

(https://databricks.com)

**1. A partir del archivo csv Case, determine las tres ciudades con más casos confirmados de la enfermedad. La salida debe contener tres columnas: provincia, ciudad y casos confirmados. El resultado debe contener exactamente los tres nombres de ciudades con más casos confirmados ya que no se admiten otros valores.**

2

```
// Importar librerías necesarias
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._

// 1. Definir la ruta del archivo y el esquema
val fileLocation = "/FileStore/tables/case.csv"

val schema = StructType(Array(
  StructField("case_id", StringType, true),
  StructField("province", StringType, true),
  StructField("city", StringType, true),
  StructField("group", StringType, true),
  StructField("infection_case", StringType, true),
  StructField("confirmed", IntegerType, true), // Castear a Integer
  StructField("latitude", DoubleType, true), // Castear a Double
  StructField("longitude", DoubleType, true) // Castear a Double
))

// 2. Leer el archivo CSV con el esquema definido y aplicar filtros
val df = spark.read
  .format("csv")
  .option("header", "true") // El archivo tiene encabezado
  .option("sep", ";")       // Usar ';' como separador
  .schema(schema)           // Aplicar el esquema definido
  .load(fileLocation)
  .filter(col("city") != "-") // Filtrar las filas donde city es "-"

// 3. Crear una vista temporal para consultas SQL
df.createOrReplaceTempView("case_table")
println("Vista temporal 'case_table' creada correctamente.")
```

▶  df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [case\_id: string, province: string ... 6 more fields]

Vista temporal 'case\_table' creada correctamente.

```
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._
fileLocation: String = /FileStore/tables/case.csv
schema: org.apache.spark.sql.types.StructType = StructType(StructField(case_id,StringType,true),StructField(province,StringType,true),StructField(city,StringType,true),StructField(group,StringType,true),StructField(infection_case,StringType,true),StructField(confirmed,IntegerType,true),StructField(latitude,DoubleType,true),StructField(longitude,DoubleType,true))
df: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [case_id: string, province: string ... 6 more fields]
```

3

```
// 4. Solución con SQL
val topCitiesSQL = spark.sql(
  """
  SELECT
    province,
    city,
    confirmed
  FROM case_table
  ORDER BY confirmed DESC
  LIMIT 3
  """
)

println("Resultado usando SQL:")
topCitiesSQL.show()
```

► topCitiesSQL: org.apache.spark.sql.DataFrame = [province: string, city: string ... 1 more field]

Resultado usando SQL:

province	city	confirmed
Daegu	Nam-gu	4511
Gyeongsangbuk-do	from other city	566
Daegu	Dalseong-gun	196

topCitiesSQL: org.apache.spark.sql.DataFrame = [province: string, city: string ... 1 more field]

4

```
// 5. Solución con Scala
val topCitiesScala = df
  .select("province", "city", "confirmed") // Seleccionar columnas relevantes
  .orderBy(col("confirmed").desc)         // Ordenar por casos confirmados en orden descendente
  .limit(3)                               // Tomar las 3 primeras filas

println("Resultado usando Scala:")
topCitiesScala.show()
```

► topCitiesScala: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [province: string, city: string ... 1 more field]

Resultado usando Scala:

province	city	confirmed
Daegu	Nam-gu	4511
Gyeongsangbuk-do	from other city	566
Daegu	Dalseong-gun	196

topCitiesScala: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [province: string, city: string ... 1 more field]

## 2. Cree un dataframe a partir del archivo csv PatientInfo. Asegúrese de que su dataframe no contenga pacientes duplicados.

- ¿Cuántos pacientes tienen informado por quién se contagiaron(columna infected\_by)? Obtenga solo los pacientes que tengan informado por quién se contagió.
- A partir de la salida del inciso anterior obtenga solo los pacientes femeninos. La salida no debe contener las columnas released\_date y deceased\_date.
- Establezca el número de particiones del dataframe resultante del inciso anterior en dos. Escriba el dataframe resultante en un archivo parquet, la salida debe estar particionada por la provincia y el modo de escritura debe ser overwrite.

6

```
// Importar librerías necesarias
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._

// 1. Definir la ruta del archivo y el esquema
val patientFileLocation = "/FileStore/tables/patient_info.csv"

val patientSchema = StructType(Array(
  StructField("patient_id", StringType, true),
  StructField("sex", StringType, true),
  StructField("age", StringType, true),
  StructField("country", StringType, true),
  StructField("province", StringType, true),
  StructField("city", StringType, true),
  StructField("infection_case", StringType, true),
  StructField("infected_by", StringType, true),
  StructField("contact_number", IntegerType, true),
  StructField("symptom_onset_date", StringType, true),
  StructField("confirmed_date", StringType, true),
  StructField("released_date", StringType, true),
  StructField("deceased_date", StringType, true),
  StructField("state", StringType, true)
))

// Leer el archivo CSV con el esquema definido
val patientDF = spark.read
  .format("csv")
  .option("header", "true") // El archivo tiene encabezado
  .option("sep", ";")      // Usar ';' como separador
  .schema(patientSchema)   // Aplicar el esquema definido
  .load(patientFileLocation)
  .dropDuplicates("patient_id") // Eliminar duplicados basados en patient_id
  .withColumn("infected_by", when(col("infected_by") === "", null).otherwise(col("infected_by"))) //
  // Reemplazar '' por null en infected_by

println(s"Tabla creada desde el CSV (pacientes únicos):")
patientDF.show()
```


▶  patientDF: org.apache.spark.sql.DataFrame = [patient\_id: string, sex: string ... 12 more fields]

Tabla creada desde el CSV (pacientes únicos):

patient_id	sex	age	country	province	city	infection_case	infected_by	contact_number	symptom_onset_date	confirmed_date	released_date	deceased_date	state
1000000001	male	50s	Korea	Seoul	Gangseo-gu	overseas inflow	null	75	2020-01-22	2020-01-23	2020-02-05	null	released
1000000002	male	30s	Korea	Seoul	Jungnang-gu	overseas inflow	null	31	2020-01-11	2020-01-30	2020-03-02	null	released
1000000003	male	50s	Korea	Seoul	Jongno-gu	contact with patient	2002000001	17	2020-01-11	2020-01-30	2020-02-19	null	released
1000000004	male	20s	Korea	Seoul	Mapo-gu	overseas inflow	null	9	2020-01-26	2020-01-30	2020-02-15	null	released
1000000005	female	20s	Korea	Seoul	Seongbuk-gu	contact with patient	1000000002	2	2020-01-11	2020-01-31	2020-02-24	null	released
1000000006	female	50s	Korea	Seoul	Jongno-gu	contact with patient	1000000003	43	2020-01-11	2020-01-31	2020-02-19	null	released
1000000007	male	20s	Korea	Seoul	Jongno-gu	contact with patient	1000000003	0	2020-01-11	2020-01-31	2020-02-10	null	released

7

```
// 2. Filtrar pacientes que tienen informado por quién se contagiaron
val infectedByDF = patientDF.filter(col("infected_by").isNotNull)
println(s"Tabla de pacientes con 'infected_by' informado:")
infectedByDF.show()
```

▶  infectedByDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [patient\_id: string, sex: string ... 12 more fields]

Tabla de pacientes con 'infected\_by' informado:

patient_id	sex	age	country	province	city	infection_case	infected_by	contact_number	symptom_onset_date	confirmed_date	released_date	deceased_date	state
------------	-----	-----	---------	----------	------	----------------	-------------	----------------	--------------------	----------------	---------------	---------------	-------

patient_id	sex	age	country	province	city	infection_case	infected_by	contact_number	symptom_onset_date
confirmed_date	released_date	deceased_date	state						
1000000003	male	50s	Korea	Seoul	Jongno-gu	contact with patient	2002000001	17	nu
ll	2020-01-30	2020-02-19			null	released			
1000000005	female	20s	Korea	Seoul	Seongbuk-gu	contact with patient	1000000002	2	nu
ll	2020-01-31	2020-02-24			null	released			
1000000006	female	50s	Korea	Seoul	Jongno-gu	contact with patient	1000000003	43	nu
ll	2020-01-31	2020-02-19			null	released			
1000000007	male	20s	Korea	Seoul	Jongno-gu	contact with patient	1000000003	0	nu
ll	2020-01-31	2020-02-10			null	released			
1000000010	female	60s	Korea	Seoul	Seongbuk-gu	contact with patient	1000000003	6	nu
ll	2020-02-05	2020-02-29			null	released			
1000000013	male	80s	Korea	Seoul	Jongno-gu	contact with patient	1000000017	117	nu
ll	2020-02-16	null			null	deceased			
1000000014	female	60s	Korea	Seoul	Jongno-gu	contact with patient	1000000013	27	2020-02-
ll	2020-02-16	2020-02-13			null	released			

8

```
// 3. Filtrar solo pacientes femeninos y eliminar las columnas 'released_date' y 'deceased_date'
val femalePatientsDF = infectedByDF
  .filter(col("sex") === "female")
  .drop("released_date", "deceased_date")

println(s"Tabla de pacientes femeninos con 'infected_by' informado:")
femalePatientsDF.show()
```

► femalePatientsDF: org.apache.spark.sql.DataFrame = [patient\_id: string, sex: string ... 10 more fields]

Tabla de pacientes femeninos con 'infected\_by' informado:

patient_id	sex	age	country	province	city	infection_case	infected_by	contact_number	symptom_onset_date
confirmed_date	released_date	deceased_date	state						
1000000005	female	20s	Korea	Seoul	Seongbuk-gu	contact with patient	1000000002	2	n
ull	2020-01-31	released							
1000000006	female	50s	Korea	Seoul	Jongno-gu	contact with patient	1000000003	43	n
ull	2020-01-31	released							
1000000010	female	60s	Korea	Seoul	Seongbuk-gu	contact with patient	1000000003	6	n
ull	2020-02-05	released							
1000000014	female	60s	Korea	Seoul	Jongno-gu	contact with patient	1000000013	27	2020-02-
-06	2020-02-16	released							
1000000019	female	70s	Korea	Seoul	Jongno-gu	contact with patient	1000000021	null	n
ull	2020-02-20	released							
1000000020	female	70s	Korea	Seoul	Seongdong-gu	Seongdong-gu APT	1000000015	null	n
ull	2020-02-20	released							
1000000029	female	20s	Korea	Seoul	Jongno-gu	Eunpyeong St. Mar...	1000000028	null	2020-02-
-11	2020-02-26	released							

9

Archivo Parquet escrito en la ubicación: /FileStore/output/patient\_info\_parquet  
 outputParquetLocation: String = /FileStore/output/patient\_info\_parquet

