Activities

Instructions for the activities that must be performed during this challenge are presented below. You will need to complete all the activities in this topic and submit the noted deliverables (each earmarked by " DELIVERABLE" below) for submitting your work for grading.

Architecture

The following activities explore the architecture of the solution.

1. Review requirements.

2. **DELIVERABLE**: Update the **WeatherStationArchitecture** Architecture Diagram and add subsystems that satisfies the requirements (use PowerPoint or Visio). Save this version as **WindFarmArchitecture-[YOUR STUDENT ID]** for assessment. Save this file in the **Lab2** folder within your GitHub repository (ensure you add, commit and push your changes)

   - List all systems and subsystems chosen for the solution.
   - Explain why the subsystem was chosen.

3. **Update** the existing **WeatherStationThreatModel** threat model and note any key factors that drive configuration of the solution.

   - **DELIVERABLE**: Save the updated threat model as a new file **WindFarmThreatModel-[YOUR STUDENT ID]** for assessment. Save this file in the **Lab2** folder within your GitHub repository (ensure you add, commit and push your changes)

Provisioning the Azure Resources

Provision the additional Azure Resources specified within the updated Architecture Diagram. Continue to utilize the following naming conventions:

- All resources to be located within a single resource group: **IoTCapstoneRG**
- All resources will follow a naming convention of:

  `iot[resource mnemonic][studentid]`

  For example:

  `iothub21f3a359`

  Note: As some resources, such as *Data lake Storage* and *Azure Storage Account*, have restrictions on what characters may be used in the name - don't use hyphens and always use lowercase characters. For example:

  `iotdls21f3a359` or `iotstore21f3a359`

Here is a list of resources and the required mnemonics which should be used **unless** an explicit name for a resource is provided:

Resource	**Mnemonic**
Azure Function	func
Azure Logic App	logic
Azure SQL Server	ss
Azure Stream Analytics	asa
Blob Storage	blob
Cosmos DB	cosmos
Data Lake Analytics	dla
Data Lake Storage	dls
Device Provisioning Service	dps
IoT Hub	iothub
Machine Learning	ml
Power BI	pbi

Resource	Mnemonic
Time Series Insights	tsi
Web App	webapp

1. Continue to use the resource group named: **IoTCapstoneRG**.

2. Create all the additional resources identified in the architecture diagram you updated earlier.

   - The resources must be created within the **IoTCapstoneRG** resource group.
   - Use the naming convention as specified above.
   - Configure the resources to meet the requirements.

     Important: You will not receive credit for your work if you do not follow the naming conventions.

Wind Farm Simulation

1. Install **Wind Farm Dashboard** App as directed in the **Lab Setup** task in the **Introduction**.

2. Connect **Wind Farm Dashboard** App to IoT Hub.

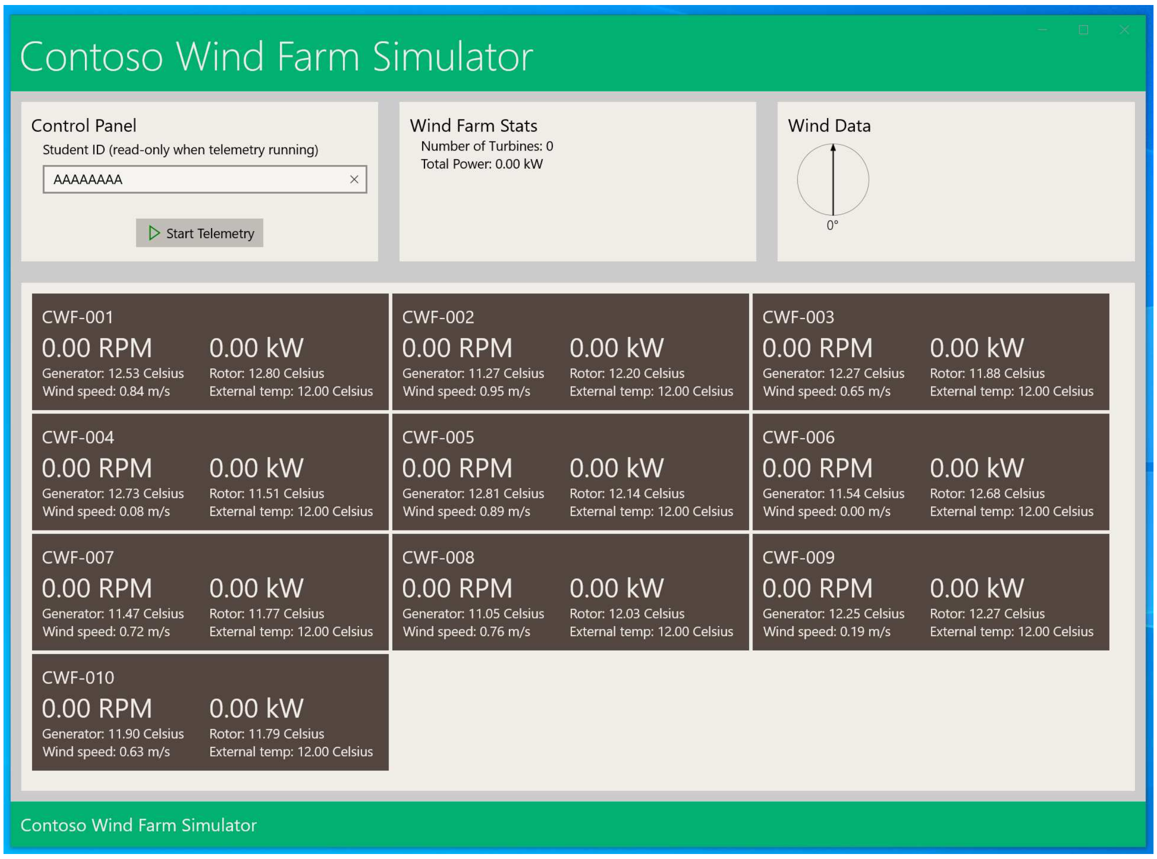
   - You will need to create 10 devices:

     - **CWF-001**.
     - **CWF-002**.
     - **CWF-003**.
     - **CWF-004**.
     - **CWF-005**.
     - **CWF-006**.

    - **CWF-007**.
    - **CWF-008**.
    - **CWF-009**.
    - **CWF-010**.

> **Tip:** Review the following materials:
>
> - **Course**: *DEV326x IoT Data Analytics and Storage*
> - **Module:** *Advanced Analytics*
> - **Lab:** *Device Management and Analytics*
> - **Topic:** *Managing IoT Devices, tags and desired configurations with IoT Hub*
>
> and
>
> - **Course**: *DEV312x Business Intelligence for IoT Solutions*
> - **Module:** *Time Series Data*
> - **Lab:** *Producing Simulated Data*
> - **Topic:** *Set up the device simulation*

- Start the **Wind Farm Dashboard** App.


Contoso Wind Farm Simulator

Control Panel
Student ID (read-only when telemetry running)
AAAAAAAA

Wind Farm Stats
Number of Turbines: 0
Total Power: 0.00 kW

Wind Data
0°

CWF-001
0.00 RPM 0.00 kW
Generator: 12.53 Celsius Rotor: 12.80 Celsius
Wind speed: 0.84 m/s External temp: 12.00 Celsius

CWF-002
0.00 RPM 0.00 kW
Generator: 11.27 Celsius Rotor: 12.20 Celsius
Wind speed: 0.95 m/s External temp: 12.00 Celsius

CWF-003
0.00 RPM 0.00 kW
Generator: 12.27 Celsius Rotor: 11.88 Celsius
Wind speed: 0.65 m/s External temp: 12.00 Celsius

CWF-004
0.00 RPM 0.00 kW
Generator: 12.73 Celsius Rotor: 11.51 Celsius
Wind speed: 0.08 m/s External temp: 12.00 Celsius

CWF-005
0.00 RPM 0.00 kW
Generator: 12.81 Celsius Rotor: 12.14 Celsius
Wind speed: 0.89 m/s External temp: 12.00 Celsius

CWF-006
0.00 RPM 0.00 kW
Generator: 11.54 Celsius Rotor: 12.68 Celsius
Wind speed: 0.00 m/s External temp: 12.00 Celsius

CWF-007
0.00 RPM 0.00 kW
Generator: 11.47 Celsius Rotor: 11.77 Celsius
Wind speed: 0.72 m/s External temp: 12.00 Celsius

CWF-008
0.00 RPM 0.00 kW
Generator: 11.05 Celsius Rotor: 12.03 Celsius
Wind speed: 0.76 m/s External temp: 12.00 Celsius

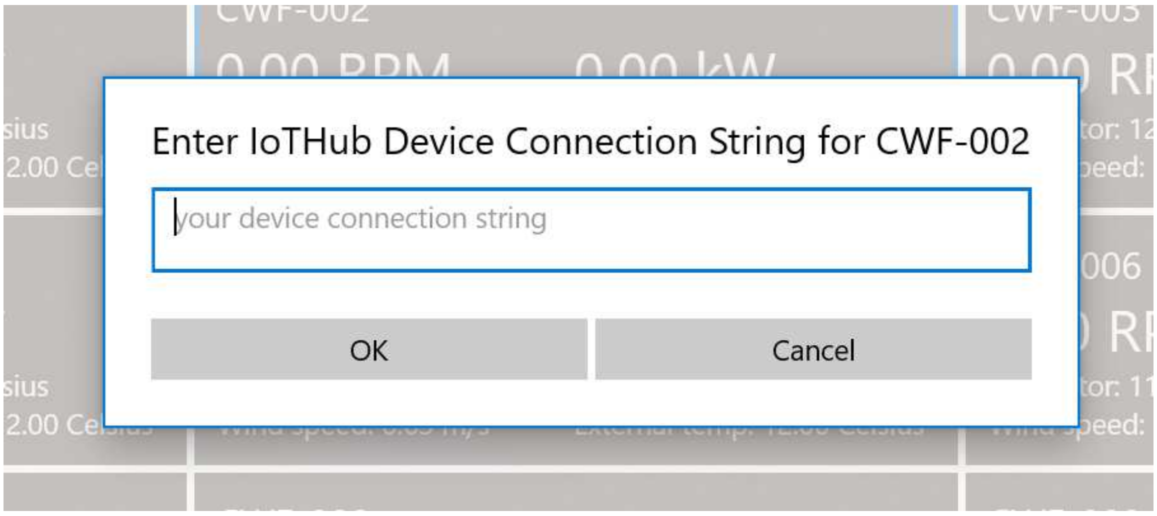
CWF-009
0.00 RPM 0.00 kW
Generator: 12.25 Celsius Rotor: 12.27 Celsius
Wind speed: 0.19 m/s External temp: 12.00 Celsius

CWF-010
0.00 RPM 0.00 kW
Generator: 11.90 Celsius Rotor: 11.79 Celsius
Wind speed: 0.63 m/s External temp: 12.00 Celsius

Contoso Wind Farm Simulator

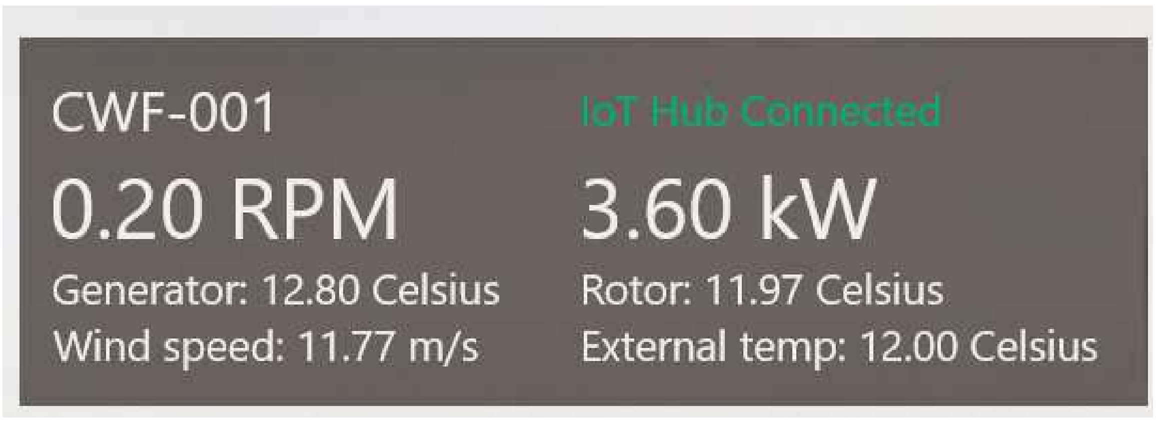
- Enter your student ID in to the appropriate field - this will be saved between sessions so you wil only need to do this once.

- Click on each of the Turbine tiles - a dialog will be displayed. Copy the **Device Connection String** for each device into the dialog and close it - this connection string will be saved between sessions so you will only need to do this once.


Enter IoTHub Device Connection String for CWF-002

your device connection string

- Start the Telemetry so that data begins to be sent to your IoT Hub - note that each turbine should display an **IoT Hub Connected** message.


CWF-001

IoT Hub Connected

0.20 RPM

3.60 kW

Generator: 12.80 Celsius

Rotor: 11.97 Celsius

Wind speed: 11.77 m/s

External temp: 12.00 Celsius

Now you have telemetry data flowing to you IoT Hub, the next step is to start gaining some insights into the data.

Time Series Insights

First you will configure Time Series Insights and examine the telemetry in near real-time.

1. Create a Time Series Insights resource and configure it to connect to the IoT Hub.

   - **Note:** Pay attention to the SKU you select - TSI can get expensive quickly.

     > **Tip:** Review the following materials:
     >
     > - **Course**: *DEV312x Business Intelligence for IoT Solutions*
     > - **Module:** *Time Series Insights*
     > - **Lab:** *Provisioning Time Series Insights*

2. Use the Time Series Explorer to create the **Interesting Telemetry** query detailed below. Set the query to display the last 30 minutes. Use an interval of 2 seconds (or as small as possible if 2 seconds is unavailable).

   - **Interesting Telemetry**. Create a query that displays the following measures:
     - **AVG** Telemetry.Power **SPLIT BY** Telemetry.Name with a display name of **Power**
     - **AVG** Telemetry.WindSpeed **SPLIT BY** Telemetry.Name with a display name of **WindSpeed**
     - **AVG** Telemetry.LowSpeedShaftRpm **SPLIT BY** Telemetry.Name with a

display name of **RPM**

- **AVG** Telemetry.GeneratorTemperatureCelsius **SPLIT BY** Telemetry.Name with a display name of **Temp**

3. Save the **Interesting Telemetry** query and wait until there is 15 - 20 minutes of data displayed.

4. Examine the displayed data - do you notice a particular turbine and telemetry that appears to be operating outside of the others?

   - **DELIVERABLE**: Select the turbine and data element that seems to be a problem and take a screen shot of the view and save it as **InterestingTelemetry-[YOUR STUDENT ID]-anomaly.png** for submission. Save this file in the **Lab2** folder within your GitHub repository (ensure you add, commit and push your changes). Ensure that the screenshot includes the following:
     - All of the terms on the left hand side of the view.
     - The charts should be displayed stacked.
     - The chart title should be visible.
     - The charts should be displaying at least 10 minutes of data.
     - The suspect telemetry term and turbine should be selected.

5. Create another query and name is **SuspectTelemetry**. Set the query to display the last 30 minutes. Use an interval of 2 seconds (or as small as possible if 2 seconds is unavailable). Starting with no initial terms, add two terms to the query and change the display to overlay the terms rather than stack them:

   - The **AVG** measure you suspect is out of the ordinary, named the same as the measure in the **Interesting Telemetry** query. Add a **WHERE** clause that only displays the suspect turbine:
     - The **WHERE** clause will be similar to `[telemetry.name] has 'CWF-XXX'`.
   - The same **AVG** measure you added above **without** a **WHERE** value or **SPLIT BY** - this will display the average for that measure across all turbines. Name this measure **AVG**.
     - **Hints:**
       - You should see a marked difference - if not, you may have the incorrect measure.

    - You may need to adjust the increment slider and time frame for data collection to see the differences between turbines. Remember that the more data that is sent to the IoT hub, the shorter the time frame you may need to examine given that you're looking at averages.

6. **DELIVERABLE**: Take a screen shot of the **SuspectTelemetry** view and save it as **SuspectTelemetry-[YOUR STUDENT ID].png** for submission. Save this file in the **Lab2** folder within your GitHub repository (ensure you add, commit and push your changes). Ensure that the screenshot includes the following:

    - All of the terms on the left hand side of the view - ensure the name of the turbine and the measure is clearly visible.
    - The chart title should be visible.
    - The chart should be displaying at least 10 minutes of data.

Next you will use PowerBI to to perform some transformations of your data and generate new insights.

Power BI

You will use CosmosDB as a Warm Data store and then leverage the Power BI desktop to generate some insights.

1. Create and configure CosmosDB for use as your Warm Storage.

    Note: The **Wind Farm Dashboard** will stream a large volume of data that will exceed the default 400 RUs capacity of a container. Set the RUs to 5000 to ensure all the data is captured. Remember to delete the resources when you are no longer using them as you will incur more cost with the higher RUs.

2. Configure your Azure Streaming Analytics instance to stream telemetry to CosmosDB.

    - Confirm data is flowing to Cosmos DB

Tip: Review the following materials:

- **Course**: *DEV326x IoT Data Analytics and Storage*
- **Module:** *Warm Storage*
- **Lab:** *Getting Started with Warm Storage*

3. Install Power BI and use the Cosmos DB account and database as the source of data.

   - Get Data - Choose **Azure** and select **Azure Cosmos DB**, then click **Connect**.
   - In the **Azure Cosmos DB** dialog, supply the following:
     - URL - you will find this value on the **Overview** pane of the **Cosmos DB Account** you created in Azure.
     - Database - You can remind yourself of the database name by using the **Browse** capability under **Collections** of the **Cosmos DB Account** you created in Azure.
     - Collection - You can remind yourself of the database name by using the **Browse** capability under **Collections** of the **Cosmos DB Account** you created in Azure.

   Tip: Review the following materials:

   - **Course**: *DEV312x Business Intelligence for IoT Solutions*
   - **Module:** *Power BI and IoT*

     Note: You will need to ensure that a minimum of 10 minutes or so of data has flown into Cosmos DB before you you continue with the reporting aspect.

4. A dialog will display the data - however, as the data is stored as JSON, you will need to help Power BI understand the data. Click **Edit** to update the query.

5. The **Power Query Editor** will show a single column of data. In the header of the column, to the right of the **Document** title is a button that expands the record - click it.

   - A selection popup appears that lists the columns that can be expanded. These are the top level properties in the JSON. Unselect all columns except:

- A selection popup appears that lists the columns that can be expanded. These are the top level properties in the JSON. Unselect all columns except:
  - telemetry
  - EventEnqueuedUtcTime

  Click **OK**. You will now see two columns with the **Document.EventProcessUtcTime** showing actual data.

6. This time, we need to expand the **Document.telemetry** column - click the expand button.
   - In the popup, unselect all columns except:
     - windSpeed
     - lowSpeedShaftRpm
     - power
     - name

   Click **OK**. You will see these columns now displayed, again containing data.

7. Take another look at the column headers - to the left of column name is an icon. It is currently showing **ABC123** in every column - this means that it is using a general purpose data type rather than identifying the underlying data type from the JSON data. We need to address that.

8. Right-click on each column header in turn, mouse over **Change type** and choose the appropriate data type as shown below:
   - Document.telemetry.windSpeed - Decimal Number
   - Document.telemetry.lowSpeedShaftRpm - Decimal Number
   - Document.telemetry.power - Decimal Number
   - Document.telemetry.name - Text
   - Document.EventEnqueuedUtcTime - Date/Time

   Note that the column icon changes to reflect the column data type.

9. We now need to add three calculated columns to enrich our data:

- PowerRatio - this column will calculate the ratio between the power measure and the lowPowerShaftRpm.
- Hour - by default, PowerBI will only display a Date dimension - by adding a custom column that exposes hour, we can drill into hours.
- Minute - by adding a custom column that exposes minute, we can drill into values by minute.

10. In the **Power Query Editor** ribbon, select the **Add Column** tab and click **Custom Column**.

    - Change the **New column name** to **PowerRatioInitial**.
    - Change the **Custom column formula** to `[Document.telemetry.power]/[Document.telemetry.lowSpeedShaftRpm]`
    - Click **OK** to add the column.

    This creates a column populated with result of dividing power by lowSpeedShaftRpm - we can use this to assess the generator efficiency.

11. Right-click the **PowerRatioInitial** column header and change the type to **Decimal Number**.

12. In the **Power Query Editor** ribbon, select the **Add Column** tab and click **Custom Column**.

    - Change the **New column name** to **PowerRatio**.
    - Change the **Custom column formula** to `if [PowerRatioInitial] = Number.PositiveInfinity or [PowerRatioInitial] = Number.NegativeInfinity or Number.IsNaN([PowerRatioInitial]) or [PowerRatioInitial] > 25 then 20 else [PowerRatioInitial]`
    - Click **OK** to add the column.

    This creates a column populated with either the valid result from **PowerRatioInitial** or zero.

13. Right-click the **PowerRatio** column header and change the type to **Decimal Number**.

14. To add additional columns for time values, select the

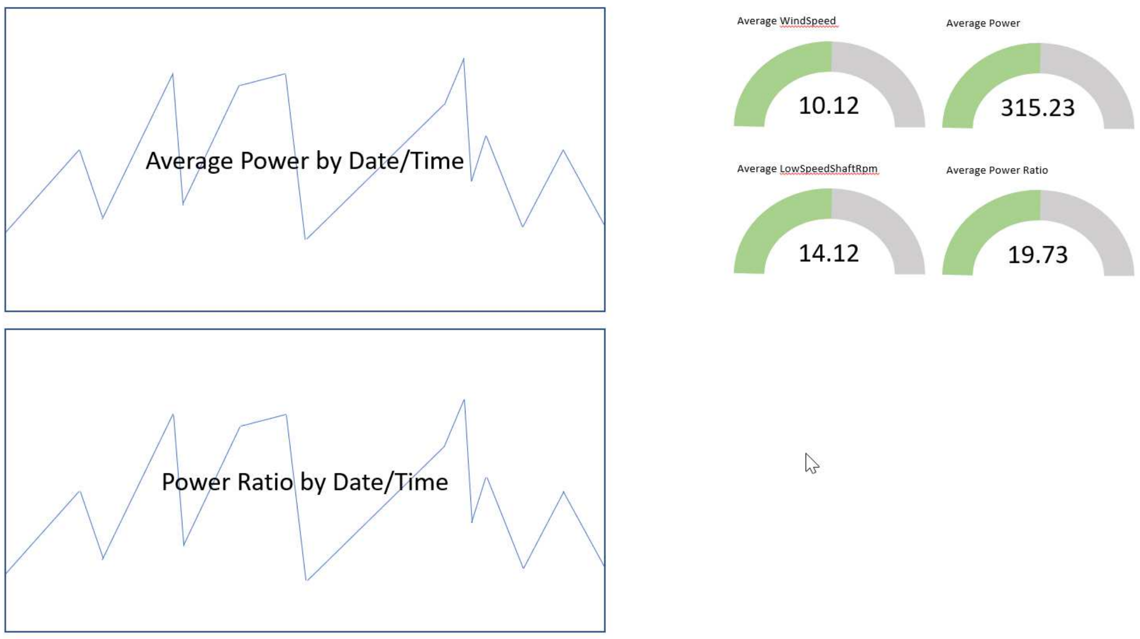
Document.EventEnqueuedUtcTime column

15. In the ribbon, in the **From Date & Time** group, click **Time** and then click **Hour** and **Hour** - a column called **Hour** is added and contains only the hour portion of the **Document.EventEnqueuedUtcTime** value.

16. In the ribbon, in the **From Date & Time group**, click **Time** and then click **Minute** - a column called **Minute** is added and contains only the minute portion of the **Document.EventEnqueuedUtcTime value**.

17. We have now finished editing the query and have shaped the data as we need. In the ribbon, select the **Home** tab, and click **Close and Apply**. The Query editor will close and the report design surface will be displayed.

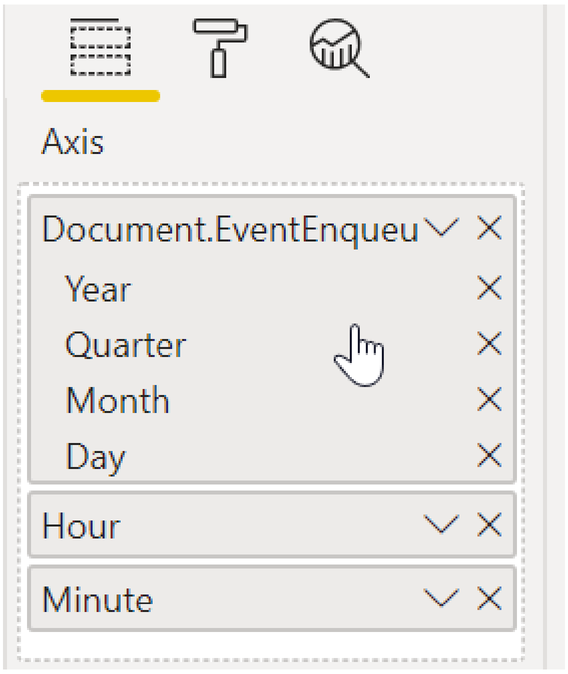
18. Create a report and layout the charts as shown:



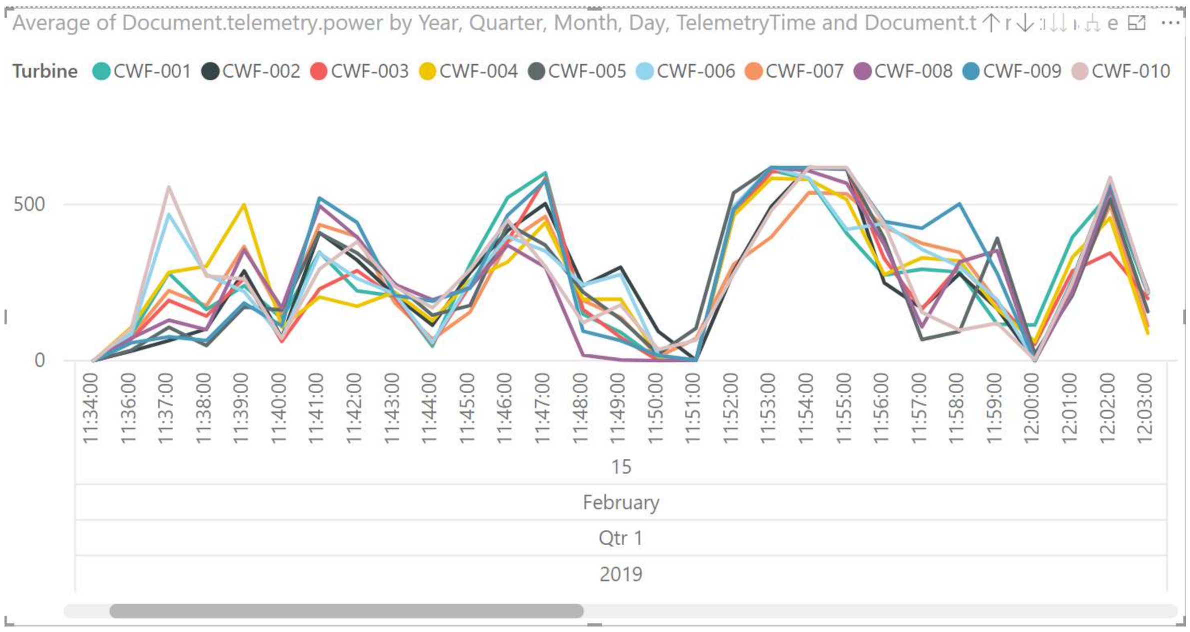
- Use gauges to show:
  - Average Power
  - Average WindSpeed
  - Average LowPowerShaftRpm
  - Average Power Ratio
- Use Line charts to show:
  - Average Power by Turbine over Date/Time

- Average Power Ratio by Turbine over Date/Time

> **Hint**: To be able to drill in to both Date and Time on the axis of the line charts, you will need to drag the **Document.EventEnqueuedUtcTime**, the **Hour** column and the **Minute column on to the Axis**:



If done correctly, the Axis of the average Power chart will look similar to:



Note: In the image above, the data is drilled into the minute level and is displaying around 20 minutes of data.

19. Once you have created the charts and are able to see the telemetry data from

Cosmos DB, examine the Power Ratio Chart - can you identify a Wind Turbine that is operating at a lower Power Ratio? The correct Wind Turbine will have a Power Ratio of approx. 17 vs. approx. 20.

20. **DELIVERABLE**: Take a screen shot of the **PowerBI Dashboard** view and save it as **PowerBIDashboard-[YOUR STUDENT ID].png** for submission. Save this file in the **Lab2** folder within your GitHub repository (ensure you add, commit and push your changes). Ensure that the screenshot includes the following:

    - A complete view of all charts on your dashboard.
    - Ensure the Wind Turbine with the low power ratio is clearly visible.

21. Once you have created and configured the resources, generate an Azure Resource Management script that documents the created resources. Do this by navigating to the **Overview** page of the **IoTCapstoneRG** resource group. Select all of the items in the resource group and then select **Export template** in the toolbar. Once generated, select **Download** - the downloaded file will be named similar to **ExportedTemplate-IoTCapstoneRG.zip**.

    - **DELIVERABLE**: Rename this file to **ExportedTemplate-IoTCapstoneRG-WindFarm-[YOUR STUDENT ID].zip** and set aside for submission for assessment. Save this file in the **Lab2** folder within your GitHub repository (ensure you add, commit and push your changes).