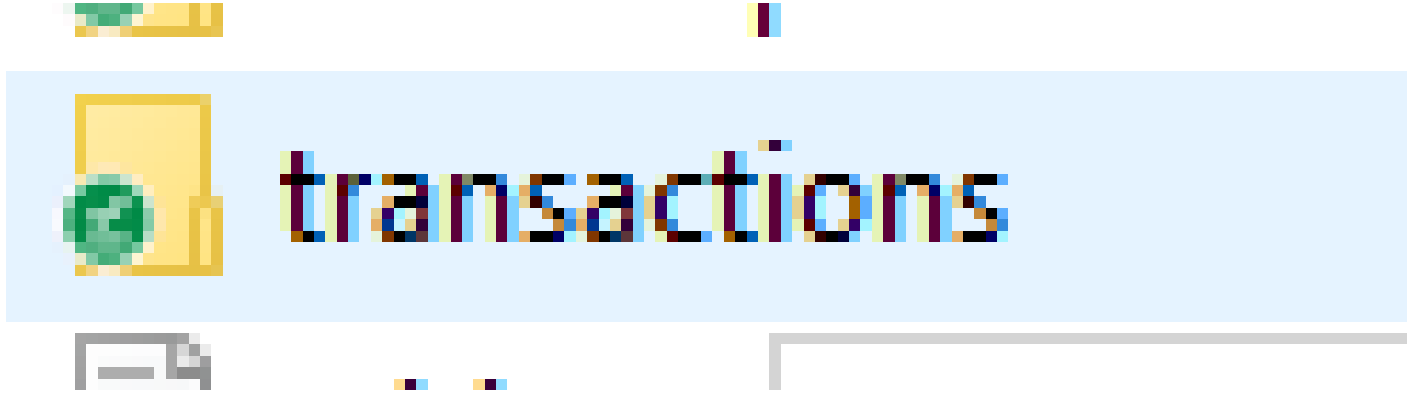
transactions

Ejercicio práctico

- Tomar la salida del ejemplo de transacciones y ajustar por el tipo de cambio
  - Archivo de tipo de cambios en repositorio
- Se puede asumir que los valores en la tabla de transacciones están en dólares y deben ser ajustados al tipo de cambio de los dos clientes (uno en Colones otro en Euros)
- La salida del programa modificado debe contener una columna con los valores monetarios ajustados

```
uscar (Ctrl+Mayús+F)
1   from datetime import datetime
2
3   from pyspark.sql import SparkSession
4   from pyspark.sql.functions import col, date_format, udf
5   from pyspark.sql.types import (DateType, IntegerType, FloatType, StringType,
6                                  StructField, StructType, TimestampType)
7
```

```
8    spark = SparkSession.builder.appName("Read Transactions").getOrCreate()
9
```

```
10  csv_schema = StructType([StructField('customer_id', IntegerType()),
11                           StructField('amount', FloatType()),
12                           StructField('purchased_at', TimestampType()),
13                           ])
14
15  dataframe = spark.read.csv("transactions.csv",
16                             schema=csv_schema,
17                             header=True)
18
19  dataframe.show()
20
```

```
+-----------+------+-------------------+
|customer_id|amount|       purchased_at|
+-----------+------+-------------------+
|          1|  55.0|2020-03-01 09:00:00|
|          1| 125.0|2020-03-01 10:00:00|
|          1|  32.0|2020-03-02 13:00:00|
|          1|  64.0|2020-03-02 15:00:00|
|          1| 128.0|2020-03-03 10:00:00|
|          2| 333.0|2020-03-01 09:00:00|
|          2| 334.0|2020-03-01 09:01:00|
|          2| 333.0|2020-03-01 09:02:00|
|          2|  11.0|2020-03-03 20:00:00|
|          2|  44.0|2020-03-03 20:15:00|
+-----------+------+-------------------+
```

```
21  # Add a new column by formatting the original date
22
23  formatted_df = dataframe.withColumn("date_string",
24                                      date_format(col("purchased_at"),
25                                                  'MM/dd/yyyy'))
26  formatted_df.show()
27
```

```
+-----------+------+-------------------+-----------+
|customer_id|amount|       purchased_at|date_string|
+-----------+------+-------------------+-----------+
|          1|  55.0|2020-03-01 09:00:00| 03/01/2020|
|          1| 125.0|2020-03-01 10:00:00| 03/01/2020|
|          1|  32.0|2020-03-02 13:00:00| 03/02/2020|
|          1|  64.0|2020-03-02 15:00:00| 03/02/2020|
|          1| 128.0|2020-03-03 10:00:00| 03/03/2020|
|          2| 333.0|2020-03-01 09:00:00| 03/01/2020|
|          2| 334.0|2020-03-01 09:01:00| 03/01/2020|
|          2| 333.0|2020-03-01 09:02:00| 03/01/2020|
|          2|  11.0|2020-03-03 20:00:00| 03/03/2020|
|          2|  44.0|2020-03-03 20:15:00| 03/03/2020|
+-----------+------+-------------------+-----------+
```

```
28  # Create a user defined function
29  string_to_date = \
30      udf(lambda text_date: datetime.strptime(text_date, '%m/%d/%Y'),
31          DateType())
32
33  typed_df = formatted_df.withColumn(
34      "date", string_to_date(formatted_df.date_string))
35  typed_df.show()
36  typed_df.printSchema()
37
```

```
+-----------+------+-------------------+-----------+----------+
|customer_id|amount|       purchased_at|date_string|      date|
+-----------+------+-------------------+-----------+----------+
|          1|  55.0|2020-03-01 09:00:00| 03/01/2020|2020-03-01|
|          1| 125.0|2020-03-01 10:00:00| 03/01/2020|2020-03-01|
|          1|  32.0|2020-03-02 13:00:00| 03/02/2020|2020-03-02|
|          1|  64.0|2020-03-02 15:00:00| 03/02/2020|2020-03-02|
|          1| 128.0|2020-03-03 10:00:00| 03/03/2020|2020-03-03|
|          2| 333.0|2020-03-01 09:00:00| 03/01/2020|2020-03-01|
|          2| 334.0|2020-03-01 09:01:00| 03/01/2020|2020-03-01|
|          2| 333.0|2020-03-01 09:02:00| 03/01/2020|2020-03-01|
|          2|  11.0|2020-03-03 20:00:00| 03/03/2020|2020-03-03|
|          2|  44.0|2020-03-03 20:15:00| 03/03/2020|2020-03-03|
+-----------+------+-------------------+-----------+----------+
```

```
root
 |-- customer_id: integer (nullable = true)
 |-- amount: float (nullable = true)
 |-- purchased_at: timestamp (nullable = true)
 |-- date_string: string (nullable = true)
 |-- date: date (nullable = true)
```

```
38    # Group By and Select the data already aggregated
39    sum_df = typed_df.groupBy("customer_id", "date").sum()
40    sum_df.show()
41
```

```
+-----------+----------+------------------+------------+
|customer_id|      date|sum(customer_id)|sum(amount)|
+-----------+----------+------------------+------------+
|          1|2020-03-03|                 1|       128.0|
|          2|2020-03-01|                 6|      1000.0|
|          1|2020-03-02|                 2|        96.0|
|          1|2020-03-01|                 2|       180.0|
|          2|2020-03-03|                 4|        55.0|
+-----------+----------+------------------+------------+
```

```
42    stats_df = \
43        sum_df.select(
44            col('customer_id'),
45            col('date'),
46            col('sum(amount)').alias('amount'))
47
48    stats_df.printSchema()
49    stats_df.show()
50
```

```
root
 |-- customer_id: integer (nullable = true)
 |-- date: date (nullable = true)
 |-- amount: double (nullable = true)
```

```
+-----------+----------+------+
|customer_id|      date|amount|
+-----------+----------+------+
|          1|2020-03-03| 128.0|
|          2|2020-03-01|1000.0|
|          1|2020-03-02|  96.0|
|          1|2020-03-01| 180.0|
|          2|2020-03-03|  55.0|
+-----------+----------+------+
```

```
51    # Load separate file where we store user names...
52    name_schema = StructType([StructField("id", IntegerType()),
53                              StructField("name", StringType()),
54                              StructField("currency", StringType())])
55
56    names_df = spark.read.csv('names.csv',
57                              schema=name_schema,
58                              header=True)
59
60    names_df.printSchema()
61    names_df.show()
62
```

```
root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- currency: string (nullable = true)
```

```
+---+----+--------+
| id|name|currency|
+---+----+--------+
|  1|John|     CRC|
|  2|Jane|     EUR|
+---+----+--------+
```

```
63    # ...and join to the aggregates
64    joint_df = stats_df.join(names_df, stats_df.customer_id == names_df.id)
65    joint_df.printSchema()
66    joint_df.show()
67
```

```
root
 |-- customer_id: integer (nullable = true)
 |-- date: date (nullable = true)
 |-- amount: double (nullable = true)
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- currency: string (nullable = true)
```

```
+-----------+----------+------+---+----+--------+
|customer_id|      date|amount| id|name|currency|
+-----------+----------+------+---+----+--------+
|          1|2020-03-03| 128.0|  1|John|     CRC|
|          2|2020-03-01|1000.0|  2|Jane|     EUR|
|          1|2020-03-02|  96.0|  1|John|     CRC|
|          1|2020-03-01| 180.0|  1|John|     CRC|
|          2|2020-03-03|  55.0|  2|Jane|     EUR|
+-----------+----------+------+---+----+--------+
```

```
67  ######## Adding exchange rate adjustment
68  exchange_schema = StructType([StructField('currency', StringType()),
69                                StructField('rate', FloatType()),
70                                ])
71
72  exchange_df = spark.read.csv("exchange_rates.csv",
73                               schema=exchange_schema,
74                               header=True)
75
76  with_exchange_rates = joint_df.join(
77      exchange_df, joint_df['currency'] == exchange_df['currency'])
78
79  with_exchange_rates.show()
80
```

```
+-----------+----------+------+---+----+--------+--------+-----+
|customer_id|      date|amount| id|name|currency|currency| rate|
+-----------+----------+------+---+----+--------+--------+-----+
|          1|2020-03-03| 128.0|  1|John|     CRC|     CRC|600.0|
|          2|2020-03-01|1000.0|  2|Jane|     EUR|     EUR|  0.9|
|          1|2020-03-02|  96.0|  1|John|     CRC|     CRC|600.0|
|          1|2020-03-01| 180.0|  1|John|     CRC|     CRC|600.0|
|          2|2020-03-03|  55.0|  2|Jane|     EUR|     EUR|  0.9|
+-----------+----------+------+---+----+--------+--------+-----+
```

```
81  currency_adjusted_df = with_exchange_rates.withColumn(
82      'adjusted_amount',
83      with_exchange_rates.amount * with_exchange_rates.rate)
84
85  currency_adjusted_df.show()
86
```

```
+-----------+----------+------+---+----+--------+--------+-----+------------------+
|customer_id|      date|amount| id|name|currency|currency| rate|   adjusted_amount|
+-----------+----------+------+---+----+--------+--------+-----+------------------+
|          1|2020-03-03| 128.0|  1|John|     CRC|     CRC|600.0|           76800.0|
|          2|2020-03-01|1000.0|  2|Jane|     EUR|     EUR|  0.9| 899.999976158142 1|
|          1|2020-03-02|  96.0|  1|John|     CRC|     CRC|600.0|           57600.0|
|          1|2020-03-01| 180.0|  1|John|     CRC|     CRC|600.0|          108000.0|
|          2|2020-03-03|  55.0|  2|Jane|     EUR|     EUR|  0.9|49.499998688697815|
+-----------+----------+------+---+----+--------+--------+-----+------------------+
```

```
final_df = \
    currency_adjusted_df.select(
        col('customer_id'),
        col('name'),
        col('date'),
        col('adjusted_amount'))

final_df.show()

```

```
+-----------+----+----------+------------------+
|customer_id|name|      date|   adjusted_amount|
+-----------+----+----------+------------------+
|          1|John|2020-03-03|           76800.0|
|          2|Jane|2020-03-01| 899.999761581421|
|          1|John|2020-03-02|           57600.0|
|          1|John|2020-03-01|          108000.0|
|          2|Jane|2020-03-03|49.499998688697815|
+-----------+----+----------+------------------+
```