Project work, part 2 - Data Sources

Orie Kimura

October 24, 2025

General

Links to your public GitHub repository and Streamlitapp for the compulsory work.

https://github.com/oriekimura123/IND320-Projectwork
https://oriekimura-ind320-projectwork.streamlit.app/

Tasks

Accounts and repositories

Both acounts and repositories are created and resed.

Local database: Cassandra

Spark-Cassandra connection works and the Cassandra database can be accessed from the Jupyter Notebook and uset to store data from the API.

Remote database: MongoDB

Tested that I can manupilate data from Python.
The MongoDB database stores data that has been trimmed/curated/prepared through the Jupyter Notebook and Spark filtering.
These data can be accessed directly from the Streamlit app.

API

the API connection at https://api.elhub.no
I am goitng to get data from https://api.elhub.no/energy-data-api#/price-areas

Tasks, Jupyter Notebook

1. Use the Elhub API to retrieve hourly production data for all price areas using PRODUCTION_PER_GROUP_MBA_HOUR for all days and hours of the year 2021.

In [1]:

```python
# --- Environment Setup ---
import os

# Paths configuration
SPARK_HOME = "C:\\Spark\\spark-3.5.1-bin-hadoop3"
HADOOP_HOME = "C:\\Hadoop\\hadoop-3.3.1"
JAVA_HOME = "C:\\Program Files\\Microsoft\\jdk-21.0.8.9-hotspot"

# Set environment variables
os.environ.update({
    "SPARK_HOME": SPARK_HOME,
    "HADOOP_HOME": HADOOP_HOME,
    "JAVA_HOME": JAVA_HOME,
    "PATH": os.environ["PATH"] + os.pathsep + os.path.join(SPARK_HOME, "bin"),
    "PYSPARK_PYTHON": "python",
    "PYSPARK_DRIVER_PYTHON": "python",
    "PYSPARK_HADOOP_VERSION": "without",
    "SPARK_CONF_DIR": os.path.join(SPARK_HOME, "conf")
})

# Database configuration
CASSANDRA_KEYSPACE = "my_keyspace"
CASSANDRA_TABLE = "production_data"
MONGO_DATABASE = "elhub_data"
MONGO_COLLECTION = "production_data_hourly"

print("Environment variables set successfully")
```

```
Environment variables set successfully
```

In [2]:

```python
# fetch data from api.elhub.no for all 5 price areas in Norway for the year 2021
import requests
import json
from dateutil.relativedelta import relativedelta
import time
from typing import List, Dict
from urllib.parse import quote
from datetime import date
from datetime import datetime, timezone, timedelta
from dateutil.relativedelta import relativedelta
import pandas as pd

# Base URL without the specific price area
BASE_URL = "https://api.elhub.no/energy-data/v0/price-areas"

# Define the dataset parameter separately
DATASET_PARAM = "dataset=PRODUCTION_PER_GROUP_MBA_HOUR"

# List of all five Norwegian price areas
PRICE_AREAS: List[str] = ["NO1", "NO2", "NO3", "NO4", "NO5"]

# Define the start and end of the full year
START_YEAR = date(2021, 1, 1)
END_YEAR = date(2021, 12, 31)

MAX_RETRIES = 3
```

```python
DELAY_SECONDS = 0.5

# --- Main Data Storage ---
all_data: List[Dict] = []
total_records_collected = 0

# --- Start Iterating Through Each Price Area ---
for area in PRICE_AREAS:
    # print(f"\n==================== Starting Fetch for AREA: {area} ==============
    current_start_date = START_YEAR

    # Loop month by month for the current area
    while current_start_date <= END_YEAR:
        # Calculate the monthly date range
        next_month_start = current_start_date + relativedelta(months=1)
        current_end_date = next_month_start - relativedelta(days=1)

        if current_end_date > END_YEAR:
            current_end_date = END_YEAR

        # Define timezone (+01:00 for Norway, UTC+1)
        TZ_OFFSET = timezone(timedelta(hours=1))

        # Convert date (which has no time) to datetime with timezone
        start_dt = datetime.combine(current_start_date, datetime.min.time(), tzinfo
        end_dt = datetime.combine(current_end_date + timedelta(days=1), datetime.mi

        # Create proper ISO-8601 timestamps and encode them
        start_date_str = quote(start_dt.isoformat(), safe="")
        end_date_str = quote(end_dt.isoformat(), safe="")

        # Construct the full API URL
        url = (
            f"{BASE_URL}/{area}?{DATASET_PARAM}"
            f"&startDate={start_date_str}&endDate={end_date_str}"
        )

        # print(f"-> Requesting data for {start_date_str} to {end_date_str} in {are

        # --- Retry Logic ---
        for attempt in range(MAX_RETRIES):
            try:
                if attempt > 0:
                    print(f"   (Attempt {attempt + 1}/{MAX_RETRIES}) Retrying in {D
                    time.sleep(DELAY_SECONDS)

                # Make the API request
                response = requests.get(url, verify=False)
                response.raise_for_status()

                # Extract and process data
                data = response.json().get('data', [])

                # Add the priceArea field to each record before appending, to ensur
                for record in data:
                    record['priceArea'] = area
```

```
            all_data.extend(data)
            total_records_collected += len(data)

            # print(f"<- Successfully retrieved {len(data)} records for {area}.
            break # Success, move to the next month

        except requests.exceptions.HTTPError as errh:
            print(f"HTTP Error for {area} ({start_date_str}): {errh}")
            if response.status_code < 500 or attempt == MAX_RETRIES - 1:
                break # Stop retrying on client errors (4xx) or max attempts
        except requests.exceptions.RequestException as err:
            print(f"General Request Error for {area} ({start_date_str}): {err}"
            if attempt == MAX_RETRIES - 1:
                break

    # Move to the next month
    current_start_date = next_month_start
```

```
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
```

```
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
```

```
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
```

```
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
```

```
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
```

```
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
C:\Users\oriek\miniconda3\envs\DVD_new\lib\site-packages\urllib3\connectionpool.py:1
097: InsecureRequestWarning: Unverified HTTPS request is being made to host 'api.elh
ub.no'. Adding certificate verification is strongly advised. See: https://urllib3.re
adthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
```

In [3]:

```python
# download the data as a JSON file to the downloads folder
import json
import os

repo_root = os.path.abspath(os.path.join(os.path.dirname(''), '..'))

DOWNLOADS_FOLDER = os.path.join(repo_root, 'downloads')
FILE_PATH = os.path.join(DOWNLOADS_FOLDER, 'Production_per_group_mad_hours_2021.jso

# Make sure the folder exists (Important!)
os.makedirs(DOWNLOADS_FOLDER, exist_ok=True)

# Write JSON to file using the ABSOLUTE path
with open(FILE_PATH, 'w') as f:
    json.dump(all_data, f, indent=4)

print(f"The file is written to: {FILE_PATH}")
```

```
The file is written to: C:\Users\oriek\IND320\IND320-Projectwork-Orie\downloads\Prod
uction_per_group_mad_hours_2021.json
```

Extract only the list in productionPerGroupMbaHour, convert to a DataFrame, and insert the data into Cassandra using Spark.

In [4]:

```python
# Extract all productionPerGroupMbaHour lists from all_data and flatten into a sing

production_data_list = []
for record in all_data:
        attributes = record.get('attributes', {})
        production_data = attributes.get('productionPerGroupMbaHour', [])
        production_data_list.extend(production_data)

# Print the first 3 items to check the structure and content
# print total number of records
# import json
```

```
# print(json.dumps(production_data_list[:3], indent=4))
# print(f"\nTotal hourly production records collected: {len(production_data_list)}"
```

In [5]:
```
# Set up Cassandra connection and create keyspace

from cassandra.cluster import Cluster
from cassandra.cluster import ConnectionException

# IMPORTANT: Ensure your Docker container is running and wait ~90 seconds
# before running this cell to allow Cassandra to fully start.
try:
    # Use 127.0.0.1 to connect to the Docker container from your host machine
    cluster = Cluster(['127.0.0.1'])
    session = cluster.connect()
    print("Connection established successfully.")

    # Create Keyspace
    keyspace_query = """
        CREATE KEYSPACE IF NOT EXISTS CASSANDRA_KEYSPACE
        WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }
    """
    session.execute(keyspace_query)
    print(f"Keyspace {CASSANDRA_KEYSPACE} confirmed.")

except ConnectionException as e:
    print(f"ERROR: Failed to connect to Cassandra. Check Docker status. Error: {e}"
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Connection established successfully.
Keyspace my_keyspace confirmed.
```

In [6]:
```
# Set environment variables for Spark, Hadoop
# Mongo DB setup
import streamlit as st
try:
    MONGO_URI = st.secrets["mongo"]["uri"]
except KeyError:
    st.error("MongoDB URI not found in Streamlit secrets. Check your .streamlit/sec
    st.stop()
```

In [7]:
```
# Set up SparkSession for Cassandra and MongoDB
from pyspark.sql import SparkSession

spark = (
    SparkSession.builder
    .appName("CassandraToMongo")
    # Cassandra config
    .config("spark.cassandra.connection.host", "localhost")
    .config("spark.cassandra.connection.port", "9042")
    .config('spark.sql.extensions', 'com.datastax.spark.connector.CassandraSparkExt
    .config('spark.sql.catalog.mycatalog', 'com.datastax.spark.connector.datasource
    .config("spark.jars.packages", "com.datastax.spark:spark-cassandra-connector_2.
    .getOrCreate()
)
```

```python
# Spark config
spark_master = spark.sparkContext.master
spark_conf = spark.sparkContext.getConf().getAll()
```

In [8]:
```python
# Create the Spark DataFrame from the list

from pyspark.sql.types import StructType, StructField, StringType, DoubleType

# Define the schema explicitly for robustness
schema = StructType([
    StructField("priceArea", StringType(), False),
    StructField("productionGroup", StringType(), False),
    StructField("startTime", StringType(), False),
    StructField("endTime", StringType(), False),
    StructField("lastUpdatedTime", StringType(), False),
    StructField("quantityKwh", DoubleType(), False)
])

# Create the Spark DataFrame from the list of dicts
df_production_data = spark.createDataFrame(production_data_list, schema)
```

In [9]:
```python
# Create a new table for production_data (first time only)
session.set_keyspace(CASSANDRA_KEYSPACE)
session.execute(f"DROP TABLE IF EXISTS {CASSANDRA_KEYSPACE}.{CASSANDRA_TABLE};")

# Create the table using the structure based on your DataFrame columns
session.execute(f"CREATE TABLE IF NOT EXISTS {CASSANDRA_KEYSPACE}.{CASSANDRA_TABLE}
                priceArea text, productiongroup text, startTime text, endTime text,
```

Out[9]:
```
<cassandra.cluster.ResultSet at 0x2812b34df30>
```

In [10]:
```python
# Verify the DataFrame schema and show a few rows
print("Spark DataFrame Schema:")
df_production_data.printSchema()
print(df_production_data.columns)
print("Show the first 5 rows to verify")
df_production_data.show(5)
```

```
Spark DataFrame Schema:
root
 |-- priceArea: string (nullable = false)
 |-- productionGroup: string (nullable = false)
 |-- startTime: string (nullable = false)
 |-- endTime: string (nullable = false)
 |-- lastUpdatedTime: string (nullable = false)
 |-- quantityKwh: double (nullable = false)

['priceArea', 'productionGroup', 'startTime', 'endTime', 'lastUpdatedTime', 'quantit
yKwh']
Show the first 5 rows to verify
+---------+---------------+--------------------+--------------------+---------------
-----+-----------+
|priceArea|productionGroup|           startTime|             endTime|     lastUpdate
dTime|quantityKwh|
+---------+---------------+--------------------+--------------------+---------------
-----+-----------+
|      NO1|          hydro|2021-01-01T00:00:...|2021-01-01T01:00:...|2024-12-20T10:3
5:...|  2507716.8|
|      NO1|          hydro|2021-01-01T01:00:...|2021-01-01T02:00:...|2024-12-20T10:3
5:...|  2494728.0|
|      NO1|          hydro|2021-01-01T02:00:...|2021-01-01T03:00:...|2024-12-20T10:3
5:...|  2486777.5|
|      NO1|          hydro|2021-01-01T03:00:...|2021-01-01T04:00:...|2024-12-20T10:3
5:...|  2461176.0|
|      NO1|          hydro|2021-01-01T04:00:...|2021-01-01T05:00:...|2024-12-20T10:3
5:...|  2466969.2|
+---------+---------------+--------------------+--------------------+---------------
-----+-----------+
only showing top 5 rows
```

In [11]:

```python
# Add a unique integer ID column named 'ind' to the DataFrame
# ind skhould be LongType and the primary key in Cassandra

from pyspark.sql.functions import monotonically_increasing_id
from pyspark.sql.types import LongType

df_final = df_production_data.withColumn(
    "ind",
    monotonically_increasing_id().cast(LongType())
)
```

In [12]:

```python
# Change column names to lowercase to match Cassandra table schema
df_to_write = df_final \
  .withColumnRenamed("priceArea","pricearea") \
  .withColumnRenamed("productionGroup","productiongroup") \
  .withColumnRenamed("startTime","starttime") \
  .withColumnRenamed("endTime","endtime") \
  .withColumnRenamed("lastUpdatedTime","lastupdatedtime") \
  .withColumnRenamed("quantityKwh","quantitykwh")
```

In [13]:

```python
# Write the DataFrame to Cassandra
df_to_write.write \
```

```
    .format("org.apache.spark.sql.cassandra") \
    .options(keyspace=CASSANDRA_KEYSPACE, table=CASSANDRA_TABLE) \
    .mode("append") \
    .save()
```

In [14]:
```
# Verify data was written to Cassandra by reading it back
# Read the data from Cassandra into a DataFrame (df_read)
df_read = spark.read \
    .format("org.apache.spark.sql.cassandra") \
    .options(table=CASSANDRA_TABLE, keyspace=CASSANDRA_KEYSPACE) \
    .load()

# Show the schema to confirm types were read correctly
print("Schema after reading from Cassandra:")
df_read.printSchema()

# Show the first 5 rows of data
print("\nFirst 5 rows read from Cassandra:")
df_read.show(5)

#  Use the .count() action
row_count = df_read.count()
print(f"Total number of rows in the table: {row_count}")
```

```
Schema after reading from Cassandra:
root
 |-- ind: long (nullable = false)
 |-- endtime: string (nullable = true)
 |-- lastupdatedtime: string (nullable = true)
 |-- pricearea: string (nullable = true)
 |-- productiongroup: string (nullable = true)
 |-- quantitykwh: double (nullable = true)
 |-- starttime: string (nullable = true)


First 5 rows read from Cassandra:
+-----------+--------------------+--------------------+---------+---------------+---
--------+-------------------+
|        ind|             endtime|     lastupdatedtime|pricearea|productiongroup|qua
ntitykwh|           starttime|
+-----------+--------------------+--------------------+---------+---------------+---
--------+-------------------+
|94489294015|2021-11-10T07:00:...|2024-10-27T01:10:...|      NO5|           wind|
0.0|2021-11-10T06:00:...|
|       9464|2021-03-07T12:00:...|2024-12-20T10:35:...|      NO1|        thermal|  2
6976.592|2021-03-07T11:00:...|
|60129552114|2021-02-06T11:00:...|2024-12-20T10:35:...|      NO4|          other|
0.053|2021-02-06T10:00:...|
|34359751627|2021-12-14T04:00:...|2024-10-27T05:17:...|      NO2|          other|
30.4|2021-12-14T03:00:...|
|34359740061|2021-09-18T11:00:...|2024-12-20T10:35:...|      NO2|          hydro|  5
791213.0|2021-09-18T10:00:...|
+-----------+--------------------+--------------------+---------+---------------+---
--------+-------------------+
only showing top 5 rows

Total number of rows in the table: 215353
```

2. Use Spark to extract the columns priceArea, productionGroup, startTime, and quantityKwh from Cassandra.

In [15]:

```
# Copy only columns pricearea, productionsgorup, sarttime and quantitykwh from df_r
df_extracted = df_read.select(
    "pricearea",
    "productiongroup",
    "starttime",
    "quantitykwh"
)

print("\nFirst 5 rows of Extracted Data:")
df_extracted.show(5)
```

```
First 5 rows of Extracted Data:
+---------+---------------+--------------------+-----------+
|pricearea|productiongroup|           starttime|quantitykwh|
+---------+---------------+--------------------+-----------+
|      NO1|        thermal|2021-10-17T15:00:...|  29912.145|
|      NO3|        thermal|2021-09-21T14:00:...|        0.0|
|      NO1|          hydro|2021-05-31T00:00:...|  2736639.2|
|      NO3|          solar|2021-08-10T04:00:...|     25.363|
|      NO3|        thermal|2021-09-09T12:00:...|        0.0|
+---------+---------------+--------------------+-----------+
only showing top 5 rows
```

3. Create the following plots:
   A pie chart for the total production of the year form a choosen price area, where each piece of the pie is one of the production groups.

In [16]:
```python
import matplotlib.pyplot as plt
from pyspark.sql.functions import col
from ipywidgets import interact, Dropdown
from IPython.display import display

# print("Starting aggregation...")

# Aggreger alle data ONCE
df_agg_all = df_read.groupBy("pricearea", "productiongroup") \
                    .sum("quantitykwh") \
                    .withColumnRenamed("sum(quantitykwh)", "Total_Kwh")

# Convert to Pandas
pdf_agg_all = df_agg_all.toPandas()

# Get the list of areas
price_area_list = pdf_agg_all['pricearea'].unique().tolist()
price_area_list.sort()

# Add 'Total' as the first choice in the list
price_area_list.insert(0, "Total")

# Get all unique production groups from dataset
production_groups = pdf_agg_all['productiongroup'].unique().tolist()
production_groups.sort() # Sort for consistency
```

In [17]:
```python
# plotting functions
# Pre-aggregate the data for all areas to speed up filtering later

COLOR_MAP = {
    "hydro": "#1f77b4",   # blue
    "wind": "#17becf",    # cyan/light blue
    "thermal": "#7f7f7f", # gray
    "solar": "#e377c2",   # pink
    "other": "#2ca02c"    # green
}
```

```python
def plot_production_pie_fast(selected_area):
    """
    Filters data and draws the pie chart with CONSISTENT colors.
    """

    # Data Filtering/Aggregation by pricearea
    if selected_area == "Total":
        pdf = pdf_agg_all.groupby('productiongroup')['Total_Kwh'].sum().reset_index
        title_area = "All areas in Norway"
    else:
        pdf = pdf_agg_all[pdf_agg_all['pricearea'] == selected_area]
        title_area = selected_area

    plt.figure(figsize=(12, 8))


    if not pdf.empty and pdf['Total_Kwh'].sum() > 0:
        # Draw the pie chart
        pie_colors = [COLOR_MAP.get(group, "#AAAAAA") for group in pdf['productiong

        wedges, texts, autotexts = plt.pie(
            pdf['Total_Kwh'],
            autopct='%1.1f%%',
            startangle=90,
            wedgeprops={'edgecolor': 'black'},
            textprops={'fontsize' : 18},
            pctdistance=0.9,
            colors=pie_colors,
        )

        # Create labels for the explanation
        labels = [f'{l}, {s/pdf["Total_Kwh"].sum():1.1%}' for l, s in zip(pdf['prod

        # Add the explanation (legend)
        plt.legend(
            wedges,
            labels,
            title="Produksjonsgruppe, Prosent",
            loc="center left",
            bbox_to_anchor=(1, 0, 0.5, 1)
        )

        plt.title(f'Total Production per Productiongroup per Pricearea: {title_area
        plt.axis('equal')
        plt.show()
    else:
        plt.close()
        display(f"No production data found for pricearea: {title_area}")
```

In [18]:

```python
%matplotlib inline
from ipywidgets import interact, Dropdown

# Get the list of unique price ranges from Pandas DataFrame (pdf_agg_all)
price_area_list = pdf_agg_all['pricearea'].unique().tolist()
price_area_list.sort()
```

```python
# Add 'Total' as the first choice in the list
price_area_list.insert(0, "Total")

# Create the Dropdown widget
area_dropdown = Dropdown(
    options=price_area_list,
    value=price_area_list[0], # Sets 'Total' as default value
    escription='Pricearea:',
)

# Use 'interact' to connect the dropdown value to the plotting function
# The pie chart will automatically update when you select a new range
interact(plot_production_pie_fast, selected_area=area_dropdown);
```

```
interactive(children=(Dropdown(description='selected_area', options=('Total', 'NO1', 'NO2', 'NO3', 'NO4', 'NO5…
```

A line plot for the first month of the year for a chooen price area. Make separete lines for each production group.

In [19]:
```python
# Select the data for the first month of 2021
from pyspark.sql.functions import col, lit, substring

df_jan_in_string = df_extracted.filter(
    (substring(col("starttime"), 1, 4) == lit("2021")) &
    (substring(col("starttime"), 6, 2) == lit("01"))
)

df_jan_in_string.show(5)
```

```
+---------+---------------+-------------------+-----------+
|pricearea|productiongroup|          starttime|quantitykwh|
+---------+---------------+-------------------+-----------+
|      NO1|           wind|2021-01-30T12:00:...|  78337.055|
|      NO2|          solar|2021-01-08T12:00:...|   1022.223|
|      NO1|          other|2021-01-26T08:00:...|        0.0|
|      NO1|           wind|2021-01-20T08:00:...|  18937.072|
|      NO5|          other|2021-01-25T03:00:...|        0.0|
+---------+---------------+-------------------+-----------+
only showing top 5 rows
```

In [20]:
```python
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

from ipywidgets import interact, Dropdown

def plot_monthly_production(selected_area):
    """
    Filters the January data by selected range and draws a line graph.
    """

    # Filter/Aggregate based on selected range ("Total" or specific pricearea)
    if selected_area == "Total":
        # Aggregate the sum of production for all areas per hour and group
        df_plot = df_jan_in_string.groupBy("starttime", "productiongroup")\
```

```python
            .sum("quantitykwh").withColumnRenamed("sum(quantitykwh)", "quantitykwh"
        title_area = "All areas in Norway"
    else:
        # Filter for the specific area
        df_plot = df_jan_in_string.filter(col("pricearea") == selected_area)
        title_area = selected_area

    # Sort the data before converting to Pandas
    df_plot = df_plot.orderBy(col("starttime"))

    # Convert the plot-ready data to Pandas
    pdf = df_plot.toPandas()

    plt.figure(figsize=(14, 7))

def plot_monthly_production(selected_area):
    """
    Filters the January data by selected range and draws a Dual Y-Axis LINE chart.
    Hydro is plotted on the left axis, and all other groups are plotted on the
    right axis to resolve the scale imbalance issue.
    """

    # Filter/Aggregate based on selected range ("Total" or specific pricearea)
    if selected_area == "Total":
        # Aggregate the sum of production for all areas per hour and group
        df_plot = df_jan_in_string.groupBy("starttime", "productiongroup")\
            .sum("quantitykwh").withColumnRenamed("sum(quantitykwh)", "quantitykwh"
        title_area = "All areas in Norway"
    else:
        # Filter for the specific area
        df_plot = df_jan_in_string.filter(col("pricearea") == selected_area)
        title_area = selected_area

    # Sort the data before converting to Pandas
    df_plot = df_plot.orderBy(col("starttime"))

    # Convert the plot-ready data to Pandas
    pdf = df_plot.toPandas()

    plt.figure(figsize=(14, 7))

    # Define the dominant group (assuming Hydro is the one crushing the scale)
    DOMINANT_GROUP = 'hydro'

    # Lists to store lines and labels for a combined legend
    lines = []
    labels = []

    if not pdf.empty:

        # Ensure 'starttime' is datetime for plotting
        if pdf['starttime'].dtype != 'datetime64[ns]':
             pdf['starttime'] = pd.to_datetime(pdf['starttime'])

        # Create the primary axis (ax1) and secondary axis (ax2)
        ax1 = plt.gca() # Primary Y-axis (for Hydro)
```

```
    ax2 = ax1.twinx() # Secondary Y-axis (for minor groups)

    # Iterate over all unique groups
    for group in pdf['productiongroup'].unique():
        group_data = pdf[pdf['productiongroup'] == group]
        current_color = COLOR_MAP.get(group, "#AAAAAA")

        if group == DOMINANT_GROUP:
            # Plot the dominant group on the primary axis (ax1)
            line, = ax1.plot(
                group_data['starttime'],
                group_data['quantitykwh'],
                label=group,
                color=current_color,
                linewidth=2.0
            )
            lines.append(line)
            labels.append(group)
        else:
            # Plot all minor groups on the secondary axis (ax2)
            line, = ax2.plot(
                group_data['starttime'],
                group_data['quantitykwh'],
                label=group,
                color=current_color,
                linewidth=1.0,
                linestyle='--' # Use a dashed line for minor groups for visual
            )
            lines.append(line)
            labels.append(group)

    # --- X-AXIS AND TITLE FORMATTING (Fix for clutter) ---

    # Format the X-axis for the primary axis (ax1)
    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%d'))  # SHOW ONLY DAY
    ax1.xaxis.set_major_locator(mdates.DayLocator(interval=5))
    ax1.xaxis.set_minor_locator(mdates.DayLocator(interval=1))

    # Set colors and labels for both Y-axes
    ax1.set_ylabel(f'{DOMINANT_GROUP} Production (kWh)', color=COLOR_MAP.get(DO
    ax1.tick_params(axis='y', labelcolor=COLOR_MAP.get(DOMINANT_GROUP, "#000000

    ax2.set_ylabel('Minor Group Production (kWh)', color='gray')
    ax2.tick_params(axis='y', labelcolor='gray')

    # Set common axis properties
    ax1.set_title(f'Production per Group in January 2021 - Area: {title_area}',
    ax1.set_xlabel("Day of Month") # Reverted label back to just Day
    ax1.tick_params(axis='x', rotation=45) # Reduced rotation slightly

    # Add a combined legend using lines and labels collected from both axes
    ax1.legend(lines, labels, title="Productionsgroup", bbox_to_anchor=(1.05, 1

    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
```

```
    else:
        plt.close()
        display(f"No production data found for Area: {title_area} in January 2021."
```

In [21]:
```
# Create the Dropdown widget (using the existing list including "Total")
area_dropdown = Dropdown(
    options=price_area_list,
    value=price_area_list[0],
    description='PriceArea:',
)

# Connect the dropdown to the plotting function
interact(plot_monthly_production, selected_area=area_dropdown);
```

```
interactive(children=(Dropdown(description='PriceArea:', options=('Total', 'NO1', 'N
O2', 'NO3', 'NO4', 'NO5'),…
```

4. Insert the Spark-extracted data into your MongoDB

In [22]:
```
# insert the data into MongoDB
df_extracted.write \
    .format("mongodb") \
    .mode("overwrite") \
    .option("database", MONGO_DATABASE) \
    .option("collection", MONGO_COLLECTION) \
    .save()
```

In [23]:
```
# Verify data was written to MongoDB by reading it back
try:
    df_mongo_data = (
        spark.read
        .format("mongodb")
        # specify the database and collection here, as the connection URI is set gl
        .option("database", MONGO_DATABASE)
        .option("collection", MONGO_COLLECTION)
        .load()
    )

    # print("\n Successfully loaded DataFrame from MongoDB Atlas.")

    #print("\n--- DataFrame Schema (Inferred from MongoDB) ---")
    # df_mongo_data.printSchema()

    print("\n--- First 5 Rows of Data ---")
    df_mongo_data.show(5)

except Exception as e:
    print(f"\n An error occurred during the read operation.")
    print(f"Please double-check your database/collection names and network access:

finally:
    # Stop the Spark session when finished
    spark.stop()
```

```
--- First 5 Rows of Data ---
+--------------------+---------+---------------+----------+--------------------+
|                 _id|pricearea|productiongroup|quantitykwh|           starttime|
+--------------------+---------+---------------+----------+--------------------+
|68f7e725d80586d43...|      NO5|          hydro| 1387041.0|2021-09-19T08:00:...|
|68f7e725d80586d43...|      NO3|          other|     2.187|2021-11-10T16:00:...|
|68f7e725d80586d43...|      NO5|           wind|       0.0|2021-10-18T14:00:...|
|68f7e725d80586d43...|      NO4|          solar|       0.0|2021-12-29T17:00:...|
|68f7e725d80586d43...|      NO4|          solar|     0.493|2021-03-19T15:00:...|
+--------------------+---------+---------------+----------+--------------------+
only showing top 5 rows
```

Tasks, Streamlit app

Create a Streamlit app including:

- requirements.txt (for package dependencies)
- Four pages (with dummy headers and test contents for now)
- The front/home page should have a sidebar menu with navigation options to the other pages.
  -- On the second page: A table showing the imported data (see below). Use the row-wise LineChartColumn() to display the first month of the data series. There should be one row in the table for each column of the imported data. -- On the third page: A plot of the imported data (see below), including header, axis titles and other relevant formatting.
  A drop-down menu (st.selectbox) choosing any single column in the CSV or all columns together. A selection slider (st.select_slider) to select a subset of the months. Defaults should be the first month.

Data should be read from a local CSV-file (open-meteo-subset.csv, available in the Files here in Canvas), using caching for app speed.

My Streamlit app is https://oriekimura-ind320-projectwork-part.streamlit.app/

Log Describing the Compulsory Work

Reorganizing the Working Folder/File Structure

I realized that my folder and file structure for the project was disorganized and inefficient, so I decided to completely reorganize it. The initial chaos resulted from my limited understanding of how to structure a project that integrates multiple tools effectively.

I reorganized both the folder structure and the internal code organization within the Jupyter Notebook (.ipynb) and Python (.py) files. Before starting Project 2, I verified that all files functioned correctly across my local PC, GitHub, and Streamlit.io.

I did not delete my initial GitHub repository, but from now on, all further project work will be developed within the new structure.

Old GitHub link and Streamlit link
https://github.com/oriekimura123/IND320-Projectwork-part1
https://oriekimura-ind320-projectwork-part1.streamlit.app/

New GitHub link and Streamlit link
https://github.com/oriekimura123/IND320-Projectwork
https://oriekimura-ind320-projectwork.streamlit.app/

Coding in Jupyter Notebook

The required tasks were copied and pasted in Markdown format into the notebook. I then wrote and tested the code step by step, using assistance from AI tools.

The line chart has two Dual Y-Axis because Gemini recommends me a readable line chart using dual Y-acsis.

I faced significant challenges in several areas:

- Finding compatible versions of Cassandra and Spark
- Creating a functional Spark–Cassandra–MongoDB connector
- Writing data to Cassandra

Spark–Cassandra–MongoDB Connector

When I first created the Spark–Cassandra connector, Spark implicitly used a Docker-related master for Cassandra and did not account for MongoDB also requiring a master connection. This caused several issues that I later resolved to make the combined connector function properly.

I placed the following .jar files into C:\Spark\spark-3.5.1-bin-hadoop3\jars:

- bson-5.6.1.jar
- mongodb-driver-core-5.6.1.jar
- mongodb-driver-sync-5.6.1.jar
- mongo-spark-connector_2.12-10.5.0.jar

I then forced Spark to use local[*] by adding these lines to C:\Spark\spark-3.5.1-bin-hadoop3\conf\spark-defaults.conf:

```
spark.master local[*]
spark.mongodb.connection.uri mongodb+srv://<user_name>:@/?
```

spark.mongodb.read.connection.uri mongodb+srv://<user_name>:@/?
spark.mongodb.write.connection.uri mongodb+srv://<user_name>:@/?

I also configured the environment variable for the Spark configuration directory:

In [24]:
```
os.environ["SPARK_CONF_DIR"] = r"C:\Spark\spark-3.5.1-bin-hadoop3\conf"
```

Then, I set up the environment as follows:

In [25]:
```
# --- Environment Setup ---
import os

SPARK_HOME = "C:\\Spark\\spark-3.5.1-bin-hadoop3"
HADOOP_HOME = "C:\\Hadoop\\hadoop-3.3.1"
JAVA_HOME = "C:\\Program Files\\Microsoft\\jdk-21.0.8.9-hotspot"

os.environ.update({
    "SPARK_HOME": SPARK_HOME,
    "HADOOP_HOME": HADOOP_HOME,
    "JAVA_HOME": JAVA_HOME,
    "PATH": os.environ["PATH"] + os.pathsep + os.path.join(SPARK_HOME, "bin"),
    "PYSPARK_PYTHON": "python",
    "PYSPARK_DRIVER_PYTHON": "python",
    "PYSPARK_HADOOP_VERSION": "without",
    "SPARK_CONF_DIR": os.path.join(SPARK_HOME, "conf")
})
```

Coding the Streamlit App

After completing the core tasks, I ran the application locally. It was challenging to implement filtering logic that accurately reflected user selections across both the pie chart (by pricearea) and the line chart (by productiongroup and month).

Commit and Push to GitHub / Streamlit Cloud Updates

I had to adjust the code to correctly handle both absolute and relative paths to ensure consistent behavior between my local environment and Streamlit Cloud.

Brief Description of AI Usage

I used AI tools to generate initial versions of the code, to refine and customize it, and to debug errors.

Coding, Improvement, and Tuning

I described my requirements, and AI tools suggested initial drafts. While some of the generated code worked well, others required significant modification.

Debugging

I used AI tools frequently for debugging, particularly for issues related to the Spark-Cassandra-MongoDB connector and writing data to Cassandra.

I noticed a key limitation: the error message and the code itself were often insufficient for the AI to solve the problem, leading to failed suggestions.

For example, when I tried to write data to Cassandra, both ChatGPT and Gemini provided many suggestions based on the error code, but none of them fixed the issue. The actual problem was case sensitivity in Cassandra, a detail neither I nor the AI tools initially considered. The AI tools failed to suggest checking case sensitivity and continued offering irrelevant code changes.

Spark Connection:

As mentioned, I had to take numerous steps to get the Spark-Cassandra-MongoDB connector functioning correctly. Both ChatGPT and Gemini provided theoretically correct setup methods for both connectors simultaneously, stating "it should work." However, they failed to account for the specific execution context.

It took several days to begin checking where the .jar files were stored and how the choice of .jar files folder affected execution. It took another two days to determine the correct location and, finally, one more day to discover how to properly utilize them (by creating a configuration file and referencing it in the Spark session builder). The error codes in VS Code did not provide possible causes for the underlying context issue.

Ultimately, I realized that to use AI tools efficiently, I need to have a deeper, system-level understanding of how the entire code stack operates. It is difficult to know in advance what level of foundational knowledge I must possess and where the limitations of the AI tools lie.