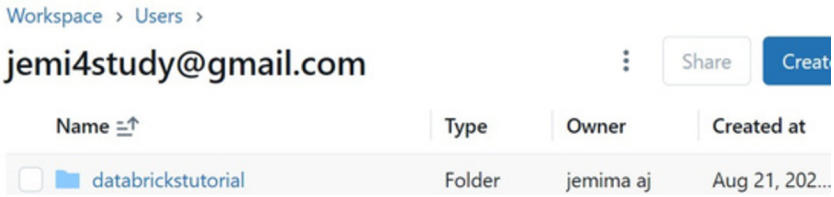
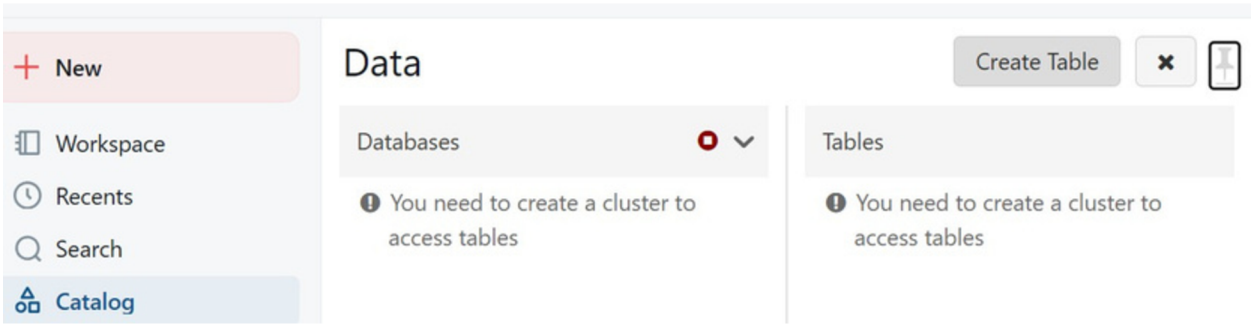
Pyspark Exercises

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



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

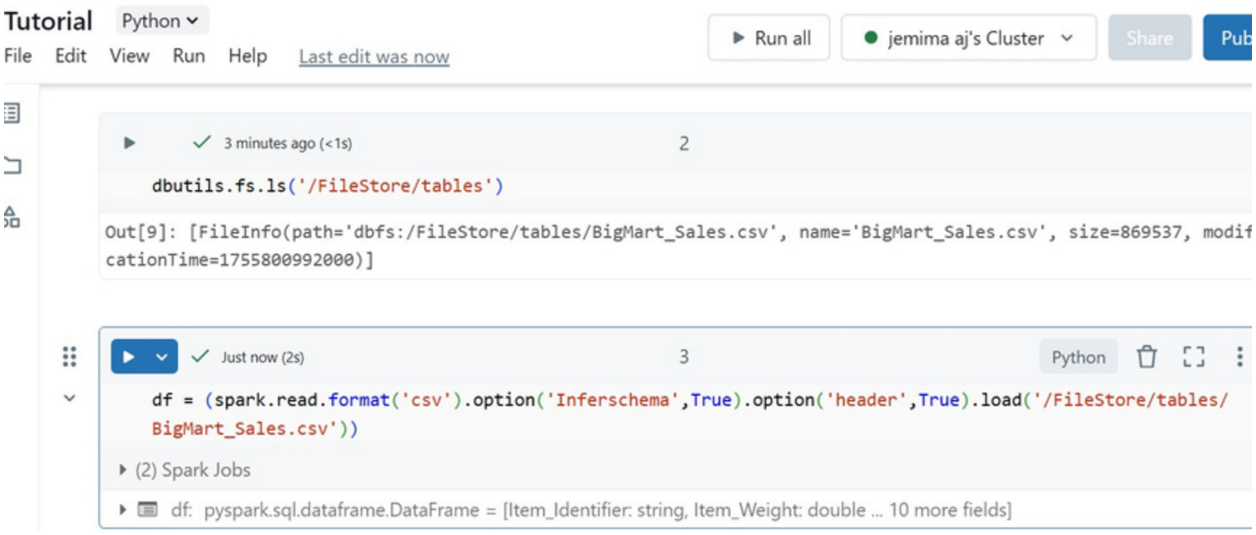


jilu.default.big_mart_sales

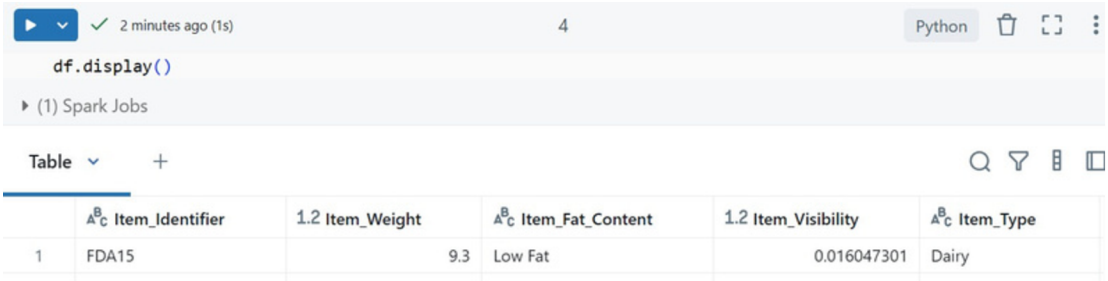
/Volumes/jemi/default/jilu/BigMart Sales.csv

/FileStore/tables/BigMart_Sales.csv

4. Data reading

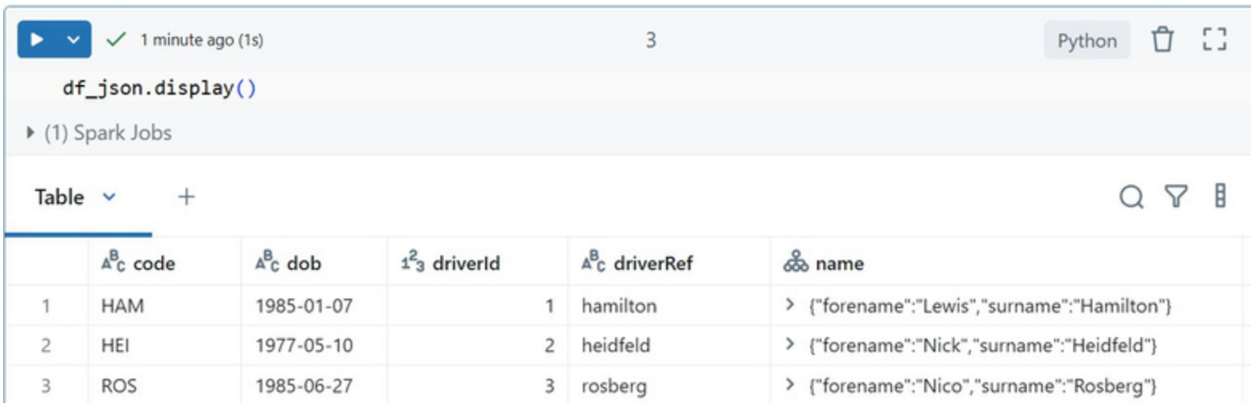


5. Display the data



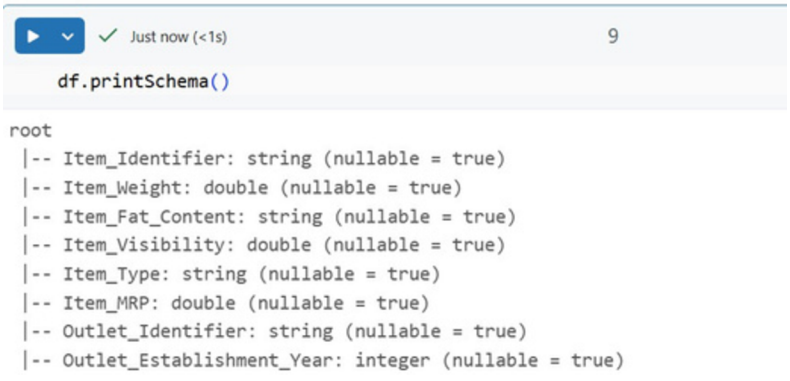
6. Upload the .json file and try to read it

/FileStore/tables/drivers.json



7. Define Schema

SCHEMA DEFINITION



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

	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
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_strct_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 Inferschma

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

```
df.select('Item_Identifier','Item_weight','Item_Fat_content').display()
```

(1) Spark Jobs

Table

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

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

(1) Spark Jobs

Table

	Item_Identifier	Item_weight	Item_Fat_content
1	FDA15	9.3	Low Fat

11. ALIAS

```
df.select(col('Item_Identifier').alias('Item_ID')).display()
```

(1) Spark Jobs

Table

	Item_ID
1	FDA15
2	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

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
1	DRC01	5.92	Regular	0.019278216	Soft Drinks
2	FDX07	19.2	Regular	0	Fruits and Vegetabl
3	FDP36	10.395	Regular	0	Baking Goods
4	FDQ10	13.65	Regular	0.012741089	Snack Foods

Scenerio 2:

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

(1) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
1	DRC01	5.92	Regular	0.019278216	Soft Drinks
2	DRZ11	8.85	Regular	0.113123893	Soft Drinks

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
1	96.9726	OUT045	2002	null	Tier 2
2	187.8214	OUT017	2007	null	Tier 2
3	45.906	OUT017	2007	null	Tier 2

13. Withcoloumrenamed- to change the coloumn

Just now (1s) 36

```
df.withColumnRenamed('Item_weight','Item_wt').display()
```

▸ (1) Spark Jobs

Table +

	Item_Identifier	Item_wt	Item_Fat_Content
1	FDA15	9.3	Low Fat
2	DRC01	5.92	Regular

14.Withcoloumn

Scenerio 1 create coloum Flag and put a constantvalue

Just now (1s) 39 Python

```
df=df.withColumn('flag',lit('new'))
df.display()
```

▸ (1) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [Item_Identifier: string, Item_Weight: double ... 11 more fields]

Table +

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

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

Just now (1s) 41 Python

```
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
1	Tier 1	Supermarket Type1	3735.138	new	2323.22556000000...
2	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

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	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

```
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

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visi...	Item_Type
121	FDL20	17.1	Low Fat	0	Fruits and Vegetables
122	DRK49	null	LF	0	Soft Drinks

Just now (1s) 51

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

(1) Spark Jobs

Table +

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

Just now (1s) 52

```
#Scenerio3: Based on multiple colomn
df.sort(['Item_Weight','Item_Visibility'],ascending = [0,0]).display()
```

(1) Spark Jobs

Table +

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

Just now (1s) 53

```
#Scenerio4: Based on multiple colomn one ascending one descending
df.sort(['Item_Weight','Item_Visibility'],ascending = [0,1]).display()
```

(1) Spark Jobs

Table +

	Item_Identifier	Item_Weight	Item_Fat_Con...	Item_Visibility
1	FDR13	9.895	Regular	0.028696932
2	FDR13	9.895	Regular	0.028765486

17.LIMIT

Just now (1s) 55 Python

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

(1) Spark Jobs

Table +

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
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

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

(1) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Type	Item_MRP
1	FDA15	9.3	Low Fat	Dairy	249.8092

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

(1) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_MRP	Outlet_Identif
1	FDA15	9.3	Low Fat	249.8092	OUT049
2	DRC01	5.92	Regular	48.2692	OUT018

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

(2) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
1	FDR12	null	Regular	0.031382044
2	NCW29	14.0	Low Fat	0.028907832

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

(2) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
1	FDP36	10.395	Regular	0
2	FDO23	17.85	Low Fat	0

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

(2) Spark Jobs

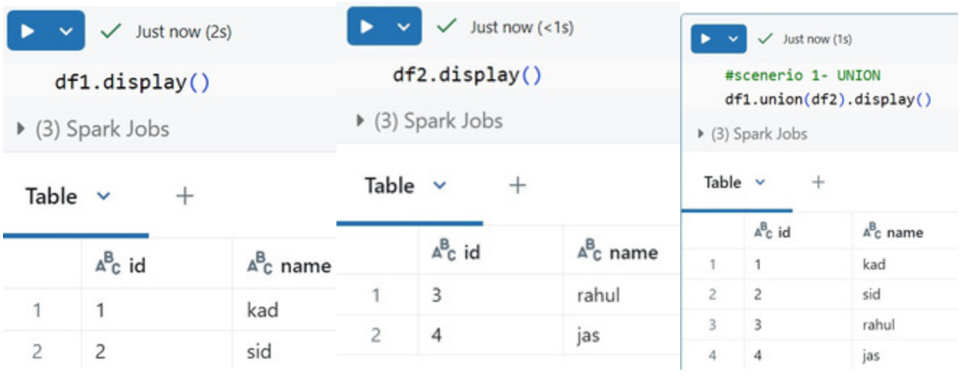
Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
1	FDR12	null	Regular	0.031382044
2	NCW29	14.0	Low Fat	0.028907832

INTERMEDIATE FUNCTIONS

19.UNIONS

Prepare the dataframes



```
# 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

Table

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

UNION by name will help combine accordingly

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

(3) Spark Jobs

Table

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

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

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

(1) Spark Jobs

Table

	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

	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

	upper(Item_type)
1	DAIRY
2	SOFT DRINKS
3	MEAT

```
# ALIAS with LOwer
df.select(upper('Item_Type').alias('upper_Item_Type')).display()
```

(1) Spark Jobs

Table

	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

	Outlet_Location_Type	Outlet_Type	Item_Outlet_...	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

	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	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

	_Type	Outlet_Type	Item_Outlet_Sales	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

	_Type	Outlet_Type	Item_Outlet_Sales	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

```python
# 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

```python
#DATE FORMAT

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

df.display()
```

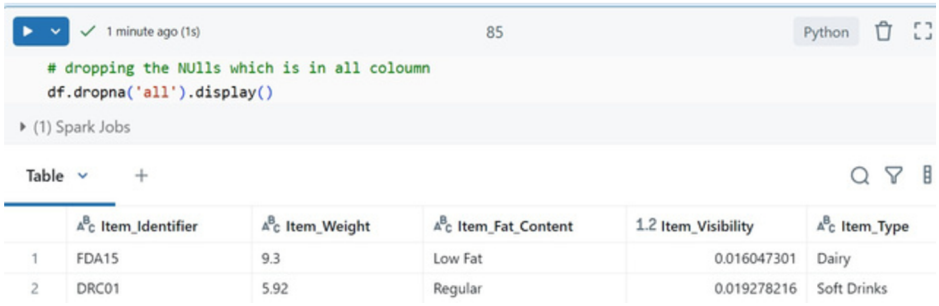
(1) Spark Jobs

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



Just now (1s) 86

```
# dropping the recors in any coloumn
df.dropna('any').display()
```

(1) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
1	FDA15	9.3	Low Fat	0.016047301
2	DRC01	5.92	Regular	0.019278216

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

(1) Spark Jobs

Table

	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
1	249.8092	OUT049	1999	Medium
2	48.2692	OUT018	2009	Medium

Just now (1s) 89

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

(1) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	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

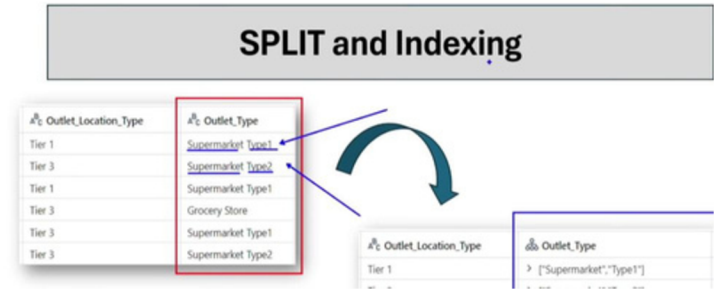
```
#Replacing/Fill the null value- using subset on Outlet_Size coloumn
df.fillna('NotAvailable',subset=['Outlet_size']).display()
```

(1) Spark Jobs

Table

	pe	1.2 Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
47	Hygiene	153.3024	OUT045	2002	NotAvailable
48		265.2226	OUT045	2002	NotAvailable

23. SPLIT and Indexing



Just now (1s) 92

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

(1) Spark Jobs

Table

	_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
1	1999	Medium	Tier 1	["Supermarket","Ty...
2	2009	Medium	Tier 3	["Supermarket","Ty...

Just now (1s) 93

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

(1) Spark Jobs

Table

	r	Outlet_Size	Outlet_Location_Type	Outlet_Type
1	1999	Medium	Tier 1	Type1
2	2009	Medium	Tier 3	Type2

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

		Outlet_Size	Outlet_Location_Type	Outlet_Type	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

	ıment_Year	Outlet_Size	Outlet_Location_Type	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()
```

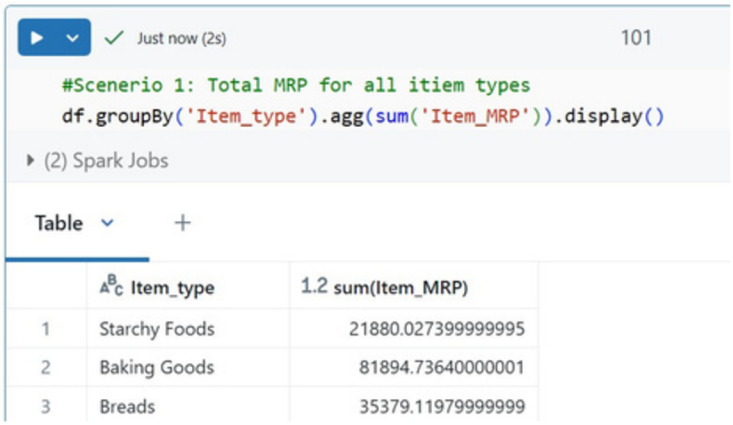
(1) Spark Jobs

Table

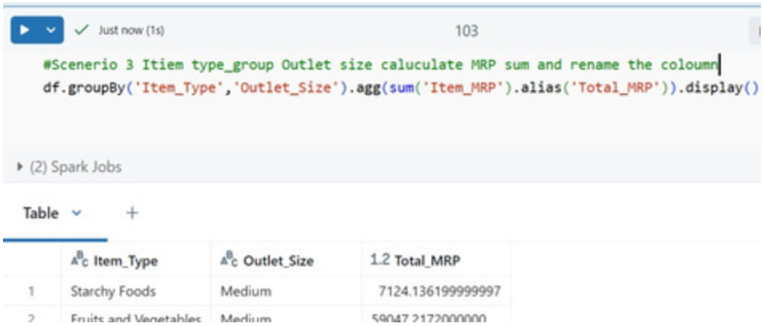
		flag	curr_date	week_after	week_before	datediff	Type1_flag
1	8	new	2025-08-23	2025-08-30	16-08-2025	7	true
2	8	new	2025-08-23	2025-08-30	16-08-2025	7	false

Item_Type	Item_MRP	Outlet_Identifier
Dairy	249.8092	OUT049
Soft Drinks	48.2692	OUT018
Meat	141.618	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
#Scenerio 1: Total MRP for all itiem types
df.groupBy('Item_type').agg(sum('Item_MRP')).display()
(2) Spark Jobs
Table
Item_type
sum(Item_MRP)
1 Starchy Foods 21880.027399999995
2 Baking Goods 81894.73640000001
3 Breads 35379.11979999999


Just now (1s)
102
#Scenerio 2: Average MRP for all itiem types
df.groupBy('Item_type').agg(avg('Item_MRP')).display()
(2) Spark Jobs
Table
Item_type
avg(Item_MRP)
1 Starchy Foods 147.83802297297294
2 Baking Goods 126.38076604938273


Just now (1s)
103
#Scenerio 3 Itiem type_group Outlet size caluculate MRP sum and rename the coloumn
df.groupBy('Item_Type','Outlet_Size').agg(sum('Item_MRP').alias('Total_MRP')).display()
(2) Spark Jobs
Table
Item_Type
Outlet_Size
Total_MRP
1 Starchy Foods Medium 7124.136199999997


Just now (1s)
104
#Scenerio 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
Item_Type
Outlet_Size
sum(Item_MRP)
avg(Item_MRP)
1 Starchy Foods Medium 7124.136199999997 148.4195041666666
2 Fruits and Vegetables Medium 59047.217200000014 142.9714702179177

26.collect_List

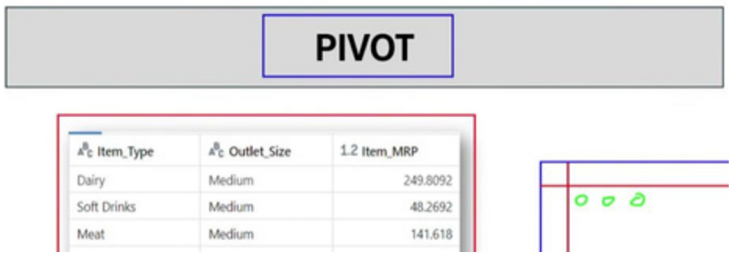

```
df_book.groupBy('user').agg(collect_list('book')).display()
```

(2) Spark Jobs

Table +

	user	collect_list(book)
1	user1	["book1","book2"]
2	user2	["book2","book4"]
3	user3	["book1"]

27.PIVOT



Just now (2s) 109 Python

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

(7) Spark Jobs

Table +

	Item_Type	null	High	Medium	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

Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Si
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_fl
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

```
#Scenerio 2 -if item VEg <100 -nont expensive >100 expensive
df.withColumn('veg_exp_flag',when(((col('veg_flag')=='Veg') & (col('Item_MRP')<100)),'Veg_Inexpensive')
\
                .when((col('veg_flag')=='Veg') & (col('Item_MRP')>100),'Veg_Expensive')\
                .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'
```

```
df1.display()
```

(3) Spark Jobs

Table

	emp_id	emp_name	dept_id
1	1	gaur	d01
2	2	kit	d02
3	3	sam	d03
4	4	tim	d03
5	5	aman	d05
6	6	nad	d06

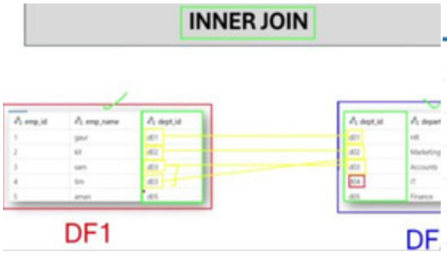
```
df2.display()
```

(3) Spark Jobs

Table

	dept_id	department
1	d01	HR
2	d02	Marketing
3	d03	Accounts
4	d04	IT
5	d05	Finance

Inner join



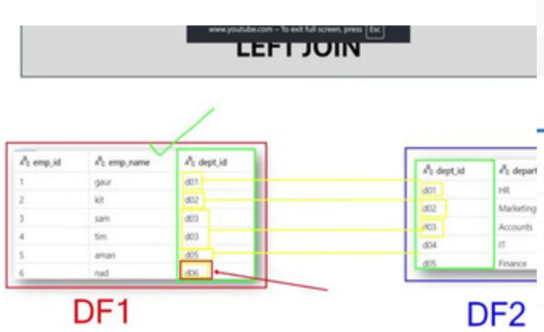

```
df1.join(df2,df1['dept_id']==df2 ['dept_id'],'inner').display()
```

(3) Spark Jobs

Table

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

LEFT JOIN



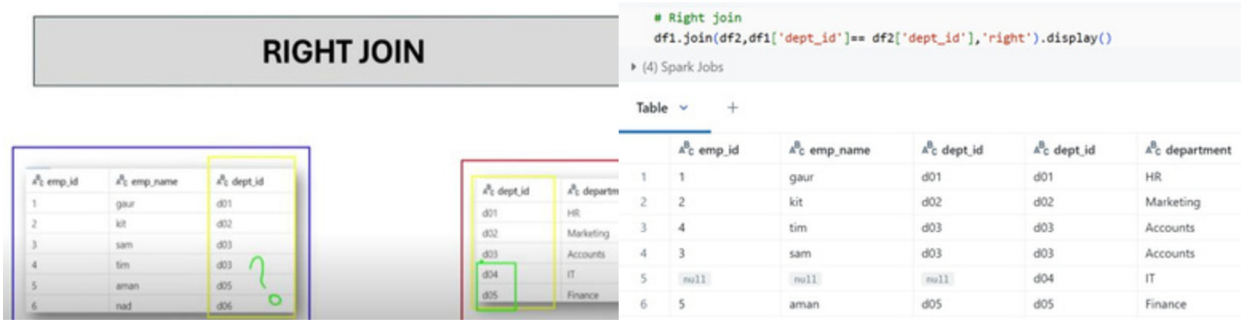

```
df1.join(df2,df1['dept_id']==df2 ['dept_id'],'left').display()
```

(6) Spark Jobs

Table

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

RIGHT JOIN



ANTI JOIN

```
#antijoin

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

(4) Spark Jobs

Table

	emp_id	emp_name	dept_id
1	6	nad	d06

29. WINDOWS FUNCTION

1 minute ago (1s) 123 Python

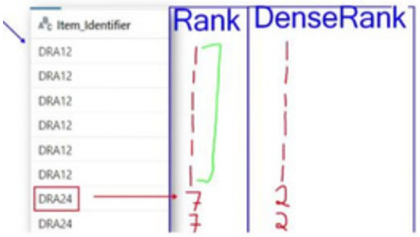
```
from pyspark.sql.window import Window
df.withColumn('rowCol',row_number().over(Window.orderBy('Item_Identifier'))).display()
```

(1) Spark Jobs

Table

	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	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

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

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

(1) Spark Jobs

Table

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	
1	NCZ54	14.65	Low Fat	0	Hou

Just now (1s) 126 Python

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

(2) Spark Jobs

Table

		Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	rank	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

Item_Type	Item_MRP
Soft Drinks	140.3154
Soft Drinks	141.6154
Soft Drinks	142.3154

CumSum
140
281.

• Cumulative Sum

```
# Cumulative Sum

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

(2) Spark Jobs

Table

	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	cumsum
1	Medium	Tier 3	Supermarket Type2	556.6088	81894.73640000001
2	Medium	Tier 3	Supermarket Type3	4064.0432	81894.73640000001
3	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

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

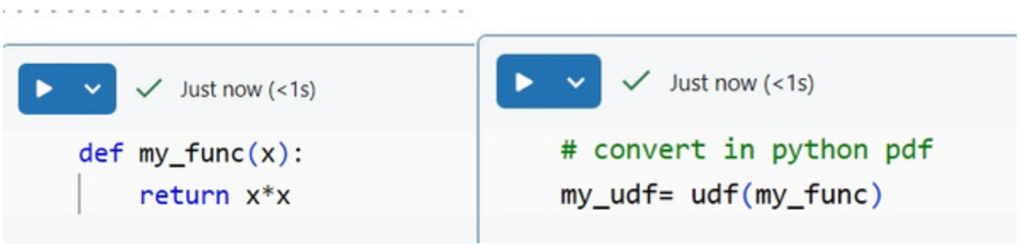
```
#following

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

(2) Spark Jobs

Table

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

30.User defined functions



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

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

✓ 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

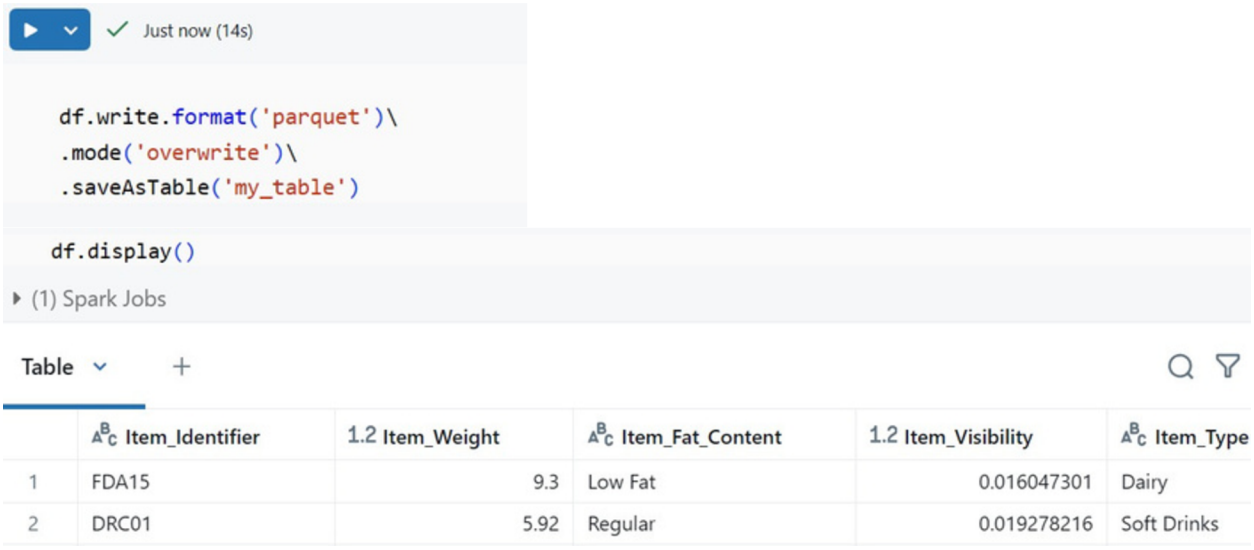
```
#Ignore
df.write.format('csv')\
.mode('ignore')\
.option('path','/FileStore/tables/CSV/data.csv')\
.save()
```

32.PARQUET FILE FORMAT

✓ Just now (5s) 14

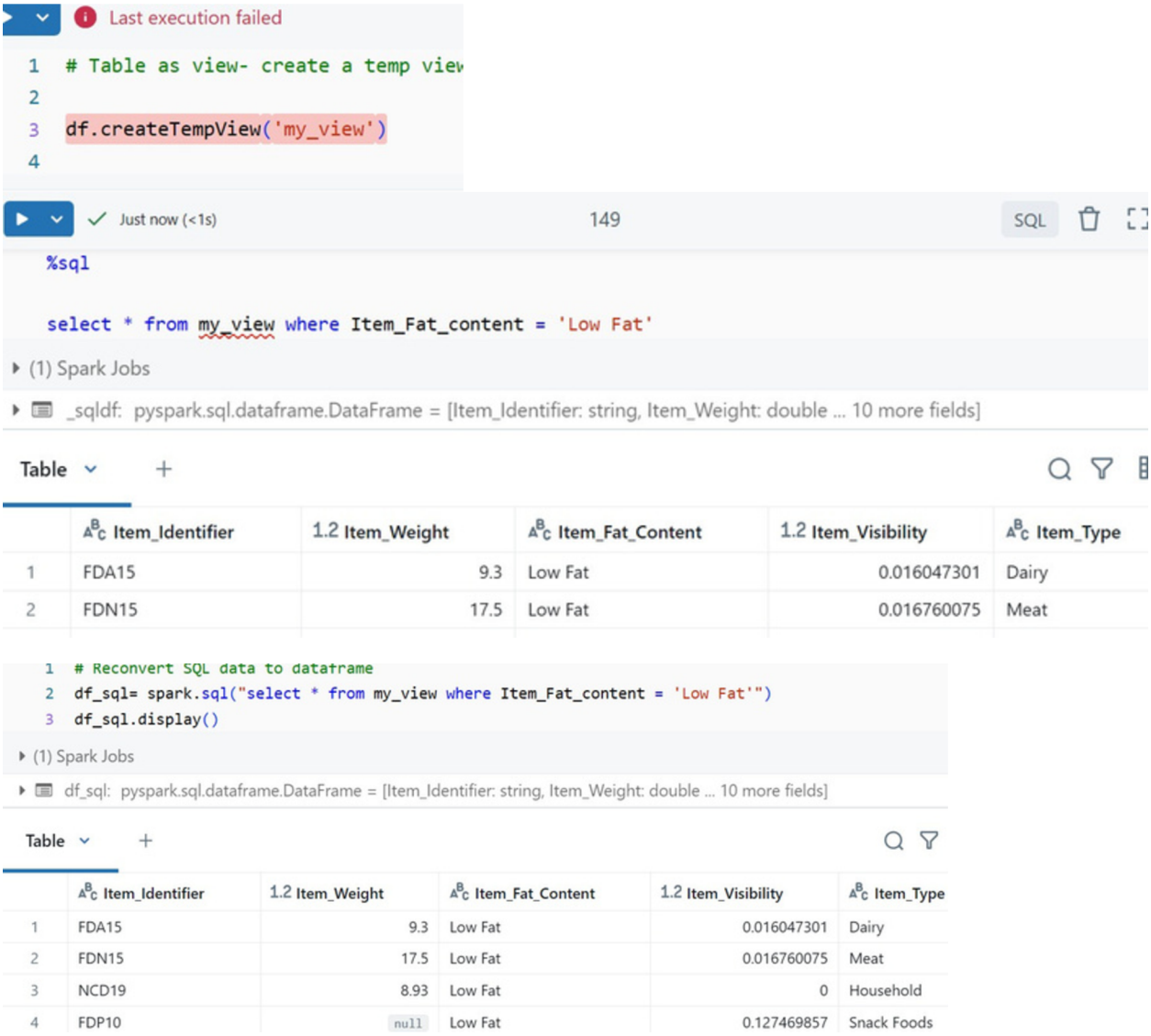
```
df.write.format('parquet')\
.mode('overwrite')\
.option('path','/FileStore/tables/CSV/data.csv')\
.save()
```

33.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

33. SPARK SQL



	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type
1	FDA15	9.3	Low Fat	0.016047301	Dairy
2	FDN15	17.5	Low Fat	0.016760075	Meat

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	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