In [ ]:
```
from pyspark.sql import SparkSession

from pyspark.sql.functions import *
from pyspark.sql.types import *

spark = SparkSession \
    .builder \
    .appName("how to read csv file") \
    .getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2023-11-22T23:33:11,991 WARN [Thread-4] org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop librar
y for your platform... using builtin-java classes where applicable
2023-11-22T23:33:14,306 WARN [Thread-4] org.apache.spark.util.Utils - Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
2023-11-22T23:33:14,307 WARN [Thread-4] org.apache.spark.util.Utils - Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
2023-11-22T23:33:14,309 WARN [Thread-4] org.apache.spark.util.Utils - Service 'SparkUI' could not bind on port 4042.
Attempting port 4043.
2023-11-22T23:33:14,311 WARN [Thread-4] org.apache.spark.util.Utils - Service 'SparkUI' could not bind on port 4043.
Attempting port 4044.
```

Cargo los datos en un dataframe de spark

In [ ]:
```
plans_refined = spark.read.csv('/user/ort/obligatorio/plans.csv', header=True)
```

Vista previa de las primeras filas

In [ ]:
```
plans_refined.show()
```

```
+----+--------+----------+
|plan|  nombre|created_at|
+----+--------+----------+
|   A|   Basic|  6/4/2021|
|   B|Original|  6/8/2021|
|   C|  Family|  6/2/2021|
|   D| Premium|  6/2/2021|
+----+--------+----------+
```

Cantidad de columnas del dataframe monthly_pays

In [ ]:
```
num_columns=len(plans_refined.columns)
num_columns
```

Out[ ]:
```
3
```

Nombre de las columnas de monthly_pays

In [ ]:
```
plans_refined.columns
```

Out[ ]:
```
['plan', 'nombre', 'created_at']
```

Descripción de los datos de la tabla

In [ ]:
```
plans_refined.describe
```

Out[ ]:
```
<bound method DataFrame.describe of DataFrame[plan: string, nombre: string, created_at: string]>
```

Schema de la tabla

In [ ]:
```
plans_refined.printSchema()
```

```
root
 |-- plan: string (nullable = true)
 |-- nombre: string (nullable = true)
 |-- created_at: string (nullable = true)
```

In [ ]:
```
plans_refined.show()
```

```
+----+--------+----------+
|plan|  nombre|created_at|
+----+--------+----------+
|   A|   Basic|  6/4/2021|
|   B|Original|  6/8/2021|
|   C|  Family|  6/2/2021|
|   D| Premium|  6/2/2021|
+----+--------+----------+
```

In [ ]:
```
spark = SparkSession.builder.appName("EjemploConversionFecha").getOrCreate()

spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
```

```
2023-11-22T23:33:30,075 WARN [Thread-4] org.apache.spark.sql.SparkSession - Using an existing Spark session; only run
time SQL configurations will take effect.
```

In [ ]:
```
plans_refined = plans_refined.withColumn("created_at", to_date(col("created_at"), "mm/dd/yyyy"))
```

In [ ]:
```
plans_refined.printSchema()
```

```
root
 |-- plan: string (nullable = true)
 |-- nombre: string (nullable = true)
 |-- created_at: date (nullable = true)
```

In [ ]:
```
plans_refined.show()
```

```
+----+--------+----------+
|plan|  nombre|created_at|
+----+--------+----------+
|   A|   Basic|2021-01-04|
|   B|Original|2021-01-08|
|   C|  Family|2021-01-02|
|   D| Premium|2021-01-02|
+----+--------+----------+
```

Valores Nulos o Faltantes

In [ ]:
```
total_nulos = plans_refined.select([sum(col(c).isNull().cast("int")).alias(c) for c in plans_refined.columns])
```

```
total_nulos.show()
```

```
+----+------+----------+
|plan|nombre|created_at|
+----+------+----------+
|   0|     0|         0|
+----+------+----------+
```

In [ ]:

```
hdfs_path = "/user/ort/obligatorio/refined/refined_plans/"
plans_refined.write.csv(hdfs_path, header=False, mode="overwrite")
```