GroupBy with Aggregate Functions

Single Column:

Use groupBy("column_name") and apply an aggregate function (e.g., sum(), avg(), count()).

```
df.groupBy("category").agg({"sales": "sum"}).show()
```

Multiple Columns:

Group by multiple columns by passing a list.

```
df.groupBy("category", "region").agg({"sales": "avg"}).show()
```

Pivot

Reshape data, turning unique values of a column into new column headers.

Used with aggregation.

```
df.groupBy("category").pivot("year").sum("sales").show()
```

Dropping Null Values

Use dropna() to remove rows containing null values.

Drop rows with nulls in **any column**:

```
df.dropna().show()
```

Drop rows with nulls in **specific columns**:

```
df.dropna(subset=["column1", "column2"]).show()
```

Drop rows based on a **threshold** of non-null values:

```
df.dropna(thresh=2).show()  # At least 2 non-null values
```

Sorting in PySpark: sortBy and orderBy

sort() Sorts a **DataFrame** based on one or more columns.

Equivalent to orderBy in PySpark; they perform the same function.

```
df.sort("column1", "column2").show()  # Default is ascending
df.sort(df["column1"].desc()).show()  # Specify descending
```

orderBy() Another way to sort a DataFrame. Allows sorting by **multiple columns**.

Offers more flexibility by supporting column expressions for sorting.

```
df.orderBy("column1", "column2").show()  # Default ascending
df.orderBy(df["column1"].desc(), df["column2"].asc()).show()  # Mixed order
```

Types of Sorting

Single Column Sorting:

```
df.orderBy("column1").show()
df.orderBy(df["column1"].desc()).show()
```

Multiple Column Sorting:

Ascending for all: `df.orderBy("column1", "column2").show()`

Mixed order: `df.orderBy(df["column1"].asc(), df["column2"].desc()).show()`

Using Expressions: Sorting can involve transformations:

```
from pyspark.sql.functions import col
df.orderBy(col("column1") + 1).show()  # Sort by column1 incremented by 1
```

create data

11:06 AM (1s)

```
from pyspark.sql import SparkSession


# Initialize SparkSession
spark = SparkSession.builder \
.appName("example") \
.getOrCreate()
# Data
simpleData = [("James","Sales","NY",90000,34,10000),
("Michael","Sales","NY",86000,56,20000),
("Robert","Sales","CA",81000,30,23000),
("Maria","Finance","CA",90000,24,23000),
("Raman","Finance","CA",99000,40,24000),
("Scott","Finance","NY",83000,36,19000),
("Jen","Finance","NY",79000,53,15000),
("Jeff","Marketing","CA",80000,25,18000),
("Kumar","Marketing","NY",91000,50,21000)
]
# Create DataFrame
schema = ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data=simpleData, schema = schema)
```

pivot()

11:06 AM (3s)

```
#using pivot function
df.groupBy("department").sum("salary").show()
df.groupBy("department").pivot("employee_name").sum("salary").show()
```

(9) Spark Jobs

```
+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|     Sales|     257000|
|   Finance|     351000|
| Marketing|     171000|
+----------+-----------+

+----------+-----+-----+-----+-----+-----+-------+-----+------+-----+
|department|James| Jeff|  Jen|Kumar|Maria|Michael|Raman|Robert|Scott|
+----------+-----+-----+-----+-----+-----+-------+-----+------+-----+
|     Sales|90000| null| null| null| null|  86000| null| 81000| null|
|   Finance| null| null|79000| null|90000|   null|99000|  null|83000|
| Marketing| null|80000| null|91000| null|   null| null|  null| null|
+----------+-----+-----+-----+-----+-----+-------+-----+------+-----+
```

salary aggregate sum before and after grouping

11:06 AM (1s) 18

```
df.groupBy("department").agg({"salary":"sum"}).show()
df.agg({"salary":"sum"}).show()  # Without group using agg on salary colums
```

(4) Spark Jobs

```
+----------+-----------+
|department|sum(salary)|
+----------+-----------+
|     Sales|     257000|
|   Finance|     351000|
| Marketing|     171000|
+----------+-----------+

+-----------+
|sum(salary)|
+-----------+
|     779000|
+-----------+
```

Working with Loan Data

Create data

11:51 AM (1s) 33

```
data_loan = spark.read.csv("/FileStore/tables/LoanData-2.csv",inferSchema=True,header=True)
```

(2) Spark Jobs

data_loan: pyspark.sql.dataframe.DataFrame = [Loan_ID: string, Gender: string ... 11 more fields]

data exploration

Markdown

12:24 PM (<1s) 35 Python

```
data_loan.limit(10).display()
```

(1) Spark Jobs

Table

	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
1	0	Graduate	No	5849	0	null	360
2	1	Graduate	No	4583	1508	128	360
3	0	Graduate	Yes	3000	0	66	360
4	0	Not Graduate	No	2583	2358	120	360
5	0	Graduate	No	6000	0	141	360
6	2	Graduate	Yes	5417	4196	267	360
7	0	Not Graduate	No	2333	1516	95	360
8	3+	Graduate	No	3036	2504	158	360
9	2	Graduate	No	4006	1526	168	360

```
11:59 AM (1s)

data_loan.printSchema()
print((data_loan.count(), len(data_loan.columns)))

(2) Spark Jobs

root
 |-- Loan_ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Married: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- Education: string (nullable = true)
 |-- Self_Employed: string (nullable = true)
 |-- ApplicantIncome: integer (nullable = true)
 |-- CoapplicantIncome: double (nullable = true)
 |-- LoanAmount: integer (nullable = true)
 |-- Loan_Amount_Term: integer (nullable = true)
 |-- Credit_History: integer (nullable = true)
 |-- Property_Area: string (nullable = true)
 |-- Loan_Status: string (nullable = true)

(614, 13)
```

groupby

```
12:05 PM (2s)

data_loan.groupBy("Education").min("ApplicantIncome").show()
data_loan.groupBy("Married").avg("CoapplicantIncome").show()
data_loan.groupBy("Loan_Status").count().show()

(6) Spark Jobs

|   Education|min(ApplicantIncome)|
+------------+--------------------+
|Not Graduate|                 210|
|    Graduate|                 150|
+------------+--------------------+

+-------+----------------------+
|Married|avg(CoapplicantIncome)|
+-------+----------------------+
|   null|    251.33333333333334|
|     No|    1316.5586854460093|
|    Yes|     1794.632964795578|
+-------+----------------------+

+-----------+-----+
|Loan_Status|count|
+-----------+-----+
|          Y|  422|
|          N|  192|
+-----------+-----+
```

12:05 PM (2s) 39

```
data_loan.groupBy("Education","Loan_Status","Married").min("CoapplicantIncome").show()
data_loan.groupBy("Loan_Status","Education").mean("ApplicantIncome").show()
data_loan.groupBy("Married","Education").count().show()
```

(6) Spark Jobs

```
+------------+-----------+-------+----------------------+

+-----------+------------+--------------------+
|Loan_Status|   Education|avg(ApplicantIncome)|
+-----------+------------+--------------------+
|          Y|    Graduate|   5751.576470588236|
|          N|Not Graduate|  3646.4423076923076|
|          N|    Graduate|  6114.5142857142855|
|          Y|Not Graduate|   3860.256097560976|
+-----------+------------+--------------------+

+-------+------------+-----+
|Married|   Education|count|
+-------+------------+-----+
|    Yes|    Graduate|  309|
|    Yes|Not Graduate|   89|
|     No|Not Graduate|   45|
|     No|    Graduate|  168|
|   null|    Graduate|    3|
+-------+------------+-----+
```

drop null()

12:07 PM (1s) 41

```
loan_copy = data_loan
loan_clean = loan_copy.na.drop()
```

(4) Spark Jobs

loan_copy: pyspark.sql.dataframe.DataFrame = [Loan_ID: string, Gender: string ... 11 more fields]

loan_clean: pyspark.sql.dataframe.DataFrame = [Loan_ID: string, Gender: string ... 11 more fields]

```
(614, 13)
(480, 13)
```

data after dropping null

+ Code + Text

12:08 PM (1s) 43

```
print((data_loan.count(), len(data_loan.columns)))
print((loan_clean.count(), len(loan_clean.columns)))
```

(4) Spark Jobs

```
(614, 13)
(480, 13)
```

groupby after null drop

12:09 PM (4s) 45

```
loan_clean.groupBy("Education").min("ApplicantIncome").show()
loan_clean.groupBy("Married").avg("CoapplicantIncome").show()
loan_clean.groupBy("Loan_Status").count().show()

loan_clean.groupBy("Education","Loan_Status","Married").min("CoapplicantIncome").show()
loan_clean.groupBy("Loan_Status","Education").mean("ApplicantIncome").show()
loan_clean.groupBy("Married","Education").count().show()
```

(12) Spark Jobs

```
+------------+--------------------+
|   Education|min(ApplicantIncome)|
+------------+--------------------+
|Not Graduate|                1442|
|    Graduate|                 150|
+------------+--------------------+

+-------+----------------------+
|Married|avg(CoapplicantIncome)|
+-------+----------------------+
|     No|    1215.8934911242604|
|    Yes|    1779.5463665229581|
+-------+----------------------+

+-----------+-----+
|Loan_Status|count|
+-----------+-----+
|          Y|  332|
|          N|  148|
+-----------+-----+
```

(12) Spark Jobs

```
+------------+-----------+-------+----------------------+
|   Education|Loan_Status|Married|min(CoapplicantIncome)|
+------------+-----------+-------+----------------------+
|Not Graduate|          Y|     No|                   0.0|
|    Graduate|          N|    Yes|                   0.0|
|Not Graduate|          N|    Yes|                   0.0|
|Not Graduate|          N|     No|                   0.0|
|    Graduate|          N|     No|                   0.0|
|Not Graduate|          Y|    Yes|                   0.0|
|    Graduate|          Y|    Yes|                   0.0|
|    Graduate|          Y|     No|                   0.0|
+------------+-----------+-------+----------------------+

+-----------+------------+--------------------+
|Loan_Status|   Education|avg(ApplicantIncome)|
+-----------+------------+--------------------+
|          Y|    Graduate|    5465.49446494465|
|          N|Not Graduate|  3654.3333333333335|
|          N|    Graduate|   6397.428571428572|
|          Y|Not Graduate|  4026.4590163934427|
+-----------+------------+--------------------+
```

```
+-------+------------+------+
|Married|   Education|count|
+-------+------------+------+
|    Yes|    Graduate|  248|
|    Yes|Not Graduate|   63|
|     No|Not Graduate|   34|
|     No|    Graduate|  135|
+-------+------------+------+
```

Joins

Data creation

```
from pyspark.sql import SparkSession


# Initialize SparkSession
spark = SparkSession.builder \
.appName("example") \
.getOrCreate()
# Data
emp = [(1,"Smith",-1,"2018","10","M",3000),(2, "Rose",1 , "2010", "20","M", 4000),(3,"Williams",1,"2010","10","M",1000),(4, "Jones",2 ,"2005","10","F",2000),(5,"Brown",2,"2010","40","",-1),(6, "Brown", 2, "2010","50",
"",-1)]
empColumns = ["emp_id","name","superior_emp_id","year_joined", "emp_dept_id","gender","salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
empDF.printSchema()
empDF.show()

dept = [("Finance",10),("Marketing",20),("Sales",30),("IT",40)]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()
deptDF.show()
```

```
(6) Spark Jobs
empDF: pyspark.sql.dataframe.DataFrame = [emp_id: long, name: string ... 5 more fields]
deptDF: pyspark.sql.dataframe.DataFrame = [dept_name: string, dept_id: long]
|    1|   Smith|           -1|      2018|        10|     M|  3000|
|    2|    Rose|            1|      2010|        20|     M|  4000|
|    3|Williams|            1|      2010|        10|     M|  1000|
|    4|   Jones|            2|      2005|        10|     F|  2000|
|    5|   Brown|            2|      2010|        40|      |    -1|
|    6|   Brown|            2|      2010|        50|      |    -1|
+-----+--------+-------------+----------+----------+------+------+
```

```
root
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)

+---------+-------+
|dept_name|dept_id|
+---------+-------+
|  Finance|     10|
|Marketing|     20|
|    Sales|     30|
|       IT|     40|
+---------+-------+
```

Inner, Outer, Full

```
12:39 PM (1s)                                                    50

#Inner join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "inner").show()
```

```
(3) Spark Jobs
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
|     1|   Smith|            -1|       2018|         10|     M|  3000|  Finance|     10|
|     3|Williams|             1|       2010|         10|     M|  1000|  Finance|     10|
|     4|   Jones|             2|       2005|         10|     F|  2000|  Finance|     10|
|     2|    Rose|             1|       2010|         20|     M|  4000|Marketing|     20|
|     5|   Brown|             2|       2010|         40|      |    -1|       IT|     40|
+------+--------+--------------+-----------+-----------+------+------+---------+-------+
```

```
#outer join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "outer").show()
```

▸ (3) Spark Jobs

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
1	Smith	-1	2018	10	M	3000	Finance	10
3	Williams	1	2010	10	M	1000	Finance	10
4	Jones	2	2005	10	F	2000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
null	null	null	null	null	null	null	Sales	30
5	Brown	2	2010	40		-1	IT	40
6	Brown	2	2010	50		-1	null	null

```
#full join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "full").show()
```

▸ (3) Spark Jobs

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
1	Smith	-1	2018	10	M	3000	Finance	10
3	Williams	1	2010	10	M	1000	Finance	10
4	Jones	2	2005	10	F	2000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
null	null	null	null	null	null	null	Sales	30
5	Brown	2	2010	40		-1	IT	40
6	Brown	2	2010	50		-1	null	null

left, left outer

▶ ✓ 12:40 PM (1s) 54

```
#Left join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "left").show()
```

▸ (6) Spark Jobs

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
1	Smith	-1	2018	10	M	3000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
3	Williams	1	2010	10	M	1000	Finance	10
4	Jones	2	2005	10	F	2000	Finance	10
5	Brown	2	2010	40		-1	IT	40
6	Brown	2	2010	50		-1	null	null

```
#Left join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "leftouter").show()
```

(6) Spark Jobs

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
1	Smith	-1	2018	10	M	3000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
3	Williams	1	2010	10	M	1000	Finance	10
4	Jones	2	2005	10	F	2000	Finance	10
5	Brown	2	2010	40		-1	IT	40
6	Brown	2	2010	50		-1	null	null

Right, Right Outer

12:41 PM (1s) 57

```
#right join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "right").show()
```

(6) Spark Jobs

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
4	Jones	2	2005	10	F	2000	Finance	10
3	Williams	1	2010	10	M	1000	Finance	10
1	Smith	-1	2018	10	M	3000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
null	null	null	null	null	null	null	Sales	30
5	Brown	2	2010	40		-1	IT	40

```
#right outer join
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "rightouter").show()
```

(6) Spark Jobs

emp_id	name	superior_emp_id	year_joined	emp_dept_id	gender	salary	dept_name	dept_id
4	Jones	2	2005	10	F	2000	Finance	10
3	Williams	1	2010	10	M	1000	Finance	10
1	Smith	-1	2018	10	M	3000	Finance	10
2	Rose	1	2010	20	M	4000	Marketing	20
null	null	null	null	null	null	null	Sales	30
5	Brown	2	2010	40		-1	IT	40

LeftSemi LeftAnti

12:42 PM (1s)

```
#leftsemijoin
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "leftsemi").show()
```

(3) Spark Jobs

```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|    name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|     1|   Smith|             -1|       2018|         10|     M|  3000|
|     3|Williams|              1|       2010|         10|     M|  1000|
|     4|   Jones|              2|       2005|         10|     F|  2000|
|     2|    Rose|              1|       2010|         20|     M|  4000|
|     5|   Brown|              2|       2010|         40|      |    -1|
+------+--------+---------------+-----------+-----------+------+------+
```

12:42 PM (2s)

```
#leftanti
empDF.join(deptDF,empDF.emp_dept_id == deptDF.dept_id, "leftanti").show()
```

(6) Spark Jobs

```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id| name|superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|     6|Brown|              2|       2010|         50|      |    -1|
+------+-----+---------------+-----------+-----------+------+------+
```

Spark SQL using Loan Data (Clean)

Create View

06:45 PM (<1s) 64

```
loan_clean.createOrReplaceTempView("loan")
loan_clean.printSchema()
```

```
root
 |-- Loan_ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Married: string (nullable = true)
 |-- Dependents: string (nullable = true)
 |-- Education: string (nullable = true)
 |-- Self_Employed: string (nullable = true)
 |-- ApplicantIncome: integer (nullable = true)
 |-- CoapplicantIncome: double (nullable = true)
 |-- LoanAmount: integer (nullable = true)
 |-- Loan_Amount_Term: integer (nullable = true)
 |-- Credit_History: integer (nullable = true)
 |-- Property_Area: string (nullable = true)
 |-- Loan_Status: string (nullable = true)
```

Select query

06:47 PM (1s) 66 Python

```
spark.sql("select * from loan").limit(10).display()
loan_clean.select("Loan_ID","ApplicantIncome","Loan_Status").show(5)
```

(2) Spark Jobs

Table

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	1.2 Coapplic
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	
7	LP001014	Male	Yes	3+	Graduate	No	3036	
8	LP001018	Male	Yes	2	Graduate	No	4006	
9	LP001020	Male	Yes	1	Graduate	No	12841	
10	LP001024	Male	Yes	2	Graduate	No	3200	

```
+--------+---------------+-----------+
| Loan_ID|ApplicantIncome|Loan_Status|
+--------+---------------+-----------+
|LP001003|           4583|          N|
|LP001005|           3000|          Y|
|LP001006|           2583|          Y|
|LP001008|           6000|          Y|
|LP001011|           5417|          Y|
+--------+---------------+-----------+
only showing top 5 rows
```

filter based on data

06:49 PM (<1s) 68 Python

```
spark.sql("""SELECT * From loan WHERE ApplicantIncome<5000""").limit(10).display()
```

(1) Spark Jobs

Table

	ne	1.2 CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
2	3000	0	66	360	1	Urban	Y
3	2583	2358	120	360	1	Urban	Y
4	2333	1516	95	360	1	Urban	Y
5	3036	2504	158	360	0	Semiurban	N
6	4006	1526	168	360	1	Urban	Y
7	3200	700	70	360	1	Urban	Y
8	3073	8106	200	360	1	Urban	Y
9	1853	2840	114	360	1	Rural	N
10	1299	1086	17	120	1	Urban	Y

10 rows | 0.33 seconds runtime Refreshed 8 minutes ago

order by

06:52 PM (<1s) 70 Python

```
spark.sql("""select * FROM loan WHERE Dependents in ('3+','2')order by Dependents """).limit(10).display()
```

(1) Spark Jobs

Table

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	1.2 Coapplic
2	LP001491	Male	Yes	2	Graduate	Yes	3316	
3	LP001245	Male	Yes	2	Not Graduate	Yes	1875	
4	LP001018	Male	Yes	2	Graduate	No	4006	
5	LP001267	Female	Yes	2	Graduate	No	1378	
6	LP001024	Male	Yes	2	Graduate	No	3200	
7	LP001318	Male	Yes	2	Graduate	No	6250	
8	LP001030	Male	Yes	2	Graduate	No	1299	
9	LP001319	Male	Yes	2	Not Graduate	No	3273	
10	LP001179	Male	Yes	2	Graduate	No	4616	

10 rows | 0.42 seconds runtime Refreshed 5 minutes ago

group by

4 minutes ago (1s) 72 Python

```
spark.sql("""SELECT Loan_Amount_Term,count(*) as count FROM loan GROUP BY Loan_Amount_Term""").show()
```

(2) Spark Jobs

```
+----------------+-----+
|Loan_Amount_Term|count|
+----------------+-----+
|           300.0|    9|
|           360.0|  411|
|           120.0|    3|
|            84.0|    3|
|           240.0|    2|
|           480.0|   12|
|            60.0|    2|
|           180.0|   36|
|            36.0|    2|
+----------------+-----+
```