SPARK

- CREAR UNA NUEVA SESION DE SPARK: spark =SparkSession.builder.master("local[*]") .getOrCreate( )
- ABRIR UN ARCHIVO, MOSTRAR EL ESQUEMA Y PROCESAR LA PRIMERA LINEA COMO HEADER:

  customers = spark.read.option("inferSchema", "true") .option("header", "true") .csv("customers.csv")

  products = spark.read.option("inferSchema", "true") .option("header", "true") .csv("products.csv")

  stock = spark.read.option("inferSchema", "true") .option("header", "true") .csv("stock.csv")

  customers.printSchema( )

  products.printSchema( )

  stock.printSchema( )
- MOSTRAR INFORMACIÓN DEL DATAFRAME:

  customers.printSchema( ) -> muestra los nombres de las columnas y el tipo de dato que contienen

  customers.show(5) -> Muestra las cinco primeras filas

  customers.dtypes -> Muestra un vector de los tipos de datos

  customers.summary( ) -> muestra una tabla con las operaciones básicas de las columnas

  Ejemplo Show:

```
+----------+--------+--------+-------+--------+-----+
|      date|    time|customer|product|quantity|price|
|05/10/2018| 2:20 PM|     100|      1|      10|  816|
|06/10/2018| 3:30 PM|     100|      1|      10|    1|
|07/10/2018| 5:20 PM|     100|      1|      10|   10|
|04/08/2018|11:38 PM|     100|      2|       8|   79|
|25/03/2018| 3:52 AM|     100|      3|       1|   91|
+----------+--------+--------+-------+--------+-----+
```

- ¿CUÁLES SON LOS PRODUCTOS EN MI DATASET?

  Customers.select(expr(“product as my_products”)) .show
- ¿CUALES SON LOS PRODUCTOS Y CANTIDADES EN MI DATASET?

  Customers.select(expr(quantity) , expr(product)) .show
- SELECCIONAR TODA LA TABLA CON *

  c.select(expr("*")) .show( )
- ¿QUÉ COMPRAS SE HAN HECHO A PRODUCTOS CON MÁS DE UNA CANTIDAD?

  Customers.selectexpr(“ * ”, “quantity>1”) .show -> añade una nueva columna que es true si esa compra tiene más de 1 unidad, o false si no.
- ¿CUÁNTOS PRODUCTOS SE HAN COMPRADO EN TOTAL?

  Customers.selectexpr(“sum(quantity) ”) .show
- ¿CUÁL ES LA MEDIA DEL PRECIO?

  Customers.selectexpr(“avg(Price) ”) .show
- ¿CUÁL ES LA DIFERENCIA ENTRE...?

  c.selectExpr("count(customer) ") .show( ) -> cuenta el número de filas en la columna customer

  c.selectExpr("count(distinct(customer)) ") .show( ) ->Cuenta el numero de filas diferentes en la columna customer.

- ¿CUÁL ES EL PRECIO MEDIO Y EL NÚMERO DE VENTAS?

  Customers.selectExpr(“avg(Price) ”, “count(customer) ”) .show
- ¿QUE COMPRAS A PRODUCTOS TIENEN MENOS DE 9 CANTIDADES?

  Customers.where(“quantity<9”) .show
- ¿QUE COMPRAS A PRODUCTOS NO LAS HA HECHO EL CUSTOMER 100?

  Customers.where(“quantity !=100) .show
- ¿QUE COMPRAS A PRODUCTOS NO SON DEL CUSTOMER 100 Y TIENEN MENOS DE 9 UNIDADES?

  Customers.where(“customer!=100”, ”quantity<9”) .show
- ¿QUÉ COMPRADORES HAN COMPRADO 7 O MÁS ÍTEMS DEL PRODUCTO 8?

  Customer.where(“quantity>=7”, ”product=8”) .show
- ¿CUÁL ES EL PRECIO MÁS CARO?

  Customers.orderby(“Price”) .show(1)
- ORDENAR DESCENDENTEMENTE POR CUSTOMER Y ASCENDENTEMENTE POR PRECIO DE PRODUCTO:

  Customer.orderby(“desc(customer) ”, asc(price) ”) .show
- ORDENAR DESCENDENTEMENTE POR CUSTOMER Y ASCENDENTEMENTE POR PRECIO DE PRODUCTO LAS COMPRAS QUE SE HAN HECHO DESPUÉS DEL DÍA 16/09/2018:

  Customer.where(“ date > ’16/09/2018’ ”.orderby(“desc(customer) ”, “asc(price) ”) .show
- ¿CUANTAS COMPRAS TENEMOS POR CADA CLIENTE?

  Customers.groupby(“customer”) .count( ) .show

  .count porque cada línea es una compra
- ¿CUÁL ES EL NUMERO TOTAL DE PRODUCTOS QUE HA COMPRADO CADA CLIENTE?

  Customers.groupbu(“customer”) .agg(expr(“sum(quantity) ”)) .show

  Se aplica agregación porque queremos saber la suma de quantity por cada customer
- POR CADA CLIENTE CALCULA LA MEDIA DE LA CANTIDAD Y LA DESVIACION TIPICA POBLACIONAL DE CANTIDAD:

  Customers.groupby(“customer”) .agg(expr(“avg(quantity) ”) , expr(“stddev_pop(quantity) ”)) .show
- POR CADA CLIENTE LA MEDIA DE CANTIDAD QUE HA COMPRADO Y EL PRECIO MÁXIMO:

  Customers.groupby(“customer”) .agg(expr(“avg(quantity) ”, expr(“max(price) ”)) .show
- UNIR DOS DATASETS:

  joined = products.join(stock, products["ID"] == stock["ID"], "inner")

  joined.show( )

Preguntas del laboratorio

- ¿cuantos elementos podemos encontrar?
  Customer.selectexpr("count(customer) ") .show

- ¿cuantos customers unicos tenemos?
  Customers.selectexpr("count(distinct(customer)) ") .show

- ¿cuantos productos han sido comprados por cada customer?
  Customers.groupby("customer") .agg("sum(quantity) as total") .show
- ordenar los customers por cantidad
  Customers.groupby("customer") .agg(expr("sum(quantity) as total"
)) .orderby("TOTAL") .show( )
- ¿cuantas veces el customer con id 100 ha comprado más de 5 items.
  Customers.where(expr("customer = 100")) .where(expr("quantity >5")) .count( )
- cuales han sido los productos comprados por el customer con el número más grande de transacciones? nos interesa el cusrtomer que ha comprado mas veces.
  Max_customer = customers.groupby("customer") .count( ) .orderBy(desc("count")) .first( ) .customer
  Dataframe no tiene fist pero groupby si, es un vector indexado por las columnas a las que puedes acceder haciendo .column
  customers.where(expr("customer = " + str(max_customer))) .select("product") .distinct( ) .show( )
  es importante hacer str(max_customer) para que lo tome como una variable definida. El distinct no se puede aplicar sin una función: por ejemplo count(distinct(...)) si seria posible pero si solo necesitamos los valores únicos de una columna hay que seleccionar la columna selecft(...) .distinct( )

- Cuantas regiones sanitarias diferentes hay y cuantos medicamentos diferentes se han dispensado?
  Recetas2.select("rsanitaria") .distinct( ) , count( )
  Recetas2.select("medicament") .distinct( ) .count( )
- Cuantas recetas se han dispensado de cada medicamento?
  Recetas2.groupby("medicament") .agg(expr("sum(nreceptes) as numero_de_recetas") .show( )

SQL

- Muestra los datos de mujeres

  Spark.sql(“select * FROM MY_Table where sexe = ‘ dona ’ ”) .show
- ¿Cuantos medicamentos diferentes se expedieron en 2022, cuantas recetas y qué coste tuvieron?

  Sparl.sql(

  “

  select medicament, sum(cast(nrecetas as int)) as numero_recetas, sum(cast(cost as int)) as cost

  from mytable

  group by medicament

  order by numero_recetas desc

  “

  Importante hacer el sum porque por cada medicamento tenemos como mínimo dos filas:

  Medicamento_a mujer nrecetas coste

  Medicamento_A hombre nreceas coste

- ¿Cuál es el mediamento mas recetado en hombre y mujer y en que región sanitaria?

  recetas2.createOrReplaceTempView("my_table")

  consulta = """

  SELECT medicament, rsanitaria, sexe, nreceptes as numero_recetas

  FROM my_table

  WHERE sexe == 'Dona' or sexe == 'Home'

  GROUP BY medicament, rsanitaria, sexe, nreceptes

  ORDER BY sexe

  """

  result = spark.sql(consulta)

  Window_spec = Window.partitionBy("sexe") .orderBy(F.desc("numero_recetas"))

  result.withColumn("rank", F.rank( ) .over(Window_spec)) .where(expr("rank=1"))

  se hace una consulta cogiendo el medicamento, la región sanitaria el sexo y el numero de recetas agrupando por todos ellos filtrando solo por hombre y mujer y ordenando por sexo, de forma que tengamos todas las mujeres arriba y todos los hombres abajo.

  Se crea una window que particiona por sexo (por eso ordenar por sexo->más claridad) y que ordena dentro de las particiones(sexo) por numero de recetas de formadescendente. De esta forma tendremos el medicamento más pedido para mujeres arriba de y el medicamento mas pedido por hombres arriba de la sección de hombres.

Creamos una nueva columna llamada rank que se basa en el orden definido en la window. Se aplica una función de ranking que asignará de cada grupo (Sexo) el valor 1 al que tenga el valor mas grande de numero recetas. Por último solo nos quedamos con esas filas que tienen en mayor número de recetas.

- ¿Cuál es el medicamento menos recetado en hombre y mujer y en que región sanitaria?

```
recetas2.createOrReplaceTempView( "my_table")

consulta = """
SELECT medicament, rsanitaria, sexe, nreceptes as numero_recetas
FROM my_table
WHERE sexe == 'Dona' or sexe == 'Home'
GROUP BY medicament, rsanitaria, sexe, nreceptes
ORDER BY sexe
"""
result = spark.sql( consulta)

window_spec = Window.partitionBy( "sexe") .orderBy( F.desc( "numero_recetas") )

result.withColumn( "rank", F.rank( ) .over( window_spec) ) .where( expr( "rank=1") )
```

- ¿Cuál es el medicamento más caro?

  El medicamento más caro de media

```
recetas2.createOrReplaceTempView( "my_table")
consulta = """
SELECT medicament, AVG( import ) as coste_medio_medicamento
FROM my_table
GROUP BY medicament
ORDER BY coste_medio_medicamento desc
Limit 1
"""
spark.sql( consulta)
```

  El medicamento más caro de todos y en qué región es tan caro

```
recetas2.createOrReplaceTempView( "my_table")
consulta = """
SELECT medicament, rsanitaria
FROM my_table
ORDER BY import desc
LIMIT 1
"""
spark.sql( consulta) .show( )
```

- Considerando los 10 medicamentos mas recetados durante 2022 indicar el coste de este por región sanitaria.

```
recetas2.createOrReplaceTempView( "my_table")
consulta_sub = """ (
SELECT  medicament
FROM my_table
WHERE medicament != 'SENSE ESPECIFICAR'
GROUP BY medicament
ORDER BY sum( cast( nreceptes as int) )  DESC
LIMIT 10) """

consulta = """
SELECT rsanitaria,  medicament,  AVG( CAST( import AS INT) )   AS coste_medio_medicamento
FROM my_table
WHERE medicament IN""" + consulta_sub +"""
GROUP BY rsanitaria, medicament
ORDER BY rsanitaria, medicament
"""
spark.sql( consulta) .show( )
```