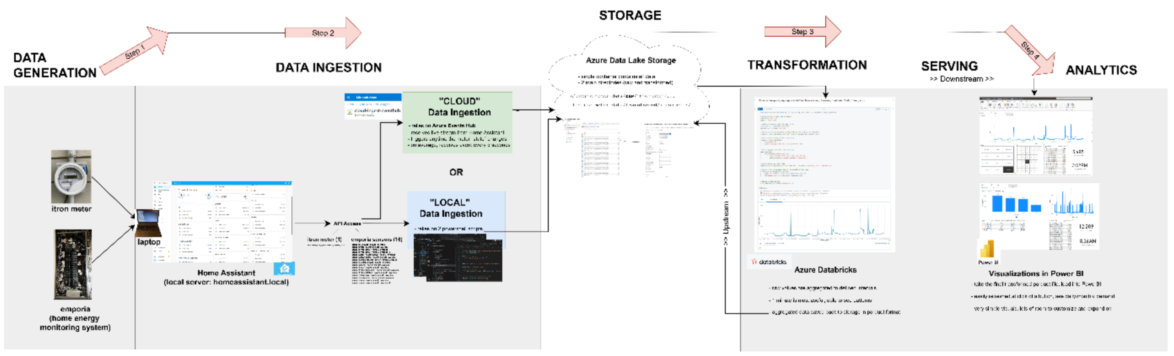
- **Project Video Demo:** Electricity Consumption Explorer Overview (9 min)
- **Google Slides:** Electricity Consumption Explorer Slides
- **GitHub Repo:** https://github.com/dfroslie-ndsu-org/f23-project-brockgion

- **Business solution**
  - **An overview of the business problem and why this was interesting to you**
    - I wanted to understand my electricity consumption patterns at home. My main motivation going into this was to understand *what exactly* contributes to my overall monthly electricity usage (I just knew my average monthly usage was ~760/kWh). It was interesting to me to use data to see precisely what's going on within my own household.
  - **The outcome of your analysis - analytics, ML, answers to your questions**
    - RESULT: I have a real-time home energy monitoring system with analytics! **I reduced my electricity use by ~7% in 3 months (~760/kWh to ~708/kWh).** I can monitor my energy usage patterns and see relative electricity % breakouts on appliances. My initial goal is to reduce electricity usage. Great start to understanding how I can potentially size a solar PV rooftop system for clean energy generation.
- **Technical solution**
  - **A high level architecture diagram**



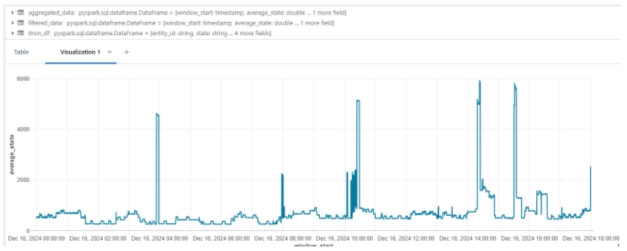
SEE FULL-SIZE DIAGRAM: Data Engineering Lifecyle - Electricity Consumption Explorer

- **Technologies used**
  - Hardware: Emporia Home Energy Monitor, xcel itron smart meter (gen5 Riva)
  - Software: Home Assistant (free, open-source)
  - Microsoft Services: Azure Data Lake Storage Gen2, Azure Key Vault, Azure Databricks, Powershell Scripts, Power BI, Azure Events Hub, Azure Function

- **Overview of your implementation for each phase of the data engineering lifecycle**

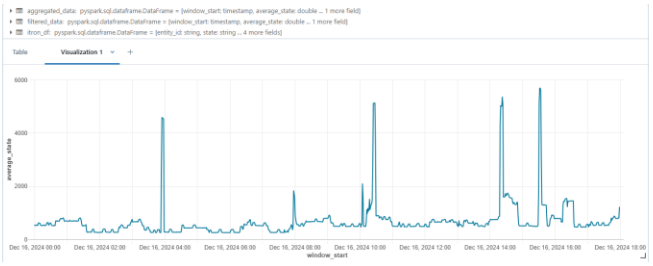
1. **Generation**
    - i. 2 data sources
        - *Itron Electric Meter* (installed on premise, at house)
        - *Emporia Home Circuit Monitoring System* (installed at breakerbox, at house)
    - ii. All data from itron meter + emporia circuits integrated into Home Assistant
        - Home Assistant required customization of configuration YAML code + 3-addons (*Xcel iTron MQTT*, *Mosquitto broker*, *Advanced SSH & Web Terminal*)
2. **Ingestion**
    - i. All data captured in Home Assistant is initially raw values (the “meter state” stored in Watts, the timestamp values stored in UTC timezone).
    - ii. To access data in Home Assistant via the API, use device names
        - Meter (1)
            - a. sensor.xcel_itron_5_instantaneous_demand_value
        - Emporia circuits (16)
            - a. sensor.fridge_basement_power_minute_average
            - b. sensor.boiler_power_minute_average
            - c. sensor.dishwasher_power_minute_average
            - d. etc.
    - iii. Two methods to ingest data and upload to Azure storage container
        - LOCAL
            - a. Two powershell scripts are used to ingest data
                - i. get_itronmeter_data.ps1
                - ii. get_emporiacircuits_data.ps1
            - b. The scripts parse data into daily .csv files and upload
                - i. 2024-12-16_sensor.xcel_itron_5_value.csv
        - CLOUD
            - a. Home Assistant captures data every ~5 seconds from the meter, sends real time updates to Azure Event Hubs
            - b. Triggers Azure Function
                - i. Azure Function appends values to single parquet file, saves raw data to azure storage container `electric-meter-data / raw / itronmeter / parquet`

3. **Transformation / Serving / Storage**
   i. All data stored in Azure Data Lake Storage (Gen2) (refer to as "adls2")
   ii. Azure Databricks can access adls2, transforms raw .csv/parquet files
   iii. Raw data is captured at most granular level (seconds), so it's useful to use Databricks to aggregate to broader intervals (15 min, hourly, etc.)
      - 1 second



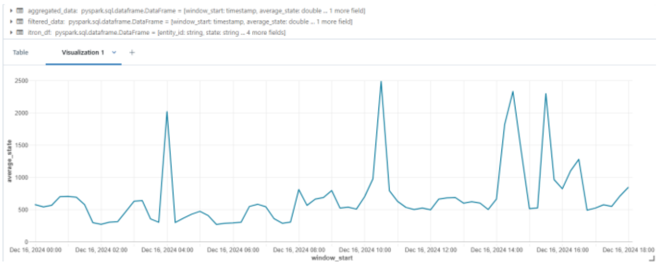
```
aggregated_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
filtered_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
itron_df: pyspark.sql.dataframe.DataFrame = [entity_id: string, state: string ... 4 more fields]
```

      - 1 minute interval



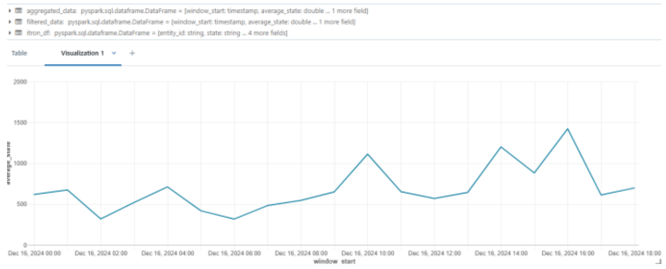
```
aggregated_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
filtered_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
itron_df: pyspark.sql.dataframe.DataFrame = [entity_id: string, state: string ... 4 more fields]
```

      - 15 minute



```
aggregated_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
filtered_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
itron_df: pyspark.sql.dataframe.DataFrame = [entity_id: string, state: string ... 4 more fields]
```

      - Hourly



```
aggregated_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
filtered_data: pyspark.sql.dataframe.DataFrame = [window_start: timestamp, average_state: double ... 1 more field]
itron_df: pyspark.sql.dataframe.DataFrame = [entity_id: string, state: string ... 4 more fields]
```

      - **Takeaway: 1 minute interval data is the best** for my intended analysis, allows manageable size with useful detail for patterns

iv. Other transformations to raw data:
- Adding a "date" field by extracting value from "Last_changed" timestamp ("YYYY-MM-DD").
- Useful to have simplified date to be able to filter on in Power BI
- BEFORE data transformation:
  a. 3 original fields
     i. *Entity_id* <-- constant value for all (sensor.xcel)
     ii. *State* <--meter value, expressed in Watts
     iii. *Last_changed* <--timestamp of recorded value
- AFTER data transformation
  a. 3 original fields + **1 added date field**
     i. Entity_id
     ii. State
     iii. Last_changed
     iv. Date

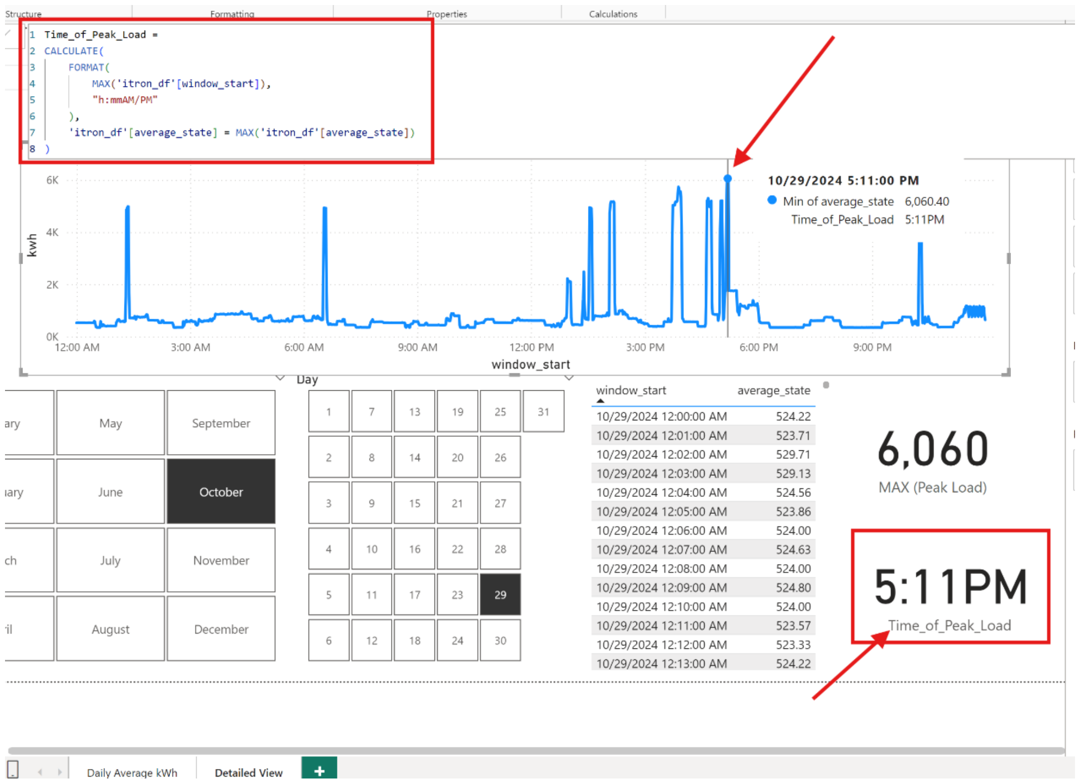
v. Databricks final step: save transformed data back into adls2 storage
- Container: `electric-meter-data / transformed / itronmeter /`
- LOCAL ingested data final saved filename:
  a. itron_meter_data_1minute_interval.parquet
- CLOUD ingested data final saved filename:
  a. cloud_ingest_itron_meter_data_1minute_interval.parquet

4. Analytics

i. Many tools available: Home Assistant, Emporia mobile app, Power BI
ii. Tool of choice for customized data visualizations: Power BI
iii. Use Power BI visualizations to easily see "peak load" patterns
iv. Accessing Data in Power BI
- The parquet files are located in adls2 storage, so the query in Power BI fetches data based on that URL
  a. https://azdbr....blob.core.windows.net/electric-meter-data/transformed/itronmeter/cloud_ingest_itron_meter_data_1minute_interval.parquet

v. Once data loaded into PowerBI, ability to add in calculated dimensions for enhanced analysis:
- Max_Peak_Load
- Time of Peak Load

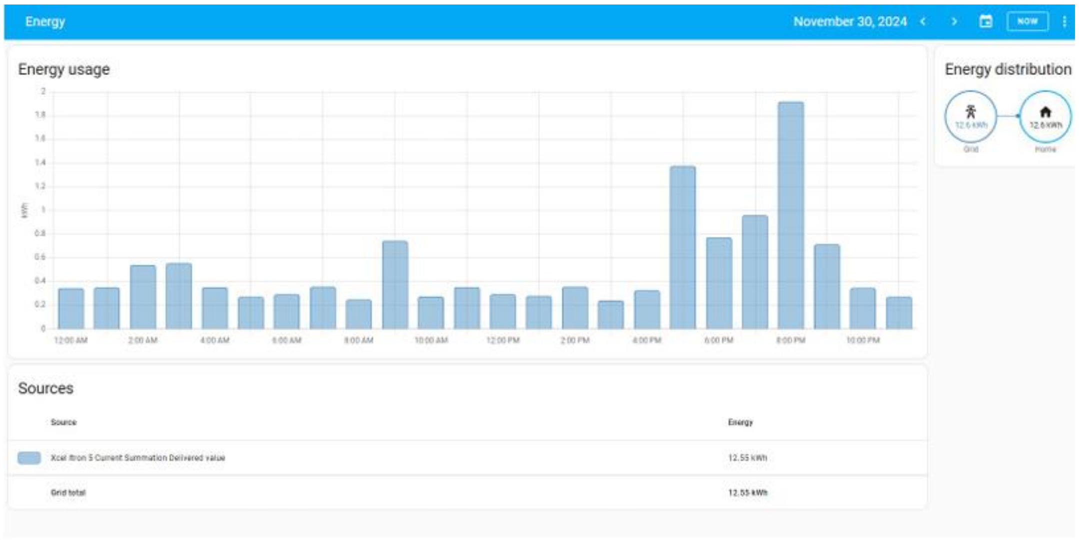
vi. PowerBI provides complete customization of page layout, able to see visuals combined with data. Very useful to quickly drill into seeing daily patterns



```
Time_of_Peak_Load =
CALCULATE(
    FORMAT(
        MAX('itron_df'[window_start]),
        "h:mmAM/PM"
    ),
    'itron_df'[average_state] = MAX('itron_df'[average_state])
)
```

window_start	average_state
10/29/2024 12:00:00 AM	524.22
10/29/2024 12:01:00 AM	523.71
10/29/2024 12:02:00 AM	529.71
10/29/2024 12:03:00 AM	529.13
10/29/2024 12:04:00 AM	524.56
10/29/2024 12:05:00 AM	523.86
10/29/2024 12:06:00 AM	524.00
10/29/2024 12:07:00 AM	524.63
10/29/2024 12:08:00 AM	524.00
10/29/2024 12:09:00 AM	524.80
10/29/2024 12:10:00 AM	524.00
10/29/2024 12:11:00 AM	523.57
10/29/2024 12:12:00 AM	523.33
10/29/2024 12:13:00 AM	524.22

Example view of PowerBI Dashboard showing time of "Peak Load" (6,060kwh at 5:11PM)



	Source	Energy
🟦	Xcel Itron 5 Current Summation Delivered value	12.55 kWh
	Grid total	12.55 kWh

Home Assistant built-in dashboard to show "Energy usage" by the hour

5. **Machine Learning – N/A**
    i. Not implemented yet. Ideally, want the ability to perform non-intrusive load monitoring.
6. **Reverse ETL – N/A**
    i. Not implemented. No need to load into other data systems yet.

Undercurrents:

- **Security**
    - Home Assistant is configured on an isolated home network (Mtr-Rdr) to keep all communications private, adhering to the principle of least privilege, where only I have root access to modify configurations. All data is securely stored in Azure Storage Containers, requiring shared key access for retrieval.
- **Data Management**
    - Data is initially captured and previewed through Home Assistant as the central monitoring hub. In the event of downtime, raw data readings can be lost, requiring manual annotation of missing events such as power outages.
- **DataOps**
    - Data ingestion began as a manual process using PowerShell scripts to retrieve data from Home Assistant. Automation is in progress with tools like Azure Event Hubs and Azure Functions, enabling near real-time ingestion and reducing manual effort.
- **Data Architecture**
    - The system captures all data at the finest level of granularity, down to one-second intervals where available, ensuring no loss of detail. Data is transformed and stored in Parquet format to optimize query performance and support scalable analytics.
- **Orchestration**
    - Daily and weekly batch processes for data retrieval and uploads are currently handled using PowerShell scripts. Aiming to use automated cloud workflows to ensure continuous and reliable data processing.
- **Software Engineering**
    - The system uses a combination of PowerShell scripts, Python scripts, and Home Assistant YAML files for ingestion, transformation, and automation. Data quality assurance is performed manually, supported by Home Assistant dashboards for anomaly detection and monitoring trends.

- **Implementation details**
  - **What cloud resources would be required to reproduce the work**
    - *Please see video demos and refer to high level architecture diagram
    - Cloud resources are all available within Microsoft Azure Portal
    - Most cloud services have a free tier, potential charge$ based on usage
      - **(4) Azure Services:**
        1. **Events Hub:** Event Hubs—Real-Time Data Ingestion | Microsoft Azure
           - NOTE: Events Hub charged on throughput units. Read more: Pricing - Event Hubs | Microsoft
           - Recommend using the "Basic" tier (~$10/mo), has max retention period of 1 day for data stream
        2. **Azure Databricks:** What is Azure Databricks? - Azure Databricks | Microsoft Learn
        3. **Azure Storage Container:** Azure Container Storage – Volume Management | Microsoft Azure
        4. **Azure Function + Azure Function App:** Azure Functions Overview | Microsoft Learn
      - **(2) Home Assistant Integrations:**
        1. Setup Home Assistant's "Azure Event Hub" Integration: home-assistant.io/integrations/azure_event_hub/
        2. Enable "nginx-proxy-manager" add-on hassio-addons/addon-nginx-proxy-manager: Nginx Proxy Manager - Home Assistant Community Add-ons
  - **What steps would be required to refresh the data, perform the transformations, and serve the data for business consumption**
  - **Using a cloud-based data ingestion approach, only 2 requirements**
    - Step 1: Run Azure Databricks, perform job to save parquet file
    - Step 2: Open PowerBI, click "refresh" data. You can view *all* data now
  - **Using a local data ingestion approach, more steps involved initially, but end result is the same**
    - Step 1: Run Powershell scripts, verify uploaded to storage container
    - Step 2: Run Azure Databricks, perform multiple jobs to batch .csv files
    - Step 3: Open PowerBI, update parquet file URL. You can view all data

- **An overview of how the code is structured in the repo**
  - All relevant code is located in the **`src` folder**. This contains the scripts necessary to run both the local/cloud data ingestion process. The `src` folder is split into two main parts:
    - **ingestion:** This contains the core scripts we worked on as part of the class. It includes everything related to pulling data from Home Assistant manually, such as PowerShell scripts for downloading and processing the data locally.
    - **cloud-ingestion**: This section shows the extra effort to use a cloud-based workflow for ingesting and processing the data automatically. It contains the code for an Azure Function that processes the data as it arrives in Azure Event Hubs
  - **PowerBI Dashboard File:**
    - PowerBI-Dashboard-Electricity-Consumption-Explorer.pbix
    - In order to user PowerBI-Dashboard .pbix file it requires Power BI Desktop because it relies on using a parquet file.
    - **Instructions to load in data**
      - Choose "Get Data > Parquet" on the Ribbon menu
      - Use this parquet file as sample data to explore (setup this container as anonymous access for testing only)
      - *NOTE: Dataset contains data 11/01/24 to 12/15/24

- All other non-code related project files are contained in **“SupplementaryInfo / ProjectFiles”** folder.
    - **Screenshots:** This contains images of various steps performed along the way of building this project. Mostly for reference.
    - **Project Initiation Doc:**
        - **Initiation:** START-Project-Initiation-CSCI-622-Electricity-Consumption-Explorer.pdf
        - **Recap:** FINAL-Project-Recap-CSCI-622-Electricity-Consumption-Explorer.pdf