Dataset & Instructions

The dataset provided for this data challenge lists the elements of local direct taxation in France at the municipal level from 1982 to 2022. It details all the local direct taxation data by tax and by beneficiary community (municipality, unions and assimilated, intercommunity, department, region). This data was collected by the Directorate General of Public Finances of France and includes in particular the following main local taxes:

- La taxe foncière sur les propriétés non bâties (TFPNB)
- La taxe foncière sur les propriétés bâties (TFPB)
- La taxe d'habitation (TH)
- La cotisation foncière des entreprises (CFE)
- La cotisation sur la valeur ajoutée des entreprises (CVAE)
- La taxe spéciale d'équipement au profit de la région d'Île-de-France et d'établissements publics (TSE)
- La taxe d'enlèvement des ordures ménagères (TEOM)
- Les impositions forfaitaires sur les entreprises de réseaux (IFER)
- La taxe sur les surfaces commerciales (Tascom)
- La taxe pour la gestion des milieux aquatiques et la prévention des inondations (Gemapi)
- La taxe additionnelle spéciale annuelle instituée au profit de la région Île-deFrance (Tasarif)

As the dataset is particularly heavy, we decided to make it accessible via SQL queries. You'll find all the necessary instructions here (it's a GitHub repo).

For more details you can find a description of each field here (it will download XLS automatically once you click).

Parquet File Format

Parquet is a modern file format used for storing large amounts of data. It organizes data in columns instead of rows, which makes it very efficient for analytics tasks. This format is especially popular in big data environments like Hadoop and Spark because it helps in quickly processing and analyzing data.

Originally a 3.6GB CSV file, this dataset has been converted into a 0.6GB Parquet file. Parquet's columnar storage format allows faster data access, reduced file size, and improved query performance. This makes it particularly suitable for handling complex queries and large datasets, providing participants with a powerful tool for their analysis.

Comparison with CSV

CSV (Comma-Separated Values) is a straightforward and widely-used format where data is stored row by row. While CSV files are easy to create and read, they aren't very efficient for large-scale data analysis, especially when dealing with complex queries or big datasets. When conducting a query that only uses a few columns, the entire dataset still needs to be read off the disk to get those column values. Parquet, on the other hand, is designed to overcome these limitations with its columnar storage approach. When conducting an analytics query, only the data that needs to be read off disk is pulled, making the query cheaper and faster.

Benefits of Parquet over CSV

1. **Columnar Storage**: Faster and more efficient data access, especially for specific columns.
2. **Compression**: Reduced file size, saving storage space and speeding up data processing.
3. **Schema Evolution**: Easily handle changes to the data structure over time.
4. **Data Types**: Supports a wide range of data types, including complex and nested data.
5. **Performance**: Optimized for analytical queries, resulting in quicker data retrieval and processing.

How we are using parquet:

Our original dataset was a 3.6GB CSV file. After converting it to parquet, it is only 0.6GB, making it cheaper to store and faster to query. In addition, queries that only use a few columns can leverage parquet to only lift required data off of disk instead of the entire dataset.

DuckDB

To facilitate the querying process, we highly recommend using DuckDB for interacting with the dataset. DuckDB is a high-performance, in-memory database engine optimized for SQL querying on local machines. We have set things up for participants to use DuckDB at no cost and with minimal friction. This is not just a tool, but an exciting opportunity to learn a new, efficient way of querying data for an ML model. This combination of Parquet and DuckDB ensures that participants can quickly and effectively analyze the data, gaining deeper insights into the fiscal dynamics of French municipalities.

DuckDB is a high-performance, in-memory database engine designed for SQL querying on a single machine without needing additional infrastructure. It allows users to run complex analytical queries efficiently on their local systems, making it perfect for data scientists and analysts. DuckDB pairs exceptionally well with Parquet files, as its engine is optimized to take full advantage of columnar data, resulting in faster and more efficient querying.

Features of DuckDB

- **Ease of Use**: Simple setup and integration with various programming languages.
- **In-Memory Processing:** Runs queries in-memory for fast, interactive data analysis.
- **High Performance:** Optimized for analytical queries and can handle large datasets efficiently.
- **Parquet Compatibility**: Designed to work seamlessly with Parquet files, leveraging columnar storage for speed.
- **SQL Support:** Allows SQL querying without the need for setting up a traditional database server.

How we are using duckDB:

DuckDB is our recommended engine for interacting with the dataset. SQL can be used to grab a subset of the data, which can be further processed in SQL or a dataframe with pandas.

Making an account

Go to the following link and sign-up.

https://thecrossbow-1717040468242.auth.us-east-1.amazoncognito.com/signup?client_id=6elhn5n4tt1p4dfa41ulljmbp3&response_type=code&scope=openid+profile&redirect_uri=https%3A%2F%2Foceandatapublicbucket.s3.amazonaws.com%2Fsuccess.html

You will need to enter a username, email, and password, and then set-up MFA with an authenticator app.

Explanation of the datasharing library

Git clone https://github.com/ChristianCasazza/datasharing
Then, follow the instructions in the README, or use the google collab
The datasharing library is set-up to provide query access to the francetax parquet file we are using for the competition.

1. You need to set up an account here first. You need to enter a ***username, email, password, and MFA authenticator app****.* You can use any authenticator app you want, such as Google Authenticator or Duo mobile.

2. After creating your account, you need to git clone this repo or use this google collab.

3. Follow the instructions in the README to set up your venv.

4. In the notebooks folder is an .ipynb file called data.ipynb. You can use this folder to query and work with the data to build your report. If working locally, you need to set your username and password in the env file. If using collab, set them in the collab secrets. The name of the variables should be OCEAN_USERNAME and OCEAN_PASSWORD

5. In the data.ipynb, you must first run the code to set your imports. Then, you need to set up your client. Run the code block [client = DataSharingClient()]. When you run this code block, you will be prompted to enter the MFA code from your authenticator app. After successfully signing in, you will be set-up with a duckdb instance with AWS credentials that provide permissions to query the francetax dataset.
    a. When using duckDB, it will be default be an in-memory db, meaning all work will be erased when creating a new section. By passing a path to a duckdb table, you can persist your work to disk storage. This could be useful if you create multiple queries saved as tables and don't want to re-run the query if you work over several days. **Note:** If your path is an existing duckdb database, the view on top of the France Tax you have created

    b. By default, it sets the default region to query as eu-west-3(Paris). If you are closer to the us-east-1 region, then set your duckDB region to us-east-1 by passing the duckdb_region parameter like client = DataSharingClient(duckdb_region='us-east-1'). Your two options for this data challenge are ***us-east-1 and eu-west-3***

6. After logging in, you can use the create_view method to create a view on top of an S3 URI. The dataset URI for this competition is included in the data.ipynb file. Creating a view ***does not*** load any data. Instead, it establishes a pointer on the dataset, which can then be queried with SQL

7. After creating the view, you can query it directly by using the view name. For example, if you name the view francetax, you can query it like *select COUNT(*) from francetax* using the **query** method

8. Any duckdb tables can be exported as CSVs or parquet files with the ***export_tables*** method

Notes and Recommendations

- Select * queries will naturally take longer to execute because it forces duckDB to read every column off disk, and thus doesn't fully leverage the advantages of parquet. If you can cut out columns in your query, then it will execute faster. In addition, use LIMIT when possible.

- Performance will vary depending on your laptop, with a laptop with 64GB RAM and 4 cores performing better than a laptop with 8GB RAM and 2 cores. The laptop hardware will affect query speed, as well as how many tables you can store.

- If you are using a persistent duckdb database, you only need to create the view on the duckdb dataset once.