Created by: Kacper Kulesza

TASK 2

Alright, let's analyze this dataset. We'll assume we have a JSON file with these fields: **APPLIANCE_ID**, **MODEL_ID**, **COOKZONE_ID**, **EVENT_DATETIME**, and **MESSAGE_TEXT**.

First, we'll need to load this data into a Pandas DataFrame. Then we can perform various analyses like checking for missing values, exploring patterns, and aggregating information. This will allow us to take a practical look at the data, facilitating our analysis of values through the ability to perform tasks on a specific example.

My jupyter notebook is here:

Question 1:

How will you determine the event that cookware was moved from one zone to another?

Provide pseudo code.

Assumption: Consider a single hob on which only one piece of cookware was placed.

Answer:

Pseudocode:

```
sorted_dataset = dataset.sort_values(by=['APPLIANCE_ID', 'EVENT_DATETIME'])

previous_appliance_id = None

previous_cookzone = None

moved_events = []


for index, row in sorted_dataset.iterrows():

    current_appliance_id = row['APPLIANCE_ID']

    current_cookzone = row['COOKZONE_ID']



    if previous_appliance_id is None or current_appliance_id != previous_appliance_id:
```

```
        previous_cookzone = current_cookzone


    else:
        if current_cookzone != previous_cookzone:
            moved_event = {
                'APPLIANCE_ID': current_appliance_id,
                'MODEL_ID': row['MODEL_ID'],
                'previous_cookzone': previous_cookzone,
                'new_cookzone': current_cookzone,
                'EVENT_DATETIME': row['EVENT_DATETIME']
            }
            moved_events.append(moved_event)
        previous_cookzone = current_cookzone
    previous_appliance_id = current_appliance_id


for event in moved_events:
    print("Appliance ID:", event['APPLIANCE_ID'])
    print("Model ID:", event['MODEL_ID'])
    print("Previous Cookzone:", event['previous_cookzone'])
    print("New Cookzone:", event['new_cookzone'])
    print("Event Datetime:", event['EVENT_DATETIME'])
    print("---------------------------")
```

STEPS ONE BY ONE:

1. First, the data is sorted by 'APPLIANCE_ID' and 'EVENT_DATETIME' in ascending order.
2. Then, variables are initialized, including a variable to store the previous cooking zone.
3. Next, a loop iterates through the sorted data, where for each row it checks whether the cooking zone has changed from the previous row for the same appliance. If so, it is considered an event of cookware being moved.
4. The move events are added to the 'moved_events' list in the form of a dictionary with required information such as APPLIANCE_ID, MODEL_ID, previous and new cooking zones, and the date and time of the event.
5. Finally, after the loop finishes, 'moved_events' are analyzed or further processed, such as printing these events to the console for verification.

We can try that exercise on my Data Sheet, we created in Jupyter Notebook. You can check that by the link I shared before(it's the same Jupyter Notebook).

Question 2.

How would you estimate the number of meals each appliance was used to prepare over a certain period?

Meal Estimation on Induction Hobs: A Theoretical Approach

Data Source: JSON messages generated by induction hobs.

Key Attributes:

- APPLIANCE_ID: Unique identifier for the hob.
- COOKZONE_ID: Identifier for the specific cooking zone (burner).
- EVENT_DATETIME: Timestamp of the event.

Meal Definition:

A "meal" - distinct cooking session initiated by activating the hob.

Cook Zone Analysis:

Changes in COOKZONE_ID during an active session (hob is on) suggest a potential transition to a new cooking session. This accounts for users switching between cook zones for different dishes within a single meal prep.

Identifying Relevant Events:

- Activated: Hob is turned on (cooking session starts).
- Deactivated: Hob is turned off (cooking session ends).
- CookZone Change: Active hob experiences a change in the selected cook zone.

Meal Count Calculation:

1. Group JSON messages by APPLIANCE_ID to isolate events for each hob.
2. Count the occurrences of "Deactivated" followed by "Activated" transitions, representing the end and beginning of distinct cooking sessions.
3. Include the first "Activated" event within the time window as a potential meal initiation.
4. If a cook zone change occurs while the hob is active (between Activated and Deactivated), this signifies a new cooking meal.

Time Windowing:

The analysis can be performed over various timeframes (daily, weekly, monthly) depending on the desired level of detail in the results.

To ensure accurate meal counts and durations, we consider the certain time window between the "Activated" and "Deactivated" events as the period during which a meal was prepared.

Question 4.

Imagine you are interested in being able to calculate the energy consumption and related cost for each induction hob in Poland in a given time period.

a. Where would you ideally get this information from?

If you're living in Poland and want to calculate the energy consumption and related costs for each induction hob you can get this information from many websites or manulas. Here I am going to present what I found:

1. **Electricity Providers**:
    - Contact "PGE Polska Grupa Energetyczna," one of the largest electricity providers in Poland. They can provide your specific electricity tariff information.
    - We use their customer service line
2. **Online Energy Calculators**:
    - Use the Kalkulator Energii on the website of the Urząd Regulacji Energetyki
    - This tool allows you to enter the power consumption of your induction hob, usage time, and electricity rate to calculate costs.
3. **Energy Efficiency Labels**:
    - Check the energy efficiency label on your Bosch or whirpool or any other induction hob, which might show an estimated annual energy consumption.
4. **Government Energy Websites**:
    - Visit the website of the Polish Ministry of Climate and Environment for energy-related publications and resources.
5. **Appliance Manuals**:
    - Example: Refer to the user manual of your Bosch induction hob.
    - The manual might state the power rating, such as "Rated power: 2.2 kW."

We can also try to calculate the power usage by ourselves. We should use simple formula found on internet or maybe even in Physics books:
Energy Cost=Power (kW)×Time (hours)×Electricity Rate (PLN/kWh)

For example, if your induction hob is rated at 2.2 kW, and you use it for 1 hour a day, and your electricity rate is 0.70 PLN per kWh:

Energy Cost=2.2 kW×1 hour×0.70 PLN/kWh=1.54 PLN per dayEnergy Cost=2.2kW×1hour×0.70PLN/kWh=1.54PLN per day

b. How would you store and organize additional data?

Pandas

Alright, so if you're diving into this energy data journey, pandas is like our Swiss knife for handling data in Python.

Imagine you have a bunch of information about these induction hobs in Poland, like their models, power ratings, how long they're used each day, and the electricity rates.

```
import pandas as pd

data = {
   'Hob Model': ['Bosch Serie 8', 'Siemens', 'AEG'],
   'Power (kW)': [1.8, 2.2, 2.5],
   'Daily Usage (hours)': [2, 1, 1.5],
   'Electricity Rate (PLN/kWh)': [0.65, 0.70, 0.80],
}

df = pd.DataFrame(data)

print(df)
```

And that’s it, it is really simple to use and gives as millions of options to manipulate the data, I used many of those in exercises before.

EXCEL:

Now, if you prefer a more visual approach or maybe want to share this with others who are more comfortable with Excel, we can do that too. Excel is a suitable tool for this task due to its user-friendly interface and calculation capabilities. With built-in formulas, such as SUM and IF, you can easily calculate energy costs based on power usage and rates. Excel's charting features also allow for clear visualization of cost comparisons between different induction hobs. Its flexibility, scalability, and ability to share and collaborate make it a practical choice for organizing and analyzing this data.

MySQL

Now, let's say you're dealing with a lot of hobs, and you need a robust system to handle all this data. That's where MySQL, our trusty database, comes into play. It is probably the most complex tool for data manipulation and storage. MySQL shines when managing large volumes of data for numerous induction hobs. Its structured database format allows for efficient storage and retrieval of information on hundreds or even thousands of hobs. With MySQL, you can ensure data integrity and security, essential for handling sensitive energy consumption and cost data. Its ability to handle complex queries and join operations makes it ideal for analyzing trends and patterns across different hobs over time. In summary, MySQL's robustness and scalability make it an excellent choice for a comprehensive and secure data repository in this scenario.

```
CREATE TABLE InductionHobs (
    HobID INT AUTO_INCREMENT PRIMARY KEY,
    HobModel VARCHAR(255),
    PowerKW FLOAT,
    DailyUsageHours FLOAT,
    ElectricityRate DECIMAL(5, 2),
    DailyCost DECIMAL(10, 2)
);
```

Here, we're creating a table called 'InductionHobs'. Each hob gets a unique ID, and we're storing its model, power, daily usage, electricity rate, and we even have a spot for the daily cost.

Summing up:
So whether you're tinkering with pandas in Python, organizing in Excel, or building a robust data fortress in MySQL, these tools are like our trusty companions in the quest to understand energy consumption and costs for these induction hobs in Poland.

c. Draw a proposed architecture of the IT solution. Assume you can use AWS cloud services and other technical tools of choice. Describe major components and briefly justify why you selected those.

AWS S3 (Simple Storage Service):

S3 provides scalable, secure, and durable object storage. We can store raw data from induction hobs and processed data.

AWS Glue:

Glue offers a fully managed ETL service. AWS Glue Studio is a graphical interface that makes it easy to create, run, and monitor data integration jobs. We can use Glue crawlers to discover and catalog metadata about the raw data stored in S3.

AWS Athena:

Athena allows ad-hoc querying of data in S3 without the need to set up and manage servers.

AWS RDS/MySQL RDS

RDS provides a managed relational database. We can store structured data like hob models, power ratings, and electricity rates. This data can be used for reference and to enrich the analysis.

AWS Lambda:

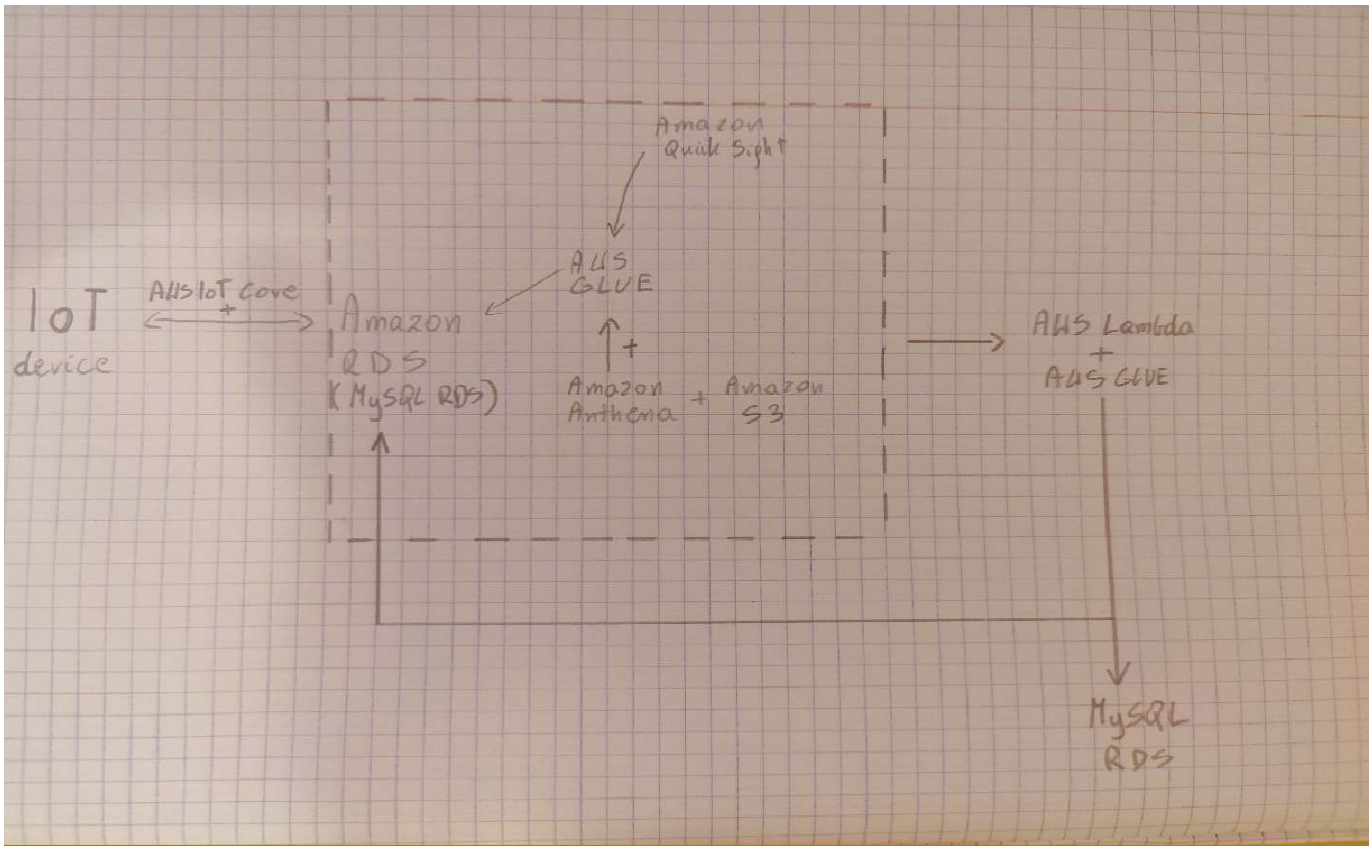
Lambda functions can be triggered by events. We can use Lambda to perform tasks like updating the RDS database with new hob models or sending notifications based on cost thresholds.

Amazon Quicksight:

Quicksight integrates seamlessly with other AWS services. We can create interactive dashboards and visualizations to present key metrics and trends related to energy consumption and costs.

AWS IoT Core:
IoT Core provides a managed service for securely connecting and managing IoT devices. Induction hobs can send real-time usage data to IoT core, enabling near real-time monitoring and analysis.

Below You can see the architecture I have tried to design.


Amazon
Quick Sight
AWS
GLUE
IoT
device
AWS IoT Core
+
Amazon
RDS
(MySQL RDS)
Amazon
Anthena
+
Amazon
S3
AWS Lambda
+
AWS GLUE
MySQL
RDS