

CasoPractico 6.1 Python ▾

File Edit View Run Help Last edit was 11 hours ago Provide feedback

Run all Terminated Share Publish

PREGUNTAS CASO NEGOCIO SPARK

DataFrame Clientes

```
1 # DataFrame Clientes
2 columns = ['id_cliente','nombre','edad','sexo','provincia','pais']
3 data = [(1,'Pablo Perez',26,'M','Madrid','España'),(2,'Eduardo Redondo',58,'M','Bogota','Colombia'),(3,'Roberto Salazar',68,'M','Monterrey','Mexico'),
4 ('4,'Pedro Conde',32,'M','Madrid','España'),(5,'Ana Robles',41,'F','Valladolid','España'),(6,'David Roldan',74,'M','Guadalajara','Mexico'),
5 ('Carmen Araujo',19,'F','Medellin','Colombia'),(8,'Sofia Rodriguez',49,'F','Barcelona','España'),(9,'David Carrasco',24,'M','Cali','Colombia'),
6 ('10,'Juan Jose Sanchez',55,'M','Barcelona','España'),(11,'Laura Diaz',33,'F','Buenos Aires','Argentina'),(12,'Andrea Hernandez',22,'F','Caracas','Venezuela'),
7 ('13,'Victor Ruiz',53,'M','Valladolid','España'),(14,'Melisa Aguado',38,'F','Bogota','Colombia'),(15,'Cristian Cuadrado',52,'M','Guadalajara','Mexico'),
8 ('16,'Cristina Sanz',49,'F','Madrid','España'),(17,'Jorge Recio',18,'M','Valladolid','España'),(18,'Laura Luis',44,'F','Cali','Colombia'),
9 ('19,'Juan Carlos Iglesias',38,'M','Barcelona','España'),(20,'Oscar Rico',22,'M','Valladolid','España'),(21,'Fatima Cuevas',29,'F','Cali','Colombia'),
10 ('22,'Clara Suarez',21,'F','Sevilla','España'),(23,'Fernando Gomez',78,'F','Monterrey','Mexico'),(24,'Ruben Garcia',68,'M','Sevilla','España'),
11 ('25,'Inaki Barcero',29,'F','Guadalajara','Mexico'),(26,'Celia Castro',47,'F','Barcelona','España'),(27,'Roberto Varaldo',64,'F','Cali','Colombia'),
12 ('28,'Walter Ramon',54,'M','Cali','Colombia'),(29,'Romina Verde',29,'F','Guadalajara','Mexico'),(30,'Maria Rodriguez',67,'F','Guadalajara','Mexico')]
```

df_clientes = spark.createDataFrame(data, columns)
df_clientes.write.mode("overwrite").option("encoding", "UTF-8").saveAsTable("df_clientes")
display(df_clientes)
df_clientes.show(n=30)

(11) Spark Jobs

df_clientes: pyspark.sql.dataframe.DataFrame = [id_cliente: long; nombre: string ... 4 more fields]

Command took 12.11 seconds — by julianvazquez171@gmail.com at 2/11/2023, 22:16:13 on My Cluster

DataFrame Ofertas estandar (truncate=False)

DataFrame facturas_mes_ant (truncate = false)

DataFrame facturas_mes_actual (truncate = false)

DataFrame Consumos diarios

CasoPráctico 6.1 Python

```

1 df_resumen_consumo = df_resumen.groupby("grupo_edad").agg(
2     round(avg("consumo_datox"), 2).alias("media"),
3     round(avg("medios_sms"), 2).alias("datos"),
4     first("nombre").alias("nombre"),
5     first("edad").alias("edad")
6 )
7
8 # Obtener el maximo de SMS enviados para cada grupo de edad
9 df_max_sms = df_resumen_consumo.groupby("grupo_edad").agg(
10    max("datos").alias("max_sms")
11 )
12
13 # Unir los DataFrames para obtener los registros deseados para cada grupo de edad
14 df_resumen_consumo = df_resumen_consumo.join(
15    df_max_sms, "grupo_edad"
16 ).where(col("datos") == col("max_sms")).select(
17    "nombre",
18    "edad",
19    "grupo_edad",
20    "max_sms",
21    "datos"
22 )
23

```

CasoPráctico 6.1 Python

```

1 df_resumen_consumo = pyspark.sql.DataFrame.DataFrame = [id_cliente: long, MB: double ... 3 more fields]
2 df_max_sms = pyspark.sql.DataFrame.DataFrame = [grupo_edad: integer, max_sms: double]
3 df_max_datos_y_sms = pyspark.sql.DataFrame.DataFrame = [nombre: string, edad: long ... 3 more fields]
4
5 +-----+
6 | nombre|edad|grupo_edad| MB|datos|
7 +-----+
8 | Carmen Arausa| 19| 1|1338.24| 0.54|
9 | Eduardo Redondo| 58| 3|473.29| 5.25|
10 | Roberto Salazar| 68| 4|171.19| 6.16|
11 | Pablo Perez| 26| 2|1082.89| 1.13|
12

```

CasoPráctico 6.1 Python

```

1 df_clientes_nuevos = df_clientes.join(df_f1, "id_cliente", "left_anti") # Obtenemos clientes nuevos de este mes
2 df_importe_total = df_f1.groupby("id_cliente").agg(sum("importe").alias("importe_total_mes_actual")) # Importe total mes actual para cada cliente nuevo
3 df_mins_total = df_f1.filter(minute_of( llamada_noviembre ) >= 0) # Filtramos las llamadas realizadas en el mes de Agosto
4 df_mins_total = df_f1.filter(minute_of( llamada_noviembre ) >= 0).between("2020-08-01", "2020-08-31").groupby("id_cliente").agg(
5     sum("mins_movil").alias("total_minutos_movil"),
6     sum("mins_fijo").alias("total_minutos_fijo")
7 )
8 # Unimos DFs para obtener DF final
9 df_clientes_total = df_clientes_nuevos.join(df_importe_total, "id_cliente").join(df_mins_total, "id_cliente").select(
10    "nombre",
11    "edad",
12    "importe_total_mes_actual",
13    (col("total_minutos_movil") + col("total_minutos_fijo")).alias("total_minutos")
14 ).orderBy("total_minutos")
15
16 df_clientes_total.show()
17

```

CasoPráctico 6.1 Python

```

1 df_clientes_exist = df_clientes.join(df_cd, "id_cliente", "inner").join(df_f0, "id_cliente", "inner") # Join entre clientes y consumos del mes anterior
2 df_clientes_exist = df_clientes_exist.withColumn("SMS", when(col("SMS").isNotNull(), 0).otherwise(col("SMS"))) # Rellenamos con 0 los SMS vacíos
3 df_clientes_exist=df_clientes_exist.groupby("nombre", "edad").agg(
4     (sum(when(col("SMS") == 0, 1).otherwise(0))).alias("n_dias_sin_sms")
5 ) # Calculamos el num de días sin SMS para cada cliente
6
7
8 df_clientes_exist.show(10)
9

```

(6) Spark Jobs

```

1 df_clientes_exist = pyspark.sql.DataFrame.DataFrame = [nombre: string, edad: long ... 1 more field]
2 +-----+
3 | nombre|edad|n_dias_sin_sms|
4 +-----+
5 | Pablo Perez| 26| 42|
6 | Eduardo Redondo| 58| 7|
7 | David Robles| 74| 1|
8 | Ana Robles| 41| 6|
9 | Roberto Salazar| 68| 3|
10 | Silvia Rodriguez| 49| 2|
11 | Luis Rodriguez| 28| 21|
12 | Carlos Ruiz| 59| 19|
13 | Pablo Lopez| 48| 11|
14 | Cristian Cuadrado| 52| 10|
15 | Victor Ruiz| 53| 7|
16 | Alvaro Monroy| 75| 31|
17 | Mercedes Lopez| 39| 20|
18 | Jorge Recio| 18| 22|
19 | Laura Lujz| 44| 14|
20 | Clara Suarez| 21| 52|
21 | Ines Barcerol| 29| 26|
22 | Fernanda Gomez| 78| 21|

```

9. Queremos obtener un Coeficiente de Ponderación que nos permita evaluar a cada cliente en función de su consumo para identificar los clientes más atractivos que forman parte de nuestra compañía. ESTE

CasoPráctico 6.1 Python v Last edit was 11 hours ago Provide feedback

9. Queremos obtener un Coeficiente de Ponderación que nos permita evaluar a cada cliente en función de su consumo para identificar los clientes más atractivos que forman parte de nuestra compañía. ESTE CÁLCULO SÓLO SE REALIZARÁ PARA LOS CLIENTES QUE TENGAN UNA SOLA OFERTA CONTRATADA CON LA COMPAÑÍA.

Este coeficiente se obtendrá en base a los consumos diarios, y por tanto sólo se tendrán en cuenta los clientes que existen en el mes de Agosto, ya que no tenemos datos de consumo del mes de Julio.

Las ponderaciones que se darán a cada uno de los consumos son las siguientes:

- 0.4 -> llamadas desde teléfono móvil.
- 0.3 -> datos_móviles_MB
- 0.2 -> llamadas desde teléfono fijo
- 0.1 -> SMS enviados

Código 31:

Los pasos a seguir son los siguientes:

1. Obtener la suma de todos los días de cada uno de los 4 consumos para cada cliente.
2. Obtenir el máximo del cálculo anterior de todos los clientes para poder obtener un valor entre 0 y 1 para cada uno de los 4 consumos (recordar que sólo aplica para los clientes con UNA SOLA OFERTA CONTRATADA en el mes de Agosto. El cliente con mayor consumo de datos tendrá un valor de 1 en la columna "datos_móviles_0_1")
3. Multiplicar estas columnas obtenidas con valor entre 0 y 1 por su ponderación correspondiente (por ejemplo la columna "datos_móviles_0_1" se multiplicará por la ponderación de datos móviles 0.4).
4. Por último se calculará la suma de las 4 ponderaciones obtenidas para calcular el Coeficiente de ponderación buscado (coeficiente_cliente), casteado a 3 decimales.
5. Mostrar DF resultante ordenado por este coeficiente en sentido descendente, de manera que el primer registro corresponderá al cliente más atractivo. El DF resultante tendrá 3 columnas: nombre, edad, coeficiente_cliente.

Código 32:

```

1 # Obtenemos la suma de todos los tipos de consumos para cada cliente
2 df_suma_cd = df_cd.groupby("id_cliente").agg(
3     sum("minis_movil").alias("total_llamadas_movil"),
4     sum("MB").alias("total_MB"),
5     sum("minis_fijo").alias("total_llamadas_fijo"),
6     sum("sms").alias("total_sms")
7 )
8 # Calculamos el maximo de las sumas de consumo entre todos los clientes
9 max_values = df_suma_cd.agg(
10     max("total_llamadas_movil"),
11     max("total_MB"),
12     max("total_llamadas_fijo"),
13     max("total_sms")
14 ).collect()[0]
15 # Normalizamos los valores entre 0 y 1
16 df_suma_cd = df_suma_cd.withColumn("coef_llamadas_movil", col("total_llamadas_movil")/max_values[0]) \
17     .withColumn("coef_datos_moviles_MB", col("total_MB") / max_values[1]) \
18     .withColumn("coef_llamadas_fijo", col("total_llamadas_fijo") / max_values[2]) \
19     .withColumn("coef_SMS_enviados", col("total_sms") / max_values[3])

```

Código 33:

10. Queremos averiguar la fecha en la que entre los 3 clientes que más datos consumen de cada grupo_edad llegan a un consumo de 20 GB de datos móviles. En caso de que algún grupo_edad no llegue entre los 3 clientes a 20 GB en todo el mes, se asignará "null" como valor de esta columna (Recordar que 1 GB = 1024 MB).

Código 34:

10. Queremos averiguar la fecha en la que entre los 3 clientes que más datos consumen de cada grupo_edad llegan a un consumo de 20 GB de datos móviles. En caso de que algún grupo_edad no llegue entre los 3 clientes a 20 GB en todo el mes, se asignará "null" como valor de esta columna (Recordar que 1 GB = 1024 MB).

Obtener un DF que contiene 4 registros, uno para cada grupo_edad y 3 columnas: grupo_edad, fecha_20_GB, datos_móviles_total_grupo_3_clientes. (datos_móviles_total_grupo_3_clientes representa el total de datos consumidos en MB por los 3 clientes del grupo hasta final de mes)

ANEXO

```
Last login: Fri Nov  3 10:28:15 on ttys000
julianvazquezsampedro@MacBook-Air-de-Julian ~ % script log.txt # Comienza la grabación de la sesión en el archivo log.txt
spark2-submit nombre_del_script.py # Ejecuta el comando spark2-submit con tu script .py
exit # Salir del modo de grabación de la sesión
Script started, output file is log.txt
script: #: No such file or directory
Script done, output file is log.txt
zsh: command not found: spark2-submit
exit: too many arguments
julianvazquezsampedro@MacBook-Air-de-Julian ~ %
```