

2024

EFREI  
PARIS

---

# DATA ENGINEERING ON CLOUD

---

Rahma  
ALBEKBASHY

Doha  
HAJJOU

# INTRODUCTION

Ce projet a pour objectif de traiter les résultats des élections européennes à Paris pour les années 2014, 2019 et 2024 en utilisant des services cloud.

Nous avons utilisé Azure Data Lake Storage Gen2 pour stocker les données brutes, Azure Databricks pour les transformer et Power BI pour les visualiser.

# 01

# ÉTAPE 1 : CRÉATION D'UN DATA LAKE (AZURE DATA LAKE STORAGE GEN2)

Pour gérer efficacement les données des résultats des élections européennes de Paris des années 2014, 2019 et 2024, nous avons opté pour l'utilisation d'Azure Data Lake Storage Gen2.

Voici les étapes suivies pour créer et configurer le Data Lake :

## 1. Création d'un Compte de Stockage Azure :

- Nous avons commencé par accéder au portail Azure et navigué vers la section "Créer une ressource".
- Sous les services Azure, nous avons sélectionné "Comptes de stockage" et suivi le processus pour créer un nouveau compte de stockage. Le compte de stockage a été nommé `stockageprojet2024` pour une identification claire.






•

## 2. Activation d'Azure Data Lake Storage Gen2 :

- Une fois le compte de stockage créé, nous avons activé Azure Data Lake Storage Gen2 sur ce compte.
- Cela permet d'utiliser les fonctionnalités avancées de gestion des données offertes par ADLS Gen2, telles que la hiérarchisation des fichiers et la sécurité granulaire.


Microsoft Azure


Search resources, services, and docs (G+)





doha.hajjou@efrei.E


Azure services


  
Create a resource


  
Azure Databricks


  
All resources


  
Subscriptions


  
Azure DevOps organizations

  
Microsoft Entra ID

  
Cost Management ...

  
Storage accounts




  
Help + support

  
More services

Resources

Recent

Favorite

Name	Type	Last Viewed
 GroupeDeRessourcesProjet	Resource group	a few seconds ago
 databricks-projet2024	Azure Databricks Service	10 minutes ago
 stockageprojet2024	Storage account	41 minutes ago

See all

Navigate

## ÉTAPE 2 : CRÉATION DES CONTENEURS DE STAGING ET DATA WAREHOUSE

**Pour organiser et traiter les données des résultats des élections, nous avons structuré notre Data Lake en utilisant des conteneurs spécifiques pour la zone de staging et la zone de Data Warehouse.**

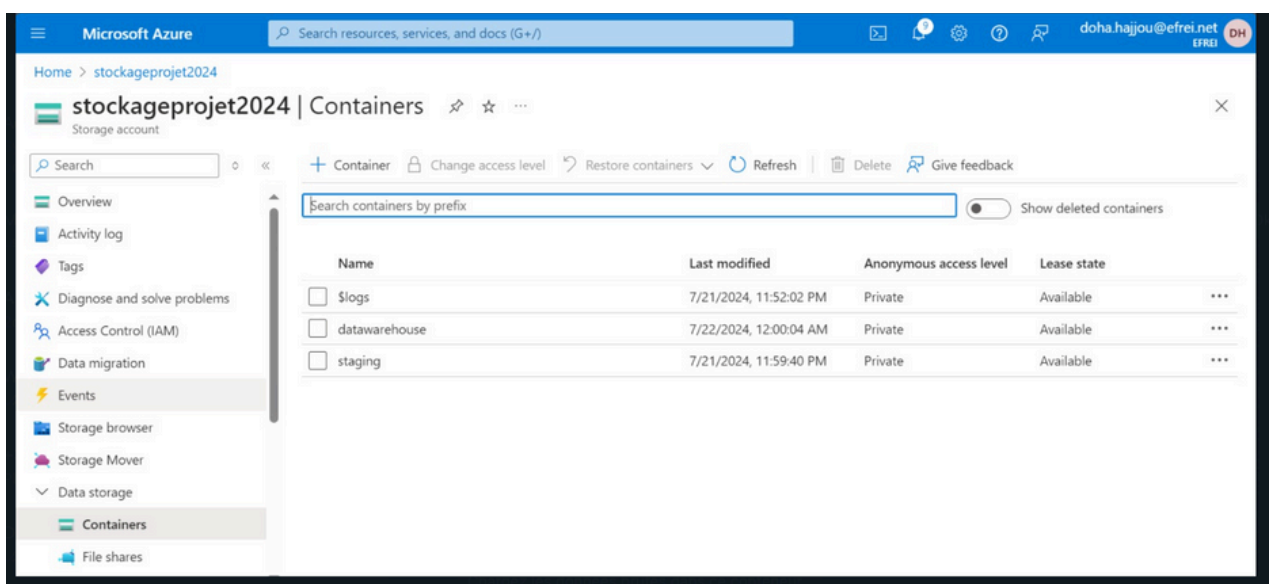
### 1. Zone de Staging :

- Nous avons créé un conteneur nommé staging dans notre compte de stockage Azure Data Lake (stockageprojet2024).
- Dans ce conteneur, nous avons chargé les données brutes des résultats des élections de 2014, 2019, et 2024. Cela nous permet de centraliser toutes les données sources en un seul endroit avant de les transformer.

•

### 2. Zone de Data Warehouse :

- Un autre conteneur nommé datawarehouse a été créé pour stocker les données transformées et prêtes à être utilisées pour des analyses ultérieures.
- Nous avons utilisé Azure Databricks pour traiter et transformer les données depuis la zone de staging vers la zone de Data Warehouse.
- Azure Databricks fournit un environnement évolutif et collaboratif pour effectuer ces transformations de manière efficace.

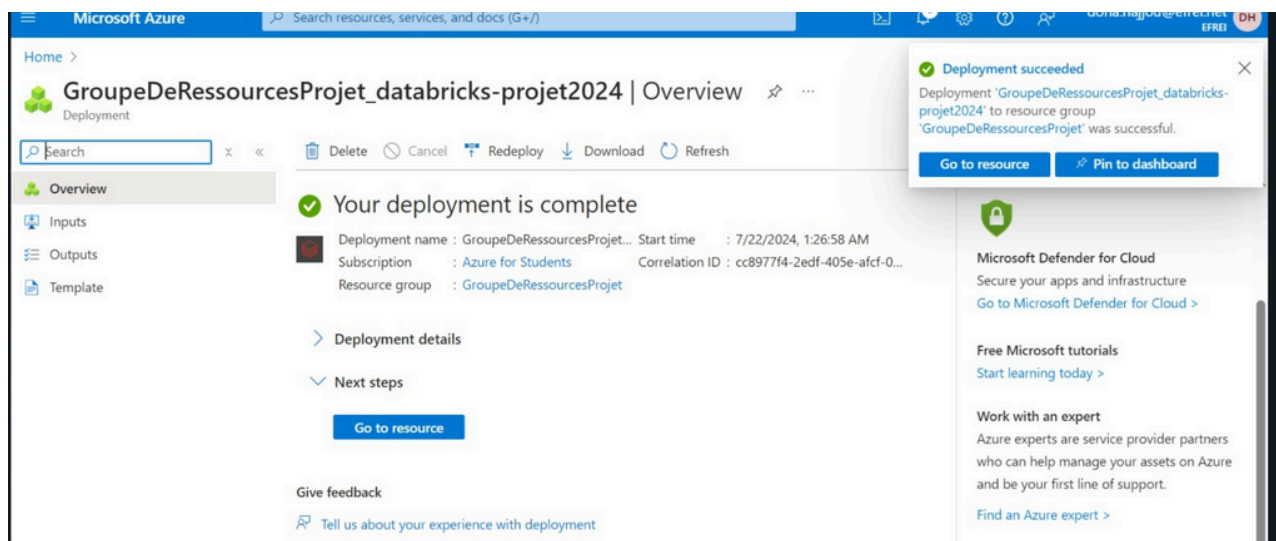


## ÉTAPE 3 : CHARGEMENT DES DONNÉES BRUTES DANS LA ZONE DE STAGING

### VOILÀ :

Pour commencer le traitement des données des résultats des élections, nous avons d'abord chargé les données brutes dans la zone de staging. Cela permet de centraliser toutes les données sources en un seul endroit avant de les transformer et de les charger dans le Data Warehouse.

- Chargement des Données Brutes :
  - Nous avons déployé un cluster Azure Databricks dans le groupe de ressources GroupeDeRessourcesProjet pour faciliter le traitement des données.
  - Le déploiement a été réalisé avec succès, comme indiqué dans la capture d'écran ci-dessous :



- Transfert des Données :
  - Une fois le cluster déployé, nous avons utilisé Azure Databricks pour transférer les fichiers CSV contenant les résultats des élections de 2014, 2019, et 2024 vers le conteneur staging.
  - Les données brutes ont été chargées en utilisant des scripts de traitement des données, garantissant que toutes les informations sont disponibles pour l'étape de transformation suivante.

## PARTIE 1

### 1. Initialisation de la Session Spark :

- Nous avons commencé par initialiser une session Spark et configuré l'accès au stockage Azure.
- Les chemins des fichiers CSV ont été définis pour les années 2014, 2019, et 2024.

```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F

Initialize Spark session
spark = SparkSession.builder \
    .appName("ProjetElectoral") \
    .config("spark.master", "local[*]") \
    .getOrCreate()

Set the Azure storage account key directly in the Spark session configuration
spark.conf.set("fs.azure.account.key.stockageprojet2024.dfs.core.windows.net", "I1P+e9o9Rt6m0adrgoB8rzls0u2AHn3lSFaPDE1YMyQZ0msQs6aRJLu+zmDk09YwqL9MJHGU3BD0+AstWebm9A==")

Define the paths to the CSV files
csv_files = {
    "data24": "abfss://staging@stockageprojet2024.dfs.core.windows.net/election-europeene-paris-2024.csv",
    "data14": "abfss://staging@stockageprojet2024.dfs.core.windows.net/elections-europeennes-2014.csv",
    "data19": "abfss://staging@stockageprojet2024.dfs.core.windows.net/elections-europeennes-2019-Paris.csv"

Read the CSV files with the correct delimiter and infer schema
dataframes = {}
for name, path in csv_files.items():
    name = spark.read.option("header", "true").option("sep", ";").option("inferSchema", "true").csv(path)
    dataframes[name] = name

Verify the schema to ensure columns are read correctly
for name, df in dataframes.items():
    print(f"Schema of {name}:")
    df.printSchema()
    df.show(5, truncate=False)
```

### Lecture et Transformation des Données :

- Lecture des fichiers CSV avec le bon délimiteur et inférence du schéma.
- Nettoyage des données : suppression des valeurs nulles, conversion des types de colonnes, calcul de nouvelles colonnes, et filtrage des lignes.

```
# Function to clean and transform dataframes
def clean_and_transform(df):
    # Drop rows with any null values
    df = df.dropna()

    # Convert columns to appropriate data types (example)
    df = df.withColumn("column1", F.col("column1").cast(IntegerType()))
    df = df.withColumn("column2", F.col("column2").cast(DoubleType()))
    df = df.withColumn("date_column", F.to_date(F.col("date_column"), 'dd/MM/yyyy'))

    # Calculate new columns if necessary (example: vote percentage)
    if "total_votes" in df.columns and "valid_votes" in df.columns:
        df = df.withColumn("vote_percentage", (F.col("valid_votes") / F.col("total_votes")) * 100)

    # Filter out rows based on conditions (example: valid votes > 0)
    df = df.filter(F.col("valid_votes") > 0)

    return df
```

# ÉTAPE 4 : TRANSFORMATION DES DONNÉES AVEC AZURE DATABRICKS

## PARTIE 2

Analyse des Données :

- Création de graphiques pour visualiser les données des élections.
- Histogrammes des votes par candidat, répartition des votes par arrondissement, et évolution des votes au fil des années.

```
# Read the CSV files with the correct delimiter and infer schema
dataframes = {
    name: spark.read.option("header", "true").option("sep", ";").option("inferSchema", "true").csv(path)
    for name, path in csv_files.items()
}

# Function to rename columns to simpler names
def rename_columns(df):
    for col in df.columns:
        new_col = col.replace(':', '_').replace(' ', '_').replace('/', '_').replace('\\', '_').replace('-', '_')
        df = df.withColumnRenamed(col, new_col)
    return df

# Rename columns for all dataframes
dataframes = {name: rename_columns(df) for name, df in dataframes.items()}

# Fill missing values or drop rows/columns with missing values
for name, df in dataframes.items():
    if 'nb_votant' in df.columns:
        df = df.fillna({'nb_votant': 0})
    if 'date' in df.columns:
        df = df.dropna(subset=['date'])
    if 'NB_VOTANT' in df.columns:
        df = df.fillna({'NB_VOTANT': 0})
    if 'DATE' in df.columns:
        df = df.dropna(subset=['DATE'])
    dataframes[name] = df

# Verify the cleaned data
for name, df in dataframes.items():
    print(f"Cleaned schema of {name}:")
    df.printSchema()
    df.show(5, truncate=False)

# Optionally, write the cleaned data back to storage
df.write.mode("overwrite").option("header", "true").csv(f"abfss://datawarehouse@stockageprojet2024.dfs.core.windows.net/cleaned_{name}.csv")

# Display dataframes
display(dataframes["data24"])
display(dataframes["data14"])
display(dataframes["data19"])
```

112 Spark Jobs

```
# Function to count null values
def count_nulls(df):
    return df.select([(F.sum(F.col(c).isNull().cast("int")).alias(c) for c in df.columns)])

# Count null values for each DataFrame
for key, df in dataframes.items():
    null_counts = count_nulls(df)
    print(f"Null values in {key} DataFrame:")
    null_counts.show(truncate=False)

# Example: Adding a new column 'total_votes' as a sum of some columns
for name, df in dataframes.items():
    if 'nb_b1' in df.columns and 'nb_nul' in df.columns and 'nb_exprim' in df.columns:
        df = df.withColumn('total_votes', F.col('nb_b1') + F.col('nb_nul') + F.col('nb_exprim'))
        dataframes[name] = df

# Clean and fill null values in 'geo_shape' and 'geo_point_2d' columns
for name, df in dataframes.items():
    if 'geo_shape' in df.columns:
        df = df.withColumn("geo_shape", F.when(F.col("geo_shape").isNull(), "unknown").otherwise(F.col("geo_shape")))
    if 'geo_point_2d' in df.columns:
        df = df.withColumn("geo_point_2d", F.when(F.col("geo_point_2d").isNull(), "unknown").otherwise(F.col("geo_point_2d")))
    dataframes[name] = df

# Write cleaned DataFrames to Parquet
output_path = {
    "data24": "abfss://datawarehouse@stockageprojet2024.dfs.core.windows.net/cleaned_data/data24",
    "data14": "abfss://datawarehouse@stockageprojet2024.dfs.core.windows.net/cleaned_data/data14",
    "data19": "abfss://datawarehouse@stockageprojet2024.dfs.core.windows.net/cleaned_data/data19"
}

# Write the DataFrames in Parquet format
for key, df in dataframes.items():
    df.write.mode("overwrite").parquet(output_path[key])
```



# ÉTAPE 4 : TRANSFORMATION DES DONNÉES AVEC AZURE DATABRICKS

## PARTIE 2

Encore beaucoup de nettoyage pour arriver a des dataframe structurée

```
import matplotlib.pyplot as plt
import pandas as pd

# Exemple de création de plots avec matplotlib

# Convertir un DataFrame Spark en Pandas DataFrame
def convert_to_pandas(spark_df):
    return spark_df.toPandas()

# Histogramme des votes pour un candidat spécifique
def plot_candidate_votes(df, candidate):
    pdf = convert_to_pandas(df.select('num_arrond', candidate))
    pdf = pdf[pdf[candidate].notnull()] # Filter out null values
    plt.figure(figsize=(10, 6))
    plt.hist(pdf[candidate], bins=30, color='blue', edgecolor='black')
    plt.title(f'Histogram of votes for {candidate}')
    plt.xlabel('Number of Votes')
    plt.ylabel('Frequency')
    plt.show()

# Répartition des votes par arrondissement
def plot_votes_by_arrond(df):
    pdf = convert_to_pandas(df.groupBy('num_arrond').sum('nb_votant').withColumnRenamed('sum(nb_votant)', 'total_votes'))
    plt.figure(figsize=(10, 6))
    plt.bar(pdf['num_arrond'], pdf['total_votes'], color='green')
    plt.title('Total Votes by Arrondissement')
    plt.xlabel('Arrondissement')
    plt.ylabel('Total Votes')
    plt.show()

# Évolution du nombre de votants sur les années
def plot_votes_by_year(dataframes):
    votes_by_year = []
    for name, df in dataframes.items():
        year = int(name[-2:])
        total_votes = df.select(F.sum('nb_votant').alias('total_votes')).collect()[0]['total_votes']
        votes_by_year.append((year, total_votes))
    votes_by_year.sort()
    years, total_votes = zip(*votes_by_year)
    plt.figure(figsize=(10, 6))
    plt.plot(years, total_votes, marker='o')
    plt.title('Total Votes Over the Years')
    plt.xlabel('Year')
    plt.ylabel('Total Votes')
    plt.show()

# Exemple d'utilisation des fonctions pour créer des plots
plot_candidate_votes(dataframes['data24'], 'bardella_jordan')
plot_votes_by_arrond(dataframes['data24'])
plot_votes_by_year(dataframes)
```

Table



## PARTIE 3

Analyse des Données :

- Création de graphiques pour visualiser les données des élections.
- Histogrammes des votes par candidat, répartition des votes par arrondissement, et évolution des votes au fil des années.

```
import matplotlib.pyplot as plt
import pandas as pd

# Exemple de création de plots avec matplotlib

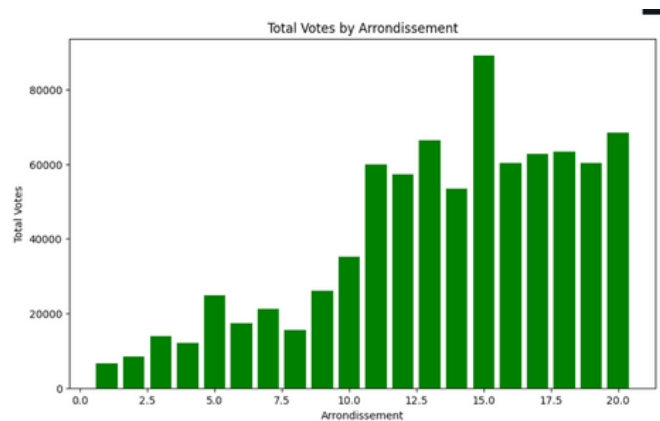
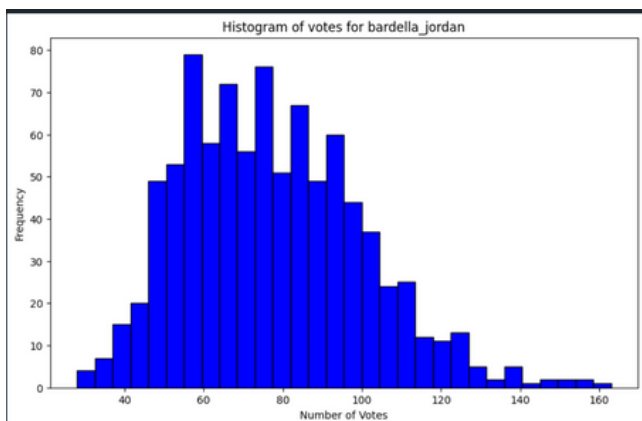
# Convertir un DataFrame Spark en Pandas DataFrame
def convert_to_pandas(spark_df):
    return spark_df.toPandas()

# Histogramme des votes pour un candidat spécifique
def plot_candidate_votes(df, candidate):
    pdf = convert_to_pandas(df.select('num_arrond', candidate))
    pdf = pdf[pdf[candidate].notnull()] # Filter out null values
    plt.figure(figsize=(10, 6))
    plt.hist(pdf[candidate], bins=30, color='blue', edgecolor='black')
    plt.title(f'Histogram of votes for {candidate}')
    plt.xlabel('Number of Votes')
    plt.ylabel('Frequency')
    plt.show()

# Répartition des votes par arrondissement
def plot_votes_by_arrond(df):
    pdf = convert_to_pandas(df.groupby('num_arrond').sum('nb_votant').withColumnRenamed('sum(nb_votant)', 'total_votes'))
    plt.figure(figsize=(10, 6))
    plt.bar(pdf['num_arrond'], pdf['total_votes'], color='green')
    plt.title('Total Votes by Arrondissement')
    plt.xlabel('Arrondissement')
    plt.ylabel('Total Votes')
    plt.show()

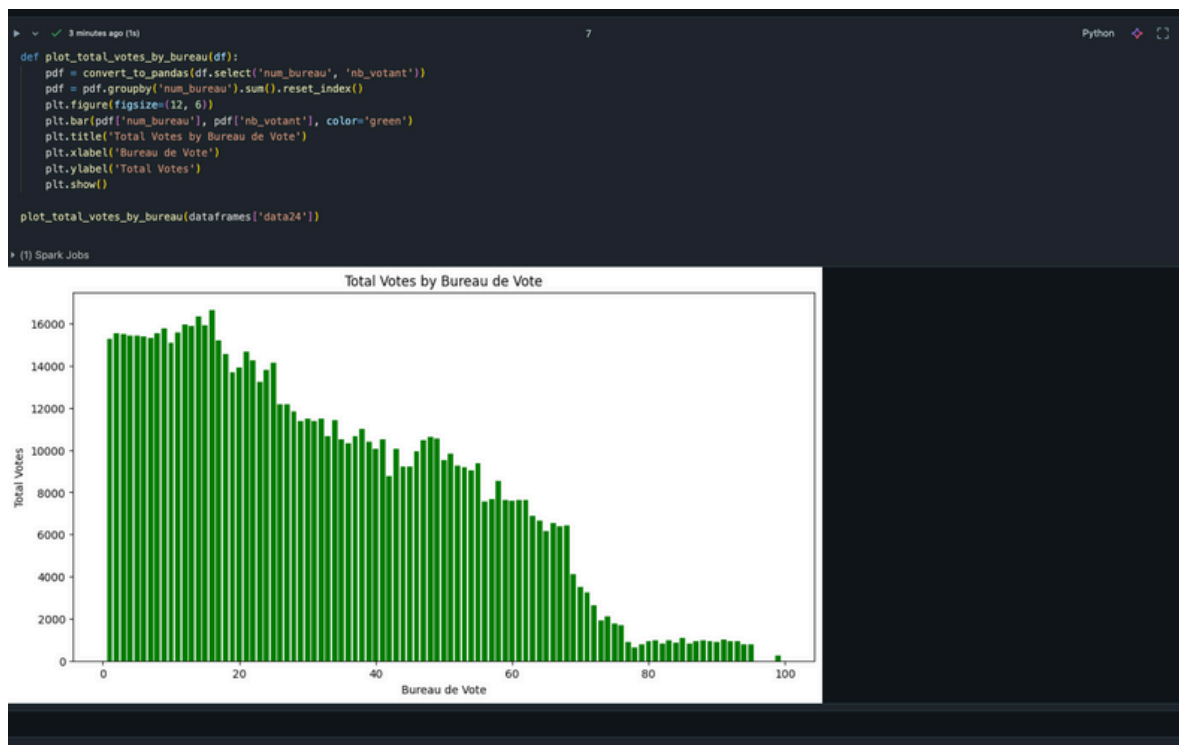
# Évolution du nombre de votants sur les années
def plot_votes_by_year(dataframes):
    votes_by_year = []
    for name, df in dataframes.items():
        year = int(name[-2:])
        total_votes = df.select(F.sum('nb_votant').alias('total_votes')).collect()[0]['total_votes']
        votes_by_year.append((year, total_votes))
    votes_by_year.sort()
    years, total_votes = zip(*votes_by_year)
    plt.figure(figsize=(10, 6))
    plt.plot(years, total_votes, marker='o')
    plt.title('Total Votes Over the Years')
    plt.xlabel('Year')
    plt.ylabel('Total Votes')
    plt.show()

# Exemple d'utilisation des fonctions pour créer des plots
plot_candidate_votes(dataframes['data24'], 'bardella_jordan')
plot_votes_by_arrond(dataframes['data24'])
plot_votes_by_year(dataframes)
```



# ÉTAPE 4 : TRANSFORMATION DES DONNÉES AVEC AZURE DATABRICKS

## PARTIE 3



## PARTIE 4

Modèle de Machine Learning :

- Préparation des données pour l'entraînement du modèle.
- Création et entraînement d'un modèle de régression linéaire pour prédire le nombre de votants (nb\_exprim).

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator

# Étape 1: Préparation des données
def prepare_data(df):
    # Sélectionner les colonnes pertinentes
    df = df.select('nb_exprim', 'nb_votant', 'nb_inscr').dropna()

    # Créer une colonne features qui contient les features sous forme de vecteur
    assembler = VectorAssembler(inputCols=['nb_votant', 'nb_inscr'], outputCol='features')
    df = assembler.transform(df)
    return df

data = prepare_data(dataframes['data24'])

# Étape 2: Création et entraînement du modèle
lr = LinearRegression(featuresCol='features', labelCol='nb_exprim')

# Diviser les données en training et test sets
train_data, test_data = data.randomSplit([0.8, 0.2], seed=1234)

# Entraîner le modèle
lr_model = lr.fit(train_data)

# Étape 3: Évaluation du modèle
predictions = lr_model.transform(test_data)

# Évaluer les performances du modèle
evaluator = RegressionEvaluator(predictionCol='prediction', labelCol='nb_exprim', metricName='rmse')
rmse = evaluator.evaluate(predictions)
r2 = evaluator.evaluate(predictions, (evaluator.metricName: "r2"))

print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R2: {r2}")

# Étape 4: Affichage des résultats
predictions.select("nb_exprim", "prediction", "features").show(5)

# Optionnel: Afficher les coefficients et l'intercept
print(f"Coefficients: {lr_model.coefficients}")
print(f"Intercept: {lr_model.intercept}")
```

```
> data: pyspark.sql.dataframe.DataFrame
> train_data: pyspark.sql.dataframe.DataFrame
> test_data: pyspark.sql.dataframe.DataFrame
> predictions: pyspark.sql.dataframe.DataFrame

Root Mean Squared Error (RMSE): 4.018329169315168
R2: 0.9993138884143484

|nb_exprim| prediction| features|
|-----|-----|-----|
| 336| 332.52317634132174| [340.0,760.0]|
| 428| 437.52681861119623| [449.0,1283.0]|
| 551| 554.0257414754984| [565.0,1291.0]|
| 563| 557.2849370579889| [570.0,1437.0]|
| 573| 569.1295950530903| [586.0,1790.0]|

only showing top 5 rows

Coefficients: [1.0137023234074294,-0.012392575579086476]
Intercept: -2.7172561770985424
```

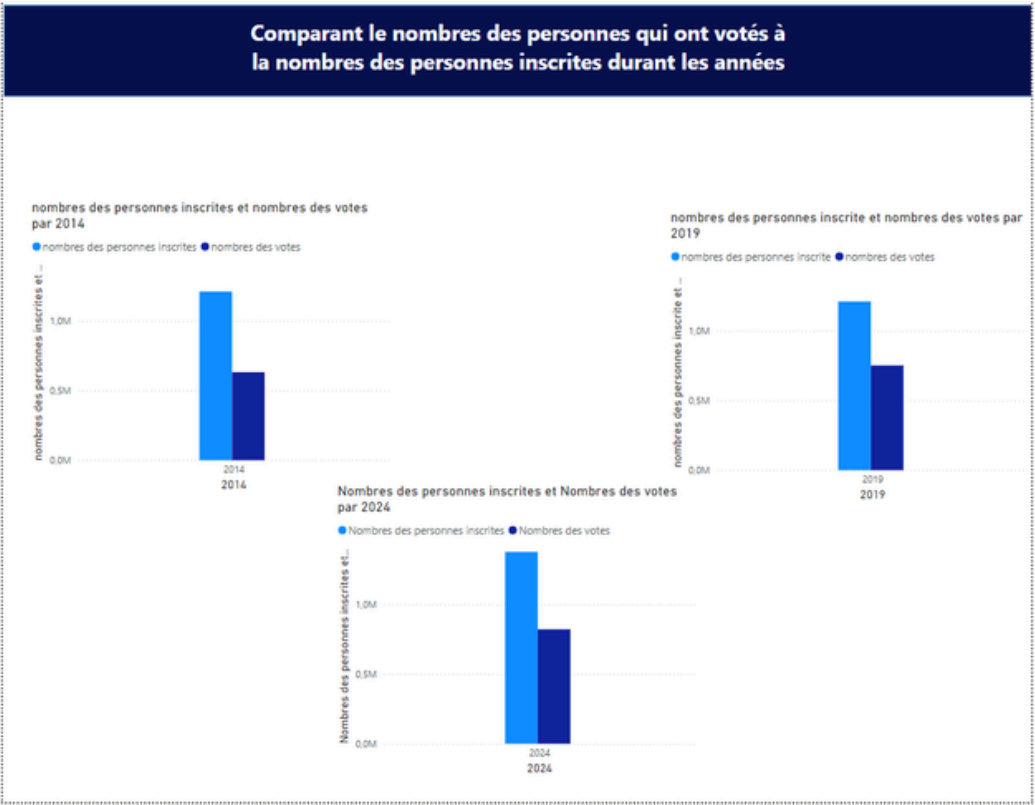
**LE MODÈLE MONTRE UNE BONNE PERFORMANCE AVEC UN RMSE DE 4.01 ET UN R2 DE 0.999.**

## ÉTAPE 5 : CONNEXION À POWER BI ET CRÉATION DU DASHBOARD

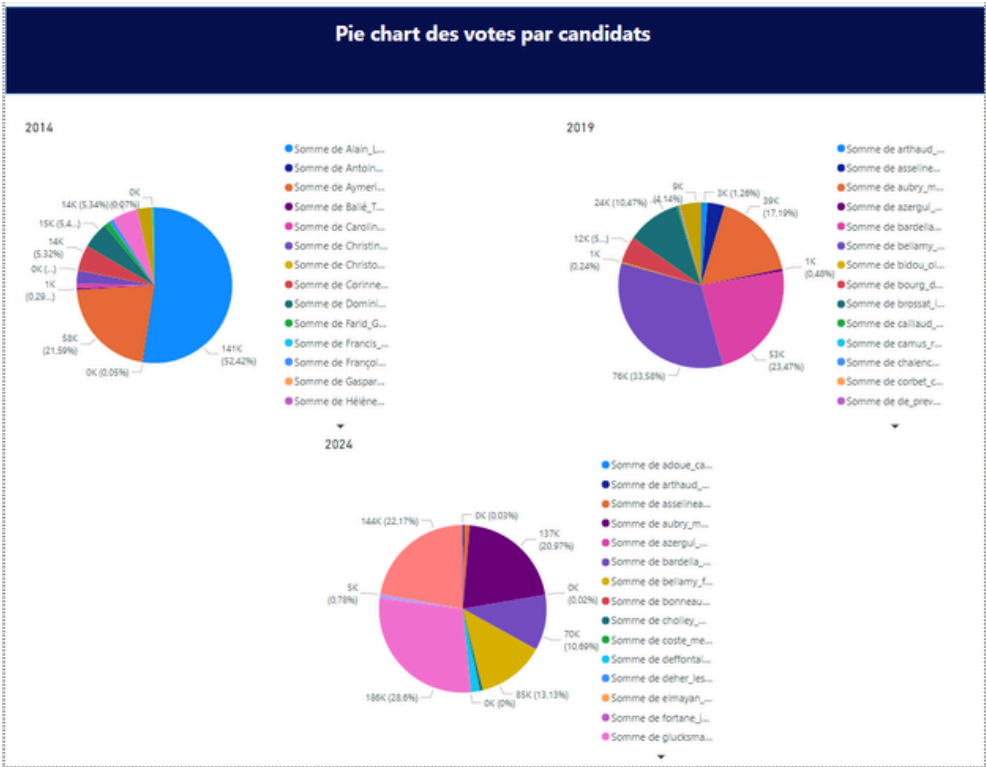
### **POUR VISUALISER LES DONNÉES DES ÉLECTIONS EUROPÉENNES DE PARIS ET PERMETTRE UNE ANALYSE INTERACTIVE, NOUS AVONS UTILISÉ POWER BI. VOICI LES ÉTAPES SUIVIES POUR SE CONNECTER À AZURE BLOB STORAGE ET CRÉER UN DASHBOARD :**

- Connexion à Azure Blob Storage :
  - Nous avons configuré les accès nécessaires pour permettre à Power BI de se connecter à notre Azure Blob Storage.
  - Les données transformées ont été chargées dans Power BI en utilisant les connexions disponibles pour Azure Blob Storage.
- Création des Visualisations :
  - Nous avons géré les relations entre les différentes tables de données pour s'assurer que les visualisations reflètent correctement les relations et les données sous-jacentes.
  - Un dashboard interactif a été créé, comprenant plusieurs types de visualisations pour analyser les résultats des élections sur différentes années.
- Visualisations Incluant :
  - Carte des résultats par arrondissement : montre la répartition géographique des votes par arrondissement pour l'année 2014.





GRAPHIQUES COMPARATIFS : COMPARE LE NOMBRE DE PERSONNES INSCRITES ET LE NOMBRE DE VOTES POUR LES ANNÉES 2014, 2019 ET 2024.



DIAGRAMMES CIRCULAIRES DES VOTES PAR CANDIDAT : VISUALISE LA RÉPARTITION DES VOTES ENTRE LES DIFFÉRENTS CANDIDATS POUR CHAQUE ANNÉE (2014, 2019, 2024).

# CONCLUSION

Finalelement

Ce projet de Data Engineering sur le Cloud a permis de traiter, analyser et visualiser les données des élections européennes à Paris en utilisant des technologies avancées telles qu'Azure Data Lake Storage Gen2, Azure Databricks et Power BI. Les étapes suivies ont permis de démontrer comment des solutions cloud peuvent être utilisées pour gérer efficacement de grandes quantités de données et en extraire des insights précieux.

Merci !