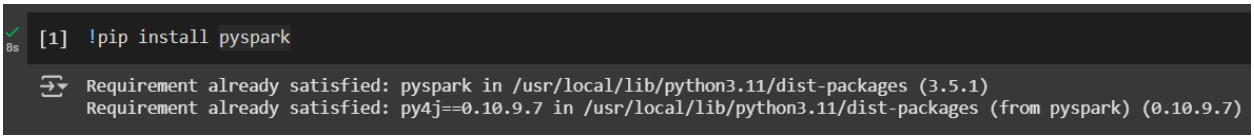
PySpark Documentation – My learning

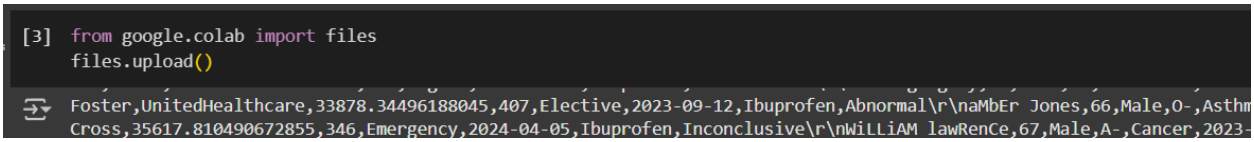
PySpark is the Python API for Apache Spark. It enables real-time, large-scale data processing in a distributed environment using Python and provides a PySpark shell for interactively analyzing your data.PySpark combines Python's learnability and ease of use with the power of Apache Spark to enable data processing and analysis at any size for everyone familiar with Python.

First, we used the command `! pip install pyspark` to the PySpark library using the Python package manager `pip.` The exclamation mark (!) at the beginning tells the notebook environment to run the command as a shell rather than Python code.



Then, we will import PySpark with the command `import pyspark.` This command brings the PySpark module into the Python environment so that its classes and functions can be accessed. PySpark is the Python interface for Apache Spark, allowing developers to write Spark applications using Python code.

The code `from google.colab import files` followed by `files.upload()` is used in Google Colab to upload files from the local computer into the Colab environment. The first line imports the `files` module from Colab's built-in utilities, which provides functions for file operations like upload and download. The `files.upload()` function opens a file browser dialog allowing users to select one or more files from their system.



When working in Google Colab, especially with PySpark, we often **upload datasets** (like CSV or JSON files) from our local computer. But once uploaded, it's not always obvious where they are or if the upload was successful. So we use import, which lets us use Python file system functions and `os.listdir()` check which files are present in the current folder. We use it to avoid

file-not-found errors and make sure they are set up correctly before loading data into Spark.

```
[6]  import os
     print(os.listdir())

['.config', 'healthcare_dataset.csv', 'sample_data']
```

After locating our file in the system, we can perform various data operations on it with the help of Spark, but we have to make a Spark session first, as A **SparkSession** is required to use PySpark functionalities like loading data, running transformations, and executing actions.t's the main **entry point** for working with structured data (like DataFrames). Where:

- `from pyspark.sql import SparkSession` – Imports the `SparkSession` class from PySpark's SQL module. `SparkSession` is the entry point to using DataFrames and SQL in PySpark.
- `SparkSession.builder` – Begins the process of creating a new Spark session.
- `.appName("Health Care")` – Sets the name of your application (helpful for tracking in logs or Spark UI).
- `.getOrCreate()` – Returns an existing Spark session if one exists, otherwise it creates a new one.

```
[4]  from pyspark.sql import SparkSession

     spark = SparkSession.builder.appName("Health Care").getOrCreate()
```

Now, the line of code `df = spark.read.csv("/content/healthcare_dataset.csv", header=True, inferSchema=True)` is used to load a CSV file into a PySpark DataFrame, which is the primary data structure for working with structured data in PySpark. The `header=True` argument ensures that the first row of the file is used as the column names, while `inferSchema=True` automatically detects and assigns the correct data types (such as integer, float, or string) to each column based on the data.

```
[7]  df = spark.read.csv("/content/healthcare_dataset.csv", header=True, inferSchema=True)
     df.show(5)

+-------------+---+------+----------+-----------------+----------------+---------------+--------------------+------------------+------------------+
|         Name|Age|Gender|Blood Type|Medical Condition|Date of Admission|         Doctor|            Hospital|Insurance Provider|    Billing Amount|
+-------------+---+------+----------+-----------------+----------------+---------------+--------------------+------------------+------------------+
|Bobby JacksOn| 30|  Male|        B-|           Cancer|      2024-01-31|  Matthew Smith|     Sons and Miller|       Blue Cross|18856.281305978155|
| LesLie TErRy| 62|  Male|        A+|          Obesity|      2019-08-20|Samantha Davies|            Kim Inc|          Medicare|33643.327286577885|
|  DaNnY sMiTH| 76|Female|        A-|          Obesity|      2022-09-22|Tiffany Mitchell|           Cook PLC|             Aetna|27955.096078842456|
| andrEw waTtS| 28|Female|        O+|         Diabetes|      2020-11-18|    Kevin Wells|Hernandez Rogers ...|          Medicare| 37909.78240987528|
|adrIENNE bEll| 43|Female|       AB+|           Cancer|      2022-09-19| Kathleen Hanna|         White-White|             Aetna|14238.317813937623|
+-------------+---+------+----------+-----------------+----------------+---------------+--------------------+------------------+------------------+
only showing top 5 rows
```

Then we perform some operations like the df.printSchema() method in PySpark is used to **display the structure of a DataFrame**, showing the names of the columns along with their **data types.** It is useful for quickly inspecting the schema of the dataset after it has been loaded,

helping to ensure that data types like integer, string, or float have been correctly inferred during the import process.

```
[8]  df.printSchema()

root
 |-- Name: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Blood Type: string (nullable = true)
 |-- Medical Condition: string (nullable = true)
 |-- Date of Admission: date (nullable = true)
 |-- Doctor: string (nullable = true)
 |-- Hospital: string (nullable = true)
 |-- Insurance Provider: string (nullable = true)
 |-- Billing Amount: double (nullable = true)
 |-- Room Number: integer (nullable = true)
 |-- Admission Type: string (nullable = true)
 |-- Discharge Date: date (nullable = true)
 |-- Medication: string (nullable = true)
 |-- Test Results: string (nullable = true)
```

.

The code `df_filtered = df.filter(df['age'] > 30)` is used to **filter** rows in the DataFrame `df` based on a condition. In this case, the condition is that the `age` column should be greater than 30. The `filter()` function allows you to specify a condition to narrow down the data, and it returns a new DataFrame (`df_filtered`) that only contains the rows that satisfy the condition.

```
[9]  df_filtered = df.filter(df['age'] > 30)
     df_filtered.show()
```

Name	Age	Gender	Blood Type	Medical Condition	Date of Admission	Doctor	Hospital	Insurance Provider	Billing Amount	Room Number	Admission Type
LesLie TErRy	62	Male	A+	Obesity	2019-08-20	Samantha Davies	Kim Inc	Medicare	33643.327286577885	265	Emergency
DaNnY sMiTH	76	Female	A-	Obesity	2022-09-22	Tiffany Mitchell	Cook PLC	Aetna	27955.096078842456	205	Emergency
adrIENNE bELl	43	Female	AB+	Cancer	2022-09-19	Kathleen Hanna	White-White	Aetna	14238.317813937623	458	Urgent
EMILY JOHNSOn	36	Male	A+	Asthma	2023-12-20	Taylor Newton	Nunez-Humphrey	UnitedHealthcare	48145.11095104189	389	Urgent
JASmINe aGuIlaR	82	Male	AB+	Asthma	2020-07-01	Daniel Ferguson	Sons Rich and	Cigna	50119.222791548505	316	Elective
ChRISTopher BerG	58	Female	AB-	Cancer	2021-05-23	Heather Day	Padilla-Walker	UnitedHealthcare	19784.63106221073	249	Elective
mIchElle daniELs	72	Male	O+	Cancer	2020-04-19	John Duncan	Schaefer-Porter	Medicare	12576.795609050234	394	Urgent
aaRon MARtiNeZ	38	Female	A-	Hypertension	2023-08-13	Douglas Mayo	Lyons-Blair	Medicare	7999.586879604188	288	Urgent
connOR HANsEn	75	Female	A+	Diabetes	2019-12-12	Kenneth Fletcher	Powers Miller, an...	Cigna	43282.28335770435	134	Emergency
rObeRt bAuer	68	Female	AB+	Asthma	2020-05-22	Theresa Freeman	Rivera-Gutierrez	UnitedHealthcare	33207.706633729606	309	Urgent
bROOkE brady	44	Female	AB+	Cancer	2021-10-08	Roberta Stewart	Morris-Arellano	UnitedHealthcare	40701.599227308754	182	Urgent
MS. nAtaliE gAMble	46	Female	AB-	Obesity	2023-01-01	Maria Dougherty	Cline-Williams	Blue Cross	12263.357425021362	465	Elective
haley perkins	63	Female	A+	Arthritis	2020-06-23	Erica Spencer	Cervantes-Wells	UnitedHealthcare	24499.847903736576	114	Elective
mRS. jamiE cAMPBELl	38	Male	AB-	Obesity	2020-03-08	Justin Kim	Torres, and Harri...	Cigna	17440.465444124675	449	Urgent
LuKE BuRgEss	34	Female	A-	Hypertension	2021-03-04	Justin Moore Jr.	Houston PLC	Blue Cross	18843.02301783416	260	Elective
dANIEL schmIdt	63	Male	B+	Asthma	2022-11-15	Denise Galloway	Hammond Ltd	Cigna	23762.203579059587	465	Elective
tIMOTHY burNs	67	Female	A-	Asthma	2023-06-28	Krista Smith	Jones LLC	Blue Cross	42.5145885533243	115	Elective
ChRISTOPHEr BRiGhT	48	Male	B+	Asthma	2020-01-21	Gregory Smith	Williams-Davis	Aetna	17695.911622343818	295	Urgent
KatHRYn StewArt	58	Female	O+	Arthritis	2022-05-12	Vanessa Newton	Clark-Mayo	Aetna	5998.10290819591	327	Urgent
dR. EilEEn thomPsoN	59	Male	A+	Asthma	2021-08-02	Donna Martinez MD	and Sons Smith	Aetna	25250.052428216135	119	Urgent

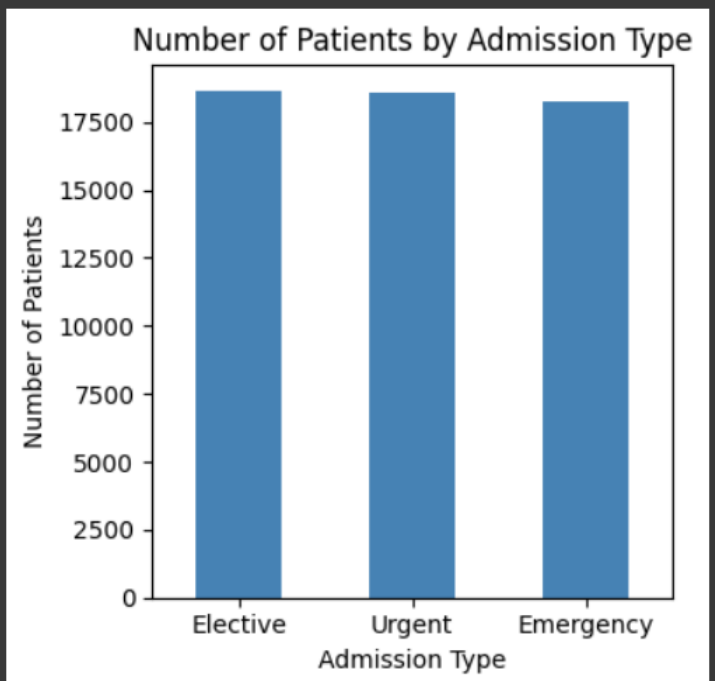
only showing top 20 rows

For graphs, we can use various Python libraries like Matplotlib. The line `import matplotlib.pyplot as plt` imports the `matplotlib` library, specifically the `pyplot` module, which is commonly used for creating visualizations like charts and graphs in Python.

This module provides functions to plot various types of graphs, such as line plots, bar charts, and histograms, which help in the **visual analysis** of data. The next line, `pdf = df.toPandas()`, converts the PySpark DataFrame `df` into a **Pandas DataFrame**.

```
[10] import matplotlib.pyplot as plt
     pdf = df.toPandas()
```

```
plt.figure(figsize=(4,4))
admission_counts.plot(kind='bar', color='steelblue')
plt.title('Number of Patients by Admission Type')
plt.xlabel('Admission Type')
plt.ylabel('Number of Patients')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



And we can use another visualization library called Seaborn, which is built on top of `matplotlib` and is known for creating visually appealing and informative statistical graphics with minimal code. The second line imports `matplotlib.pyplot` as `plt`, a core plotting library used for creating a variety of static, animated, and interactive plots. PySpark's DataFrame is not directly compatible with these visualization libraries
. The third line `pdf = df.toPandas()` is used to **convert the PySpark DataFrame into a Pandas DataFrame**.

```
plt.figure(figsize=(6,4))
sns.boxplot(x='Medical Condition', y='Billing Amount', data=pdf)
plt.title('Billing Amount Distribution by Medical Condition')
plt.xticks(rotation=45)
plt.xlabel('Medical Condition')
plt.ylabel('Billing Amount')
plt.tight_layout()
plt.show()
```