```
# Instalar SDK Java 8

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Descargar Spark 3.2.2

!wget -q https://archive.apache.org/dist/spark/spark-3.2.3/spark-3.2.3-bin-hadoop3.2.tgz

# Descomprimir el archivo descargado de Spark

!tar xf spark-3.2.3-bin-hadoop3.2.tgz

# Establecer las variables de entorno

import os

os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.2.3-bin-hadoop3.2"

# Instalar la librería findspark

!pip install -q findspark

# Instalar pyspark

!pip install -q pyspark
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 4.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 18.1 MB/s eta 0:00:00
  Building wheel for pyspark (setup.py) ... done
```

```
#Inicio una sesion de spark

import findspark
findspark.init()
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext
```

```
data = spark.read.parquet('./data/convertir')

data.printSchema()

data.show(truncate=False)
```

```
root
 |-- date: string (nullable = true)
 |-- timestamp: string (nullable = true)
 |-- date_str: string (nullable = true)
 |-- ts_str: string (nullable = true)

+----------+----------------------+----------+----------------+
|date      |timestamp             |date_str  |ts_str          |
+----------+----------------------+----------+----------------+
|2021-01-01|2021-01-01 20:10:50.723|01-01-2021|18-08-2021 46:58|
+----------+----------------------+----------+----------------+
```

Se ha guardado correctamente

```
from pyspark.sql.functions import col, to_date, to_timestamp
```

```
data1 = data.select(
    to_date(col('date')).alias('date1'),
    to_timestamp(col('timestamp')).alias('ts1'),
    to_date(col('date_str'), 'dd-MM-yyyy').alias('date2'),
    to_timestamp(col('ts_str'), 'dd-MM-yyyy mm:ss').alias('ts2')

)

data1.show(truncate=False)

data1.printSchema()
```

```
+----------+----------------------+----------+-------------------+
|date1     |ts1                   |date2     |ts2                |
+----------+----------------------+----------+-------------------+
|2021-01-01|2021-01-01 20:10:50.723|2021-01-01|2021-08-18 00:46:58|
+----------+----------------------+----------+-------------------+

root
 |-- date1: date (nullable = true)
 |-- ts1: timestamp (nullable = true)
 |-- date2: date (nullable = true)
 |-- ts2: timestamp (nullable = true)
```

```
#cambiar el formato de fecha
from pyspark.sql.functions import date_format

data1.select(
    date_format(col('date1'), 'dd-MM-yyyy')
).show()
```

```
+------------------------------+
|date_format(date1, dd-MM-yyyy)|
+------------------------------+
|                    01-01-2021|
+------------------------------+
```

```
#Cambio de df

df = spark.read.parquet('./data/calculo.parquet')

df.show()
```

```
+------+-------------+------------+-------------------+
|nombre|fecha_ingreso|fecha_salida|       baja_sistema|
+------+-------------+------------+-------------------+
|  Jose|   2021-01-01|  2021-11-14|2021-10-14 15:35:59|
|Mayara|   2021-02-06|  2021-11-25|2021-11-25 10:35:55|
+------+-------------+------------+-------------------+
```

```
#diferencia de fechas

from pyspark.sql.functions import datediff, months_between, last_day

df.select(
    col('nombre'),
    datediff(col('fecha_salida'), col('fecha_ingreso')).alias('dias'),
    months_between(col('fecha_salida'), col('fecha_ingreso')).alias('meses'),
    last_day(col('fecha_salida')).alias('ultimo_dia_mes')
).show()
```

```
+------+----+-----------+--------------+
|nombre|dias|      meses|ultimo_dia_mes|
+------+----+-----------+--------------+
|  Jose| 317|10.41935484|    2021-11-30|
|Mayara| 292| 9.61290323|    2021-11-30|
+------+----+-----------+--------------+
```

Se ha guardado correctamente

```
from pyspark.sql.functions import date_add, date_sub

df.select(
    col('nombre'),
    col('fecha_ingreso'),
    date_add(col('fecha_ingreso'), 14).alias('mas_14_dias'),
    date_sub(col('fecha_ingreso'), 1).alias('menos_1_dia')
).show()
```

```
+------+-------------+-----------+-----------+
|nombre|fecha_ingreso|mas_14_dias|menos_1_dia|
+------+-------------+-----------+-----------+
|  Jose|   2021-01-01| 2021-01-15| 2020-12-31|
|Mayara|   2021-02-06| 2021-02-20| 2021-02-05|
+------+-------------+-----------+-----------+
```

```
from pyspark.sql.functions import year, month, dayofmonth, dayofyear, hour, minute, second

df.select(
    col('baja_sistema'),
    year(col('baja_sistema')),
    month(col('baja_sistema')),
    dayofmonth(col('baja_sistema')),
    dayofyear(col('baja_sistema')),
    hour(col('baja_sistema')),
    minute(col('baja_sistema')),
    second(col('baja_sistema'))
).show()
```

```
+-------------------+------------------+-------------------+------------------------+-----------------------+-------------------+---
|       baja_sistema|year(baja_sistema)|month(baja_sistema)|dayofmonth(baja_sistema)|dayofyear(baja_sistema)|hour(baja_sistema)|mir
+-------------------+------------------+-------------------+------------------------+-----------------------+-------------------+---
|2021-10-14 15:35:59|              2021|                 10|                      14|                    287|                 15|
|2021-11-25 10:35:55|              2021|                 11|                      25|                    329|                 10|
+-------------------+------------------+-------------------+------------------------+-----------------------+-------------------+---
```

```
# Funciones para trabajo con strings

data = spark.read.parquet('./data/data.parquet')

data.show() #espacios a la izquierda y a la derecha

from pyspark.sql.functions import ltrim, rtrim, trim #para eliminar espacios

data.select(
    ltrim('nombre').alias('ltrim'),#eliminar espacio a la izquierda
    rtrim('nombre').alias('rtrim'),#eliminar espacio a la derecha
    trim('nombre').alias('trim')#eliminar espacio a ambos lados
).show()

from pyspark.sql.functions import col, lpad, rpad

data.select(
    trim(col('nombre')).alias('trim')
).select(
    lpad(col('trim'), 8, '-').alias('lpad'),
    rpad(col('trim'), 8, '=').alias('rpad')
).show()
```

```
+-------+
| nombre|
+-------+
| Spark |
+-------+

+------+------+-----+
| ltrim| rtrim| trim|
+------+------+-----+
|Spark | Spark|Spark|
+------+------+-----+

+--------+--------+
|    lpad|    rpad|
+--------+--------+
|---Spark|Spark===|
```



```
df1 = spark.createDataFrame([('Spark', 'es', 'maravilloso')], ['sujeto', 'verbo', 'adjetivo'])

df1.show()

from pyspark.sql.functions import concat_ws, lower, upper, initcap, reverse

df1.select(
    concat_ws(' ', col('sujeto'), col('verbo'), col('adjetivo')).alias('frase')
).select(
    col('frase'),
    lower(col('frase')).alias('minuscula'),
    upper(col('frase')).alias('mayuscula'),
    initcap(col('frase')).alias('initcap'),
    reverse(col('frase')).alias('reversa')
).show()
```

```
+------+-----+-----------+
|sujeto|verbo|   adjetivo|
+------+-----+-----------+
| Spark|   es|maravilloso|
+------+-----+-----------+

+--------------------+--------------------+--------------------+--------------------+--------------------+
|               frase|           minuscula|           mayuscula|             initcap|             reversa|
+--------------------+--------------------+--------------------+--------------------+--------------------+
|Spark es maravilloso|spark es maravilloso|SPARK ES MARAVILLOSO|Spark Es Maravilloso|osollivaram se krapS|
+--------------------+--------------------+--------------------+--------------------+--------------------+
```

```
from pyspark.sql.functions import regexp_replace

df2 = spark.createDataFrame([(' voy a casa por mis llaves',)], ['frase'])

df2.show(truncate=False)
```

```
#remplazar palabras
df2.select(
    regexp_replace(col('frase'), 'voy|por', 'ir').alias('nueva_frase')
).show(truncate=False)
```

```
+--------------------------+
|frase                     |
+--------------------------+
| voy a casa por mis llaves|
+--------------------------+

+------------------------+
|nueva_frase             |
+------------------------+
| ir a casa ir mis llaves|
+------------------------+
```

```
# Funciones para trabajo con colecciones

data = spark.read.parquet('./data/parquet/')

data.show(truncate=False)

data.printSchema()
```

```
+-----+-------------------------------------------+
|dia  |tareas                                     |
+-----+-------------------------------------------+
|lunes|[hacer la tarea, buscar agua, lavar el auto]|
+-----+-------------------------------------------+

root
 |-- dia: string (nullable = true)
 |-- tareas: array (nullable = true)
 |    |-- element: string (containsNull = true)
```

```
from pyspark.sql.functions import col, size, sort_array, array_contains
```

Se ha guardado correctamente

```
data.select(
    size(col('tareas')).alias('tamaño'),
    sort_array(col('tareas')).alias('arreglo_ordenado'),
    array_contains(col('tareas'), 'buscar agua').alias('buscar_agua')
).show(truncate=False)

from pyspark.sql.functions import explode

#Asociar tareas al día de la semana

data.select(
    col('dia'),
    explode(col('tareas')).alias('tareas')
).show()
```

```
+------+-------------------------------------------+-----------+
|tamaño|arreglo_ordenado                           |buscar_agua|
```

```
+------+-----------------------------------------+-----------+
|3     |[buscar agua, hacer la tarea, lavar el auto]|true       |
+------+-----------------------------------------+-----------+

+-----+--------------+
|  dia|        tareas|
+-----+--------------+
|lunes|hacer la tarea|
|lunes|   buscar agua|
|lunes| lavar el auto|
+-----+--------------+
```

```
# Formato JSON

json_df_str = spark.read.parquet('./data/json')

json_df_str.show(truncate=False)

json_df_str.printSchema()
```

```
+-------------------------------------------------------------------------+
|tareas_str                                                               |
+-------------------------------------------------------------------------+
|{"dia": "lunes","tareas": ["hacer la tarea","buscar agua","lavar el auto"]}|
+-------------------------------------------------------------------------+

root
 |-- tareas_str: string (nullable = true)
```

```
from pyspark.sql.types import StructType, StructField, StringType, ArrayType

#necesitamos crear un esquema json para poder leer el json

schema_json = StructType(
    [
     StructField('dia', StringType(), True),
     StructField('tareas', ArrayType(StringType()), True)
    ]
)

from pyspark.sql.functions import from_json, to_json

json_df = json_df_str.select(
    from_json(col('tareas_str'), schema_json).alias('por_hacer')
)

json_df.printSchema()
```

```
root
 |-- por_hacer: struct (nullable = true)
 |    |-- dia: string (nullable = true)
 |    |-- tareas: array (nullable = true)
 |    |    |-- element: string (containsNull = true)
```

```
Se ha guardado correctamente

s'),
    col('por_hacer').getItem('tareas').getItem(0).alias('primer_tarea') #obtener la primera tarea en la posición 0
).show(truncate=False)
```

```
+-------------+-----------------------------------------+--------------+
|por_hacer.dia|por_hacer.tareas                         |primer_tarea  |
+-------------+-----------------------------------------+--------------+
|lunes        |[hacer la tarea, buscar agua, lavar el auto]|hacer la tarea|
+-------------+-----------------------------------------+--------------+
```

```
#Inversamente: convertir un string en un json
json_df.select(
    to_json(col('por_hacer'))
).show(truncate=False)
```

```
+-------------------------------------------------------------------------+
|to_json(por_hacer)                                                       |
```

```
+--------------------------------------------------------------------------+
|{"dia":"lunes","tareas":["hacer la tarea","buscar agua","lavar el auto"]}|
+--------------------------------------------------------------------------+
```

```
# Funciones when, coalesce y lit

data = spark.read.parquet('./data/data2')

data.show()

from pyspark.sql.functions import col, when, lit, coalesce

data.select(
    col('nombre'),
    when(col('pago') == 1, 'pagado').when(col('pago') == 2, 'sin pagar').otherwise('sin iniciar').alias('pago')
).show()

data.select(
    coalesce(col('nombre'), lit('sin nombre')).alias('nombre')
).show()
```

```
+------+----+
|nombre|pago|
+------+----+
|  Jose|   1|
| Julia|   2|
| Katia|   1|
|  null|   3|
|  Raul|   3|
+------+----+
```

```
+------+-----------+
|nombre|       pago|
+------+-----------+
|  Jose|     pagado|
| Julia|  sin pagar|
| Katia|     pagado|
|  null|sin iniciar|
|  Raul|sin iniciar|
+------+-----------+
```

```
+----------+
|    nombre|
+----------+
|      Jose|
|     Julia|
|     Katia|
|sin nombre|
|      Raul|
+----------+
```

```
# Funciones definidas por el usuario UDF

def cubo(n):
    return n * n * n

from pyspark.sql.types import LongType

spark.udf.register('cubo', cubo, LongType())

spark.range(1,10).createOrReplaceTempView('df_temp')

                                    ubo FROM df_temp").show()
```



```
+---+----+
| id|cubo|
+---+----+
|  1|   1|
|  2|   8|
|  3|  27|
|  4|  64|
|  5| 125|
|  6| 216|
|  7| 343|
|  8| 512|
|  9| 729|
+---+----+
```

```
def bienvenida(nombre):
    return ('Hola {}'.format(nombre))
```

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

bienvenida_udf = udf(lambda x: bienvenida(x), StringType())

df_nombre = spark.createDataFrame([('Jose',), ('Julia',)], ['nombre'])

df_nombre.show()
```

```
+------+
|nombre|
+------+
|  Jose|
| Julia|
+------+
```

```
from pyspark.sql.functions import col

df_nombre.select(
    col('nombre'),
    bienvenida_udf(col('nombre')).alias('bie_nombre')
).show()
```

```
+------+----------+
|nombre|bie_nombre|
+------+----------+
|  Jose| Hola Jose|
| Julia|Hola Julia|
+------+----------+
```

```
@udf(returnType=StringType())
def mayuscula(s):
    return s.upper()

df_nombre.select(
    col('nombre'),
    mayuscula(col('nombre')).alias('may_nombre')
).show()
```

```
+------+----------+
|nombre|may_nombre|
+------+----------+
|  Jose|      JOSE|
| Julia|     JULIA|
+------+----------+
```

```
# Las UDF de pandas tienen un mejor rendimiento
import pandas as pd

from pyspark.sql.functions import pandas_udf

def cubo_pandas(a: pd.Series) -> pd.Series:
    return a * a * a

                                                  returnType=LongType())

x = pd.Series([1, 2, 3])

print(cubo_pandas(x))

df = spark.range(5)

df.select(
    col('id'),
    cubo_udf(col('id')).alias('cubo_pandas')
).show()
```

```
0     1
1     8
2    27
dtype: int64
+---+-----------+
| id|cubo_pandas|
+---+-----------+
```

```
|  0|          0|
|  1|          1|
|  2|          8|
|  3|         27|
|  4|         64|
+---+-----------+
```

```
# Funciones de ventana

import findspark
findspark.init()
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

df = spark.read.parquet('./data/data3')

df.show()

from pyspark.sql.window import Window
from pyspark.sql.functions import desc, row_number, rank, dense_rank, col

windowSpec = Window.partitionBy('departamento').orderBy(desc('evaluacion')) #particion por departamento
```

```
+-------+----+------------+----------+
| nombre|edad|departamento|evaluacion|
+-------+----+------------+----------+
| Lazaro|  45|      letras|        98|
|   Raul|  24|  matemática|        76|
|  Maria|  34|  matemática|        27|
|   Jose|  30|     química|        78|
| Susana|  51|     química|        98|
|   Juan|  44|      letras|        89|
|  Julia|  55|      letras|        92|
|  Kadir|  38|arquitectura|        39|
| Lilian|  23|arquitectura|        94|
|   Rosa|  26|      letras|        91|
|   Aian|  50|  matemática|        73|
|Yaneisy|  29|      letras|        89|
|Enrique|  40|     química|        92|
|    Jon|  25|arquitectura|        78|
|  Luisa|  39|arquitectura|        94|
+-------+----+------------+----------+
```

```
# row_number
#parada dar el número de filas secuencial

df.withColumn('row_number', row_number().over(windowSpec)).filter(col('row_number').isin(1,2)).show() #limitando a 2 filas, si no limita
```

```
+-------+----+------------+----------+----------+
| nombre|edad|departamento|evaluacion|row_number|
+-------+----+------------+----------+----------+
| Lilian|  23|arquitectura|        94|         1|
|  Luisa|  39|arquitectura|        94|         2|
| Lazaro|  45|      letras|        98|         1|
|  Julia|  55|      letras|        92|         2|
|   Raul|  24|  matemática|        76|         1|
|   Aian|  50|  matemática|        73|         2|
| Susana|  51|     química|        98|         1|
                                    92|         2|
                                    -------+----------+
```

Se ha guardado correctamente

```
# rank
# problema con los empates?? rangos que desaparecen
df.withColumn('rank', rank().over(windowSpec)).show()
```

```
+-------+----+------------+----------+----+
| nombre|edad|departamento|evaluacion|rank|
+-------+----+------------+----------+----+
| Lilian|  23|arquitectura|        94|   1|
|  Luisa|  39|arquitectura|        94|   1|
|    Jon|  25|arquitectura|        78|   3|
|  Kadir|  38|arquitectura|        39|   4|
| Lazaro|  45|      letras|        98|   1|
|  Julia|  55|      letras|        92|   2|
|   Rosa|  26|      letras|        91|   3|
|   Juan|  44|      letras|        89|   4|
```

```
|Yaneisy|  29|      letras|        89|   4|
|   Raul|  24|  matemática|        76|   1|
|   Aian|  50|  matemática|        73|   2|
|  Maria|  34|  matemática|        27|   3|
| Susana|  51|     química|        98|   1|
|Enrique|  40|     química|        92|   2|
|   Jose|  30|     química|        78|   3|
+-------+----+------------+----------+----+
```

```
# dense_rank
# soluciona los empates
df.withColumn('dense_rank', dense_rank().over(windowSpec)).show()
```

```
+-------+----+------------+----------+----------+
| nombre|edad|departamento|evaluacion|dense_rank|
+-------+----+------------+----------+----------+
| Lilian|  23|arquitectura|        94|         1|
|  Luisa|  39|arquitectura|        94|         1|
|    Jon|  25|arquitectura|        78|         2|
|  Kadir|  38|arquitectura|        39|         3|
| Lazaro|  45|      letras|        98|         1|
|  Julia|  55|      letras|        92|         2|
|   Rosa|  26|      letras|        91|         3|
|   Juan|  44|      letras|        89|         4|
|Yaneisy|  29|      letras|        89|         4|
|   Raul|  24|  matemática|        76|         1|
|   Aian|  50|  matemática|        73|         2|
|  Maria|  34|  matemática|        27|         3|
| Susana|  51|     química|        98|         1|
|Enrique|  40|     química|        92|         2|
|   Jose|  30|     química|        78|         3|
+-------+----+------------+----------+----------+
```

```
# Agregaciones con especificaciones de ventana
#no es necesario el orderby
windowSpecAgg = Window.partitionBy('departamento')

from pyspark.sql.functions import min, max, avg

(df.withColumn('min', min(col('evaluacion')).over(windowSpecAgg))
.withColumn('max', max(col('evaluacion')).over(windowSpecAgg))
.withColumn('avg', avg(col('evaluacion')).over(windowSpecAgg))
.withColumn('row_number', row_number().over(windowSpec))
 ).show()
```

```
+-------+----+------------+----------+---+---+------------------+----------+
| nombre|edad|departamento|evaluacion|min|max|               avg|row_number|
+-------+----+------------+----------+---+---+------------------+----------+
| Lilian|  23|arquitectura|        94| 39| 94|             76.25|         1|
|  Luisa|  39|arquitectura|        94| 39| 94|             76.25|         2|
|    Jon|  25|arquitectura|        78| 39| 94|             76.25|         3|
|  Kadir|  38|arquitectura|        39| 39| 94|             76.25|         4|
| Lazaro|  45|      letras|        98| 89| 98|              91.8|         1|
|  Julia|  55|      letras|        92| 89| 98|              91.8|         2|
|   Rosa|  26|      letras|        91| 89| 98|              91.8|         3|
|   Juan|  44|      letras|        89| 89| 98|              91.8|         4|
|Yaneisy|  29|      letras|        89| 89| 98|              91.8|         5|
|   Raul|  24|  matemática|        76| 27| 76|58.666666666666664|         1|
|   Aian|  50|  matemática|        73| 27| 76|58.666666666666664|         2|
|  Maria|  34|  matemática|        27| 27| 76|58.666666666666664|         3|
| Susana|  51|     química|        98| 78| 98| 89.33333333333333|         1|
|Enrique|  40|     química|        92| 78| 98| 89.33333333333333|         2|
|   Jose|  30|     química|        78| 78| 98| 89.33333333333333|         3|
+-------+----+------------+----------+---+---+------------------+----------+
```

Se ha guardado correctamente

```
# Catalyst Optimizer

#data = spark.read.parquet('./data/data3')
data = spark.read.parquet('./data/vuelos.parquet')

data.printSchema()

data.show()

from pyspark.sql.functions import col

nuevo_df = (data.filter(col('MONTH').isin(6,7,8))
            .withColumn('dis_tiempo_aire', col('DISTANCE') / col('AIR_TIME'))
).select(
```

```
    col('AIRLINE'),
    col('dis_tiempo_aire')
).where(col('AIRLINE').isin('AA', 'DL', 'AS'))

nuevo_df.explain(True)
```

0 s completado a las 20:22

Se ha guardado correctamente