ASSIGNMENT SOLUTION

The following Common Boilerplate code to create a Spark Session has to be executed before running the queries.

from pyspark.sql import SparkSession

from pyspark.sql import SparkSession

import getpass

username = getpass.getuser()

spark= SparkSession. \

builder. \

config('spark.ui.port','0'). \

config("spark.sql.warehouse.dir", f"/user/{username}/warehouse"). \

enableHiveSupport(). \

master('yarn'). \

getOrCreate()

[1]:
```
from pyspark.sql import SparkSession
import getpass
username = getpass.getuser()
spark= SparkSession. \
builder. \
config('spark.ui.port','0'). \
config("spark.sql.warehouse.dir", f"/user/{username}/warehouse"). \
enableHiveSupport(). \
master('yarn'). \
getOrCreate()
```

[2]:
```
spark
```

[2]: **SparkSession - hive**

SparkContext

Spark UI

Version	`v3.0.1`
Master	`yarn`
AppName	`pyspark-shell`

Question 1

Create a DataFrame using the sample data and schema

df = spark.createDataFrame(data, schema=["season", "windspeed"])

Print the schema of the DataFrame

df.printSchema()

Show the contents of the DataFrame

df.show()

[4]:
```
data = [("Spring", 12.3),
("Summer", 10.5),
("Autumn", 8.2),
("Winter", 15.1)]
```

[5]:
```
df = spark.createDataFrame(data, schema=["season", "windspeed"])
```

[6]:
```
df.printSchema()
```

```
root
 |-- season: string (nullable = true)
 |-- windspeed: double (nullable = true)
```

[7]:
```
df.show()
```

```
+------+---------+
|season|windspeed|
+------+---------+
|Spring|     12.3|
|Summer|     10.5|
|Autumn|      8.2|
|Winter|     15.1|
+------+---------+
```

Question 2

ArrayType():

In the provided schema, the ArrayType is used to define a field that can contain multiple values of the same data type, while StructType is used to define a field that contains a structured collection of fields with different data types.

In the case of the books field, it is an array because a library can have multiple books. Each book is represented by a struct with fields like book_id, book_name, author, and copies_available. By using ArrayType(StructType([...])), we can define that the books field is an array containing multiple struct objects representing different books.

Similarly, the members field is also an array because a library can have multiple members, and each member can borrow multiple books.

Each member is represented by a struct with fields like member_id, member_name, age, and books_borrowed. The books_borrowed field is again an array of book IDs that the member has borrowed.

If we were to use StructType instead of ArrayType, it would imply that there can be only one book or member, which is not the case for these fields.

Therefore, using ArrayType(StructType([...])) allows us to define a collection of books and members within the library schema.

```
from pyspark.sql.types import *

schema = StructType([
    StructField("library_name", StringType()),
    StructField("location", StringType()),
    StructField("books", ArrayType(
        StructType([
            StructField("book_id", StringType()),
            StructField("book_name", StringType()),
            StructField("author", StringType()),
            StructField("copies_available", IntegerType())
        ])
    )),
    StructField("members", ArrayType(
        StructType([
            StructField("member_id", StringType()),
            StructField("member_name", StringType()),
            StructField("age", IntegerType()),
            StructField("books_borrowed", ArrayType(StringType()))
        ])
    ))
])

library_df =
spark.read.schema(schema).json("/public/trendytech/datasets/library_data.jso
n")
```

```
from pyspark.sql.types import *
```

```
schema = StructType([
    StructField("library_name", StringType()),
    StructField("location", StringType()),
    StructField("books", ArrayType(
        StructType([
            StructField("book_id", StringType()),
            StructField("book_name", StringType()),
            StructField("author", StringType()),
            StructField("copies_available", IntegerType())
        ])
    )),
    StructField("members", ArrayType(
        StructType([
            StructField("member_id", StringType()),
            StructField("member_name", StringType()),
            StructField("age", IntegerType()),
            StructField("books_borrowed", ArrayType(StringType()))
        ])
    ))
])
```

library_df.printSchema()

```
library_df = spark.read.schema(schema).json("/public/trendytech/datasets/library_data.json")
```

```
library_df.printSchema()
```

```
root
 |-- library_name: string (nullable = true)
 |-- location: string (nullable = true)
 |-- books: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- book_id: string (nullable = true)
 |    |    |-- book_name: string (nullable = true)
 |    |    |-- author: string (nullable = true)
 |    |    |-- copies_available: integer (nullable = true)
 |-- members: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- member_id: string (nullable = true)
 |    |    |-- member_name: string (nullable = true)
 |    |    |-- age: integer (nullable = true)
 |    |    |-- books_borrowed: array (nullable = true)
 |    |    |    |-- element: string (containsNull = true)
```

Question 3

a) train_df=spark.read \

.format("csv") \

.option("header","true") \

.option("inferSchema","true") \

.load("/public/trendytech/datasets/train.csv")

```
train_df=spark.read \
            .format("csv") \
            .option("header","true") \
            .option("inferSchema","true") \
            .load("/public/trendytech/datasets/train.csv")
```

```
train_df.printSchema()
```

```
root
 |-- train_number: integer (nullable = true)
 |-- train_name: string (nullable = true)
 |-- seats_available: integer (nullable = true)
 |-- passenger_name: string (nullable = true)
 |-- age: integer (nullable = true)
 |-- ticket_number: string (nullable = true)
 |-- seat_number: string (nullable = true)
```

dropped_df = train_df.drop("passenger_name", "age")

dropped_df.show()

```
dropped_df = train_df.drop("passenger_name", "age")
```

```
dropped_df.show()
```

```
+------------+----------+---------------+-------------+-----------+
|train_number|train_name|seats_available|ticket_number|seat_number|
+------------+----------+---------------+-------------+-----------+
|         123|   Express|            100|         T123|         A1|
|         123|   Express|            100|         T124|         B2|
|         456| Superfast|            150|         T125|         C3|
|         456| Superfast|            150|         T126|         D4|
|         789|     Local|             50|         T127|         E5|
|         789|     Local|             50|         T128|         F6|
|         789|     Local|             50|         T129|         G7|
+------------+----------+---------------+-------------+-----------+
```

b) df = dropped_df.dropDuplicates(["train_number", "ticket_number"])

num_rows = df.count()

print("Number of rows after removing duplicates:", num_rows)

```
df = dropped_df.dropDuplicates(["train_number", "ticket_number"])
num_rows = df.count()
print("Number of rows after removing duplicates:", num_rows)
```

```
Number of rows after removing duplicates: 7
```

c) distinct_departments = df.select("train_name").distinct()

num_departments = distinct_departments.count()

print("Number of unique train names:", num_departments)

```
distinct_departments = df.select("train_name").distinct()
num_departments = distinct_departments.count()
print("Number of unique train names:", num_departments)
```

```
Number of unique train names: 3
```

Question 4

1.

schema="store_id integer,product string,quantity integer,revenue double"

df_permissive = spark.read.schema(schema).option("mode","permissive").json("/public/trendytech/datasets/sales_data.json")

num_records_permissive = df_permissive.count()

print("Number of records read:", num_records_permissive)

```
schema="store_id integer,product string,quantity integer,revenue double"
df_permissive = spark.read.schema(schema).option("mode","permissive").json("/public/trendytech/datasets/sales_data.json")
num_records_permissive = df_permissive.count()
print("Number of records read:", num_records_permissive)
```

```
Number of records read: 22
```

2.

df_dropmalformed = spark.read.option("mode", "dropmalformed").schema(schema).json("/public/trendytech/datasets/sales_data.json")

df_dropmalformed.show()

num_records_dropmalformed = df_dropmalformed.count()

num_corrupt_records_dropmalformed = num_records_permissive - num_records_dropmalformed

print("Number of records read:", num_records_dropmalformed)

print("Number of dropped malformed records:", num_corrupt_records_dropmalformed)

```
df_dropmalformed = spark.read.option("mode", "dropmalformed").schema(schema).json("/public/trendytech/datasets/sales_data.json")
df_dropmalformed.show()
num_records_dropmalformed = df_dropmalformed.count()
num_corrupt_records_dropmalformed = num_records_permissive - num_records_dropmalformed
print("Number of records read:", num_records_dropmalformed)
print("Number of dropped malformed records:", num_corrupt_records_dropmalformed)
```

```
+--------+----------+--------+-------+
|store_id|   product|quantity|revenue|
+--------+----------+--------+-------+
|       1|     Apple|      10|  100.0|
|       2|    Banana|      15|   75.0|
|       3|    Orange|      12|   90.0|
|       4|     Mango|       8|  120.0|
|       5|     Grape|      20|  150.0|
|       6|Watermelon|       5|   50.0|
|       7|Strawberry|      18|  108.0|
|       8| Pineapple|      14|  140.0|
|       9|    Cherry|       7|  105.0|
|      10|      Pear|       9|   81.0|
|      11| Blueberry|      11|   88.0|
|      12|      Kiwi|      16|  128.0|
|      13|     Peach|      13|   91.0|
|      14|      Plum|       6|   54.0|
|      15|     Lemon|      10|   70.0|
|      16| Raspberry|      17|  136.0|
|      17|   Coconut|       4|   80.0|
|      18|   Avocado|      11|   99.0|
|      19|Blackberry|       8|   64.0|
+--------+----------+--------+-------+

Number of records read: 21
Number of dropped malformed records: 1
```

3.

df_failfast = spark.read.option("mode","failfast").schema(schema).json("/user/itv005357/sales_data.json")

df_failfast.show()

```
df_failfast = spark.read.option("mode","failfast").schema(schema).json("/public/trendytech/datasets/sales_data.json")
df_failfast.show()
```

```
---------------------------------------------------------------------------
Py4JJavaError                             Traceback (most recent call last)
<ipython-input-34-b722af7f5e5f> in <module>
      1 df_failfast = spark.read.option("mode","failfast").schema(schema).json("/public/trendytech/datasets/sales_data.json")
----> 2 df_failfast.show()

/opt/spark-3.0.1-bin-hadoop3.2/python/pyspark/sql/dataframe.py in show(self, n, truncate, vertical)
    438         """
    439         if isinstance(truncate, bool) and truncate:
--> 440             print(self._jdf.showString(n, 20, vertical))
    441         else:
    442             print(self._jdf.showString(n, int(truncate), vertical))

/opt/spark-3.0.1-bin-hadoop3.2/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in __call__(self, *args)
   1303         answer = self.gateway_client.send_command(command)
   1304         return_value = get_return_value(
-> 1305             answer, self.gateway_client, self.target_id, self.name)
   1306
   1307         for temp_arg in temp_args:

/opt/spark-3.0.1-bin-hadoop3.2/python/pyspark/sql/utils.py in deco(*a, **kw)
    126     def deco(*a, **kw):
    127         try:
--> 128             return f(*a, **kw)
    129         except py4j.protocol.Py4JJavaError as e:
    130             converted = convert_exception(e.java_exception)
```

```
Py4JJavaError: An error occurred while calling o220.showString.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 18.0 failed 4 times, most recent failure: Lost task 0.3 i
stage 18.0 (TID 621, w02.itversity.com, executor 1): org.apache.spark.SparkException: Malformed records are detected in record parsing. Parse
ode: FAILFAST. To process malformed records as null result, try setting the option 'mode' as 'PERMISSIVE'.
        at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:70)
```

Question 5

1.

```
schema='patient_id integer,admission_date date,discharge_date
date,diagnosis string,doctor_id integer,total_cost float'


hosp_df=spark.read \
        .format("csv") \
        .option("header","true") \
        .schema(schema) \
        .option("dateFormat","MM-dd-yyyy") \
        .load("/public/trendytech/datasets/hospital.csv")


hospital_df = hosp_df.drop("doctor_id")
hospital_df.show()
```

```
schema='patient_id integer,admission_date date,discharge_date date,diagnosis string,doctor_id integer,total_cost float'
```

```
hosp_df=spark.read \
            .format("csv") \
            .option("header","true") \
            .schema(schema) \
            .option("dateFormat","MM-dd-yyyy") \
            .load("/public/trendytech/datasets/hospital.csv")
```

```
hospital_df = hosp_df.drop("doctor_id")
hospital_df.show()
```

```
+----------+--------------+--------------+-------------+----------+
|patient_id|admission_date|discharge_date|    diagnosis|total_cost|
+----------+--------------+--------------+-------------+----------+
|         1|    2022-01-01|    2022-01-10|    Pneumonia|    5000.0|
|         2|    2022-02-05|    2022-02-09| Appendicitis|    7000.0|
|         3|    2022-03-12|    2022-03-18|Fractured Arm|    3500.0|
|         4|    2022-04-02|    2022-04-08| Heart Attack|   15000.0|
|         5|    2022-05-05|    2022-05-07|    Influenza|    2500.0|
|         6|    2022-06-10|    2022-06-15| Appendicitis|    8000.0|
|         7|    2022-07-20|    2022-07-25|    Pneumonia|    5500.0|
|         8|    2022-08-25|    2022-09-01| Heart Attack|   20000.0|
|         9|    2022-09-15|    2022-09-22|Fractured Leg|    6000.0|
|        10|    2022-10-05|    2022-10-10| Appendicitis|    7500.0|
|        11|    2022-11-02|    2022-11-05|    Influenza|    2800.0|
|        12|    2022-12-10|    2022-12-18|    Pneumonia|    6000.0|
|        13|    2023-01-02|    2023-01-09| Heart Attack|   18000.0|
|        14|    2023-02-14|    2023-02-18| Appendicitis|    7200.0|
|        15|    2023-03-20|    2023-03-28|Fractured Arm|    3800.0|
|        16|    2023-04-05|    2023-04-11|    Influenza|    2700.0|
|        17|    2023-05-08|    2023-05-11| Heart Attack|   16000.0|
|        18|    2023-06-15|    2023-06-20|    Pneumonia|    4800.0|
|        19|    2023-07-22|    2023-07-27|Fractured Leg|    6500.0|
|        20|    2023-08-10|    2023-08-16| Appendicitis|    7800.0|
+----------+--------------+--------------+-------------+----------+
only showing top 20 rows
```

2.

```
hospital_new_df = hospital_df.withColumnRenamed("total_cost", "hospital_bill")

hospital_new_df.show()
```

```
hospital_new_df = hospital_df.withColumnRenamed("total_cost", "hospital_bill")
hospital_new_df.show()
```

```
+----------+--------------+--------------+-------------+-------------+
|patient_id|admission_date|discharge_date|    diagnosis|hospital_bill|
+----------+--------------+--------------+-------------+-------------+
|         1|    2022-01-01|    2022-01-10|    Pneumonia|       5000.0|
|         2|    2022-02-05|    2022-02-09| Appendicitis|       7000.0|
|         3|    2022-03-12|    2022-03-18|Fractured Arm|       3500.0|
|         4|    2022-04-02|    2022-04-08| Heart Attack|      15000.0|
|         5|    2022-05-05|    2022-05-07|    Influenza|       2500.0|
|         6|    2022-06-10|    2022-06-15| Appendicitis|       8000.0|
|         7|    2022-07-20|    2022-07-25|    Pneumonia|       5500.0|
|         8|    2022-08-25|    2022-09-01| Heart Attack|      20000.0|
|         9|    2022-09-15|    2022-09-22|Fractured Leg|       6000.0|
|        10|    2022-10-05|    2022-10-10| Appendicitis|       7500.0|
|        11|    2022-11-02|    2022-11-05|    Influenza|       2800.0|
|        12|    2022-12-10|    2022-12-18|    Pneumonia|       6000.0|
|        13|    2023-01-02|    2023-01-09| Heart Attack|      18000.0|
|        14|    2023-02-14|    2023-02-18| Appendicitis|       7200.0|
|        15|    2023-03-20|    2023-03-28|Fractured Arm|       3800.0|
|        16|    2023-04-05|    2023-04-11|    Influenza|       2700.0|
|        17|    2023-05-08|    2023-05-11| Heart Attack|      16000.0|
|        18|    2023-06-15|    2023-06-20|    Pneumonia|       4800.0|
|        19|    2023-07-22|    2023-07-27|Fractured Leg|       6500.0|
|        20|    2023-08-10|    2023-08-16| Appendicitis|       7800.0|
+----------+--------------+--------------+-------------+-------------+
only showing top 20 rows
```

3.

```
from pyspark.sql.functions import expr

hospital_expr_df = hospital_new_df.withColumn("duration_of_stay", expr("datediff(discharge_date, admission_date)"))

hospital_expr_df.show()
```

```
from pyspark.sql.functions import expr
hospital_expr_df = hospital_new_df.withColumn("duration_of_stay", expr("datediff(discharge_date, admission_date)"))
hospital_expr_df.show()
```

```
+----------+--------------+--------------+-------------+-------------+----------------+
|patient_id|admission_date|discharge_date|    diagnosis|hospital_bill|duration_of_stay|
+----------+--------------+--------------+-------------+-------------+----------------+
|         1|    2022-01-01|    2022-01-10|    Pneumonia|       5000.0|               9|
|         2|    2022-02-05|    2022-02-09| Appendicitis|       7000.0|               4|
|         3|    2022-03-12|    2022-03-18|Fractured Arm|       3500.0|               6|
|         4|    2022-04-02|    2022-04-08| Heart Attack|      15000.0|               6|
|         5|    2022-05-05|    2022-05-07|    Influenza|       2500.0|               2|
|         6|    2022-06-10|    2022-06-15| Appendicitis|       8000.0|               5|
|         7|    2022-07-20|    2022-07-25|    Pneumonia|       5500.0|               5|
|         8|    2022-08-25|    2022-09-01| Heart Attack|      20000.0|               7|
|         9|    2022-09-15|    2022-09-22|Fractured Leg|       6000.0|               7|
|        10|    2022-10-05|    2022-10-10| Appendicitis|       7500.0|               5|
|        11|    2022-11-02|    2022-11-05|    Influenza|       2800.0|               3|
|        12|    2022-12-10|    2022-12-18|    Pneumonia|       6000.0|               8|
|        13|    2023-01-02|    2023-01-09| Heart Attack|      18000.0|               7|
|        14|    2023-02-14|    2023-02-18| Appendicitis|       7200.0|               4|
|        15|    2023-03-20|    2023-03-28|Fractured Arm|       3800.0|               8|
|        16|    2023-04-05|    2023-04-11|    Influenza|       2700.0|               6|
|        17|    2023-05-08|    2023-05-11| Heart Attack|      16000.0|               3|
|        18|    2023-06-15|    2023-06-20|    Pneumonia|       4800.0|               5|
|        19|    2023-07-22|    2023-07-27|Fractured Leg|       6500.0|               5|
|        20|    2023-08-10|    2023-08-16| Appendicitis|       7800.0|               6|
+----------+--------------+--------------+-------------+-------------+----------------+
only showing top 20 rows
```

4.

```
hospital_price_df = hospital_expr_df.withColumn("adjusted_total_cost", expr("CASE WHEN diagnosis LIKE 'Heart Attack' THEN hospital_bill * 1.5 WHEN diagnosis LIKE 'Appendicitis' THEN hospital_bill * 1.2 ELSE hospital_bill END"))

hospital_price_df.show()
```

```
+----------+--------------+--------------+-------------+-------------+----------------+-------------------+
|patient_id|admission_date|discharge_date|    diagnosis|hospital_bill|duration_of_stay|adjusted_total_cost|
+----------+--------------+--------------+-------------+-------------+----------------+-------------------+
|         1|    2022-01-01|    2022-01-10|    Pneumonia|       5000.0|               9|             5000.0|
|         2|    2022-02-05|    2022-02-09| Appendicitis|       7000.0|               4|             8400.0|
|         3|    2022-03-12|    2022-03-18|Fractured Arm|       3500.0|               6|             3500.0|
|         4|    2022-04-02|    2022-04-08| Heart Attack|      15000.0|               6|            22500.0|
|         5|    2022-05-05|    2022-05-07|    Influenza|       2500.0|               2|             2500.0|
|         6|    2022-06-10|    2022-06-15| Appendicitis|       8000.0|               5|             9600.0|
|         7|    2022-07-20|    2022-07-25|    Pneumonia|       5500.0|               5|             5500.0|
|         8|    2022-08-25|    2022-09-01| Heart Attack|      20000.0|               7|            30000.0|
|         9|    2022-09-15|    2022-09-22|Fractured Leg|       6000.0|               7|             6000.0|
|        10|    2022-10-05|    2022-10-10| Appendicitis|       7500.0|               5|             9000.0|
|        11|    2022-11-02|    2022-11-05|    Influenza|       2800.0|               3|             2800.0|
|        12|    2022-12-10|    2022-12-18|    Pneumonia|       6000.0|               8|             6000.0|
|        13|    2023-01-02|    2023-01-09| Heart Attack|      18000.0|               7|            27000.0|
|        14|    2023-02-14|    2023-02-18| Appendicitis|       7200.0|               4|             8640.0|
|        15|    2023-03-20|    2023-03-28|Fractured Arm|       3800.0|               8|             3800.0|
|        16|    2023-04-05|    2023-04-11|    Influenza|       2700.0|               6|             2700.0|
|        17|    2023-05-08|    2023-05-11| Heart Attack|      16000.0|               3|            24000.0|
|        18|    2023-06-15|    2023-06-20|    Pneumonia|       4800.0|               5|             4800.0|
|        19|    2023-07-22|    2023-07-27|Fractured Leg|       6500.0|               5|             6500.0|
|        20|    2023-08-10|    2023-08-16| Appendicitis|       7800.0|               6|             9360.0|
+----------+--------------+--------------+-------------+-------------+----------------+-------------------+
only showing top 20 rows
```

5.

hospital_final_df = hospital_price_df.select("patient_id", "diagnosis", "hospital_bill", "adjusted_total_cost")

hospital_final_df.show()

```
hospital_final_df = hospital_price_df.select("patient_id", "diagnosis", "hospital_bill", "adjusted_total_cost")
hospital_final_df.show()
```

```
+----------+-------------+-------------+-------------------+
|patient_id|    diagnosis|hospital_bill|adjusted_total_cost|
+----------+-------------+-------------+-------------------+
|         1|    Pneumonia|       5000.0|             5000.0|
|         2| Appendicitis|       7000.0|             8400.0|
|         3|Fractured Arm|       3500.0|             3500.0|
|         4| Heart Attack|      15000.0|            22500.0|
|         5|    Influenza|       2500.0|             2500.0|
|         6| Appendicitis|       8000.0|             9600.0|
|         7|    Pneumonia|       5500.0|             5500.0|
|         8| Heart Attack|      20000.0|            30000.0|
|         9|Fractured Leg|       6000.0|             6000.0|
|        10| Appendicitis|       7500.0|             9000.0|
|        11|    Influenza|       2800.0|             2800.0|
|        12|    Pneumonia|       6000.0|             6000.0|
|        13| Heart Attack|      18000.0|            27000.0|
|        14| Appendicitis|       7200.0|             8640.0|
|        15|Fractured Arm|       3800.0|             3800.0|
|        16|    Influenza|       2700.0|             2700.0|
|        17| Heart Attack|      16000.0|            24000.0|
|        18|    Pneumonia|       4800.0|             4800.0|
|        19|Fractured Leg|       6500.0|             6500.0|
|        20| Appendicitis|       7800.0|             9360.0|
+----------+-------------+-------------+-------------------+
only showing top 20 rows
```