Projeto 11

Um pipeline **ETL (Extract, Transform, and Load)** extrai dados de fontes, transforma-os e carrega-os em um sistema de armazenamento.

Ele ajuda a criar formatos de dados limpos e utilizáveis para análise.

O PySpark é ideal para construir pipelines ETL para processamento de dados em larga escala.

Ele oferece computação distribuída, alto desempenho e lida com dados estruturados e não estruturados de forma eficiente.

Este projeto o levará pela construção de um pipeline ETL usando o PySpark.

Construindo um Pipeline ETL usando PySpark

O conjunto de dados que usaremos para construir um ETL Pipeline contém dados relacionados à temperatura para vários países de 1961 a 2022.

As colunas incluem identificadores como ObjectId, Country, ISO2 e ISO3, juntamente com dados de temperatura anuais, como F1961, F1962, etc., como valores de ponto flutuante.

Algumas colunas contêm valores ausentes.

Você pode baixar este conjunto de dados **aqui.**

Estudo de Caso

Este estudo de caso utiliza dois conjuntos de dados abrangentes para explorar o impacto global das emissões de carbono.

O primeiro conjunto de dados, **Mudança Anual de Temperatura da Superfície (Annual Surface Temperature Change)** , fornece anomalias de temperatura em vários países em relação a uma linha de base, medida em graus Celsius.

O conjunto de dados abrange de 1960 a 2022 e inclui observações anuais, oferecendo uma perspectiva temporal sobre como as temperaturas globais mudaram ao longo de décadas.

Os principais recursos incluem nomes de países e valores de mudança de temperatura para cada ano.

O segundo conjunto de dados, **Concentrações Atmosféricas Mensais de CO_2 (Monthly Atmospheric CO_2 Concentrations)**, inclui medições globais de CO_2 de 1958 a 2024.

Os dados são representados em partes por milhão (ppm), com observações mensais que permitem uma análise temporal detalhada.

Os principais recursos incluem a data da medição e os valores de concentração de CO_2.

Juntos, esses conjuntos de dados fornecem uma base sólida para analisar tendências climáticas e entender a ligação entre emissões de carbono e aquecimento global.

Desenvolveremos um Pipeline ETL usando PySpark para processar este conjunto de dados para lidar com as seguintes tarefas:

> **1. Extrair:** Carregar o conjunto de dados do arquivo CSV;
>
> **2. Transformar:** Limpar os dados, lidar com valores ausentes e pivotar dados de temperatura anuais para análise;
>
> **3. Carregar:** Salvar os dados processados em um novo formato de armazenamento (por exemplo, Parquet ou um banco de dados).

Solução

Etapa 1: Configurando o ambiente e inicializando uma sessão do PySpark

Certifique-se de que o PySpark esteja instalado e configurado.

Execute o seguinte comando para instalar o PySpark se ele ainda não estiver instalado:

In [1]:
```
#pip install pyspark
```

Inicialize uma sessão PySpark para permitir a interação com a estrutura Spark:

In [2]:
```
# Importa a classe SparkSession do módulo pyspark.sql
from pyspark.sql import SparkSession

# Configura a SparkSession
builder = SparkSession.builder
builder = builder.appName("ETL Pipeline")  # Define o nome do aplicativo
spark = builder.getOrCreate()              # Cria ou reutiliza uma SparkSession existente

# Exibe uma mensagem indicando que a SparkSession foi inicializada com sucesso
print("SparkSession inicializada com sucesso!")
```

```
SparkSession inicializada com sucesso!
```

Etapa 2: Extrair - Carregar o Conjunto de Dados

A próxima etapa é carregar o conjunto de dados em um PySpark DataFrame:

In [3]:
```
# Define o caminho do arquivo CSV
file_path = "C:/Users/giova/Projetos/Projeto_11/temperature.csv"

# Carrega o arquivo CSV em um DataFrame Spark
# - header=True indica que o arquivo contém cabeçalho
```

```
# - inferSchema=True permite que o Spark infira automaticamente os tipos das colunas
df = spark.read.csv(file_path, header=True, inferSchema=True)
```

In [4]:
```
# Exibe o esquema do DataFrame (nomes das colunas e seus tipos de dados)
df.printSchema()
```

```
root
 |-- ObjectId: integer (nullable = true)
 |-- Country: string (nullable = true)
 |-- ISO2: string (nullable = true)
 |-- ISO3: string (nullable = true)
 |-- F1961: double (nullable = true)
 |-- F1962: double (nullable = true)
 |-- F1963: double (nullable = true)
 |-- F1964: double (nullable = true)
 |-- F1965: double (nullable = true)
 |-- F1966: double (nullable = true)
 |-- F1967: double (nullable = true)
 |-- F1968: double (nullable = true)
 |-- F1969: double (nullable = true)
 |-- F1970: double (nullable = true)
 |-- F1971: double (nullable = true)
 |-- F1972: double (nullable = true)
 |-- F1973: double (nullable = true)
 |-- F1974: double (nullable = true)
 |-- F1975: double (nullable = true)
 |-- F1976: double (nullable = true)
 |-- F1977: double (nullable = true)
 |-- F1978: double (nullable = true)
 |-- F1979: double (nullable = true)
 |-- F1980: double (nullable = true)
 |-- F1981: double (nullable = true)
 |-- F1982: double (nullable = true)
 |-- F1983: double (nullable = true)
 |-- F1984: double (nullable = true)
 |-- F1985: double (nullable = true)
 |-- F1986: double (nullable = true)
 |-- F1987: double (nullable = true)
 |-- F1988: double (nullable = true)
 |-- F1989: double (nullable = true)
 |-- F1990: double (nullable = true)
 |-- F1991: double (nullable = true)
 |-- F1992: double (nullable = true)
 |-- F1993: double (nullable = true)
 |-- F1994: double (nullable = true)
 |-- F1995: double (nullable = true)
 |-- F1996: double (nullable = true)
 |-- F1997: double (nullable = true)
 |-- F1998: double (nullable = true)
 |-- F1999: double (nullable = true)
 |-- F2000: double (nullable = true)
 |-- F2001: double (nullable = true)
 |-- F2002: double (nullable = true)
 |-- F2003: double (nullable = true)
 |-- F2004: double (nullable = true)
 |-- F2005: double (nullable = true)
 |-- F2006: double (nullable = true)
 |-- F2007: double (nullable = true)
 |-- F2008: double (nullable = true)
 |-- F2009: double (nullable = true)
 |-- F2010: double (nullable = true)
 |-- F2011: double (nullable = true)
 |-- F2012: double (nullable = true)
 |-- F2013: double (nullable = true)
 |-- F2014: double (nullable = true)
 |-- F2015: double (nullable = true)
 |-- F2016: double (nullable = true)
 |-- F2017: double (nullable = true)
 |-- F2018: double (nullable = true)
 |-- F2019: double (nullable = true)
 |-- F2020: double (nullable = true)
 |-- F2021: double (nullable = true)
 |-- F2022: double (nullable = true)
```

In [5]:
```python
# Exibe as primeiras 5 linhas do DataFrame no console
df.show(5)
```

```
+--------+--------------------+----+----+------+------+------+------+------+-----+------+------+------+------+------+------+------+------+------+------+------+------+-----+------+------+------+------+------+------+------+------+-----+------+---
--+------+------+------+-----+------+------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|ObjectId|             Country|ISO2|ISO3| F1961| F1962| F1963| F1964| F1965|F1966| F1967| F1968| F1969| F1970| F1971| F1972| F1973| F1974| F1975| F1976| F1977| F1978|F1979| F1980| F1981| F1982| F1983| F1984| F1985| F1986| F1987|F1988| F1989|F19
90| F1991| F1992| F1993|F1994| F1995| F1996|F1997|F1998|F1999|F2000|F2001|F2002|F2003|F2004|F2005|F2006|F2007|F2008|F2009|F2010|F2011|F2012|F2013|F2014|F2015|F2016|F2017|F2018|F2019|F2020|F2021|F2022|
+--------+--------------------+----+----+------+------+------+------+------+-----+------+------+------+------+------+------+------+------+------+------+------+------+-----+------+------+------+------+------+------+------+------+-----+------+---
--+------+------+------+-----+------+------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|       1|Afghanistan, Isla...|  AF| AFG|-0.113|-0.164| 0.847|-0.764|-0.244|0.226|-0.371|-0.423|-0.539| 0.813| 0.619|-1.124| 0.232|-0.489|-0.445|-0.286| 0.513| 0.129|0.361|   0.6| 0.483|-0.346| 0.164| 0.145| 0.283|-0.141| 0.391|0.919|-0.205| 0.
73|-0.168|-0.294|  0.22| 0.43| 0.359|-0.116|0.471|0.675|1.198|0.993|1.311|1.365|0.587|1.373|0.401| 1.72|0.675|0.704|0.895|1.613|1.397|0.223|1.281|0.456|1.093|1.555| 1.54|1.544| 0.91|0.498|1.327|2.012|
|       2|             Albania|  AL| ALB| 0.627| 0.326| 0.075|-0.166|-0.388|0.559|-0.074| 0.081|-0.013|-0.106|-0.195|-0.069|-0.288|-0.139|-0.211|-0.683| 0.545|-0.814|0.203|-0.414|-0.351| 0.173|-0.128| -0.27|-0.103| 0.569|-0.106| 0.37|-0.066|0.7
95|-0.269| 0.106| 0.076| 1.33|-0.172|-0.038|0.075|0.795| 0.67|1.065|1.532|0.492| 0.97|0.444|0.189|0.345|1.316|0.978| 0.91|1.191|1.055|1.487|1.333|1.198|1.569|1.464|1.121|2.028|1.675|1.498|1.536|1.518|
|       3|             Algeria|  DZ| DZA| 0.164| 0.114| 0.077|  0.25|  -0.1|0.433|-0.026|-0.067| 0.291| 0.116|-0.385|-0.348|-0.015|-0.503|-0.539|-0.782| 0.504| 0.012|0.654| 0.232| 0.215| 0.399|  0.56|-0.004| 0.508| 0.296| 0.975|1.304| 0.386|1.2
66| 0.031|-0.312| 0.552|0.732| 0.595| 0.846|1.059|1.109|1.476| 0.82|1.856|1.258|1.585|0.988|1.264|1.395| 1.22|1.185|0.945|2.265|1.398|1.147|1.192| 1.69|1.121|1.757|1.512| 1.21|1.115|1.926| 2.33|1.688|
|       4|      American Samoa|  AS| ASM| 0.079|-0.042| 0.169| -0.14|-0.562|0.181|-0.368|-0.187| 0.132|-0.047|-0.477|-0.067|  0.33|-0.308|-0.118|-0.177| 0.156| 0.092|0.341|  0.35| 0.179|  0.28| 0.313| 0.277| 0.256| 0.394| 0.354|0.509| 0.143|0.4
97| 0.641| 0.344|-0.069|0.189| 0.755| 0.784| NULL| NULL|0.242|0.626|0.904|1.152|0.716|0.191|0.801|0.403|1.032| 0.67| NULL|1.311|0.854|0.924|1.257| 1.17|1.009|1.539|1.435|1.189|1.539| 1.43|1.268|1.256|
|       5|Andorra, Principa...|  AD| AND| 0.736| 0.112|-0.752| 0.308| -0.49|0.415| 0.637| 0.018|-0.137| 0.121|-0.326|-0.499| 0.025|-0.371| 0.246|-0.045|-0.093|-0.163|0.058|-0.188| 0.178| 1.044| 0.859|-0.157| 0.059| 0.387| 0.397|0.883| 1.162|1.7
36| 0.231| 0.386| 0.174|1.508| 1.279|  0.57|1.788|1.018|1.055| 1.05| 1.48|0.835|1.949|0.936|0.851|1.485|1.024|0.946|1.413|0.471|1.677|1.265|0.831|1.946| 1.69| 1.99|1.925|1.919|1.964|2.562|1.533|3.243|
+--------+--------------------+----+----+------+------+------+------+------+-----+------+------+------+------+------+------+------+------+------+------+------+------+-----+------+------+------+------+------+------+------+------+-----+------+---
--+------+------+------+-----+------+------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

In [6]:
```python
#Converte o DataFrame Spark em um DataFrame Pandas
pdf = df.toPandas()

# Exibe as Primeiras Linhas usando Pandas
pdf.head()
```

Out[6]:

	ObjectId	Country	ISO2	ISO3	F1961	F1962	F1963	F1964	F1965	F1966	...	F2013	F2014	F2015	F2016	F2017	F2018	F2019	F2020	F2021	F2022
0	1	Afghanistan, Islamic Rep. of	AF	AFG	-0.113	-0.164	0.847	-0.764	-0.244	0.226	...	1.281	0.456	1.093	1.555	1.540	1.544	0.910	0.498	1.327	2.012
1	2	Albania	AL	ALB	0.627	0.326	0.075	-0.166	-0.388	0.559	...	1.333	1.198	1.569	1.464	1.121	2.028	1.675	1.498	1.536	1.518
2	3	Algeria	DZ	DZA	0.164	0.114	0.077	0.250	-0.100	0.433	...	1.192	1.690	1.121	1.757	1.512	1.210	1.115	1.926	2.330	1.688
3	4	American Samoa	AS	ASM	0.079	-0.042	0.169	-0.140	-0.562	0.181	...	1.257	1.170	1.009	1.539	1.435	1.189	1.539	1.430	1.268	1.256
4	5	Andorra, Principality of	AD	AND	0.736	0.112	-0.752	0.308	-0.490	0.415	...	0.831	1.946	1.690	1.990	1.925	1.919	1.964	2.562	1.533	3.243

5 rows × 66 columns

No PySpark, estamos carregando um arquivo CSV em um DataFrame distribuído, que é semelhante a usar pandas.read_csv() para carregar dados em um Pandas DataFrame.

No entanto, diferentemente do Pandas, que usa memória e roda em uma única máquina, o PySpark manipula grandes conjuntos de dados distribuídos em um cluster.

Os métodos **df.printSchema()** e **df.show(5)** fornecem insights sobre o esquema e visualizam os dados, comparáveis a **df.info()** e **df.head()** no Pandas, mas projetados para exploração de dados escaláveis em cargas de trabalho de big data.

Etapa 3: Transformar – Limpar e Processar os Dados

Todos os conjuntos de dados exigem diferentes tipos de etapas de limpeza e processamento.

Nestes dados, substituiremos valores ausentes em colunas importantes como ISO2 ou imputaremos valores de temperatura ausentes:

In [7]:
```python
# Preenche Valores Ausentes (NaN) na Coluna "ISO2" com a String "Unknown"
# fillna recebe um dicionário onde a chave é o nome da coluna e o valor é o preenchimento desejado
df = df.fillna({"ISO2": "Unknown"})
```

In [8]:
```python
# Identificar as Colunas que Começam com 'F', Presumidamente Relacionadas a Temperaturas
temperature_columns = [col for col in df.columns if col.startswith('F')]
```

In [9]:
```python
# Remover Linhas do DataFrame onde Todas as Colunas de Temperatura têm Valores Nulos
df = df.dropna(
    subset=temperature_columns,  # Considera apenas as colunas relacionadas a temperaturas
    how="all"  # Remove apenas se todas as colunas no subset tiverem valores nulos
)
```

Em seguida, transformaremos o conjunto de dados para ter “Ano” como uma única coluna e seu valor de temperatura:

In [10]:
```
# Importa a Função "expr" que é usada para executar expressões SQL diretamente em uma coluna ou conjunto de colunas em um DataFrame PySpark
from pyspark.sql.functions import expr
```

In [11]:
```
# Reorganizar Dados de Temperatura para ter as Colunas 'Year' e 'Temperature'
df_pivot = df.selectExpr(
    # Manter as colunas originais fixas no DataFrame resultante
    "ObjectId",
    "Country",
    "ISO3",

    # Usar a função stack para pivotar os dados de temperatura
    # Gerar 62 pares para os anos de 1961 a 2022
    "stack(62, " +
    ",".join([f"'F{1961 + i}', F{1961 + i}" for i in range(62)]) +
    # Nomear as novas colunas como 'Year' e 'Temperature'
    ") as (Year, Temperature)"
)
```

In [12]:
```
# Atualizar a Coluna 'Year' Convertendo seu Valor para Inteiro
df_pivot = df_pivot.withColumn(
    "Year",  # Nome da coluna a ser substituída
    expr("int(substring(Year, 2, 4))")  # Extrair a parte numérica e converter para inteiro
)
```

In [13]:
```
# Exibe as primeiras 5 linhas do DataFrame no Console
df_pivot.show(5)
```

```
+--------+--------------------+----+----+-----------+
|ObjectId|             Country|ISO3|Year|Temperature|
+--------+--------------------+----+----+-----------+
|       1|Afghanistan, Isla...| AFG|1961|     -0.113|
|       1|Afghanistan, Isla...| AFG|1962|     -0.164|
|       1|Afghanistan, Isla...| AFG|1963|      0.847|
|       1|Afghanistan, Isla...| AFG|1964|     -0.764|
|       1|Afghanistan, Isla...| AFG|1965|     -0.244|
+--------+--------------------+----+----+-----------+
only showing top 5 rows
```

Etapa 4: Carregar – Salvar os Dados Processados

Após concluir todas as etapas de processamento, salvamos os dados transformados em um arquivo Parquet para armazenamento e consulta eficientes:

In [14]:
```
# Define o caminho de saída para o arquivo Parquet
output_path = "/processed_temperature.parquet"
```

In [15]:
```
# Salva o DataFrame no formato Parquet
# Sobrescreve o arquivo se ele já existir
# Especifica que o formato de saída será Parquet
df_pivot.write \
    .mode("overwrite") \
    .parquet(output_path)
```

Esta operação salva o DataFrame transformado como um arquivo Parquet, o que o otimiza para armazenamento e consulta em um ambiente distribuído.

Podemos carregar o arquivo Parquet salvo para garantir que os dados foram salvos corretamente:

In [16]:
```
# Carrega o Arquivo Parquet Salvo
processed_df = spark.read.parquet(output_path)
```

In [17]:
```
# Exibe as primeiras 5 linhas do Arquivo Salvo
processed_df.show(5)
```

```
+--------+--------------------+----+----+-----------+
|ObjectId|             Country|ISO3|Year|Temperature|
+--------+--------------------+----+----+-----------+
|       1|Afghanistan, Isla...| AFG|1961|     -0.113|
|       1|Afghanistan, Isla...| AFG|1962|     -0.164|
|       1|Afghanistan, Isla...| AFG|1963|      0.847|
|       1|Afghanistan, Isla...| AFG|1964|     -0.764|
|       1|Afghanistan, Isla...| AFG|1965|     -0.244|
+--------+--------------------+----+----+-----------+
only showing top 5 rows
```

Resumo

O PySpark é ideal para construir pipelines ETL (Extract, Transform, and Load) para processamento de dados em larga escala.

Ele oferece computação distribuída, alto desempenho e manipula dados estruturados e não estruturados de forma eficiente.

Espero que este projeto sobre "Como Construir um Pipeline ETL" usando Python tenha sido útil para seu aprendizado.

Referência do Projeto: https://thecleverprogrammer.com/2024/11/28/building-an-etl-pipeline-using-pyspark/
Base de Dados: https://statso.io/carbon-emissions-worldwide-case-study/
Autor: Aman Kharwal - https://amankharwal.medium.com/
Projeto Estudado por: Giovani Aloísio da Luz
Disponível em: https://www.giovani-luz.com.br/