IND320 - Assignment 4

Name: Eskil Torp Skogsholm

GitHub link: https://github.com/esksko/IND320-assignments-esksko

Streamlit app: https://ind320-esksko.streamlit.app/

AI usage

During the development of this assignment, I used several AI tools to support my workflow. GitHub Copilot assisted with code completion, while ChatGPT and Claude provided explanations, troubleshooting guidance, and code snippets. I relied more heavily on AI than usual for the SLiding Window Correlation and Forecasting pages due to time constrains. AI tools were also used to translat my plots from MAtplotlib to Plotly, which significantly reduced the time needed to refactor them.

Log

The Jupyter Notebook part of this assignment was straightforward, as a lot of the tasks were similar to previous assignments. First, I made a function that retrieved all the production and consumption data from the Elhub API and saved them into two dataframes. Then, I used Docker, Spark and Cassandra before connecting to MongoDB and making sure that I appended the data correctly without getting duplicates. I had already deleted the example data, so I did not encounter any storage issues.

The streamlit part of the assignment was more challenging. Exchanging the plots from matplotlib to Plotly was mostly straightforward except for the “All” plot on the Plots page. It was more difficult to get multiple axes to work than expected, but with a bit of tinkering, I found a decent solution. I decided to divide my pages into two sections: Weather and Energy. The weather section includes most of the API-data (Weather data) while the Energy section encompasses most of the MongoDB usage (Energy data). Getting the price areas to be consistent across pages also turned out to be a challenge. I tried to get it to work using the “key” attribute of st.radio, but I couldn’t get it to work, so I ended up doing it in a more manual way. This solution isn’t perfect, but it works.

The “Map and selectors” and “Snow drift calculation and illustration” pages were straightforward to implement by following the examples from the lectures. I decided to implement the “Elevation” bonus task on the map page. To get this to work, I just had to implement an API call to the correct open-meteo endpoint with the latitude and longitude from the map. The last two pages “Meteorology and energy production” and “Forecasting of

energy production and consumption" were more difficult. For these two pages I had to use AI tools more extensively than I usually would due to time constraints and other assignments. I am not 100% happy with the results, but I think the functionality works for the most part.

SWC correlation findings: Looking at the correlation between temperature and consumption data with these settings:

• January 2024

• Window of 168 hours (1 week)

• 0 lag

• Price area NO1

I see that there is a consistent negative correlation between the temperature and consumption data, hovering between -0.2 and -0.6 for the most part. This means that as the temperature decreases, there is an increase in energy consumption and vice versa. This negative correlation effect is most visible during in week 2/3 of January, where the correlation goes all the way down to -0.8. During this time, the temperature increased from -20C to above 0, and the consumption decreased significantly. Changing the lag by between -8 and 8 hours didn't affect the correlation very much. This indicates that the household demand for energy responds quickly to the temperature.

Tasks

In [ ]:
```python
import pandas as pd
import requests
from datetime import datetime
import calendar

def get_elhub_data(year_start, year_end, dataset):

    base_url = "https://elhub.no/api/v1/measurements"
    entity = "price-areas"
    urls = []
    all_data = []

    for year in range(year_start, year_end + 1):

        monthly_ranges = []
        for month in range(1, 13):
            # First day of the month
            start = datetime(year, month, 1, 0, 0, 0)

            # Last day of the month
            last_day = calendar.monthrange(year, month)[1]
            end = datetime(year, month, last_day, 23, 59, 59)
```

```
            # Format as ISO8601 with URL-encoded +01
            start_str = start.strftime("%Y-%m-%dT%H:%M:%S") + "%2B01"
            end_str = end.strftime("%Y-%m-%dT%H:%M:%S") + "%2B01"

            monthly_ranges.append((start_str, end_str))


        for month_start, month_end in monthly_ranges:
            url = f"https://api.elhub.no/energy-data/v0/{entity}?dataset={dataset}&
            urls.append(url)

    for url in urls:
        try:
            response = requests.get(url)
            data = response.json()

            print(f"Fetched data from {url}")

            for entry in data.get("data", []):
                attrs = entry.get("attributes", {})

                if "productionPerGroupMbaHour" in attrs:
                    monthly_list = attrs["productionPerGroupMbaHour"]
                elif "consumptionPerGroupMbaHour" in attrs:
                    monthly_list = attrs["consumptionPerGroupMbaHour"]
                else:
                    monthly_list = []

                all_data.extend(monthly_list)
        except Exception as e:
            print(f"Error fetching data from {url}: {e}")


    return pd.DataFrame(all_data)
```

In [12]:
```
pd_df = get_elhub_data(2022, 2024,  "PRODUCTION_PER_GROUP_MBA_HOUR")
```

```
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-01-01T00:00:00%2B01&endDate=2022-01-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-02-01T00:00:00%2B01&endDate=2022-02-28T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-03-01T00:00:00%2B01&endDate=2022-03-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-04-01T00:00:00%2B01&endDate=2022-04-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-05-01T00:00:00%2B01&endDate=2022-05-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-06-01T00:00:00%2B01&endDate=2022-06-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-07-01T00:00:00%2B01&endDate=2022-07-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-08-01T00:00:00%2B01&endDate=2022-08-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-09-01T00:00:00%2B01&endDate=2022-09-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-10-01T00:00:00%2B01&endDate=2022-10-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-11-01T00:00:00%2B01&endDate=2022-11-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2022-12-01T00:00:00%2B01&endDate=2022-12-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-01-01T00:00:00%2B01&endDate=2023-01-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-02-01T00:00:00%2B01&endDate=2023-02-28T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-03-01T00:00:00%2B01&endDate=2023-03-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-04-01T00:00:00%2B01&endDate=2023-04-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-05-01T00:00:00%2B01&endDate=2023-05-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-06-01T00:00:00%2B01&endDate=2023-06-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-07-01T00:00:00%2B01&endDate=2023-07-31T23:59:59%2
```

```
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-08-01T00:00:00%2B01&endDate=2023-08-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-09-01T00:00:00%2B01&endDate=2023-09-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-10-01T00:00:00%2B01&endDate=2023-10-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-11-01T00:00:00%2B01&endDate=2023-11-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2023-12-01T00:00:00%2B01&endDate=2023-12-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-01-01T00:00:00%2B01&endDate=2024-01-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-02-01T00:00:00%2B01&endDate=2024-02-29T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-03-01T00:00:00%2B01&endDate=2024-03-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-04-01T00:00:00%2B01&endDate=2024-04-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-05-01T00:00:00%2B01&endDate=2024-05-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-06-01T00:00:00%2B01&endDate=2024-06-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-07-01T00:00:00%2B01&endDate=2024-07-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-08-01T00:00:00%2B01&endDate=2024-08-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-09-01T00:00:00%2B01&endDate=2024-09-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-10-01T00:00:00%2B01&endDate=2024-10-31T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-11-01T00:00:00%2B01&endDate=2024-11-30T23:59:59%2
B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=PRODUCTION
_PER_GROUP_MBA_HOUR&startDate=2024-12-01T00:00:00%2B01&endDate=2024-12-31T23:59:59%2
B01
```

In [ ]:
```
pd_df.head()
```

Out[ ]:

	endTime	lastUpdatedTime	priceArea	productionGroup	quantityKwh	
0	2022-01-01T01:00:00+01:00	2025-02-01T18:02:57+01:00	NO1	hydro	1291422.4	01T00:0(
1	2022-01-01T02:00:00+01:00	2025-02-01T18:02:57+01:00	NO1	hydro	1246209.4	01T01:0(
2	2022-01-01T03:00:00+01:00	2025-02-01T18:02:57+01:00	NO1	hydro	1271757.0	01T02:0(
3	2022-01-01T04:00:00+01:00	2025-02-01T18:02:57+01:00	NO1	hydro	1204251.8	01T03:0(
4	2022-01-01T05:00:00+01:00	2025-02-01T18:02:57+01:00	NO1	hydro	1202086.9	01T04:0(

In [14]:
```
import os

# Setting environment variables for Spark and Hadoop
os.environ["HADOOP_HOME"] = "C:/Hadoop/hadoop-3.3.1"

os.environ["JAVA_HOME"] = "C:/Program Files/Microsoft/jdk-11.0.28.6-hotspot"

os.environ["PYSPARK_PYTHON"] = "python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "python"

os.environ["HADOOP_VERSION"] = "without"
```

In [15]:
```
# Connecting to Cassandra

from cassandra.cluster import Cluster

# Connecting to a local Cassandra instance
cluster = Cluster(["localhost"], port=9042)
session = cluster.connect()
```

In [16]:
```
from pyspark.sql import SparkSession

# Setting keyspace
session.set_keyspace("assignment_2_keyspace")

# Configuring Spark session to connect to Cassandra
spark = SparkSession.builder.appName('Assignment2SparkCassandraApp').\
    config('spark.jars.packages', 'com.datastax.spark:spark-cassandra-connector_2.1
    config('spark.cassandra.connection.host', 'localhost').\
    config('spark.sql.extensions', 'com.datastax.spark.connector.CassandraSparkExte
    config('spark.sql.catalog.mycatalog', 'com.datastax.spark.connector.datasource.
    config('spark.cassandra.connection.port', '9042').getOrCreate()
```

In [17]:
```
# Converting pandas DataFrame to Spark DataFrame

sdf = spark.createDataFrame(pd_df)
sdf.show(5)
```

```
+-------------------+-------------------+---------+--------------+-----------+---
----------------+
|            endTime|     lastUpdatedTime|priceArea|productionGroup|quantityKwh|
startTime|
+-------------------+-------------------+---------+--------------+-----------+---
----------------+
|2022-01-01T01:00:...|2025-02-01T18:02:...|      NO1|          hydro|  1291422.4|202
2-01-01T00:00:...|
|2022-01-01T02:00:...|2025-02-01T18:02:...|      NO1|          hydro|  1246209.4|202
2-01-01T01:00:...|
|2022-01-01T03:00:...|2025-02-01T18:02:...|      NO1|          hydro|  1271757.0|202
2-01-01T02:00:...|
|2022-01-01T04:00:...|2025-02-01T18:02:...|      NO1|          hydro|  1204251.8|202
2-01-01T03:00:...|
|2022-01-01T05:00:...|2025-02-01T18:02:...|      NO1|          hydro|  1202086.9|202
2-01-01T04:00:...|
+-------------------+-------------------+---------+--------------+-----------+---
----------------+
only showing top 5 rows
```

In [18]:
```
# Displaying schema and column names of the Spark DataFrame
print("DataFrame columns:")
sdf.printSchema()
print("\nColumn names:", sdf.columns)
```

```
DataFrame columns:
root
 |-- endTime: string (nullable = true)
 |-- lastUpdatedTime: string (nullable = true)
 |-- priceArea: string (nullable = true)
 |-- productionGroup: string (nullable = true)
 |-- quantityKwh: double (nullable = true)
 |-- startTime: string (nullable = true)


Column names: ['endTime', 'lastUpdatedTime', 'priceArea', 'productionGroup', 'quanti
tyKwh', 'startTime']
```

In [19]:
```
from pyspark.sql.functions import to_timestamp

# Converting startTime, endTime, and lastUpdatedTime to timestamp type

sdf = (
    sdf
    .withColumn("starttime", to_timestamp("starttime"))
    .withColumn("endtime", to_timestamp("endtime"))
    .withColumn("lastupdatedtime", to_timestamp("lastupdatedtime"))
)

# Converting all column names to lowercase
sdf = sdf.toDF(*[c.lower() for c in sdf.columns])
sdf.show(5)
```

```
+-------------------+-------------------+---------+---------------+-----------+-----
--------------+
|            endtime|    lastupdatedtime|pricearea|productiongroup|quantitykwh|
starttime|
+-------------------+-------------------+---------+---------------+-----------+-----
--------------+
|2022-01-01 01:00:00|2025-02-01 18:02:57|      NO1|          hydro|  1291422.4|2022-
01-01 00:00:00|
|2022-01-01 02:00:00|2025-02-01 18:02:57|      NO1|          hydro|  1246209.4|2022-
01-01 01:00:00|
|2022-01-01 03:00:00|2025-02-01 18:02:57|      NO1|          hydro|  1271757.0|2022-
01-01 02:00:00|
|2022-01-01 04:00:00|2025-02-01 18:02:57|      NO1|          hydro|  1204251.8|2022-
01-01 03:00:00|
|2022-01-01 05:00:00|2025-02-01 18:02:57|      NO1|          hydro|  1202086.9|2022-
01-01 04:00:00|
+-------------------+-------------------+---------+---------------+-----------+-----
--------------+
only showing top 5 rows
```

In [ ]:
```
# Printing number of rows
print(f"Number of rows: {sdf.count()}")
```

```
Number of rows: 657600
```

In [21]:
```
# Writing data into Cassandra

sdf.write\
    .format("org.apache.spark.sql.cassandra") \
    .options(keyspace="assignment_2_keyspace", table="production_per_group_hour") \
    .mode("append") \
    .save()
```

In [25]:
```
# Extracting priceArea, productionGroup, startTime, and quantityKwh from Cassandra.

extracted_df = spark.read \
    .format("org.apache.spark.sql.cassandra") \
    .options(keyspace="assignment_2_keyspace", table="production_per_group_hour") \
    .load() \
    .select("pricearea", "productiongroup", "starttime", "quantitykwh")
```

In [26]:
```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
import tomllib

# Getting username and password for MongoDB connection

with open("../.streamlit/secrets.toml", "rb") as f:
    cfg = tomllib.load(f)

USR = cfg["MongoDB"]["username"]
PWD = cfg["MongoDB"]["pwd"]

uri = f"mongodb+srv://esksko:{PWD}@ind320-esksko.5nbj7x0.mongodb.net/?retryWrites=t
```

```
# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))

# Selecting database and collection
db = client["IND320_assignment_4"] # Creates or connects to database
collection = db["production_data"] # Creates or connects to collection

# Converting spark DataFrame to pandas DataFrame, then to records for MongoDB inser
records = extracted_df.toPandas().to_dict("records")

# Inserting new records
result = collection.insert_many(records)

print(f"Inserted {len(result.inserted_ids)} records into MongoDB.")

# Verifying insertion by checking count
count = collection.count_documents({})
print(f"Total documents in collection: {count}")

# Showing sample document
print("\nSample document:")
print(collection.find_one())
```

```
Inserted 872953 records into MongoDB.
Total documents in collection: 872953

Sample document:
{'_id': ObjectId('69171b0969f823a8b878a737'), 'pricearea': 'NO2', 'productiongroup':
'solar', 'starttime': datetime.datetime(2021, 1, 1, 0, 0), 'quantitykwh': 876.556}
```

In [28]:
```
# Now doing the same for consumption data

pd_df = get_elhub_data(2021, 2024,  "CONSUMPTION_PER_GROUP_MBA_HOUR")
```

```
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-01-01T00:00:00%2B01&endDate=2021-01-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-02-01T00:00:00%2B01&endDate=2021-02-28T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-03-01T00:00:00%2B01&endDate=2021-03-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-04-01T00:00:00%2B01&endDate=2021-04-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-05-01T00:00:00%2B01&endDate=2021-05-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-06-01T00:00:00%2B01&endDate=2021-06-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-07-01T00:00:00%2B01&endDate=2021-07-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-08-01T00:00:00%2B01&endDate=2021-08-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-09-01T00:00:00%2B01&endDate=2021-09-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-10-01T00:00:00%2B01&endDate=2021-10-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-11-01T00:00:00%2B01&endDate=2021-11-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2021-12-01T00:00:00%2B01&endDate=2021-12-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-01-01T00:00:00%2B01&endDate=2022-01-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-02-01T00:00:00%2B01&endDate=2022-02-28T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-03-01T00:00:00%2B01&endDate=2022-03-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-04-01T00:00:00%2B01&endDate=2022-04-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-05-01T00:00:00%2B01&endDate=2022-05-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-06-01T00:00:00%2B01&endDate=2022-06-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-07-01T00:00:00%2B01&endDate=2022-07-31T23:59:59%
```

```
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-08-01T00:00:00%2B01&endDate=2022-08-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-09-01T00:00:00%2B01&endDate=2022-09-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-10-01T00:00:00%2B01&endDate=2022-10-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-11-01T00:00:00%2B01&endDate=2022-11-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2022-12-01T00:00:00%2B01&endDate=2022-12-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-01-01T00:00:00%2B01&endDate=2023-01-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-02-01T00:00:00%2B01&endDate=2023-02-28T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-03-01T00:00:00%2B01&endDate=2023-03-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-04-01T00:00:00%2B01&endDate=2023-04-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-05-01T00:00:00%2B01&endDate=2023-05-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-06-01T00:00:00%2B01&endDate=2023-06-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-07-01T00:00:00%2B01&endDate=2023-07-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-08-01T00:00:00%2B01&endDate=2023-08-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-09-01T00:00:00%2B01&endDate=2023-09-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-10-01T00:00:00%2B01&endDate=2023-10-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-11-01T00:00:00%2B01&endDate=2023-11-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2023-12-01T00:00:00%2B01&endDate=2023-12-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-01-01T00:00:00%2B01&endDate=2024-01-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
```

```
N_PER_GROUP_MBA_HOUR&startDate=2024-02-01T00:00:00%2B01&endDate=2024-02-29T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-03-01T00:00:00%2B01&endDate=2024-03-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-04-01T00:00:00%2B01&endDate=2024-04-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-05-01T00:00:00%2B01&endDate=2024-05-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-06-01T00:00:00%2B01&endDate=2024-06-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-07-01T00:00:00%2B01&endDate=2024-07-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-08-01T00:00:00%2B01&endDate=2024-08-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-09-01T00:00:00%2B01&endDate=2024-09-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-10-01T00:00:00%2B01&endDate=2024-10-31T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-11-01T00:00:00%2B01&endDate=2024-11-30T23:59:59%
2B01
Fetched data from https://api.elhub.no/energy-data/v0/price-areas?dataset=CONSUMPTIO
N_PER_GROUP_MBA_HOUR&startDate=2024-12-01T00:00:00%2B01&endDate=2024-12-31T23:59:59%
2B01
```

In [29]:
```
pd_df.head()
```

Out[29]:

	consumptionGroup	endTime	lastUpdatedTime	meteringPointCount	priceArea
0	cabin	2021-01-01T01:00:00+01:00	2024-12-20T10:35:40+01:00	100607	NO1
1	cabin	2021-01-01T02:00:00+01:00	2024-12-20T10:35:40+01:00	100607	NO1
2	cabin	2021-01-01T03:00:00+01:00	2024-12-20T10:35:40+01:00	100607	NO1
3	cabin	2021-01-01T04:00:00+01:00	2024-12-20T10:35:40+01:00	100607	NO1
4	cabin	2021-01-01T05:00:00+01:00	2024-12-20T10:35:40+01:00	100607	NO1

In [32]:
```
# Converting pandas DataFrame to Spark DataFrame
sdf = spark.createDataFrame(pd_df)
sdf.show(5)
```

```
+----------------+-------------------+-------------------+------------------+-----
----+-----------+-------------------+
|consumptionGroup|            endTime|    lastUpdatedTime|meteringPointCount|price
Area|quantityKwh|          startTime|
+----------------+-------------------+-------------------+------------------+-----
----+-----------+-------------------+
|           cabin|2021-01-01T01:00:...|2024-12-20T10:35:...|            100607|
NO1|  177071.56|2021-01-01T00:00:...|
|           cabin|2021-01-01T02:00:...|2024-12-20T10:35:...|            100607|
NO1|  171335.12|2021-01-01T01:00:...|
|           cabin|2021-01-01T03:00:...|2024-12-20T10:35:...|            100607|
NO1|  164912.02|2021-01-01T02:00:...|
|           cabin|2021-01-01T04:00:...|2024-12-20T10:35:...|            100607|
NO1|  160265.77|2021-01-01T03:00:...|
|           cabin|2021-01-01T05:00:...|2024-12-20T10:35:...|            100607|
NO1|  159828.69|2021-01-01T04:00:...|
+----------------+-------------------+-------------------+------------------+-----
----+-----------+-------------------+
only showing top 5 rows
```

In [33]:

```python
sdf = (
    sdf
    .withColumn("starttime", to_timestamp("starttime"))
    .withColumn("endtime", to_timestamp("endtime"))
    .withColumn("lastupdatedtime", to_timestamp("lastupdatedtime"))
)

sdf = sdf.drop("meteringPointCount")

# Converting all column names to lowercase
sdf = sdf.toDF(*[c.lower() for c in sdf.columns])
sdf.show(5)
```

```
+----------------+-------------------+-------------------+---------+-----------+----
---------------+
|consumptiongroup|            endtime|    lastupdatedtime|pricearea|quantitykwh|
starttime|
+----------------+-------------------+-------------------+---------+-----------+----
---------------+
|           cabin|2021-01-01 01:00:00|2024-12-20 10:35:40|      NO1|  177071.56|2021
-01-01 00:00:00|
|           cabin|2021-01-01 02:00:00|2024-12-20 10:35:40|      NO1|  171335.12|2021
-01-01 01:00:00|
|           cabin|2021-01-01 03:00:00|2024-12-20 10:35:40|      NO1|  164912.02|2021
-01-01 02:00:00|
|           cabin|2021-01-01 04:00:00|2024-12-20 10:35:40|      NO1|  160265.77|2021
-01-01 03:00:00|
|           cabin|2021-01-01 05:00:00|2024-12-20 10:35:40|      NO1|  159828.69|2021
-01-01 04:00:00|
+----------------+-------------------+-------------------+---------+-----------+----
---------------+
only showing top 5 rows
```

In [34]:
```
session.execute("""
CREATE TABLE IF NOT EXISTS consumption_per_group_hour (
    pricearea text,
    consumptiongroup text,
    starttime timestamp,
    endtime timestamp,
    quantitykwh double,
    lastupdatedtime timestamp,
    PRIMARY KEY ((pricearea, consumptiongroup), starttime)
);
""")
```

Out[34]:
```
<cassandra.cluster.ResultSet at 0x21c231b6d10>
```

In [35]:
```
sdf.write \
    .format("org.apache.spark.sql.cassandra") \
    .options(keyspace="assignment_2_keyspace", table="consumption_per_group_hour")
    .mode("append") \
    .save()
```

In [38]:
```
# Reading back from Cassandra to verify

df_cass = spark.read \
    .format("org.apache.spark.sql.cassandra") \
    .options(keyspace="assignment_2_keyspace", table="consumption_per_group_hour")
    .load()

print(f"Number of rows in consumption_per_group_hour: {df_cass.count()}")
df_cass.show(5)
```

```
Number of rows in consumption_per_group_hour: 876600
+---------+----------------+-------------------+-------------------+----------------
---+-----------+
|pricearea|consumptiongroup|          starttime|            endtime|    lastupdatedt
ime|quantitykwh|
+---------+----------------+-------------------+-------------------+----------------
---+-----------+
|      NO5|        tertiary|2021-01-01 00:00:00|2021-01-01 01:00:00|2024-12-20 10:3
5:40|  281465.94|
|      NO5|        tertiary|2021-01-01 01:00:00|2021-01-01 02:00:00|2024-12-20 10:3
5:40|  281856.97|
|      NO5|        tertiary|2021-01-01 02:00:00|2021-01-01 03:00:00|2024-12-20 10:3
5:40|  282018.25|
|      NO5|        tertiary|2021-01-01 03:00:00|2021-01-01 04:00:00|2024-12-20 10:3
5:40|  283190.28|
|      NO5|        tertiary|2021-01-01 04:00:00|2021-01-01 05:00:00|2024-12-20 10:3
5:40|   284519.1|
+---------+----------------+-------------------+-------------------+----------------
---+-----------+
only showing top 5 rows
```

In [39]:
```
# Pushing consumption data to MongoDB

extracted_df = df_cass.select(
```

```
    "pricearea",
    "consumptiongroup",
    "starttime",
    "quantitykwh"
)

# Converting to pandas DataFrame and then to records for MongoDB insertion

records = extracted_df.toPandas().to_dict("records")
```

In [41]:

```
# Getting username and password for MongoDB connection

with open("../.streamlit/secrets.toml", "rb") as f:
    cfg = tomllib.load(f)

USR = cfg["MongoDB"]["username"]
PWD = cfg["MongoDB"]["pwd"]

uri = f"mongodb+srv://esksko:{PWD}@ind320-esksko.5nbj7x0.mongodb.net/?retryWrites=t

# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))

# Selecting database and collection
db = client["IND320_assignment_4"] # Creates or connects to database
collection = db["consumption_data"] # Creates or connects to collection

# Converting spark DataFrame to pandas DataFrame, then to records for MongoDB inser
records = extracted_df.toPandas().to_dict("records")

# Inserting new records
result = collection.insert_many(records)

print(f"Inserted {len(result.inserted_ids)} records into MongoDB.")

# Verifying insertion by checking count
count = collection.count_documents({})
print(f"Total documents in collection: {count}")

# Showing sample document
print("\nSample document:")
print(collection.find_one())
```

```
Inserted 876600 records into MongoDB.
Total documents in collection: 876600

Sample document:
{'_id': ObjectId('69172b5f69f823a8b885f931'), 'pricearea': 'NO2', 'consumptiongrou
p': 'household', 'starttime': datetime.datetime(2021, 1, 1, 0, 0), 'quantitykwh': 14
25141.0}
```