check the databases for information, run the same queries as were used to check for the correct column names after the calculator has been test submitted. If data appears in the tables, then the setup has gone to plan. If, however, it has not; please re-read the guide to make sure that you haven’t missed anything.

Happy Deploying!