






NahumFGz /
TareaLayla



<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

TareaLayla / clase_02 / practica_a_solucion.ipynb



 **NahumFGz** feat: ✨ Semana 4 terminado af1d94d · last week 

815 lines (815 loc) · 20.3 KB

Preview

Code

Blame



Raw









```
In [0]: # Importar las bibliotecas necesarias
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lit, avg, count, max, min, sum,

# Crear una instancia de SparkSession
spark = SparkSession.builder \
    .appName("DataFrame Manipulations") \
    .getOrCreate()

# Crear un DataFrame con los datos proporcionados
data = [("Alice", 25, "New York"),
        ("Bob", 30, "Los Angeles"),
        ("Charlie", 22, "Chicago")]

schema = ["Nombre", "Edad", "Ciudad"]
df = spark.createDataFrame(data, schema)
```

```
In [0]: # 1. Mostrar el DataFrame inicial
df.show()
```

Nombre	Edad	Ciudad
Alice	25	New York
Bob	30	Los Angeles
Charlie	22	Chicago

```
In [0]: # 2. Mostrar solo los nombres
df.select("Nombre").show()
```

Nombre
Alice
Bob
Charlie

```
In [0]: # 3. Filtrar personas cuya edad sea mayor o igual a 25
df.filter(col("Edad") >= 25).show()
```

Nombre	Edad	Ciudad
Alice	25	New York
Bob	30	Los Angeles

```
In [0]: # 4. Agregar una nueva columna "Pais" con valor constante "USA"
df = df.withColumn("Pais", lit("USA"))
df.show()
```

Nombre	Edad	Ciudad	Pais
Alice	25	New York	USA
Bob	30	Los Angeles	USA
Charlie	22	Chicago	USA

Nombre	Edad	Ciudad	Pais
Alice	25	New York	USA
Bob	30	Los Angeles	USA
Charlie	22	Chicago	USA

```
In [0]: # 5. Calcular el promedio de edad
df.select(avg("Edad").alias("PromedioEdad")).show()
```

PromedioEdad
25.666666666666668

```
In [0]: # 6. Ordenar por edad en orden descendente
df.orderBy(col("Edad").desc()).show()
```

Nombre	Edad	Ciudad	Pais
Bob	30	Los Angeles	USA
Alice	25	New York	USA
Charlie	22	Chicago	USA

```
In [0]: # 7. Agrupar por ciudad y contar personas
df.groupBy("Ciudad").agg(count("*").alias("Cantidad")).show()
```

Ciudad	Cantidad
New York	1
Los Angeles	1
Chicago	1

```
In [0]: # 8. Renombrar la columna "Nombre" a "NombreCompleto"
df = df.withColumnRenamed("Nombre", "NombreCompleto")
df.show()
```

NombreCompleto	Edad	Ciudad	Pais
Alice	25	New York	USA
Bob	30	Los Angeles	USA
Charlie	22	Chicago	USA

```
In [0]: # 9. Eliminar la columna "Edad"
df_sin_edad = df.drop("Edad")
df_sin_edad.show()
```

NombreCompleto	Ciudad	Pais
Alice	New York	USA
Bob	Los Angeles	USA
Charlie	Chicago	USA

```
|      Charlie |      Chicago |    USA |
+-----+-----+-----+
```

```
In [0]: # 10. Realizar una consulta SQL para seleccionar personas mayores de 20
df.createOrReplaceTempView("personas")
spark.sql("SELECT * FROM personas WHERE Edad > 20").show()
```

```
+-----+-----+-----+-----+
|NombreCompleto|Edad|      Ciudad|Pais|
+-----+-----+-----+-----+
|      Alice |  25 |   New York | USA |
|      Bob   |  30 | Los Angeles| USA |
|    Charlie |  22 |   Chicago | USA |
+-----+-----+-----+-----+
```

```
In [0]: # 11. Calcular la suma total de las edades
df.select(sum("Edad").alias("SumaTotalEdad")).show()
```

```
+-----+
|SumaTotalEdad|
+-----+
|          77 |
+-----+
```

```
In [0]: # 12. Calcular la edad mínima y máxima
df.select(min("Edad").alias("EdadMinima"), max("Edad").alias("EdadMaxima"))
```

```
+-----+-----+
|EdadMinima|EdadMaxima|
+-----+-----+
|        22 |        30 |
+-----+-----+
```

```
In [0]: # 13. Filtrar personas cuya ciudad sea "Chicago" y edad menor de 30
df.filter((col("Ciudad") == "Chicago") & (col("Edad") < 30)).show()
```

```
+-----+-----+-----+-----+
|NombreCompleto|Edad|      Ciudad|Pais|
+-----+-----+-----+-----+
|      Charlie |  22 |   Chicago | USA |
+-----+-----+-----+-----+
```

```
In [0]: # 14. Agregar una nueva columna "EdadDuplicada" que sea el doble de la
df = df.withColumn("EdadDuplicada", col("Edad") * 2)
df.show()
```

```
+-----+-----+-----+-----+-----+
|NombreCompleto|Edad|      Ciudad|Pais|EdadDuplicada|
+-----+-----+-----+-----+-----+
|      Alice |  25 |   New York | USA |          50 |
|      Bob   |  30 | Los Angeles| USA |          60 |
|    Charlie |  22 |   Chicago | USA |          44 |
+-----+-----+-----+-----+-----+
```

```
In [0]: # 15. Convertir las edades a meses
df = df.withColumn("EdadEnMeses", col("Edad") * 12)
```

```
df.show()
```

NombreCompleto	Edad	Ciudad	Pais	EdadDuplicada	EdadEnMeses
Alice	25	New York	USA	50	300
Bob	30	Los Angeles	USA	60	360
Charlie	22	Chicago	USA	44	264

```
In [0]: # 16. Contar el número total de personas
df.select(count("*").alias("TotalPersonas")).show()
```

TotalPersonas
3

```
In [0]: # 17. Filtrar personas cuya edad sea un número par
df.filter((col("Edad") % 2) == 0).show()
```

NombreCompleto	Edad	Ciudad	Pais	EdadDuplicada	EdadEnMeses
Bob	30	Los Angeles	USA	60	360
Charlie	22	Chicago	USA	44	264

```
In [0]: # 18. Calcular la cantidad de personas por rango de edades
df.withColumn("RangoEdad",
              when(col("Edad") <= 20, "0-20")
                .when((col("Edad") > 20) & (col("Edad") <= 40), "21-40")
                .when((col("Edad") > 40) & (col("Edad") <= 60), "41-60")
                .otherwise("61+")) \
  .groupBy("RangoEdad").count().show()
```

RangoEdad	count
21-40	3

```
In [0]: # 19. Contar cuántas personas tienen el mismo nombre
df.groupBy("NombreCompleto").count().show()
```

NombreCompleto	count
Alice	1
Bob	1
Charlie	1

```
In [0]: # 20. Concatenar "Nombre" y "Ciudad" en una nueva columna "InformacionPersonal"
df = df.withColumn("InformacionPersonal", concat_ws(", ", col("NombreCompleto"), col("Ciudad")))
df.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
+-----+
|NombreCompleto|Edad|Ciudad|Pais|EdadDuplicada|EdadEnMeses|Informac
ionPersonal|
+-----+-----+-----+-----+-----+-----+-----+
+-----+
|Alice|25|New York|USA|50|300|Alice, New York|
|Bob|30|Los Angeles|USA|60|360|Bob,
Los Angeles|
|Charlie|22|Chicago|USA|44|264|Charl
ie, Chicago|
+-----+-----+-----+-----+-----+-----+-----+
```