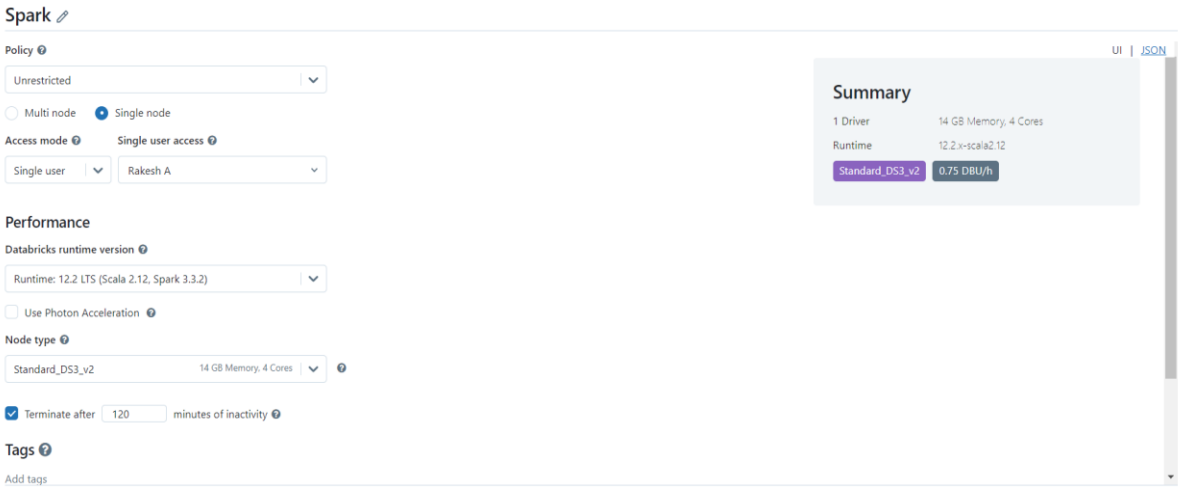
Manual Schema Vs Infer Schema

Data:

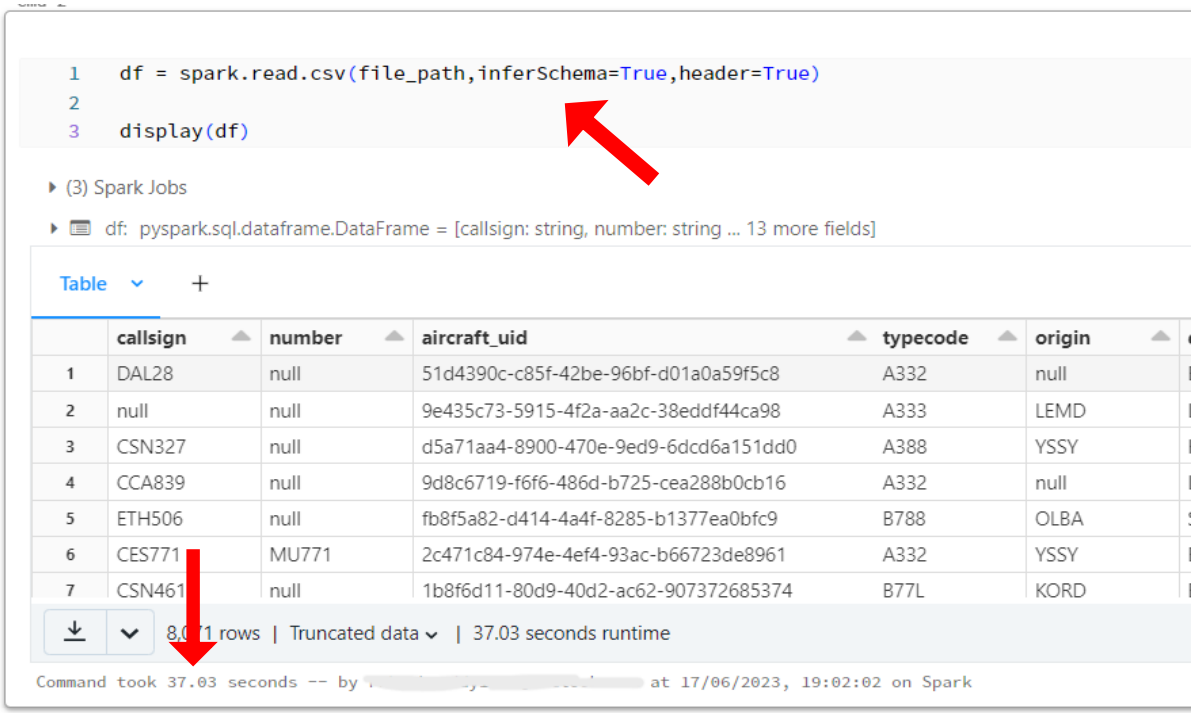
Air Traffic Dataset
Size: Around 2GB
Format: CSV

Spark Cluster Configuration:



Create a dataframe df using inferSchema

As you see we used “inferSchema=True” as parameter while reading creating the dataframe.



```
df = spark.read.csv(file_path,inferSchema=True,header=True)

display(df)
```

(3) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [callsign: string, number: string ... 13 more fields]

Table +

	callsign	number	aircraft_uid	typecode	origin
1	DAL28	null	51d4390c-c85f-42be-96bf-d01a0a59f5c8	A332	null
2	null	null	9e435c73-5915-4f2a-aa2c-38eddf44ca98	A333	LEMD
3	CSN327	null	d5a71aa4-8900-470e-9ed9-6dcd6a151dd0	A388	YSSY
4	CCA839	null	9d8c6719-f6f6-486d-b725-cea288b0cb16	A332	null
5	ETH506	null	fb8f5a82-d414-4a4f-8285-b1377ea0bfc9	B788	OLBA
6	CES771	MU771	2c471c84-974e-4ef4-93ac-b66723de8961	A332	YSSY
7	CSN461	null	1b8f6d11-80d9-40d2-ac62-907372685374	B77L	KORD

8,071 rows | Truncated data | 37.03 seconds runtime

Command took 37.03 seconds -- by [redacted] at 17/06/2023, 19:02:02 on Spark

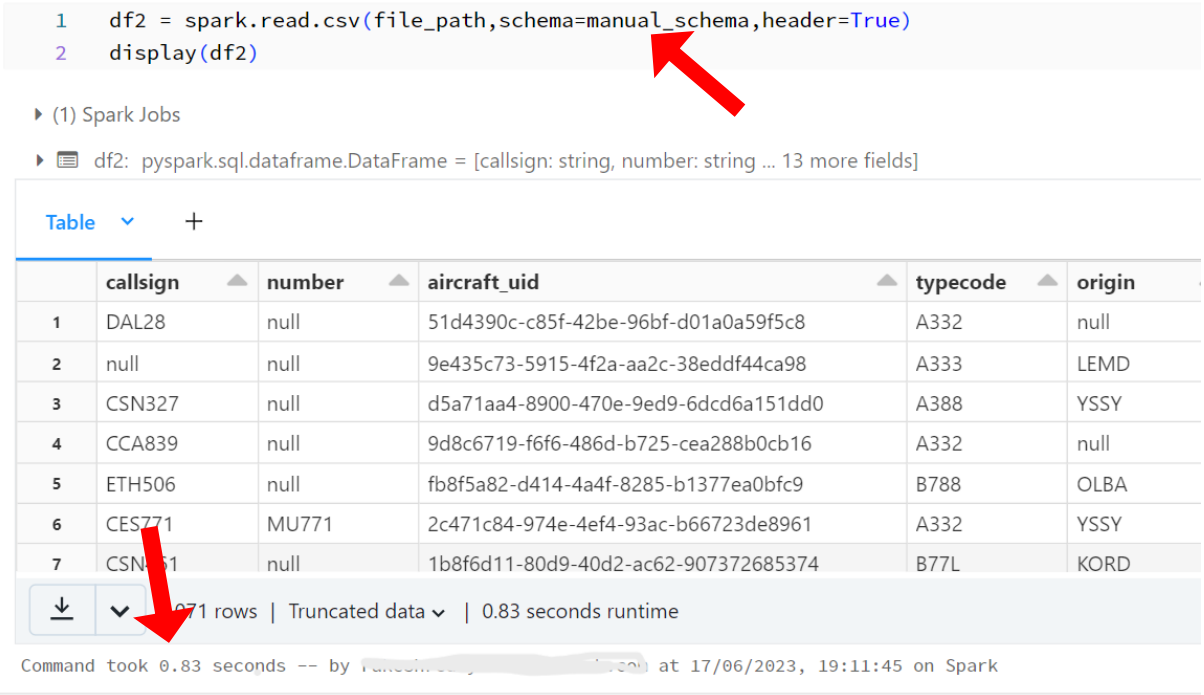
And it took 37.03 seconds to read the data.

Create a manual schema

```
from pyspark.sql.types import StructType, StructField, StringType, DoubleType, TimestampType


manual_schema = StructType([
    StructField("callsign",StringType(),True),
    StructField("number",StringType(),True),
    StructField("aircraft_uid",StringType(),True),
    StructField("typecode",StringType(),True),
    StructField("origin",StringType(),True),
    StructField("destination",StringType(),True),
    StructField("firstseen",TimestampType(),True),
    StructField("lastseen",TimestampType(),True),
    StructField("day",TimestampType(),True),
    StructField("latitude_1",DoubleType(),True),
    StructField("longitude_1",DoubleType(),True),
    StructField("altitude_1",DoubleType(),True),
    StructField("latitude_2",DoubleType(),True),
    StructField("longitude_2",DoubleType(),True),
    StructField("altitude_2",DoubleType(),True),
]
)
```

Using the above manual schema as parameter, create a new dataframe.



As you can observe there is a drastic change in the reading time.

Conclusion:

Here's what happens when you use `inferSchema`:

1. **Spark reads a sample of the data**: When you load a file into Spark using `inferSchema`, it reads a configurable number of rows (by default, the first 1000 rows) to analyze and infer the schema.

2. **Schema inference process**: Spark examines the values in each column of the sampled data to make educated guesses about the data types. It performs various checks, such as checking the data's format and patterns, to determine the most appropriate data type for each column.

3. **Schema creation**: Based on the analysis of the sample data, Spark constructs a schema that defines the data types of each column. The schema represents the structure of the data, including column names and their associated data types.

4. **Data loading**: Once the schema has been inferred, Spark loads the rest of the data from the file, using the inferred schema to parse and type-cast the values accordingly. This allows Spark to efficiently process the data based on the inferred schema.

All the above steps can be reduced if we provide a manual schema.

While manual schema can have its advantages in certain scenarios, it is not always preferred in Spark. Here are a few reasons why manual schema may not be the preferred approach:

1. Flexibility: In some cases, the structure of the data may evolve over time or vary across different datasets. Using manual schema requires you to define the schema explicitly, which can become cumbersome and less flexible when dealing with evolving or diverse data sources.

2. Schema Evolution: When the schema of the data changes, manually maintaining and updating the schema can be time-consuming and error-prone. In contrast, inferSchema can automatically adapt to changes in the data structure without requiring manual modifications.

3.Time and Effort: Constructing and maintaining a manual schema can be a time-consuming task, especially for large datasets with numerous columns. InferSchema saves time by automatically inferring the schema from the data, eliminating the need for manual schema definition.

4.Data Exploration: When initially exploring new datasets, inferSchema allows for quick data analysis without the need to define a schema manually. It provides a convenient way to understand the data structure and make informed decisions regarding subsequent data processing steps.

5.Data Source Compatibility: Different data sources may have varying schema requirements or limitations. Using manual schema for each different data source can be impractical. InferSchema, on the other hand, can adapt to different data sources, making it more compatible and versatile.

So, It's important to note that the suitability of manual schema versus inferSchema depends on the specific use case and the nature of the data being processed. Considering factors such as data variability, maintenance efforts, and adaptability can help determine the preferred approach in each scenario.

Dataset Link:
https://www.kaggle.com/datasets/ishadss/covid19-period-airtraffic-dataset