

Pyspark Exercises

1. Create a databricks community edition account
2. Create databricks folder under Worspace>>users>youremail

The screenshot shows the Databricks workspace interface. At the top, it displays 'Workspace > Users >' followed by the email address 'jemimajayaraman@gmail.com'. Below this is a 'Create' button. A table lists a single folder entry:

Name	Type	Owner	Created at
databrickstutorial	Folder	jemima aj	Aug 21, 202...

3. Create a table under catalog and load the bigmarket.sales file

The screenshot shows the Databricks Catalog interface. On the left, there's a sidebar with 'New', 'Workspace', 'Recents', 'Search', and 'Catalog' (which is selected). The main area is titled 'Data' and contains two sections: 'Databases' and 'Tables'. Both sections have a note: 'You need to create a cluster to access tables'. In the 'Tables' section, there is one entry: 'jilu.default.big_mart_sales'.

/Volumes/jemi/default/jilu/BigMart Sales.csv

/FileStore/tables/BigMart_Sales.csv

4. Data reading

The screenshot shows a Jupyter Notebook cell in Python. The code is:

```
dbutils.fs.ls('/FileStore/tables')
Out[9]: [FileInfo(path='dbfs:/FileStore/tables/BigMart_Sales.csv', name='BigMart_Sales.csv', size=869537, modificationTime=1755800992000)]
```

Below the cell, the output is shown:

```
df = (spark.read.format('csv').option('Inferschema',True).option('header',True).load('/FileStore/tables/BigMart_Sales.csv'))
(2) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: double ... 10 more fields]
```

5. Display the data

2 minutes ago (1s) 4 Python ⚡ ⚡ ⚡

```
df.display()
```

(1) Spark Jobs

Table +

A ^B _C Item_Identifier	1.2 Item_Weight	A ^B _C Item_Fat_Content	1.2 Item_Visibility	A ^B _C Item_Type
1 FDA15	9.3	Low Fat	0.016047301	Dairy

6. Upload the .json file and try to read it

/FileStore/tables/drivers.json

Just now (1s) 2 Python ⚡ ⚡ ⚡

```
df_json = spark.read.format('json').option('Inferschema',True)\n    .option('header',True)\n    .option('multiline',False).load('/FileStore/tables/drivers.json')
```

1 minute ago (1s) 3 Python ⚡ ⚡ ⚡

```
df_json.display()
```

(1) Spark Jobs

Table +

A ^B _C code	A ^B _C dob	1 ² ₃ driverId	A ^B _C driverRef	name
1 HAM	1985-01-07		1 hamilton	> {"forename": "Lewis", "surname": "Hamilton"}
2 HEI	1977-05-10		2 heidfeld	> {"forename": "Nick", "surname": "Heidfeld"}
3 ROS	1985-06-27		3 rosberg	> {"forename": "Nico", "surname": "Rosberg"}

7. Define Schema

SCHEMA DEFINITION

Just now (<1s) 9

```
df.printSchema()
```

```
root\n |-- Item_Identifier: string (nullable = true)\n |-- Item_Weight: double (nullable = true)\n |-- Item_Fat_Content: string (nullable = true)\n |-- Item_Visibility: double (nullable = true)\n |-- Item_Type: string (nullable = true)\n |-- Item_MRP: double (nullable = true)\n |-- Outlet_Identifier: string (nullable = true)\n |-- Outlet_Establishment_Year: integer (nullable = true)
```

Just now (<1s) 12

```
from pyspark.sql import SparkSession

# Define schema using DDL
my_ddl_schema = """
Item_Identifier STRING,
Item_Weight STRING,
Item_Fat_Content STRING,
Item_Visibility DOUBLE,
Item_Type STRING,
Item_MRP DOUBLE,
Outlet_Identifier STRING,
Outlet_Establishment_Year INT,
Outlet_Size STRING,
```

2 minutes ago (1s) 13

```
df = spark.read.format('csv') \
    .schema(my_ddl_schema) \
    .option('header', True) \
    .load('/FileStore/tables/Bigmart_Sales.csv')

df.display(5)
```

(1) Spark Jobs

```
df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: string ... 10 more fields]
```

Table +

	A _B C Item_Identifier	A _B C Item_Weight	A _B C Item_Fat_Content	1.2 Item_Visibility	A _B C Item_Type
1	FDA15	9.3	Low Fat	0.016047301	Dairy
2	DRC01	5.92	Regular	0.019278216	Soft Drinks
3	FDN15	17.5	Low Fat	0.016760075	Meat

8. Struct type schema

Just now (<1s)

```
from pyspark.sql.types import *
from pyspark.sql.functions import *
```

```
my_struct_schema = StructType([
    StructField('item_Identifier', StringType(), True),
    StructField('item_weight', StringType(), True),
    StructField('item_fat_content', StringType(), True),
    StructField('item_visibility', StringType(), True),
    StructField('item_mrp', StringType(), True),
    StructField('outlet_identifier', StringType(), True),
    StructField('outlet_establishment_year', StringType(), True),
    StructField('outlet_size', StringType(), True),
    StructField('outlet_location_type', StringType(), True),
    StructField('outlet_type', StringType(), True),
    StructField('item_outlet_sales', StringType(), True)
])
```

9. Reset using InferSchema

Just now (<1s) 17

```
df = spark.read.format('csv') \
    .schema(my_struct_schema) \
    .option('header', True) \
    .load('/FileStore/tables/Bigmart_Sales.csv')
```

df.printSchema()

```
root
 |-- Item_Identifier: string (nullable = true)
 |-- Item_Weight: string (nullable = true)
 |-- Item_Fat_Content: string (nullable = true)
```

PYSPARK TRANSFORMATION

10.Pyspark Transformation -SELECT

Select 1st 3 coloums

23
df.select('Item_Identifier','Item_weight','Item_Fat_content').display()
24
df.select(col('Item_Identifier'),col('Item_weight'),col('Item_Fat_content')).display()

Item_Identifier	Item_weight	Item_Fat_content
FDA15	9.3	Low Fat
DRC01	5.92	Regular
FDN15	17.5	Low Fat

Item_Identifier	Item_weight	Item_Fat_content
FDA15	9.3	Low Fat
~	~	~

11. ALIAS

df.select(col('Item_Identifier').alias('Item_ID')).display()
1) Spark Jobs

Item_ID
FDA15
DRC01

12.FILTER

- 1) Filter the data with fat content = Regular ✓
- 2) Slice the data with item type = Soft Drinks and weight < 10
- 3) Fetch the data with Tier in (Tier1 or Tier 2) and Outlet Size is Null ✓

Scenerio 1:

df.filter(col('Item_Fat_Content')=='Regular').display()
1) Spark Jobs

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
DRC01	5.92	Regular	0.019278216	Soft Drinks
FDX07	19.2	Regular	0	Fruits and Vegetabl
FDP36	10.395	Regular	0	Baking Goods
FDN10	17.65	Regular	0.012741089	Snack Foods

Scenerio 2:

df.filter((col('Item_Type')=='Soft Drinks') & (col('Item_Weight')< 10)).display()
1) Spark Jobs

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
DRC01	5.92	Regular	0.019278216	Soft Drinks
DR711	8.85	Regular	0.113123893	Soft Drinke

Scenerio 3:

```
df.filter((col('Outlet_size').isNull()) & (col('Outlet_Location_Type').isin('Tier 1','Tier 2'))).display()
▶ (1) Spark Jobs
```

Table +

MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type
96.9726	OUT045	2002	null	Tier 2
187.8214	OUT017	2007	null	Tier 2
45.906	OUT017	2007	null	Tier 2

13. Withcoloumrenamed- to change the column

```
▶ Just now (1s) 36
df.withColumnRenamed('Item_weight','Item_wt').display()
▶ (1) Spark Jobs
```

Table +

Item_Identifier	Item_wt	Item_Fat_Content
FDA15	9.3	Low Fat
DRC01	5.92	Regular

14. Withcolumn

Scenerio 1 create coloum Flag and put a constantvalue

```
▶ Just now (1s) 39
df=df.withColumn('flag',lit('new'))
df.display()
▶ (1) Spark Jobs
df: pyspark.sql.DataFrame = [Item_Identifier: string, Item_Weight: double ... 11 more fields]
```

Table +

Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	flag
High	Tier 3	Supermarket Type1	994.7052	new
Medium	Tier 3	Supermarket Type2	556.6088	new

Scenerio 2: create a column based on 2 other coloumns multiply function

```
▶ Just now (1s) 41
df.withColumn('multiply',col('Item_weight')* col('Item_MRP')).display()
▶ (1) Spark Jobs
Filter displayed data
```

Table +

Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	flag	multiply
Tier 1	Supermarket Type1	3735.138	new	2323.225560000000...
Tier 3	Supermarket Type2	443.4228	new	285.753663999999...

Scenerio3: Modify the existing coloumn in tem fat LF -low fat and Reg the regular

regexp_replace()

```
Just now (1s) 43 Python
df.withColumn('Item_Fat_Content', regexp_replace (col ('Item_Fat_Content'), "Regular", 'Reg')).\
| withColumn('Item_Fat_Content', regexp_replace (col ('Item_Fat_Content'), "Low Fat", 'LF')). display()
```

▶ (1) Spark Jobs

Table +

	A ^B c Item_Identifier	1.2 Item_Weight	A ^B c Item_Fat_Content	1.2 Item_Visibility	A ^B c Item_Type
1	FDA15	9.3	LF	0.016047301	Dairy
2	DRC01	5.92	Reg	0.019278216	Soft Drinks

15. Typecast

```
df = df.withColumn('Item_Weight', col('Item_Weight').cast(StringType()))
```

▶ df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: string ... 11 columns]

```
Just now (<1s) 46
df.printSchema()
```

root
|-- Item_Identifier: string (nullable = true)
|-- Item_Weight: string (nullable = true)

16.sorting

```
Just now (1s) 50 Python
df.sort(col('Item_Visibility').asc()).display()
```

▶ (1) Spark Jobs

Table +

	A ^B c Item_Identifier	A ^B c Item_Weight	A ^B c Item_Fat_Content	1.2 Item_Vis...	⋮	A ^B c Item_Type
121	FDL20	17.1	Low Fat	0	0	Fruits and Vegetables
122	DRK49	null	LF	0	0	Soft Drinks

Just now (1s) 51

```
#Scenario 2
df.sort(col('Item_weight').desc()).display()
```

▶ (1) Spark Jobs

Table +

	A _c Item_Identifier	A _c Item_Weight	A _c Item_Fat_Content
1	FDR13	9.895	Regular
2	DRD49	9.895	Low Fat

Just now (1s) 52

```
#Scenario3: Based on multiple column
df.sort(['Item_Weight','Item_Visibility'],ascending = [0,1]).display()
```

▶ (1) Spark Jobs

Table +

	A _c Item_Identifier	A _c Item_Weight	A _c Item_Fat_Content	1.2 Item_Visibility
1	DRD49	9.895	Low Fat	0.168780385
2	DRD49	9.895	Low Fat	0.16817143

Just now (1s) 53

```
#Scenario4: Based on multiple column one ascending one descending
df.sort(['Item_Weight','Item_Visibility'],ascending = [0,1]).display()
```

▶ (1) Spark Jobs

Table +

	A _c Item_Identifier	A _c Item_Weight	A _c Item_Fat_Content	1.2 Item_Visibility
1	FDR13	9.895	Regular	0.028696932
2	FDR13	9.895	Regular	0.028765486

17.LIMIT

Just now (1s) 55

```
df.limit(5).display()
```

▶ (1) Spark Jobs

Table +

	A _c Item_Identifier	A _c Item_Weight	A _c Item_Fat_Content	1.2 Item_Visibility	A _c Item_Type
1	FDR13	9.895	Low Fat	0.028696932	Dairy
2	DRC01	5.92	Regular	0.019278216	Soft Drinks
3	FDN15	17.5	Low Fat	0.016760075	Meat
4	FDX07	19.2	Regular	0	Fruits and Vegetables

18.DROP

Just now (1s) 57 Python

```
df.drop('Item_Visibility').display()
```

▶ (1) Spark Jobs

Table +

	A ^B C Item_Identifier	A ^B C Item_Weight	A ^B C Item_Fat_Content	A ^B C Item_Type	1.2 Item_MRP
1	FDA15	9.3	Low Fat	Dairy	249.8092

Just now (1s) 58 Python

```
# SCenerio 2
df.drop('Item_Visibility','Item_type').display()
```

▶ (1) Spark Jobs

Table +

	A ^B C Item_Identifier	A ^B C Item_Weight	A ^B C Item_Fat_Content	1.2 Item_MRP	A ^B C Outlet_Identif
1	FDA15	9.3	Low Fat	249.8092	OUT049
2	DRC01	5.92	Regular	48.2692	OUT018

Just now (3s) 59 Python

```
#Scenerio3:DROP DUPLICATES
df.dropDuplicates().display()
```

▶ (2) Spark Jobs

Table +

	A ^B C Item_Identifier	A ^B C Item_Weight	A ^B C Item_Fat_Content	1.2 Item_Visibility
1	FDR12	null	Regular	0.031382044
2	NCW29	14.0	Low Fat	0.028907832

Just now (2s) 60 Python

```
#Scenerio3:DROP DUPLICATES only in 1 column
df.drop_duplicates(subset=['Item_type']).display()
```

▶ (2) Spark Jobs

Table +

	A ^B C Item_Identifier	A ^B C Item_Weight	A ^B C Item_Fat_Content	1.2 Item_Visibility
1	FDP36	10.395	Regular	0
2	FDO23	17.85	Low Fat	0

Just now (1s) 61 Python

```
#Scenerio4:DROP DUPLICATES without subset
df.distinct().display()
```

▶ (2) Spark Jobs

Table +

	A ^B C Item_Identifier	A ^B C Item_Weight	A ^B C Item_Fat_Content	1.2 Item_Visibility
1	FDR12	null	Regular	0.031382044
2	NCW29	14.0	Low Fat	0.028907832

INTERMEDIATE FUNCTIONS

19.UNIONS

Prepare the dataframes

```
▶ ✓ 1 minute ago (2s) 65

data1 = [('1','kad'),
          ('2','sid')]
schema1 = 'id STRING, name STRING'

df1 = spark.createDataFrame(data1,schema1)

data2 = [('3','rahul'),
          ('4','jas')]
schema2 = 'id STRING, name STRING'

df2 = spark.createDataFrame(data2,schema2)
```

▶ df1: pyspark.sql.dataframe.DataFrame = [id: string, name: string]
▶ df2: pyspark.sql.dataframe.DataFrame = [id: string, name: string]

The screenshot shows the PySpark notebook interface with three code cells and their corresponding output.

- Cell 1:** Displays the creation of two dataframes, df1 and df2, from lists of tuples. The schema for both is defined as 'id STRING, name STRING'.
- Cell 2:** Displays the execution of `df1.display()` and `df2.display()`, both resulting in 3 Spark Jobs.
- Cell 3:** Displays the execution of `#scenerio 1- UNION df1.union(df2).display()`, resulting in 3 Spark Jobs.

DataFrames:

- df1:**

	A ^B id	A ^B name
1	1	kad
2	2	sid
- df2:**

	A ^B id	A ^B name
1	3	rahul
2	4	jas
- Union Result:**

	A ^B id	A ^B name
1	1	kad
2	2	sid
3	3	rahul
4	4	jas

Just now (1s)

```
# altert the dataframe
data1 = [ ('kad','1'), ('sid','2')]
schema1 = 'name STRING, id STRING'
df1 = spark.createDataFrame(data1,schema1)
df1.display()
```

df1.union(df2).display()

(3) Spark Jobs

	A _c name	A _c id
1	kad	1
2	sid	2
3	3	rahul
4	4	jas

UNION by name will help combine accordingly

Just now (1s)

```
# Union by name
df1.unionByName(df2).display()
```

(3) Spark Jobs

Table +

	A _c name	A _c id
1	kad	1
2	sid	2
3	rahul	3
4	ias	4

20. STRING initcap(),upper(),Lower()

initcap()

```
df.select(initcap('Item_type')).display()
```

(1) Spark Jobs

Table +

	A _c initcap(Item_type)
1	Dairy
2	Soft Drinks
3	Meat
4	Fruits And Vegetables
5	Household

lower()

```
df.select(lower('Item_type')).di
```

(1) Spark Jobs

Table +

	A _c lower(Item_type)
1	dairy
2	soft drinks
3	meat
4	fruits and vegetables
5	household

upper()

```
df.select(upper('Item_type')).display()
```

(1) Spark Jobs

Table +

	A _c upper(Item_type)
1	DAIRY
2	SOFT DRINKS

ALIAS with Lower

```
df.select(upper('Item_Type').alias('upper_Item_Type')).display()
```

(1) Spark Jobs

Table +

	A _c upper_Item_Type
1	DAIRY
2	SOFT DRINKS

21.DATE

Just now (1s) 78 Python

```
#current_date
df.withColumn('Current_Date',current_date()).display()
```

▶ (1) Spark Jobs

Table + Q Y

	A ^B c Outlet_Location_Type	A ^B c Outlet_Type	1.2 Item_Outlet_Sales	A ^B c flag	Current_Date
1	Tier 1	Supermarket Type1	3735.138	new	2025-08-23
2	Tier 3	Supermarket Type2	443.4228	new	2025-08-23

Just now (1s) 79 Python

```
# DATE_add()
df = df.withColumn('week_after',date_add('Current_date',7))
df.display()
```

▶ (1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: string ... 12 more fields]

Table + Q Y

	A ^B c Outlet_Location_Type	A ^B c Outlet_Type	1.2 Item_Outlet_Sales	A ^B c flag	week_after
1	Tier 1	Supermarket Type1	3735.138	new	2025-08-30
2	Tier 3	Supermarket Type2	443.4228	new	2025-08-30

Just now (1s) 80 Python

```
# DATE_sub()
df.withColumn('week_before',date_sub('Current_date',7))
df.display()
```

▶ (1) Spark Jobs

Table + Q Y

	_Type	A ^B c Outlet_Type	1.2 Item_Outlet_Sales	A ^B c flag	week_after	week_befor
1		Supermarket Type1	3735.138	new	2025-08-30	2025-08-16
2		Supermarket Type2	443.4228	new	2025-08-30	2025-08-16

DATE_add() still you want to subtract
df.withColumn('week_before',date_sub('Current_date',-7))
df.display()

▶ (1) Spark Jobs

Table + Q Y

	_Type	A ^B c Outlet_Type	1.2 Item_Outlet_Sales	A ^B c flag	week_after	week_before
1		Supermarket Type1	3735.138	new	2025-08-30	2025-08-16
2		Supermarket Type2	443.4228	new	2025-08-30	2025-08-16

```
# Date DIFF

df = df.withColumn('datediff', datediff('week_after', 'curr_date'))

df.display()

▶ (1) Spark Jobs
▶ df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: string ... 15 more fields]

Table +
```

	Item_Outlet_Sales	flag	curr_date	week_after	week_before	datediff
1	3735.138	new	2025-08-23	2025-08-30	2025-08-16	7
2	443.4228	new	2025-08-23	2025-08-30	2025-08-16	7

```
#DATE FORMAT

df = df.withColumn('week_before', date_format('week_before', 'dd-MM-yyyy'))

df.display()

(1) Spark Jobs
▶ df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: string ... 15 more fields]
```

Table +

	Item_Outlet_Sales	flag	curr_date	week_after	week_before
1	3735.138	new	2025-08-23	2025-08-30	16-08-2025
2	443.4228	new	2025-08-23	2025-08-30	16-08-2025

22. Handiling the NULL values

1 minute ago (s) 85 Python

```
# dropping the Nulls which is in all column
df.dropna('all').display()
```

▶ (1) Spark Jobs

Table +

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
1	FDA15	9.3	Low Fat	0.016047301	Dairy
2	DRC01	5.92	Regular	0.019278216	Soft Drinks

Just now (1s) 86 P

```
# dropping the records in any column
df.dropna('any').display()
```

▶ (1) Spark Jobs

Table +

	A ^B _C Item_Identifier	A ^B _C Item_Weight	A ^B _C Item_Fat_Content	1.2 Item_Visibility
1	FDA15	9.3	Low Fat	0.016047301
2	DRC01	5.92	Regular	0.019278216

```
# drop records with subset
df.dropna(subset=['Outlet_Size']).display()
```

▶ (1) Spark Jobs

Table +

	1.2 Item_MRP	A ^B _C Outlet_Identifier	1 ² ₃ Outlet_Establishment_Year	A ^B _C Outlet_Size
1	249.8092	OUT049		1999 Medium
2	48.2692	OUT018		2009 Medium

Just now (1s) 89 P

```
# Replacing/Fill the null value
df.fillna('NotAvailable').display()
```

▶ (1) Spark Jobs

Table +

	A ^B _C Item_Identifier	A ^B _C Item_Weight	A ^B _C Item_Fat_Content	1.2 Item_Visibility
63	FDF09	NotAvailable	Low Fat	0.012090074
64	FDY40	NotAvailable	Regular	0.15028599
65	FDY45	NotAvailable	Low Fat	0.026015519

Just now (1s) 90 Python trash

```
#Replacing/Fill the null value- using subset on Outlet_Size column
df.fillna('NotAvailable',subset=['Outlet_size']).display()
```

▶ (1) Spark Jobs

Table + Q Y

	pe	1.2 Item_MRP	A ^B C Outlet_Identifier	1 ² 3 Outlet_Establishment_Year	A ^B C Outlet_Size
47	Hygiene	153.3024	OUT045		2002 NotAvailable
48		265.2226	OUT045		2002 NotAvailable

23. SPLIT and Indexing

SPLIT and Indexing

```
▶ Just now (1s) 92 Python trash
```

```
# Split into list
df.withColumn('Outlet_Type',split('Outlet_type',' ')).display()
```

▶ (1) Spark Jobs

Table +

_Year	A ^B C Outlet_Size	A ^B C Outlet_Location_Type	Outlet_Type
1	1999	Medium	Tier 1
2	2009	Medium	Tier 3

Just now (1s) 93 Python trash

```
# Indexing with Split
df.withColumn('Outlet_Type',split('Outlet_type',' ')[1]).display()
```

▶ (1) Spark Jobs

Table +

r	A ^B C Outlet_Size	A ^B C Outlet_Location_Type	A ^B C Outlet_Type
1	1999	Medium	Tier 1
2	2009	Medium	Tier 3

23. EXPLODE

Just now (1s) 95 Python

```
# Split into list in df_exp
df_exp = df.withColumn('Outlet_Type', split('Outlet_type', " ")).display()
```

▶ (1) Spark Jobs

Table + Q

	A ^B _C Outlet_Size	A ^B _C Outlet_Location_Type	A ^B _C Outlet_Type	1.2 Item_Outlet_Sales
1	1999 Medium	Tier 1	> ["Supermarket", "Ty...]	3735.13;
2	2009 Medium	Tier 3	> ["Supermarket", "Ty...]	443.422;

Just now (1s) 96

```
#Explode
df_exp = df.withColumn('Outlet_Type', split('Outlet_type', ' '))
df_exp.withColumn('Outlet_Type', explode('Outlet_Type')).display()
```

▶ (1) Spark Jobs

▶ df_exp: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: string ... 15 more

Table +

	Item_Year	A ^B _C Outlet_Size	A ^B _C Outlet_Location_Type	A ^B _C Outlet_Type
1	1999	Medium	Tier 1	Supermarket
2	1999	Medium	Tier 1	Type1

24.ARRAY contains

Just now (1s) 98 Python

```
df_exp.withColumn('Type1_flag', array_contains('Outlet_Type', 'Type1')).display()
```

▶ (1) Spark Jobs

Table + Q Y E [

	A ^B _C flag	A ^B _C curr_date	A ^B _C week_after	A ^B _C week_before	A ^B _C datediff	A ^B _C Type1_flag
1	8 new	2025-08-23	2025-08-30	16-08-2025	7	true
2	8 new	2025-08-23	2025-08-30	16-08-2025	7	false

A ^B _C Item_Type	1.2 Item_MRP	A ^B _C Outlet_Identifier
Dairy	249.8092	OUT049
Soft Drinks	48.2692	OUT018
Meat	141.6118	OUT049
Fruits and Vegetables	182.095	OUT010
Household	53.8614	OUT013
Baking Goods	51.4008	OUT018
Snack Foods	57.6588	OUT013
Snack Foods	107.7622	OUT027
Frozen Foods	96.9726	OUT045
Frozen Foods	187.8214	OUT017

25.Groupby

Just now (2s) 101

```
#Scenario 1: Total MRP for all item types
df.groupBy('Item_type').agg(sum('Item_MRP')).display()
```

(2) Spark Jobs

Table +

	A ^B Item_type	1.2 sum(Item_MRP)
1	Starchy Foods	21880.027399999995
2	Baking Goods	81894.73640000001
3	Breads	35379.11979999999

Just now (1s) 102

```
#Scenario 2: Average MRP for all item types
df.groupBy('Item_type').agg(avg('Item_MRP')).display()
```

(2) Spark Jobs

Table +

	A ^B Item_type	1.2 avg(Item_MRP)
1	Starchy Foods	147.83802297297294
2	Baking Goods	126.38076604938273

Just now (1s) 103

```
#Scenario 3 Item type_group Outlet size calculate MRP sum and rename the column
df.groupBy('Item_Type','Outlet_Size').agg(sum('Item_MRP').alias('Total_MRP')).display()
```

(2) Spark Jobs

Table +

	A ^B Item_Type	A ^B Outlet_Size	1.2 Total_MRP
1	Starchy Foods	Medium	7124.136199999997
2	Fruits and Vegetables	Medium	59047.217200000014

Just now (1s) 104

```
#Scenario 4 in scenerion 3 want both total MRP and avg MRP
df.groupBy('Item_Type','Outlet_Size').agg(sum('Item_MRP'),avg('Item_MRP')).display()
```

(2) Spark Jobs

Table +

	A ^B Item_Type	A ^B Outlet_Size	1.2 sum(Item_MRP)	1.2 avg(Item_MRP)
1	Starchy Foods	Medium	7124.136199999997	148.4195041666666
2	Fruits and Vegetables	Medium	59047.217200000014	142.9714702179177

26.collect_List

```
data = [('user1','book1'), ('user1','book2'), ('user2','book2'), ('user2','book4'), ('user3','book1')]

schema = 'user string, book string'

df_book = spark.createDataFrame(data,schema)

df_book.display()
```

#	user	book
1	user1	book1
2	user1	book2
3	user2	book2
4	user2	book4
5	user3	book1

```
df_book.groupBy('user').agg(collect_list('book')).display()
```

▶ (2) Spark Jobs

Table +

	A ^B c user	Ⓐ collect_list(book)
1	user1	> ["book1","book2"]
2	user2	> ["book2","book4"]
3	user3	> ["book1"]

27.PIVOT

```
df.groupby('Item_Type').pivot('Outlet_Size').agg(avg('Item_MRP')).display()
```

▶ (7) Spark Jobs

Just now (2s) 109 Python

Table +

	A ^B c Item_Type	1.2 null	1.2 High	1.2 Medium	1.2 Small
1	Starchy Foods	140.480004651162...	158.157073684210...	148.4195041666666	150.2701736842105
2	Breads	139.048616666666...	133.75896	140.8610385542169	145.5236507042254
3	Baking Goods	126.669398918918...	129.202043835616...	126.178568472906...	125.213363636363...
4	Fruits and Vegetables	142.575160458452...	145.572870422535...	142.9714702179177	148.313369512195...

28. WHEN-OTERWISE

WHEN-OTHERWISE

Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
Dairy ✓	249.8092	OUT049	1999	Medium
Soft Drinks ✗	48.2692	OUT018	2009	Medium

Just now (1s) 111 Python ⚡

```
df= df.withColumn('veg_flag', when(col('Item_Type')=='Meat','non_veg').otherwise('veg')).display()
```

▶ (1) Spark Jobs

Table +

	flag	curr_date	week_after	week_before	datediff	veg_flag
1	.138	new	2025-08-25	2025-09-01	18-08-2025	7 veg
2	.228	new	2025-08-25	2025-09-01	18-08-2025	7 veg
3	7.27	new	2025-08-25	2025-09-01	18-08-2025	7 non_veg

```
1 #Scenario 2 -if item VEG <100 -not expensive >100 expensive
2 df.withColumn('veg_exp_flag',when((col('veg_flag')=='Veg') & (col('Item_MRP')<100),'Veg_Inexpensive')
3 \ .when((col('veg_flag')=='Veg') & (col('Item_MRP')>100),'Veg_Expensive')\
4 .otherwise('Non_Veg')).display()
```

Table +

	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	flag	veg_flag	veg_exp_flag
1	Tier 1	Supermarket Type1	3735.138	new	Veg	Veg_Expensive
2	Tier 3	Supermarket Type2	443.4228	new	Veg	Veg_Inexpensive

29.Joins

```
# Create a dataframe
dataj1 = [(1,'gaur','d01'),
          (2,'kit','d02'),
          (3,'sam','d03'),
          (4,'tim','d03'),
          (5,'aman','d05'),
          (6,'nad','d06')]

schemaj1 = 'emp_id STRING, emp_name STRING, dept_id STRING'

df1 = spark.createDataFrame(dataj1,schemaj1)

dataj2 = [(('d01','HR'),),
          ('d02','Marketing'),
          ('d03','Accounts'),
          ('d04','IT'),
          ('d05','Finance')]

schemaj2 = 'dept_id STRING, department STRING'
```

Just now (<1s)

df1.display()

(3) Spark Jobs

A^B_c emp_id	A^B_c emp_name	A^B_c dept_id	A^B_c dept_id	A^B_c department
1	gaur	d01	1	HR
2	kit	d02	2	Marketing
3	sam	d03	3	Accounts
4	tim	d03	4	IT
5	aman	d05	5	Finance
6	nad	d06		

Inner join

```
df1.join(df2,df1['dept_id']==df2 ['dept_id'],'inner').display()
```

(3) Spark Jobs

INNER JOIN

DF1

DF2

A^B_c emp_id	A^B_c emp_name	A^B_c dept_id	A^B_c dept_id	A^B_c department
1	gaur	d01	1	HR
2	kit	d02	2	Marketing
3	sam	d03	3	Accounts
4	tim	d03	4	Accounts
5	aman	d05	5	Finance

LEFT JOIN

www.youtube.com - To exit full screen, press [Esc]

LEFT JOIN

DF1

DF2

A^B_c emp_id	A^B_c emp_name	A^B_c dept_id	A^B_c dept_id	A^B_c department
1	gaur	d01	1	HR
2	kit	d02	2	Marketing
3	sam	d03	3	Accounts
4	tim	d03	4	Accounts
5	aman	d05	5	Finance
6	nad	d06	null	null

df1.join(df2,df1['dept_id']==df2 ['dept_id'],'left').display()

(6) Spark Jobs

RIGHT JOIN

RIGHT JOIN

```
# Right join
df1.join(df2,df1['dept_id']== df2['dept_id'],'right').display()
```

▶ (4) Spark Jobs

Table +

	A _c emp_id	A _c emp_name	A _c dept_id	A _c dept_id	A _c department
1	gaur		d01	d01	HR
2	kit		d02	d02	Marketing
3	sam		d03	d03	Accounts
4	tim		d03	d03	Accounts
5	aman		d05	d04	IT
6	nad		d06	d05	Finance

ANTI JOIN

```
#antijoin
```

```
df1.join(df2,df1['dept_id']== df2['dept_id'],'anti').display()
```

▶ (4) Spark Jobs

Table +

	A _c emp_id	A _c emp_name	A _c dept_id
1	6	nad	d06

29. WINDOWS FUNCTION

WINDOW FUNCTIONS

•ROW NUMBER()



•RANK()

•DENSE_RANK()

1 minute ago (1s) 123 Python ⚡ ⌂ ⋮

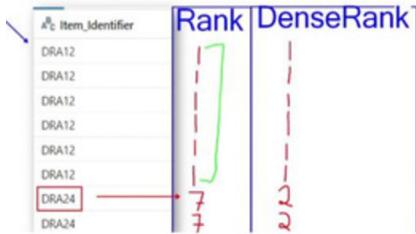
```
from pyspark.sql.window import Window
df.withColumn('rowCol',row_number().over(Window.orderBy('Item_Identifier'))).display()
```

▶ (1) Spark Jobs

Table +

	A _c Outlet_Size	A _c Outlet_Location_Type	A _c Outlet_Type	1.2 Item_Outlet_Sales	1 ² 3 rowCol
1	null	Tier 2	Supermarket Type1	2552.6772	1
2	null	Tier 2	Supermarket Type1	3829.0158	2
3	High	Tier 3	Supermarket Type1	2552.6772	3

WINDOW FUNCTIONS



Just now (1s) 124 Python ⚡ ⚡ ⚡

```
#Rank and Dense Rank
df.withColumn('rank',rank().over(Window.orderBy(col('Item_identifier')))).display()
```

▶ (1) Spark Jobs

Table +

	A _c Outlet_Size	A _c Outlet_Location_Type	A _c Outlet_Type	1.2 Item_Outlet_Sales	1 ² ₃ rank
5	09	Medium	Tier 3	Supermarket Type2	850.8924
6	98	null	Tier 3	Grocery Store	283.6308
7	07	null	Tier 2	Supermarket Type1	1146.5076
8	35	Small	Tier 1	Grocery Store	491.3604

#Rank with descending order
df.withColumn('rank',rank().over(Window.orderBy(col('Item_identifier').desc()))).display()

▶ (1) Spark Jobs

Table +

	A _c Item_Identifier	1.2 Item_Weight	A _c Item_Fat_Content	1.2 Item_Visibility	A _c
1	NCZ54	14.65	Low Fat	0	Hot

Just now (1s) 126 Python ⚡ ⚡ ⚡

```
#Rank and dense rank with descending order
df.withColumn('rank',rank().over(Window.orderBy(col('Item_identifier').desc())))\n    .withColumn('denserank',dense_rank().over(Window.orderBy(col('Item_identifier').desc()))).display()\n()
```

▶ (2) Spark Jobs

Table +

	A _c Outlet_Location_Type	A _c Outlet_Type	1.2 Item_Outlet_Sales	1 ² ₃ rank	1 ² ₃ denserank
5	Tier 1	Grocery Store	162.4552	1	1
6	Tier 3	Supermarket Type2	2599.2832	1	1
7	Tier 1	Supermarket Type1	7148.0288	1	1
8	Tier 3	Supermarket Type2	1884.214	8	2

WINDOW FUNCTIONS

• Cumulative Sum

Item_Type	Item_MRP	CumSum
Soft Drinks	140.3154	140
Soft Drinks	141.6154	281
Soft Drinks	142.3154	-

```
# Cumulative Sum

df.withColumn('cumsum',sum('Item_MRP').over(Window.orderBy('Item_Type'))).display()
```

▶ (2) Spark Jobs

Table +

Q Y E I

Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	cumsum
Medium	Tier 3	Supermarket Type2	556.6088	81894.73640000001
Medium	Tier 3	Supermarket Type3	4064.0432	81894.73640000001
Small	Tier 1	Grocery Store	214.3876	81894.73640000001

```
#preceding
df.withColumn('cumsum',sum('Item_MRP').over(Window.orderBy('Item_Type')).rowsBetween(Window.unboundedPreceding,Window.currentRow))).display()
```

▶ (2) Spark Jobs

Table +

Q Y E □

Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	cumsum
Medium	Tier 3	Supermarket Type2	556.6088	51.4008
Medium	Tier 3	Supermarket Type3	4064.0432	195.9452
Small	Tier 1	Grocery Store	214.3876	303.639
Small	Tier 1	Supermarket Type1	2576.646	364.261

#following

```
df.withColumn('totalsum',sum('Item_MRP').over(Window.orderBy('Item_Type')).rowsBetween(Window.unboundedPreceding,Window.unboundedFollowing))).display()
```

(2) Spark Jobs

Table +

Q Y E

Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	totalsum
Medium	Tier 3	Supermarket Type2	556.6088	1201681.48080000..
Medium	Tier 3	Supermarket Type3	4064.0432	1201681.48080000..

30. User defined functions

```
Just now (<1s) Just now (<1s)  
def my_func(x):  
    return x*x  
# convert in python pdf  
my_udf= udf(my_func)  
  
Just now (2s) 134 Python  
df.withColumn('mynewcol',my_udf('Item_MRP')).display()  
▶ (1) Spark Jobs  
  
Table +  


|   | Outlet_Size | Outlet_Location_Type | Outlet_Type       | Item_Outlet_Sales | mynewcol            |
|---|-------------|----------------------|-------------------|-------------------|---------------------|
| 1 | Medium      | Tier 1               | Supermarket Type1 | 3735.138          | 62404.6364046400... |
| 2 | Medium      | Tier 3               | Supermarket Type2 | 443.4228          | 2329.91566863999... |


```

31. DATA WRITING -----> SERVING LAYER

Read-----> Transformation -----> Data Writing

```
Just now (3s) 136  
df.write.format('csv')  
    .save('/FileStore/tables/CSV/data.csv')  
  
Just now (1s) 137  
#append  
df.write.format('csv')  
    .mode('append')  
    .save('/FileStore/tables/CSV/data.csv')  
  
▶ (1) Spark Jobs
```

```
Just now (1s) 138  
#append with path  
df.write.format('csv')  
    .mode('append')  
    .option('path','/FileStore/tables/CSV/data.csv')  
    .save()  
  
▶ (1) Spark Jobs
```

▶ ✓ 1 minute ago (2s)

139

```
#overwrite
df.write.format('csv')\
.mode('overwrite')\
.option('path','/FileStore/tables/CSV/data.csv')\
.save()
```

▶ (1) Spark Jobs

```
1 #error
2 df.write.format('csv')\
3 .mode('error')\
4 .option('path','/FileStore/tables/CSV/data.csv')\
5 .save()
```

! > AnalysisException: Path dbfs:/FileStore/tables/CSV/data.csv already exists.

▶ ✓ Just now (<1s)

141

```
#Ignore
df.write.format('csv')\
.mode('ignore')\
.option('path','/FileStore/tables/CSV/data.csv')\
.save()
```

32.PARQUET FILE FORMAT

PARQUET FILE FORMAT

ROW-BASED

emp_id	emp_name	dept_id
1	gaur	d01
2	kit	d02

1 gaur d01 2 kit d02

COLUMNAR

emp_id	emp_name	dept_id
1	gaur	d01
2	kit	d02
3	sam	d03
4	tim	d03
5	aman	d05

12345 gaur kit sam tim

▶ ✓ Just now (5s)

14

```
df.write.format('parquet')\
.mode('overwrite')\
.option('path','/FileStore/tables/CSV/data.csv')\
.save()
```

33.TABLE

```
▶ Just now (14s)

df.write.format('parquet')\
.mode('overwrite')\
.saveAsTable('my_table')

df.display()

▶ (1) Spark Jobs
```

Table +

	A ^B _C Item_Identifier	1.2 Item_Weight	A ^B _C Item_Fat_Content	1.2 Item_Visibility	A ^B _C Item_Type
1	FDA15	9.3	Low Fat	0.016047301	Dairy
2	DRC01	5.92	Regular	0.019278216	Soft Drinks

33. SPARK SQL

```
▶ Last execution failed

1 # Table as view- create a temp view
2
3 df.createTempView('my_view')
4

▶ Just now (<1s) 149 SQL ⚡
```

%sql

```
select * from my_view where Item_Fat_content = 'Low Fat'

▶ (1) Spark Jobs
```

_sqldf: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: double ... 10 more fields]

Table +

	A ^B _C Item_Identifier	1.2 Item_Weight	A ^B _C Item_Fat_Content	1.2 Item_Visibility	A ^B _C Item_Type
1	FDA15	9.3	Low Fat	0.016047301	Dairy
2	FDN15	17.5	Low Fat	0.016760075	Meat

```
1 # Reconvert SQL data to dataframe
2 df_sql= spark.sql("select * from my_view where Item_Fat_content = 'Low Fat'")
3 df_sql.display()
```

▶ (1) Spark Jobs

df_sql: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: double ... 10 more fields]

Table +

	A ^B _C Item_Identifier	1.2 Item_Weight	A ^B _C Item_Fat_Content	1.2 Item_Visibility	A ^B _C Item_Type
1	FDA15	9.3	Low Fat	0.016047301	Dairy
2	FDN15	17.5	Low Fat	0.016760075	Meat
3	NCD19	8.93	Low Fat	0	Household
4	FDP10	null	Low Fat	0.127469857	Snack Foods