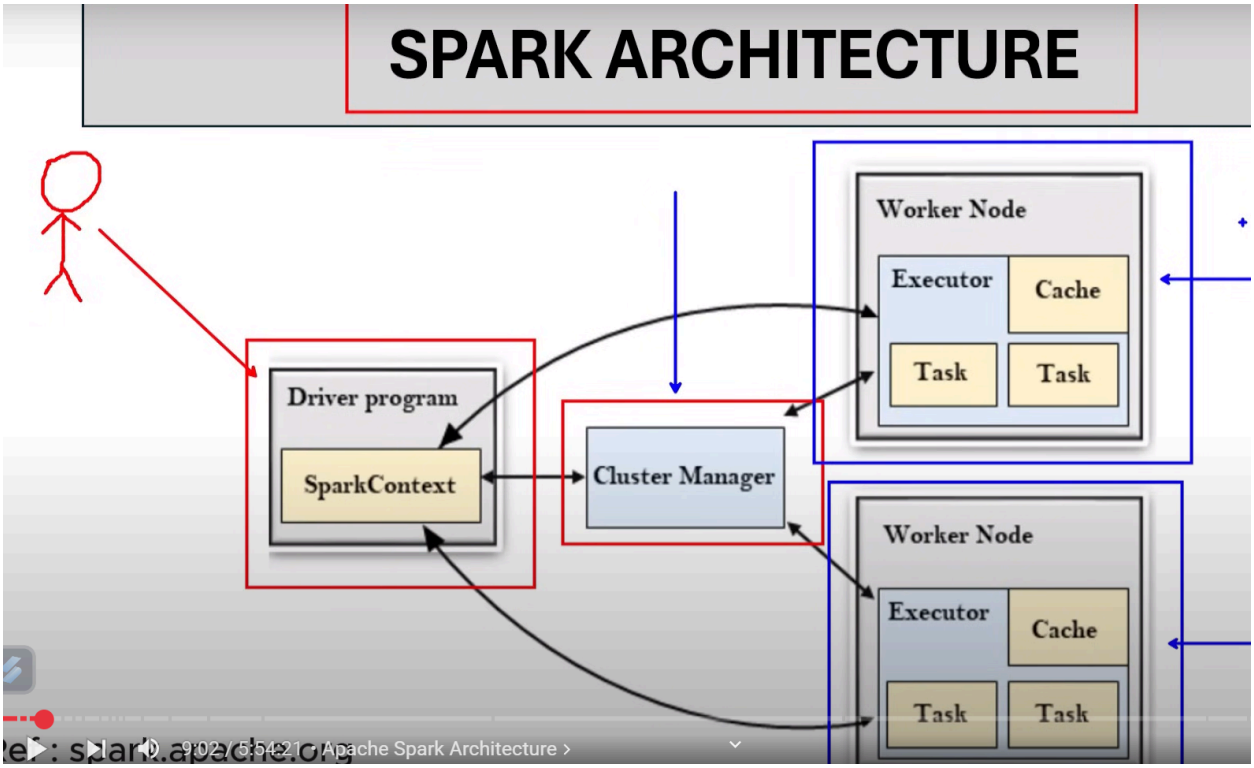
PySpark

What is Spark?

- It is the distributed engine which distribute data among the several machine or computer.

Spark Architecture



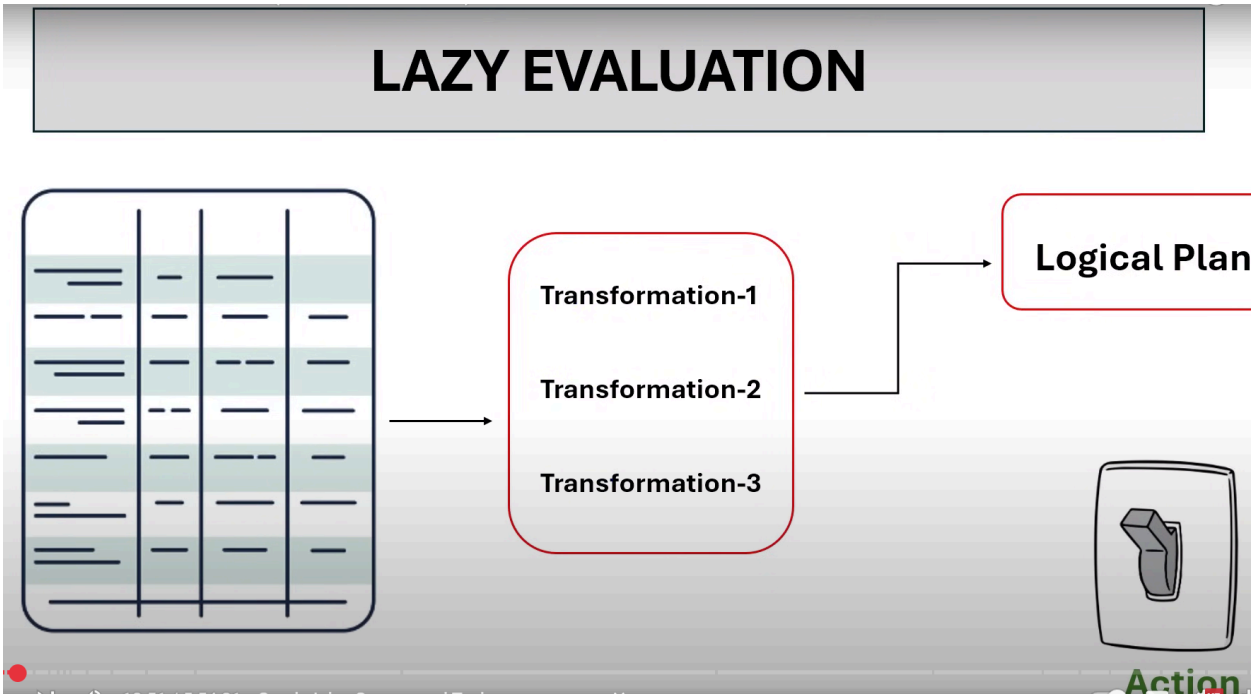
- Driver Program is computer or node so here all data breakdown into some smaller task and all jobs done on here like transforming , jobs, stages, tasks etc.
- Now these task given to cluster manager (Cluster - combination of many computer and work like one computer) and Cluster manager give these task to many worker node .

- These worker node actually execute the task and number of worker node can be many or infinity.
- This architecture is known as **Master Slave Architecture** where master is driver program and slave is worker node.

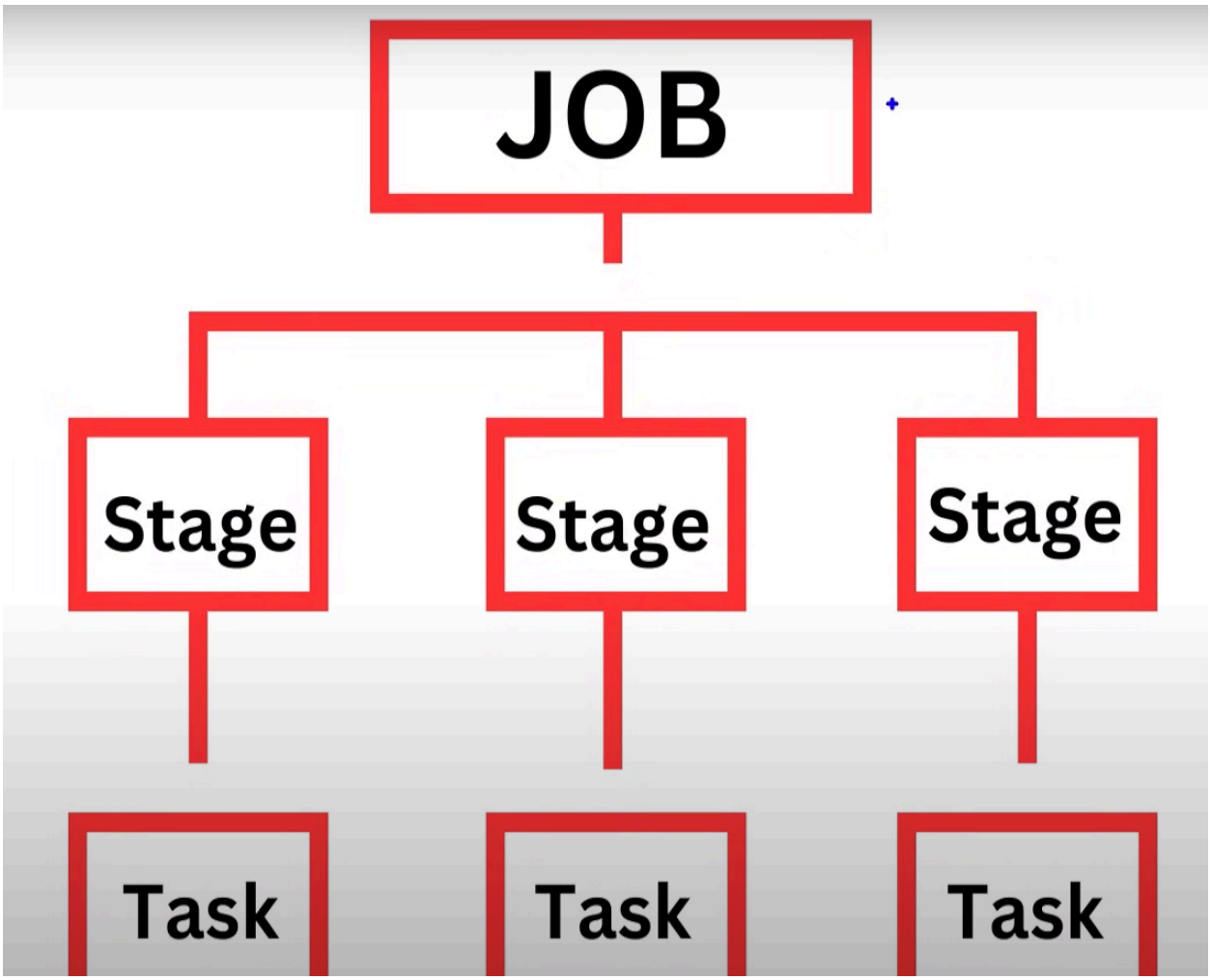
Benefits of Spark

- In-Memory computation
- lazy Evaluation
- Fault Tolerant
- partitioning

Lazy Evaluation



Hierarchy of data-



Api provided by Spark-

- python —> pyspark
- Scala
- SQL
- R

Pyspark code

Data Reading CSV

```
dbutils.fs.ls('/FileStore/tables')
#  Out[6]: [FileInfo(path='dbfs:/FileStore/tables/BigMart_Sales.csv', name='Big
```

```
Mart_Sales.csv', size=869537, modificationTime=1749895393000)]

df=spark.read.format('csv').option('inferSchema',True).option('header',True).l
oad('/FileStore/tables/BigMart_Sales.csv')

df.show()
#+--------------+----------+---------------+--------------+-------------
------+--------+----------------+-----------------------+----------+------
-------------+----------------+----------------+
|Item_Identifier|Item_Weight|Item_Fat_Content|Item_Visibility|          Item_Type
|Item_MRP|Outlet_Identifier|Outlet_Establishment_Year|Outlet_Size|Outlet_Loc
ation_Type|      Outlet_Type|Item_Outlet_Sales|
+--------------+----------+---------------+--------------+---------------
----+--------+----------------+-----------------------+----------+--------
-----------+----------------+----------------+
|         FDA15|       9.3|        Low Fat|   0.016047301|             Dairy|249.8092|
OUT049|                1999|     Medium|           Tier 1|Supermarket Type1|
3735.138|
|         DRC01|      5.92|        Regular|   0.019278216|        Soft Drinks| 48.269
2|          OUT018|                2009|     Medium|           Tier 3|Supermarket Ty
pe2|        443.4228|
|         FDN15|      17.5|        Low Fat|   0.016760075|              Meat| 141.618|
OUT049|                1999|     Medium|           Tier 1|Supermarket Type1|
2097.27|
|         FDX07|      19.2|        Regular|           0.0|Fruits and Vegeta...| 182.095|
OUT010|                1998|       null|           Tier 3|    Grocery Store|          732.
38|
|         NCD19|      8.93|        Low Fat|           0.0|         Household| 53.8614|
OUT013|                1987|       High|           Tier 3|Supermarket Type1|        9
94.7052|
|         FDP36|    10.395|        Regular|           0.0|       Baking Goods| 51.4008|
OUT018|                2009|     Medium|           Tier 3|Supermarket Type2|
556.6088|
|         FDO10|     13.65|        Regular|   0.012741089|        Snack Foods| 57.65
88|          OUT013|                1987|       High|           Tier 3|Supermarket Typ
e1|        343.5528|
```

```
|       FDP10|      null|       Low Fat|   0.127469857|       Snack Foods|107.762
2|        OUT027|              1985|     Medium|           Tier 3|Supermarket Ty
pe3|      4022.7636|
|       FDH17|      16.2|       Regular|   0.016687114|       Frozen Foods| 96.972
6|        OUT045|              2002|      null|           Tier 2|Supermarket Type
1|      1076.5986|
|       FDU28|      19.2|       Regular|    0.09444959|       Frozen Foods|187.8
214|        OUT017|              2007|      null|           Tier 2|Supermarket Typ
e1|       4710.535|
|       FDY07|      11.8|       Low Fat|           0.0|Fruits and Vegeta...| 45.5402|
OUT049|              1999|     Medium|           Tier 1|Supermarket Type1|
1516.0266|
|       FDA03|      18.5|       Regular|   0.045463773|              Dairy|144.1102|
OUT046|              1997|      Small|           Tier 1|Supermarket Type1|       2
187.153|
|       FDX32|      15.1|       Regular|    0.1000135|Fruits and Vegeta...|145.478
6|        OUT049|              1999|     Medium|           Tier 1|Supermarket Ty
pe1|      1589.2646|
|       FDS46|      17.6|       Regular|   0.047257328|       Snack Foods|119.67
82|        OUT046|              1997|      Small|           Tier 1|Supermarket Typ
e1|      2145.2076|
|       FDF32|     16.35|       Low Fat|    0.0680243|Fruits and Vegeta...|196.4
426|        OUT013|              1987|      High|           Tier 3|Supermarket Ty
pe1|      1977.426|
|       FDP49|      9.0|       Regular|   0.069088961|          Breakfast| 56.3614
|        OUT046|              1997|      Small|           Tier 1|Supermarket Type1
|      1547.3192|
|       NCB42|      11.8|       Low Fat|   0.008596051|  Health and Hygiene|11
5.3492|        OUT018|              2009|     Medium|           Tier 3|Supermar
ket Type2|      1621.8888|
|       FDP49|      9.0|       Regular|   0.069196376|          Breakfast| 54.3614
|        OUT049|              1999|     Medium|           Tier 1|Supermarket Typ
e1|       718.3982|
|       DRI11|      null|       Low Fat|   0.034237682|       Hard Drinks|113.2834
|        OUT027|              1985|     Medium|           Tier 3|Supermarket Typ
e3|      2303.668|
```

```
|        FDU02|     13.35|        Low Fat|    0.10249212|              Dairy|230.5352|
OUT035|               2004|     Small|           Tier 2|Supermarket Type1|      2
748.4224|
+--------------+----------+---------------+--------------+-------------------
----+--------+----------------+----------------------+----------+--------
-----------+----------------+----------------+
only showing top 20 rows


df.display()
```

Data Reading JSON

```
dbutils.fs.ls("/FileStore/tables")

#Out[13]: [FileInfo(path='dbfs:/FileStore/tables/BigMart_Sales.csv', name='Big
Mart_Sales.csv', size=869537, modificationTime=1749895393000),
 FileInfo(path='dbfs:/FileStore/tables/drivers.json', name='drivers.json', size=1
80812, modificationTime=1749897188000)]

 df_json=spark.read.format('json').option('inferSchema',True)\
                              .option('header',True)\
                              .option('multiLine',False)\
                                 .load('/FileStore/tables/drivers.json')


 df_json.display()
```

SCHEMA CHANGE

```
df.printSchema()

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

```
 |-- Item_Fat_Content: string (nullable = true)
 |-- Item_Visibility: double (nullable = true)
 |-- Item_Type: string (nullable = true)
 |-- Item_MRP: double (nullable = true)
 |-- Outlet_Identifier: string (nullable = true)
 |-- Outlet_Establishment_Year: integer (nullable = true)
 |-- Outlet_Size: string (nullable = true)
 |-- Outlet_Location_Type: string (nullable = true)
 |-- Outlet_Type: string (nullable = true)
 |-- Item_Outlet_Sales: double (nullable = true)
```

DDL SCHEMA

```
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,
          Outlet_Location_Type STRING,
          Outlet_Type STRING,
          Item_Outlet_Sales DOUBLE
          '''
df = spark.read.format('csv')\
   .schema(ddl_schema)\
      .option('header',True)\
         .load('/FileStore/tables/BigMart_Sales.csv')

df.display()

df.printSchema()
```

```
root
 |-- Item_Identifier: string (nullable = true)
 |-- Item_Weight: string (nullable = true)
 |-- Item_Fat_Content: string (nullable = true)
 |-- Item_Visibility: double (nullable = true)
 |-- Item_Type: string (nullable = true)
 |-- Item_MRP: double (nullable = true)
 |-- Outlet_Identifier: string (nullable = true)
 |-- Outlet_Establishment_Year: integer (nullable = true)
 |-- Outlet_Size: string (nullable = true)
 |-- Outlet_Location_Type: string (nullable = true)
 |-- Outlet_Type: string (nullable = true)
 |-- Item_Outlet_Sales: double (nullable = true)
```

StructType() Schema

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

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_Type',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),

])
```

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

df.printSchema()

df.printSchema()

root
 |-- Item_Identifier: string (nullable = true)
 |-- Item_Weight: string (nullable = true)
 |-- Item_Fat_Content: string (nullable = true)
 |-- Item_Visibility: string (nullable = true)
 |-- Item_Type: string (nullable = true)
 |-- Item_MRP: string (nullable = true)
 |-- Outlet_Identifier: string (nullable = true)
 |-- Outlet_Establishment_Year: string (nullable = true)
 |-- Outlet_Size: string (nullable = true)
 |-- Outlet_Location_Type: string (nullable = true)
 |-- Outlet_Type: string (nullable = true)
 |-- Item_Outlet_Sales: string (nullable = true)

df.display()
```

SELECT

```
   df.select('Item_Identifier','Item_Weight','Item_Visibility').display()

   df.select('Item_Identifier','Item_Weight','Item_Visibility').display()
```

ALIAS

```
df.select(col("Item_Identifier").alias("Item_Idn")).display()
```

Filter

Scenerio 1

```
df.filter(col('Item_Fat_Content')=='Regular').display()
```

Scenerio 2

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

Scenerio 3

```
df.filter((col('Outlet_Size').isNull()) & (col('Outlet_Location_type').isin('Tier 1','Tier 3'))).display()
```

withColumnRenamed

```
df.withColumnRenamed('Item_Identifier','Item_Id').display()
```

withColumn

Scenerio 1

```
df=df.withColumn("New_Col",lit(0))
df.display()

df=df.withColumn("Multiply_Col",col('Item_Weight')*col('Item_Visibility'))
df.display()
```

Scenerio 2

```
df=df.withColumn("Item_Fat_Content",regexp_replace(col("Item_Fat_Conten
t"),"Regular","reg"))\
  .withColumn("Item_Fat_Content",regexp_replace(col("Item_Fat_Content"),"Lo
w Fat","LF"))

df.display()
```

TypeCasting

```
df.display()

df.withColumn("New_Col",col("New_Col").cast(StringType()))

df.printSchema()

root
 |-- Item_Identifier: string (nullable = true)
 |-- Item_Weight: string (nullable = true)
 |-- Item_Fat_Content: string (nullable = true)
 |-- Item_Visibility: string (nullable = true)
 |-- Item_Type: string (nullable = true)
 |-- Item_MRP: string (nullable = true)
 |-- Outlet_Identifier: string (nullable = true)
 |-- Outlet_Establishment_Year: string (nullable = true)
 |-- Outlet_Size: string (nullable = true)
 |-- Outlet_Location_Type: string (nullable = true)
 |-- Outlet_Type: string (nullable = true)
 |-- Item_Outlet_Sales: string (nullable = true)
 |-- New_Col: integer (nullable = false)
 |-- Multiply_Col: double (nullable = true)
```

sort

Scenario 1

```
df.sort(col("Item_Weight").asc()).display()
```

Scenario 2

```
df.sort(col('Item_Weight').desc()).display()
```

Scenario 3

```
df.sort(['Item_Weight','Item_MRP'],ascending=[0,0]).display()
```

Scenario 4

```
df.sort(['Item_Weight','Item_MRP'],ascending=[0,1]).display()
```

limit

```
df.limit(10).display()
```

drop

scenario 1

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

Scenario 2

```
df.drop('Item_Outlet_Sales','Multiply_Col').display()
```

drop_duplicates

scenario 1

```
df.drop_duplicates().display()
```

scenario 2

```
df.drop_duplicates(subset=['Item_Type']).display()
```

union & unionByName

preparing dataframe

```
data1=[('1','abc'),('2','xyz')]
schema1='id STRING , name STRING'

pdf1=spark.createDataFrame(data1,schema1)

data2=[('3','mno'),('4','fgh')]
schema2='id STRING, name STRING'

pdf2=spark.createDataFrame(data2,schema2)
```

```
pdf1.union(pdf2).display()
```

```
data3 = [('vby','5'),('xcf','6')]
schema3='name STRING,id STRING'

pdf3=spark.createDataFrame(data3,schema3)
```

```
pdf1.unionByName(pdf3).display()
```

String Function

initcap

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

lower

```
df.select(lower('Outlet_Size')).display()
```

upper

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

Date Function

current_date

```
df=df.withColumn('curr_date',current_date())
```

date_add()

```
df=df.withColumn('week_after',date_add('curr_date',7))
```

date_sub()

```
df=df.withColumn('week_before',date_sub('curr_date',7))
```

datediff()

```
df.withColumn('date_diff',datediff('week_after','week_before')).display()
```

date_format()

```
df.withColumn('week_before',date_format('week_before','dd-MM-yyyy')).displ
ay()
```

Handling null value

dropping null value

```
df.dropna('all').display()
df.dropna('any').display()
df.dropna(subset=['Outlet_Size']).display()
```

filling null value

```
df.fillna('N/A').display()
df.fillna('N/A',subset=['Outlet_Size']).display()
```

split & indexing

split

```
df.withColumn('split_column',split("Outlet_Type"," ")).display()
```

indexing

```
df.withColumn('split_column1',split('Outlet_Type',' ')[1]).display()
```

explode

```
#column must have data in the form of list.
```

```
df.withColumn("explode_col",explode('split_column').display()
```

Array_Contains

```
df.withColumn("New_col",array_contains('split_column','Type1')).display()
```

GroupBy

Scenario 1

```
df.groupBy("Item_Type").agg(sum("Item_MRP")).display()
```

Scenario 2

```
df.groupBy("Item_Type").agg(avg("Item_MRP").alias("AVG")).display()
```

Scenario 3

```
df.groupBy("Item_Type","Outlet_Type").agg(sum("Item_MRP")).display()
```

Scenario 4

```
df.groupBy("Item_Type","Outlet_Type").agg(sum("Item_MRP"),avg("Item_MRP")).display()
```

collect_list

```
data=[("user1","book1"),("user1","book2"),("user2","book3"),("user2","book4")]

schema="user STRING, book string"
```

```
new_df=spark.createDataFrame(data,schema)

new_df.groupBy("user").agg(collect_list("book")).display()
```

pivot

```
df.groupBy("Item_Type").pivot("Outlet_Type").agg(sum("Item_MRP")).display
()
```

when-otherwise

Scenario 1

```
df=df.withColumn("Veg_Flag",when(col("Item_Type")=="Meat","Non-Veg").ot
herwise("Veg"))
```

Scenario 2

```
df.withColumn("Veg-expensense",when((col("Veg_Flag")=='Veg') & (col("Ite
m_MRP")<100),"Veg-Inexpensive" )\
            .when((col("Veg_Flag")=="Veg") & (col("Item_MRP")>100),"Veg-Expen
sive").otherwise("Non-Veg")).display()
```

Join

inner join

```
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'

df2 = spark.createDataFrame(dataj2,schemaj2)
```

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

left join

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

right join

```
df1.join(df2, df1["dept_id"]==df2["dept_id"], "right").display()
```

anti join

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

window function

row_number()

```
from pyspark.sql.window import Window

df.withColumn('rowCol',row_number().over(Window.orderBy("Item_Identifier"))).display()
```

rank()

```
df.withColumn("Rank",rank().over(Window.orderBy(col("Item_Identifier").desc()))).display()
```

dense_rank()

```
df.withColumn("DenseRank",dense_rank().over(Window.orderBy(col("Item_Identifier").desc()))).display()
```

cumulative sum()

```
df.withColumn("cumSum",sum("Item_MRP").over(Window.orderBy("Item_Type"))).display()

df.withColumn("CumSum",sum("Item_MRP").over(Window.orderBy("Item_Type").rowsBetween(Window.unboundedPreceding,Window.currentRow))).display()

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

User Defined Function (UDF)

step-1

```
def fun(a):
    return a*a*a
```

step-2

```
my_udf=udf(fun)
```

step-3

```
df.withColumn("Cube",my_udf("CumSum")).display()
```

Data Writing

CSV

```
df.write.format('csv').save('/FileStore/tables/CSV/data.csv')
```

mode

append

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

overwrite

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

error

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

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

ignore

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

Parquet File

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

Table

```
df.write.format('csv')\
    .mode('overwrite')\
            .saveAsTable('Demo_Table')
```

Spark SQL

```
df.createTempView('temp_view')

%sql

select * from temp_view where Item_MRP=51.4008

df_sql=spark.sql("select * from temp_view where Item_Fat_Content='reg'")

df_sql.display()
```