PySpark Documentation – My learning

Summary

This document describes my practical PySpark learning and implementation experience using Google Colab. I developed a PySpark pipeline for a healthcare dataset that includes data transformation, cleaning, and visualization. I also looked into how to export the finished DataFrame and how to upload it to a SQL database.

Introduction to PySpark

PySpark is the Python API for Apache Spark. It enables you to perform real-time, large-scale data processing in a distributed environment using Python. It also 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 processing and analysis of data at any size for everyone familiar with Python.

PySpark supports all of Spark's features such as Spark SQL, DataFrames, Structured Streaming, Machine Learning (MLlib) and Spark Core.

I used PySpark in Google Colab, which eliminates the need for a local setup and enables Spark code to run in a cloud environment.

Environment Setup in Google Colab

To work with PySpark in Colab:

```
!pip install pyspark

!pip install findspark
```

These packages allow PySpark to be initialized and run in the Colab notebook environment.

Libraries Used

```
import pyspark

import pandas as pd

import plotly.express as px

from google.colab import files

from pyspark.sql import SparkSession

from pyspark.sql.functions import datediff
```

Dataset Handling

1. **Uploading Dataset:**

uploaded = files.upload()

2. **Reading Data with Pandas (for preview):**

pd.read_csv('healthcare_dataset.csv')

3. **Spark Session Initialization:**

spark = SparkSession.builder.appName("Practice").getOrCreate()

4. **Reading CSV using PySpark:**

df = spark.read.option('header', 'true').csv('healthcare_dataset.csv', inferSchema=True)

df.printSchema()

df.describe().show()

Data Cleaning and Transformation

1. **Drop Unnecessary Columns:**

df = df.drop('Name', 'Doctor', 'Room Number')

2. **Derived Column - Length of Stay:**

df = df.withColumn("Length of Stay", datediff("Discharge Date", "Date of Admission"))

df.show()

Data Visualization

Using Plotly for interactive visualization:

px.pie(df.toPandas(), names='Blood Type', title='Distribution of Blood Group')

Exporting Cleaned Data to CSV (for SQL Upload)

1. **Write Data into Single File:**

df.coalesce(1).write.csv('cleaned_health-care-single', header=True, mode='overwrite')

2. **Zip the Output Folder:**

!zip -r output_csv_single.zip cleaned_health-care-single

3. **Download to Local System:**

```
files.download("output_csv_single.zip")
```

Conclusion

Through this project, I learned how to:

- Install and utilize PySpark in a virtual setting (Googe Colab).
- Analyze and purify big datasets.
- Use the PySpark APIs to carry out transformation operations.
- Use Plotly to visualize data.

I gained confidence working with large-scale data pipelines and transferring data between Spark and SQL-based systems with ease thanks to this hands-on exercise.