# Practice M3: Databases and Analytics

For this module we will need a PC running recent version of **Windows**, **Linux**, or **macOS**. It can be either physical or virtual machine

## Part 1: Azure SQL

Navigate to <https://portal.azure.com>

Enter your credentials

##### Resource group

Navigate to **resource groups** section

Create new resource group, for example **RG-SQL** in the **West Europe** region

Enter the resource group

##### SQL Server

Being in the resource group, click on the **+ Add** button to add a new resource

In the main search field enter **Azure SQL** and hit the **Enter** key

Click on the **Create Azure SQL resource**

In the drop-down menu **Resource type** of the **SQL database** section select **Database server**

Click on the **Create** button

Make sure both fields **Subscription** and **Resource group** are filled appropriately

Enter a globally unique name in the **Server name** field as it will become part of an URL

Adjust the **Location** to your liking, for example **West Europe**

In the **Server admin login** enter something different from **sa**, **root**, **admin**, etc. for example **demosa**

For a password choose something between **8** and **128** characters, for example **DemoPassword-2019**

Click on **Review + create**

Finally, click on the **Create** button

After a while we will have our new SQL server ready

Once the deployment is done, click on the **Go to resource** button

##### Create database

Click on the **+ New database** button

In the **Database name** field enter for example **db1**

Select **Sample (AdventureWorksLT)** from the **Select source** drop-down list

Click on the **Pricing tier** to change the size and pricing

Being in the **General Purpose** you can select **Serverless** or click on the **Looking for basic, standard, premium?** button

Click on different levels to examine the parameters

Select the **Basic** plan and click on **Apply**

Click on **OK** to create the database

After a while the new database will appear in the bottom table of the **Overview** screen

##### Database connectivity

Click on the **Firewall and virtual networks** option in the **Security** section

You can turn on the **Allow Azure services and resource to access this serve** option

Click on the **+ Add client IP** button to allow tools on your station to access the databases on the server

Don’t forget to click on the **Save** button

##### Connect to a database with Query editor

Click on the **SQL databases** option under the **Settings** section

Select the **DB1** database

Click on the **Query editor (preview)** option

Ensure that you enter the correct values in the **Login** and **Password** fields

Click **OK**

Expand the **Tables** node

Click on the **SalesLT.Customer** table

From the context menu choose **Select Top 1000 Rows**

You can create a new query by clicking on the **+ New Query** button

Or load an existing query by clicking on the **Open query** button

##### Connect to a database with SQL Server Management Studio (SSMS)

Click on the **Connection strings** option under the **Settings** section

As we can see, there are several prepared connection strings

Click on the **ADO.NET**

And copy the server name and the port. It should be something like **azesql.database.windows.net,1433**

Open the SSMS tool

Paste the copied string in the **Server name** section

In the **Login** field enter a valid user and the corresponding password

Click on **Connect**

Expand the **Databases** node, then the **DB1** node, and then the **Tables** node

Select the **SalesLT.Product** table and from its context menu choose **Select Top 1000 Rows**

In a similar fashion you can use other tools like **Azure Data Studio**, **SSDT** or **Visual Studio Code**

##### Restore a database

Return to the **Overview** of the SQL servers

Click on the **Import database** button

Click on **Storage** section

Select an existing storage account

If you do not see one, go and create, then return back here

Click on **+ Container**

For **Name** enter **db1** and click **OK**

Click on the new container

Click on the **Upload** button

Select the **database bacpac file** (**BGCities.bacpac**) provided with this practice to upload and click **Upload**

Select the uploaded export and click on the **Select** button

Click on the **Pricing tier** to select a different one

Let’s select **Basic** and confirm with **Apply**

Adjust the collation if needed

Enter the credentials and click **OK**

Once the process finishes, you can go to the database and query its data

##### Azure CLI: Create database

If using local shell, login first by issuing:

**az login**

First, let’s set a few default values in order to shorten our commands. We will set the region and the SQL server:

**az configure --defaults group=RG-SQL sql-server=azesql**

Now, list the existing databases:

**az sql db list**

Or just their names:

**az sql db list --query [].name**

Let’s create one database with:

**az sql db create --name DBCLI --edition Basic --capacity 5**

##### PowerShell: Create database

If not working in **Azure Cloud Shell**, then first login:

**Connect-AzAccount**

If we don’t know all the commands related to both the SQL server and the database, we can ask for them

For the SQL server:

**Get-Command \*azsqlserver\***

For the SQL databases:

**Get-Command \*azsqldatabase\***

Now, let’s list all existing databases:

**Get-AzSqlDatabase -ServerName azesql -ResourceGroupName RG-SQL**

Or just their names:

**Get-AzSqlDatabase -ServerName azesql -ResourceGroupName RG-SQL | Select -Property DatabaseName**

Create new database with:

**New-AzSqlDatabase -DatabaseName DBPS -ResourceGroupName RG-SQL -ServerName azesql -Edition Basic**

##### Connect to a database from the command line

With the help of **Azure CLI** we can get a connection string for a particular database:

**az sql db show-connection-string --client sqlcmd --name BGCities**

Now, we can connect with the help of the **sqlcmd** tool using the provided connection string:

**sqlcmd -S tcp:azesql.database.windows.net,1433 -d BGCities -U demosa -P "DemoPassword-2019" -N -l 30**

First, ask for the version of the database:

**SELECT @@version;**

**GO**

Now, change the database to **BGCities**:

**USE BGCities;**

**GO**

Ask for the list of top cities:

**SELECT \* FROM TopCities;**

**GO**

Close the session with:

**QUIT**

## Part 2: Cosmos DB

Navigate to <https://portal.azure.com>

Enter your credentials

##### Resource group

Search for resource groups

Click on **+ Add** to create a new resource group

For **Resource group** set **RG-CosmosDB**

Select the region to be **West Europe**

Click on **Review + create**

Then on **Create**

##### Azure Cosmos DB account

Go to the resource group

Click on the **+ Add** button to create new resource

Search for **Azure Cosmos DB** in the main search bar and hit **Enter**

Click on the **+ Add** or the **Create Azure Cosmos DB account**

Select the resource group we created earlier

In the **Account Name** field enter a globally unique name, for example **azecos**

For the **API** make sure the **Core (SQL)** is selected

Change the **Location** to a region close to you, for example **West Europe**

Click on **Review + create**

Click on **Create**

Once the deployment is done, click on the **Go to resource** button

Select **Create new** and for **Database id** enter **TimeTracker**

Leave the provisioning options with their default values

For **Container id** enter **TimeSlots**

In the **Partition key** enter **/category**

Click **OK**

##### Work with data

Expand the **TimeSlots** container and click on **Items** option

Then click on the **New Item** button

Add the following:

**{**

**"id": "1",**

**"category": "personal",**

**"description": "Bathroom activities",**

**"startedOn": "2019-12-09 07:30:00",**

**"slotLength": "900"**

**}**

Click on **Save**

Repeat the procedure (you can use the **Cosmos-DB-Sample-Data.txt** file) few more times

Now, that we have a few records, we can experiment with data retrieval a bit

Note that there is a query, let’s edit it. Click on the **Edit Filter**

Enter **WHERE c.category = "business"** and click on the **Apply Filter** button

Let’s try another query. Click on the **New SQL Query** button (3-rd on the toolbar)

New window will open with a prepopulated query. Click on the **Execute Query** button

On the tab **Results** you can see the output and on the tab **Query Stats** you can explore what it took to execute it

Let’s amend the query to:

SELECT c.category, COUNT(1) AS slotsCount FROM TimeSlots c GROUP BY c.category

Click on the **Execute Query** button to see the results

##### Azure CLI: Create database (!!! Cosmos sql is still in preview !!!)

If using local shell, login first by issuing:

**az login**

First, let’s list all databases:

**az cosmosdb sql database list --account-name azecos --resource-group RG-CosmosDB**

If all we want is just the name of each database, we can adjust the command to:

**az cosmosdb sql database list --account-name azecos --resource-group RG-CosmosDB --query [].name**

Now, let’s create one new database:

**az cosmosdb sql database create --account-name azecos --resource-group RG-CosmosDB --name AZDB**

And a container for storing people partitioned by city:

**az cosmosdb sql container create --account-name azecos --resource-group RG-CosmosDB --database-name AZDB --name People --partition /city --throughput 400**

Please note that the above should be changed a little bit if you are using **Azure CLI** under **Bash** or **Azure Cloud Shell**. The partition part will become:

**--partition '/city'**

##### PowerShell: Inspect Cosmos DB (!!! Lack of specialized commands !!!)

If not working in **Azure Cloud Shell**, then first login:

**Connect-AzAccount**

Here the situation is a little bit more complex. Because of the lack of specialized commands, we must pass everything through the **Resource** object. For example, to list all databases, we must execute:

**Get-AzResource -ResourceType Microsoft.DocumentDB/databaseAccounts/apis/databases `**

**-ApiVersion "2015-04-08" -ResourceGroupName "RG-CosmosDB" -Name "azecos/sql/"**

To show information for a particular database, we must execute:

**Get-AzResource -ResourceType Microsoft.DocumentDB/databaseAccounts/apis/databases `**

**-ApiVersion "2015-04-08" -ResourceGroupName "RG-CosmosDB" -Name "azecos/sql/TimeTracker"**

Should we want all information, we must execute:

**Get-AzResource -ResourceType Microsoft.DocumentDB/databaseAccounts/apis/databases `**

**-ApiVersion "2015-04-08" -ResourceGroupName "RG-CosmosDB" -Name "azecos/sql/TimeTracker" `**

**| Select \***

If we want all containers in a database, we must execute:

**Get-AzResource -ResourceType Microsoft.DocumentDB/databaseAccounts/apis/databases/containers `**

**-ApiVersion "2015-04-08" -ResourceGroupName "RG-CosmosDB" -Name "azecos/sql/TimeTracker"**

Information for a container in a database we can retrieve via:

**Get-AzResource -ResourceType Microsoft.DocumentDB/databaseAccounts/apis/databases/containers `**

**-ApiVersion "2015-04-08" -ResourceGroupName "RG-CosmosDB" `**

**-Name "azecos/sql/TimeTracker/TimeSlots" | Select \***

##### Import data from (from Azure SQL)

We will need external tool for this exercise. This is the **Azure Cosmos DB Data Migration tool**. It can be downloaded from here: <https://aka.ms/csdmtool>

Extract it to a folder of your choice

The tool supports command line mode and graphics mode. We will choose to use the second option

Navigate to the folder and double click over **dtui.exe**

On the **Welcome** screen click the **Next** button

Change the **Import from** selected value to **SQL**

Return to the Azure portal, go to the **Azure SQL server** and then to **SQL databases**

Click on the one that we will export. Let’s select the **BGCities** one

Go to the **Connection strings** option under the **Settings** section

Copy the connection string for the **ADO.NET** and paste it in the **Connection String** field of the **Data Migration Tool**

Don’t forget to substitute the **{your\_password}** with the appropriate password, for example **DemoPassword-2019**

Click on the **Verify** button to test the connectivity

Select the **Enter Query** option and paste the following SQL statement:

**SELECT CityName, Population FROM TopCities**

Click on **Next**

Make sure that **Azure Cosmos DB – Sequential record import (partitioned collection)** is selected under **Export to**

Return to the **Azure Portal**, go to the **Azure Cosmos DB**, select the **azecos** database account

Click on the **Keys** option under the **Settings** section

Copy the **PRIMARY CONNECTION STRING** value

Return to the **Data Migration Tool** and paste the value to the **Connection** field

At the end add **Database=AZDB** or **PSDB** or another one

Click on the **Verify** button to check the connectivity

Type **TopCities** in the **Collection** field

Enter **/CityName** as a **Partition Key**

Click on **Next**

Click again **Next**

Click **Import**

Close the application and return to the **Azure Portal**

go to the **Azure Cosmos DB**, select the **azecos** database account

Navigate to **Data Explorer** option

You should click on thedatabase used during the import process and under it the **TopCities** container

Expand **TopCities** and click on **Items,** you should see the list of the top 10 cities by population in Bulgaria

## Part 3: Azure Stream Analytics

Navigate to <https://portal.azure.com>

Enter your credentials

##### Resource group and storage account

Navigate to **resource groups** section

Create new resource group, for example **RG-Stream-Analytics** in the **West Europe** region

Enter the resource group

Create new storage account in the same resource group

During creation process you can leave most of or all the parameters with their default values. For example, you may want to change the **Replication** to **Locally redundant storage** to achieve lower price

##### Create a stream analytics job

Next, while still in the resource group, enter in the search bar **stream analytics** and press **Enter**

Then, click on the **Create** button

For **Job name** use **SimpleStreamAnalyticsJob**

Change the **Streaming units** to **1**

Other setting should be fine

Click on the **Create** button

##### Storage preparation

Go to the storage account, you created earlier

Click on **Containers** option in the **Blob services** section

Click on the **+ Container** button to create a new one

In the **Name** field enter **demo** and click **OK**

##### Configure input and output for a stream analytics job

Navigate back to the resource group

In the **Overview** section click on the **SimpleStreamAnalyticsJob** item

Click on the **Inputs** option in the **Job topology** section

Select **+ Add stream input** to create new input stream

In the drop-down list choose **Blob storage**

For **Input alias** enter **StreamIn**

Ensure that the **Subscription** and the **Storage account** are correctly set

Under the **Container** select **Use existing**

Select the **blob container** that you created earlier. For example, **demo**

In the **Path pattern** field enter **input/**

Click on the **Save** button

Now, it is time to configure the output stream. Click on the **Outputs** command in the **Job topology** section

Click on the **+ Add** button

In the drop-down menu select **Blob storage/Data Lake Storage Gen2**

For **Output alias** enter **StreamOut**

Pay attention to the **Subscription** and **Storage account** settings

In the **Container** section select **Use existing**

Select the **demo** container from the drop-down list

Enter **output/** in the **Path pattern** field

Click on **Save**

**NOTE:** The following actions are possible if you have some type of Power BI account. It could be trial, personal, or professional. Feel free to skip all instructions related to Power BI

Let’s add one additional output stream. This time it will be for **Power BI**

Click on the **+ Add** button

Select **Power BI** from the drop-down list

Click on the **Authorize** button

Enter your credentials for Power BI in the pop-up window

Change the **Authentication mode** to **User token**

Then, enter **PowerBI** in the **Output alias** field

In the **Dataset name** enter **TemperatureData**

For **Table name** use **Measurements**

Confirm with the **Save** button

##### Create a query

Using the **Azure Portal** or the **Microsoft Azure Storage Explorer** application upload the **sensor-sample.json** file in a folder **input/** in the **demo** container

Return to the stream analytics job

Click on the **Query** option in the **Job topology** section

Change the query to:

SELECT CONCAT(Measurement.[Index], '-', Sensor) AS ID,

       Sensor,

       Measurement.[Index] AS MeasureId,

       Measurement.Taken AS MeasuredAt,

       Measurement.Value AS Temperature

INTO StreamOut

FROM StreamIn

Here, we can note a few things:

* It is indeed an SQL-like syntaxis. In fact, is a subset of SQL
* Reserved words, when used for names, must be surrounded with square brackets
* We can use aliases for the columns
* We can create custom columns as well

Now, click on the **Save query** button

Under the query area you should see the **Input preview** pane showing the sample data we uploaded earlier

Hit the **Test query** button

The pane under the query should switch to **Test results** view showing what would we receive as output

Let’s add another query which will serve our Power BI output

Leave one empty row bellow the first query and paste:

SELECT System.Timestamp AS WindowEnd,

       Sensor,

       COUNT(\*) AS Count,

       AVG(Measurement.Value) AS Temperature

INTO PowerBI

FROM StreamIn

GROUP BY Sensor, TumblingWindow(mi, 1)

Click on **Save query** button again

Let’s click on the **Test query** again

This time, we can click on each of the two **Outputs** listed in the left section to force the **Test results** pane to show the corresponding test output

##### Adjust the data generation process

Extract the provided archive – **AzureBlobJsonGenerator-bin** in a local folder of your choice

Open the **AzureBlobJsonGenerator.exe.config** file in a text editor

Check the **BlobContainer** setting. Set it to match the container you created earlier. By default, it is **demo**

Go to the storage account

Then, navigate to the **Access Keys** option under the **Settings** section

Copy the **Connection string** value either for the **key1** or for the **key2**

Paste the value in the **StorageConnectionString** setting

Check and adjust also the **LocalTempPath** parameter, it must point to a folder on your PC, that will be used to store the files generated while the application is working

Save and close the file

##### Start the job

Return to the **Overview** mode of the stream analytics job

Click on the **Start** button

Leave the **Job output start time** to **Now** and click on **Start**

Wait for the job to start

Once it is started, go to the folder where you did extract the data generation application

Doble click on the **AzureBlobJsonGenerator.exe**

Check the **json/** folder to see if there are files being generated

##### BLOB output

After a minute or so, there should be some data on the output stream

Go to the storage account

Enter the container

Go to the **output/** folder

Click on the file to download it

Open it with a text editor

The result should match what we defined for the output stream

##### Power BI output

Navigate to <https://app.powerbi.com>

Login to your account

Click on **My workspace**

Click on **+ Create** button (in the top-right corner)

Select **Dashboard**

For **Dashboard name** enter for example **Temperature**

Click on **Create**

Click on **+ Add tile**

Under **Real-time data** select **Custom Streaming Data**

Then click **Next**

You should see the result of our Power BI output stream - **TemperatureData**

Select it and click **Next**

Click **+ Add value**

In the drop-down select **Count**

Then click on **Next**

In the **Title** field enter **Measurements** and in the **Subtitle** field enter **Per minute**

Click on **Apply**

Let’s add one more tile

Click on **+ Add tile**

Under **Real-time data** select **Custom Streaming Data**

Then click **Next**

You should see the result of our Power BI output stream - **TemperatureData**

Select it and click **Next**

Change the **Visualization Type** to **Line chart**

Click **+ Add value** in the **Axis** section and select the **WindowsEnd** option

Click **+ Add value** in the **Legend** section and select **Sensor** option

Click **+ Add value** in the **Values** section and select **Temperature** option

In the **Time window to display** select the **Last 5 Minutes**

Then click on **Next**

In the **Title** field enter **Temperature** and in the **Subtitle** field enter **For the last 5 minutes**

Click on **Apply**

##### Job monitoring

Return to the Azure portal and go to the **Overview** view of the stream analytics job

There, under **Monitoring** we can see the performance

Select **Job diagram** in the left menu

This is an alternative view of what is going on

##### Cleaning

Return to the **Overview**

Click on **Stop** button and confirm that you want to stop the job

Stop the data generation application and delete the files under the **json** folder

Delete either the whole storage account, or just the blob container

Delete the resource group