B_b_2 : Obtenir les données - Demande quotidienne d'électricité HQ

Source : Historique de la demande d'électricité au Québec

Hydro-Québec

https://www.hydroquebec.com/documents-donnees/donnees-ouvertes/demande-electricite-quebec/

Site consulté le : 2023-11-13

Description

Ensemble de données d'Hydro-Québec sur la demande d'électricité au Québec en mégawatts. Mises à jour toutes les 15 minutes, elles montrent les variations du besoin quotidien d'électricité en fonction de l'heure de la journée.

Fichiers ou services

Demande d'électricité au Québec (fichier JSON)

L'ouverture du lien nous amène à une page présentant un fichier `json`.


```
{
  "dateStart" : "2023-11-12T00:00:00",
  "dateEnd" : "2023-11-14T00:00:00",
  "recentHour" : "2023-11-13T18:30:00",
  "indexDonneePlusRecent" : 170,
  "nbDateAvecData" : 171,
  "details" : [ {
    "date" : "2023-11-12T00:00:00",
    "valeurs" : {
      "demandeTotal" : 22150.0
    }
  }, {
    "date" : "2023-11-12T00:15:00",
    "valeurs" : {
      "demandeTotal" : 21938.0
    }
  }, {
    "date" : "2023-11-12T00:30:00",
    "valeurs" : {
      "demandeTotal" : 21942.0
    }
```

```
  }, {
    "date" : "2023-11-12T00:45:00",
    "valeurs" : {
      "demandeTotal" : 21883.0
    }
  }, {
    "date" : "2023-11-12T01:00:00",
    "valeurs" : {
      "demandeTotal" : 21653.0
    }
  }, {

  ...

 {
    "date" : "2023-11-13T17:45:00",
    "valeurs" : {
      "demandeTotal" : 28019.0
    }
  }, {
    "date" : "2023-11-13T18:00:00",
    "valeurs" : {
      "demandeTotal" : 28025.0
    }
  }, {
    "date" : "2023-11-13T18:15:00",
    "valeurs" : {
      "demandeTotal" : 27593.0
    }
  }, {
    "date" : "2023-11-13T18:30:00",
    "valeurs" : {
      "demandeTotal" : 27714.0
    }
  }, {
    "date" : "2023-11-13T18:45:00",
    "valeurs" : { }
  }, {
    "date" : "2023-11-13T19:00:00",
    "valeurs" : { }
  }, {
    "date" : "2023-11-13T19:15:00",
    "valeurs" : { }
  }, {
    "date" : "2023-11-13T19:30:00",
    "valeurs" : { }
  }, {
```


Lors de la consultation à 18h52, nous pouvons voir que les données sont disponibles quasiment en temps réel, soit jusqu'à 18h30.

La journée d'hier est complètement présente.

Objectifs

Lire les données à tous les jours et les enregistrer pour utilisation future.

L'objectif serait, en 2024, d'avoir une série continue de données, à partir de l'importation historique de 2023, en plus des données quotidiennes qui seront lues à tous les jours.

In [1]:
```
# %load_ext jupyter_black

import black
import jupyter_black

jupyter_black.load(
    lab=True,
    line_length=55,
    target_version=black.TargetVersion.PY311,
)
```

In [2]:
```
import os
import pandas as pd
import json
import httpx
from datetime import date
import locale

locale.setlocale(locale.LC_ALL, "fr_CA")

import matplotlib.pyplot as plt

plt.style.use("fivethirtyeight")
col_HQ = "#FF9B00"
```

Import des données en dataframe

Nous transformons les données sous `json` vers un dataframe.

In [3]:
```
url = "https://www.hydroquebec.com/data/documents-donnees/donnees-ouvertes/j
r = httpx.get(url)
data = json.loads(r.text)
```

In [4]:
```
df = (
    pd.DataFrame.from_records(data["details"])
    .rename(columns={"valeurs": "MW"})
    .set_index("date")
)
df.index = pd.to_datetime(df.index)
df["MW"] = df["MW"].apply(
    lambda x: x.get("demandeTotal")
)
df = df.dropna()
```

In [5]:
```
df
```

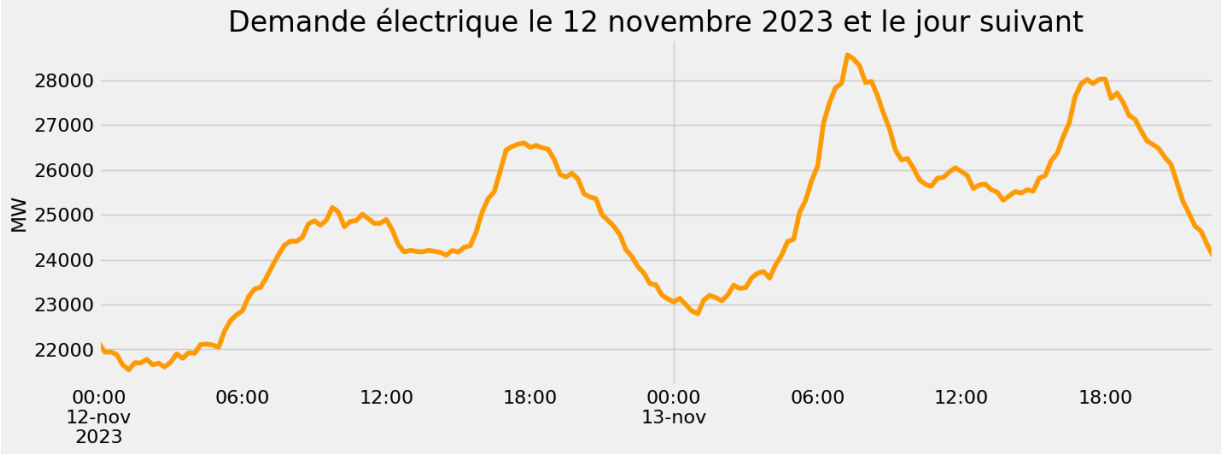
Out[5]:

date	MW
2023-11-12 00:00:00	22150.0
2023-11-12 00:15:00	21938.0
2023-11-12 00:30:00	21942.0
2023-11-12 00:45:00	21883.0
2023-11-12 01:00:00	21653.0
...	...
2023-11-13 21:30:00	25033.0
2023-11-13 21:45:00	24749.0
2023-11-13 22:00:00	24637.0
2023-11-13 22:15:00	24345.0
2023-11-13 22:30:00	24082.0

187 rows × 1 columns

Visualisation des données importées

In [6]:
```
df.plot(
    style="-",
    ms=1.5,
    fontsize=16,
    figsize=(15, 5),
    xlabel="",
    ylabel="MW",
    legend=False,
    color=col_HQ,
)
plt.title(
    f"Demande électrique le {str(df.index.date[0].strftime('%d %B %Y'))} et
    fontsize=24,
)
plt.show()
```



Regrouper par heure

Nous n'avons pas besoin d'autant de précision (15 minutes) pour la suite et nous voulons uniformiser avec les données historiques qui sont horaires. Nous effectuons donc une moyenne par heure des données recueillies.

In [7]:
```
df["date_"] = df.index.date
df["heure_"] = df.index.hour
df["dh"] = (
    df["date_"].astype(str)
    + " "
    + df["heure_"].astype(str)
    + ":00:00"
)
df
```

Out[7]:

date	MW	date_	heure_	dh
2023-11-12 00:00:00	22150.0	2023-11-12	0	2023-11-12 0:00:00
2023-11-12 00:15:00	21938.0	2023-11-12	0	2023-11-12 0:00:00
2023-11-12 00:30:00	21942.0	2023-11-12	0	2023-11-12 0:00:00
2023-11-12 00:45:00	21883.0	2023-11-12	0	2023-11-12 0:00:00
2023-11-12 01:00:00	21653.0	2023-11-12	1	2023-11-12 1:00:00
...	...	...	...	...
2023-11-13 21:30:00	25033.0	2023-11-13	21	2023-11-13 21:00:00
2023-11-13 21:45:00	24749.0	2023-11-13	21	2023-11-13 21:00:00
2023-11-13 22:00:00	24637.0	2023-11-13	22	2023-11-13 22:00:00
2023-11-13 22:15:00	24345.0	2023-11-13	22	2023-11-13 22:00:00
2023-11-13 22:30:00	24082.0	2023-11-13	22	2023-11-13 22:00:00

187 rows × 4 columns

Après avoir fait généré les dates heures sur une même colonne, nous pouvons regroupper par ce champs.

Nous utilisons la moyenne sur les 4 lectures horaires. Nous laissons aussi tomber la denière mesure, qui peut être incomplète.

In [8]:
```
df_grouped = pd.DataFrame(df.groupby(["dh"]).MW.mean())
df_grouped.index = pd.to_datetime(df_grouped.index)
df_grouped.sort_index(inplace=True)
df_grouped = df_grouped.iloc[:-1]
df_grouped.index.names = ["date"]

df_grouped
```

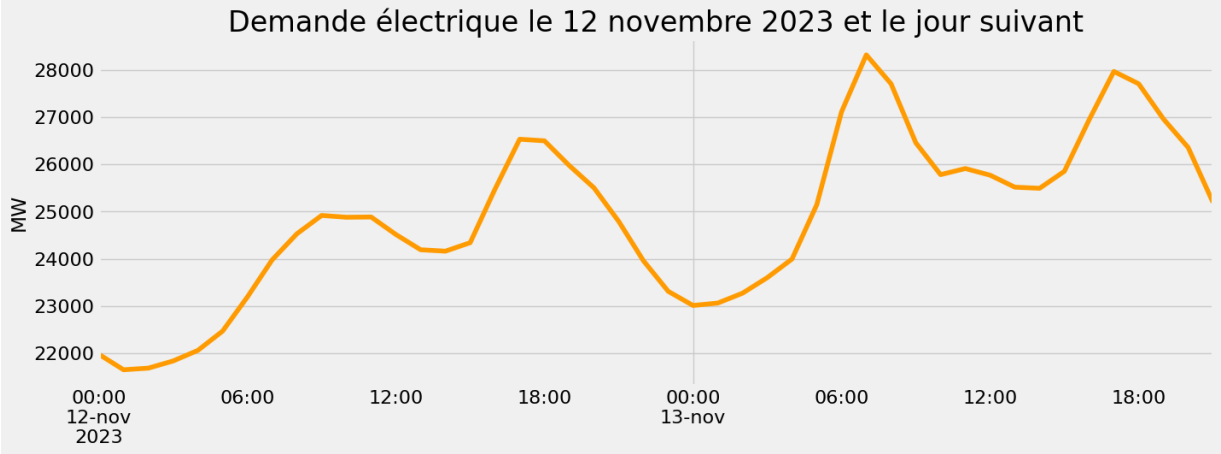
Out[8]:

date	MW
2023-11-12 00:00:00	21978.25
2023-11-12 01:00:00	21648.75
2023-11-12 02:00:00	21683.00
2023-11-12 03:00:00	21834.50
2023-11-12 04:00:00	22058.50
2023-11-12 05:00:00	22467.50
2023-11-12 06:00:00	23189.25
2023-11-12 07:00:00	23976.00
2023-11-12 08:00:00	24529.50
2023-11-12 09:00:00	24918.75
2023-11-12 10:00:00	24880.75
2023-11-12 11:00:00	24886.00
2023-11-12 12:00:00	24515.25
2023-11-12 13:00:00	24191.75
2023-11-12 14:00:00	24162.00
2023-11-12 15:00:00	24341.50
2023-11-12 16:00:00	25473.50
2023-11-12 17:00:00	26534.25
2023-11-12 18:00:00	26500.50
2023-11-12 19:00:00	25978.00
2023-11-12 20:00:00	25504.50
2023-11-12 21:00:00	24792.75
2023-11-12 22:00:00	23957.50
2023-11-12 23:00:00	23310.50
2023-11-13 00:00:00	23011.50
2023-11-13 01:00:00	23061.75
2023-11-13 02:00:00	23271.25
2023-11-13 03:00:00	23601.00
2023-11-13 04:00:00	23995.50
2023-11-13 05:00:00	25147.00
2023-11-13 06:00:00	27115.00

	MW
date	
2023-11-13 07:00:00	28322.00
2023-11-13 08:00:00	27709.00
2023-11-13 09:00:00	26458.50
2023-11-13 10:00:00	25783.75
2023-11-13 11:00:00	25912.50
2023-11-13 12:00:00	25772.50
2023-11-13 13:00:00	25518.00
2023-11-13 14:00:00	25494.50
2023-11-13 15:00:00	25854.25
2023-11-13 16:00:00	26946.25
2023-11-13 17:00:00	27970.00
2023-11-13 18:00:00	27709.50
2023-11-13 19:00:00	26964.25
2023-11-13 20:00:00	26358.75
2023-11-13 21:00:00	25198.25

Vérifions si la visualisation fait du sens après ces opérations.

In [9]:
```
df_grouped.plot(
    ms=1.5,
    fontsize=16,
    figsize=(15, 5),
    xlabel="",
    ylabel="MW",
    legend=False,
    color=col_HQ,
)
plt.title(
    f"Demande électrique le {str(df.index.date[0].strftime('%d %B %Y'))} et
    fontsize=24,
)
plt.show()
```



Nous pouvons voir que la courbe est plus lisse, mais toujours semblable.

Enregistrer les données historiques dans le fichier *interim*

Nous pouvons conserver ces données dans un fichier parquet. Ce dernier sera mis à jour de façon quotidienne.

In [10]:
```
path_to_interim_data = "../data/interim/"
file = "quotidien_demande_HQ.parquet"

df_grouped.to_parquet(
    path=os.path.join(path_to_interim_data, file),
    engine="pyarrow",
)
```

Création d'une fonction

Maintenant que nous avons effectué l'exercice d'importer une fois les données, nous voulons :

- lire le fichier enregistré ;
- lire les dernières données à jour sur le site d'HQ ;
- les concaténer ;
- enregistrer le fichier à nouveau ;
- enregistrer le graphique.

In [11]:
```
def update_demande_quotidienne_HQ():
    url = "https://www.hydroquebec.com/data/documents-donnees/donnees-ouvert
    path_to_interim_data = "../data/interim/"
    path_to_img = "../reports/figures/"
    file_parquet = "quotidien_demande_HQ.parquet"
    file_image = "quotidient_demande_HQ.png"

    # Obtenir les données sur le site HQ
```

```
    r = httpx.get(url)
    data = json.loads(r.text)

    df = (
        pd.DataFrame.from_records(data["details"])
        .rename(columns={"valeurs": "MW"})
        .set_index("date")
    )
    df.index = pd.to_datetime(df.index)
    df["MW"] = df["MW"].apply(
        lambda x: x.get("demandeTotal")
    )
    df = df.dropna()

    # Enregistrer le graphique
    df.plot(
        style="-",
        ms=1.5,
        fontsize=16,
        figsize=(15, 5),
        xlabel="",
        ylabel="MW",
        legend=False,
        color=col_HQ,
    )

    plt.title(
        f"Demande électrique le {str(df.index.date[0].strftime('%d %B %Y'))}
        fontsize=24,
    )
    plt.tight_layout()
    plt.savefig(
        os.path.join(path_to_img, file_image), dpi=300
    )

    # Regrouper par heure
    df["date_"] = df.index.date
    df["heure_"] = df.index.hour
    df["dh"] = (
        df["date_"].astype(str)
        + " "
        + df["heure_"].astype(str)
        + ":00:00"
    )
    df_grouped = pd.DataFrame(
        df.groupby(["dh"]).MW.mean()
    )
    df_grouped.index = pd.to_datetime(df_grouped.index)
    df_grouped.sort_index(inplace=True)
    df_grouped = df_grouped.iloc[:-1]
    df_grouped.index.names = ["date"]

    # Lire les données existantes et combiner avec les nouvelles données
    nouveau_df = pd.read_parquet(
        path=os.path.join(
            path_to_interim_data, file_parquet
```

```
        ),
        engine="pyarrow",
    ).combine_first(df_grouped)

    #  Enregistrer le nouveau fichier par dessus l'existant
    nouveau_df.to_parquet(
        path=os.path.join(
            path_to_interim_data, file_parquet
        ),
        engine="pyarrow",
    )

    print(
        f"Le fichier de données quotidiennes possède maintenant {nouveau_df.
    )

    return nouveau_df
```

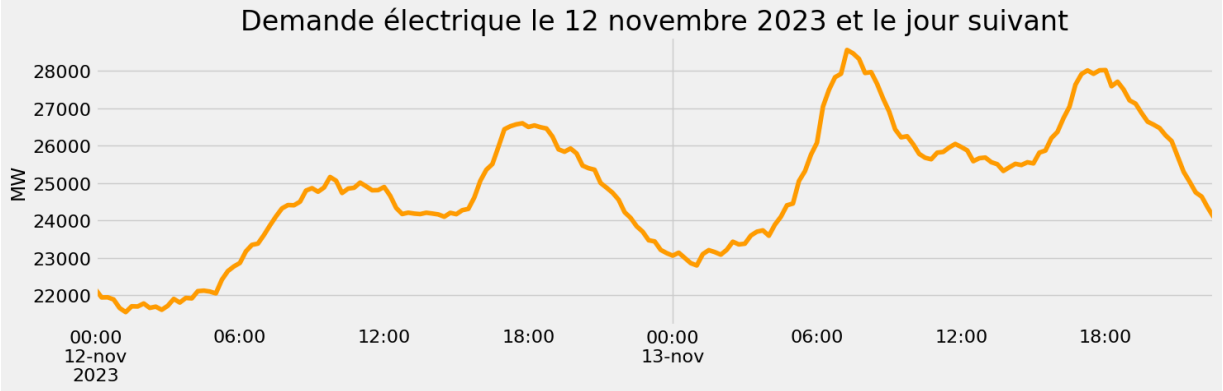
In [12]:
```
update_demande_quotidienne_HQ()
```

```
Le fichier de données quotidiennes possède maintenant 46 heures.
```

Out[12]:

date	MW
2023-11-12 00:00:00	21978.25
2023-11-12 01:00:00	21648.75
2023-11-12 02:00:00	21683.00
2023-11-12 03:00:00	21834.50
2023-11-12 04:00:00	22058.50
2023-11-12 05:00:00	22467.50
2023-11-12 06:00:00	23189.25
2023-11-12 07:00:00	23976.00
2023-11-12 08:00:00	24529.50
2023-11-12 09:00:00	24918.75
2023-11-12 10:00:00	24880.75
2023-11-12 11:00:00	24886.00
2023-11-12 12:00:00	24515.25
2023-11-12 13:00:00	24191.75
2023-11-12 14:00:00	24162.00
2023-11-12 15:00:00	24341.50
2023-11-12 16:00:00	25473.50
2023-11-12 17:00:00	26534.25
2023-11-12 18:00:00	26500.50
2023-11-12 19:00:00	25978.00
2023-11-12 20:00:00	25504.50
2023-11-12 21:00:00	24792.75
2023-11-12 22:00:00	23957.50
2023-11-12 23:00:00	23310.50
2023-11-13 00:00:00	23011.50
2023-11-13 01:00:00	23061.75
2023-11-13 02:00:00	23271.25
2023-11-13 03:00:00	23601.00
2023-11-13 04:00:00	23995.50
2023-11-13 05:00:00	25147.00
2023-11-13 06:00:00	27115.00

	MW
date	
2023-11-13 07:00:00	28322.00
2023-11-13 08:00:00	27709.00
2023-11-13 09:00:00	26458.50
2023-11-13 10:00:00	25783.75
2023-11-13 11:00:00	25912.50
2023-11-13 12:00:00	25772.50
2023-11-13 13:00:00	25518.00
2023-11-13 14:00:00	25494.50
2023-11-13 15:00:00	25854.25
2023-11-13 16:00:00	26946.25
2023-11-13 17:00:00	27970.00
2023-11-13 18:00:00	27709.50
2023-11-13 19:00:00	26964.25
2023-11-13 20:00:00	26358.75
2023-11-13 21:00:00	25198.25



Nous pouvons maintenant insérer cette fonction dans notre fichier d'import dans le but d'automatiser le tout.

Création d'un github action

Comme nous ne voulons pas faire la mise à jour manuellement tous les jours, je créé une action sur github actions avec ce *workflow* situé à `.github/workflows/update_quotidien_hq.yml` :

```
name: run update_quotidien_demande_hq
```

```yaml
on:
  schedule:
    - cron: "30 1 * * *" # “At 01:30.”

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: checkout repo content
        uses: actions/checkout@v2 # checkout the repository content

      - name: setup python
        uses: actions/setup-python@v4
        with:
          python-version: "3.11" # install the python version
needed

      - name: install python packages
        run: |
          python -m pip install --upgrade pip
          pip install -r .github/workflows/requirements_gh.txt

      - name: set locale
        run: |
          sudo apt-get update && sudo apt-get install tzdata
locales -y && sudo locale-gen fr_CA.UTF-8
          sudo localectl set-locale LANG="fr_CA.UTF-8"
          export LANG="fr_CA.UTF-8"
          sudo update-locale
          locale -a
          locale
          locale -c -k LC_NUMERIC
          localectl status

      - name: execute python script # run main.py
        env:
          SOME_SECRET: ${{ secrets.SOME_SECRET }}
        run: python src/data/quotidien_demande_hq.py

      - name: commit files
        run: |
          git config --local user.email "github-
actions[bot]@users.noreply.github.com"
          git config --local user.name "github-actions[bot]"
          git diff-index --quiet HEAD || (git commit -a -m "maj
automatique - demande quotidienne électricité" --allow-empty)

      - name: push changes
        uses: ad-m/github-push-action@master
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          branch: main
```

La mise à jour se fera automatiquement à tous les jours à 1h30 du matin.

Résumé

Dans ce notebook, nous avons :

- importé les données des données quotidiennes de la demande rendues disponibles par HQ ;
- effectué une visualisation des données importées ;
- regroupé les données de façon horaire (plutôt qu'au 15 minutes) ;
- concaténé toutes les données avec les données déjà importées précédemment ;
- sauvegardé ces données intérimaires en un format parquet, que nous allons réutiliser subséquemment ;
- créé un pipeline de données dans le dossier `src/data`, afin d'automatiser toute cette partie ;
- créé un pipeline automatique quotidien avec github action