databricks ETL PIPELINE - 1

(https://databricks.com)

ETL PIPELINE ON DATABRICKS

```
# END TO END ETL PIPELINE ON DATABRICKS
# here we are taking our dataset from (https://datahub.io/core/glacier-mass-balance)
# by using this dataset we will build an ETL PIPELINE
# we will eXtract our data
# then we will load our data into our databrick file system
# we will transform our data
```

```
# MAIN TASK IS TO SPLIT THE DATAFRAME INTO TWO BASED ON THE YEAR AND THEN PERFORM THE TRANSFORMATION FUNCTIONS ON THAT
```

Creating Spark Session

```
# starting the spark session
spark
```

SparkSession - hive

SparkContext

Spark UI

```
Version
      v3.3.2
Master
      local[8]
AppName
      Databricks Shell
```

```
# datset url
# https://datahub.io/core/glacier-mass-balance
```

Importing Libraries

```
# request is a HTTP library which is used to get the online data onto databaricks
import requests
from pyspark.sql import DataFrame
```

```
help(requests)
```

```
Help on package requests:

NAME
    requests

DESCRIPTION
    Requests HTTP Library
    ~~~~~~~~~~~~~~~~~~~~~

    Requests is an HTTP library, written in Python, for human beings.
    Basic GET usage:

       >>> import requests
       >>> r = requests.get('https://www.python.org (https://www.python.org)')
       >>> r.status_code
       200
       >>> b'Python is a programming language' in r.content
       True

    ... or POST:
```

```
help(requests.get  )
```

```
Help on function get in module requests.api:

get(url, params=None, **kwargs)
    Sends a GET request.

    :param url: URL for the new :class:`Request` object.
    :param params: (optional) Dictionary, list of tuples or bytes to send
        in the query string for the :class:`Request`.
    :param \*\*kwargs: Optional arguments that ``request`` takes.
    :return: :class:`Response <Response>` object
    :rtype: requests.Response
```

Extracting Data

```
# reading this stream of data using get function which have 2 paratmeters :
# 1) passing the url
# 2) stram parameter : if it is set to True it means that is will read the data in the stream if tokens.
# we are storing it in a variable r
# we will be opening our file with the name glaciers.csv and 'wb' means we are writing the byted of the files
with requests.get('https://datahub.io/core/glacier-mass-balance/r/glaciers.csv ' , stream = True) as r:
    with open('/dbfs/glacier.csv','wb') as f:
        for chunk in r.iter_content(chunk_size = 8192):# we are difining the chunk size
            f.write(chunk)
```

Loading data on databricks file system (dbfs)

```
# we are writing a function for get data
# # we will split the url on the basis of (/) slashes . and it will take the last element from the array of file splitted
# rest we have performed the same function that we have done in cmd 7
def get_data(url:str):
    filename = url.split('/')[-1]
    with requests.get('https://datahub.io/core/glacier-mass-balance/r/glaciers.csv', stream = True) as r:
        with open("/dbfs/{}".format(filename), 'wb') as f:
            for chunk in r.iter_content(chunk_size = 8192):
                f.write(chunk)
    return filename
```

```
file_name  = get_data('https://datahub.io/core/glacier-mass-balance/r/glaciers.csv')
```

```
file_name
```

```
Out[23]: 'glaciers.csv'
```

Reading data in DataBricks

```
#reding our file system
spark.read.format('csv').option("header","True").load("file:/dbfs/glacier.csv")
```

```
Out[26]: DataFrame[<!doctype html>: string]
```

```
# extracting the file format by splitting the file on (.)
file_format = file_name.split(".")[-1]
```

Reading Data in all the possible formats

```
# we are writing the function to read the files in all the possible format

def read_data(file_name):
    if file_format =='csv':
        df = spark.read.format(file_format).option("header","true").load("file:/dbfs/{}".format(file_name))
    elif file_format == 'json':
        df = spark.read.format(file_format).load("file:/dbfs/{}".format(file_name))
    elif file_format == 'parquet':
         df = spark.read.format(file_format).load("file:/dbfs/{}".format(file_name))
    elif file_format == 'txt':
         df = spark.read.format(file_format).load("file:/dbfs/{}".format(file_name))
    return df
```

```
# calling above function
df = read_data(file_name)
```

Displaying Data

```
# now its time to display our data
display(df)
```

Table

	Year	Mean cumulative mass balance	Number of observations
1	1945	0	null
2	1946	-1.13	1
3	1947	-3.19	1
4	1948	-3.19	1
5	1949	-3.82	3
6	1950	-4.887	3
7	1951	-5.217	3

70 rows

Creating temporary view of our data

```
# we are creating a temporary view of our dataframe so that we can perform some sql operation on that

df.createOrReplaceTempView("df")
```

Starting Sql session

```
%sql
SELECT * from df
```

Table

	Year	Mean cumulative mass balance	Number of observations
1	1945	0	null
2	1946	-1.13	1
3	1947	-3.19	1
4	1948	-3.19	1
5	1949	-3.82	3
6	1950	-4.887	3
7	1951	-5.217	3

70 rows

Creating 2 temporary views of our Data

```
%sql
create or replace temp view nintys as select * from df where year like '19%' order by year asc ;
create or replace temp view twentys as  select * from df where year like '20%' order by year asc ;
```

```
OK
```

```
nintys_df = spark.sql("select * from nintys")
twentys_df = spark.sql("select * from twentys")
```

Transforming Data

```
# we are creating a function to perform the transformation on our data

def transform_data( df: DataFrame):
    spark.sql("create or replace temp view nintys as select * from df where year like '19%' order by year asc;")
    nintys_df = spark.sql("select * from nintys")
    spark.sql("create or replace temp view twentys as select * from df where year like '20%' order by year asc;")
    twentys_df = spark.sql("select * from twentys")
    return nintys_df , twentys_df
```

```
# splittimg our data into vatiables a and b

a,b = transform_data(df)
```

```
# displaying nintys dataset
display(nintys_df)
```

Table

	Year	Mean cumulative mass balance	Number of observations
1	1945	0	null
2	1946	-1.13	1
3	1947	-3.19	1
4	1948	-3.19	1
5	1949	-3.82	3
6	1950	-4.887	3
7	1951	-5.217	3

55 rows

```
# displaying twentys dataset
display(twentys_df)
```

Table

	Year	Mean cumulative mass balance	Number of observations
1	2000	-17.727	37
2	2001	-18.032	37
3	2002	-18.726	37
4	2003	-19.984	37
5	2004	-20.703	37
6	2005	-21.405	37
7	2006	-22.595	37

15 rows

Showing file name according to the dataset

```
 # since our dataset is divided into 2 dataset so i want to show the filename according to the dataset divided

# file name for nintys dataset
 nintys_file_names = spark.sql("(select * from nintys order by Year ASC) union (select * from nintys order by Year DESC
limit 1)")

#file name for twentys dataset
twentys_file_names  = spark.sql("(select * from twentys order by Year ASC) union (select * from twentys order by Year DESC
limit 1)")
```

```
display(nintys_file_names)
```

Table

	Year	Mean cumulative mass balance	Number of observations
1	1962	-9.109	20
2	1973	-10.538	32
3	1960	-8.688	14
4	1957	-6.989	9
5	1967	-9.303	29
6	1950	-4.887	3
7	1977	-10.682	37

55 rows

```
  twentys_file_names_df= twentys_file_names.collect()
  nintys_file_names_df = nintys_file_names.collect()
```

```
# by using getitem we will be appending the first and last dates in order to get the variable name

twentys_file_names_df[0].__getitem__('Year')+ "-"+ twentys_file_names_df[1].__getitem__('Year')

# performing for nintys dataset

nintys_file_names_df[0].__getitem__('Year')+ "-"+ nintys_file_names_df[1].__getitem__('Year')
```

```
Out[70]: '1962-1973'
```

Function for creating file names

```
def create_file_names():
    nintys_file_names = spark.sql("(select * from nintys order by Year ASC) union (select * from nintys order by Year DESC
limit 1)")
    twentys_file_names  = spark.sql("(select * from twentys order by Year ASC) union (select * from twentys order by Year
DESC limit 1)")
    nintys_file_names_df = nintys_file_names.collect()
    twentys_file_names_df= twentys_file_names.collect()

    twentys_file_name  = twentys_file_names_df[0].__getitem__('Year')+ "-"+ twentys_file_names_df[1].__getitem__('Year')

    nintys_file_name  = nintys_file_names_df[0].__getitem__('Year')+ "-"+ nintys_file_names_df[1].__getitem__('Year')
    return nintys_file_name , twentys_file_name
```

```
m , n = create_file_names()
```

```
print(m,n)
```

```
1962-1973 2000-2003
```

writing data in parquet file format

```
# writing our data in parquet format
nintys_df.write.format('parquet').save("/dbfs/nintys_df.parquet")
```

```
#writing our data in parquet fomat
twentys_df.write.format('parquet').save("/dbfs/twentys_df.parquet")
```

```
Out[79]: [FileInfo(path='dbfs:/dbfs/nintys_df.parquet/', name='nintys_df.parquet/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/dbfs/twentys_df.parquet/', name='twentys_df.parquet/', size=0, modificationTime=0)]
```