```
#check that java is installed
!java -version
```

```
openjdk version "11.0.27" 2025-04-15
OpenJDK Runtime Environment (build 11.0.27+6-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.27+6-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```

```
#install pyspark
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

```
import os
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DoubleType, LongType, TimestampType
from pyspark.sql.functions import spark_partition_id
```

```
!curl -O https://raw.githubusercontent.com/Apoorva-888/Spark-Optimization/main/BigMart_Sales.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  849k  100  849k    0     0   626k      0  0:00:01  0:00:01 --:--:--  627k
```

```
spark = SparkSession.builder.appName('AutomotiveData').getOrCreate()

print(f'The Spark version is {spark.version}')
```

```
The Spark version is 3.5.1
```

```
spark.conf.set("spark.sql.adaptive.enabled","false")
spark.conf.get("spark.sql.adaptive.enabled")
```

```
'false'
```

```
spark.conf.get("spark.sql.adaptive.enabled")
```

```
'false'
```

```
df = spark.read.format("csv") \
    .option("inferSchema", True) \
    .option("header", True) \
    .load("BigMart_Sales.csv")

df.show(5)
```

```
+---------------+-----------+---------------+---------------+--------------------+--------+----------------+------------------------+
|Item_Identifier|Item_Weight|Item_Fat_Content|Item_Visibility|           Item_Type|Item_MRP|Outlet_Identifier|Outlet_Establishment_Year|
+---------------+-----------+---------------+---------------+--------------------+--------+----------------+------------------------+
|          FDA15|        9.3|        Low Fat|    0.016047301|               Dairy|249.8092|          OUT049|                    1999|
|          DRC01|       5.92|        Regular|    0.019278216|         Soft Drinks| 48.2692|          OUT018|                    2009|
|          FDN15|       17.5|        Low Fat|    0.016760075|                Meat| 141.618|          OUT049|                    1999|
|          FDX07|       19.2|        Regular|            0.0|Fruits and Vegeta...| 182.095|          OUT010|                    1998|
|          NCD19|       8.93|        Low Fat|            0.0|           Household| 53.8614|          OUT013|                    1987|
+---------------+-----------+---------------+---------------+--------------------+--------+----------------+------------------------+
only showing top 5 rows
```

```
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)
```

```
!ls -lh BigMart_Sales.csv
```

```
-rw-r--r-- 1 root root 850K Jul 21 12:28 BigMart_Sales.csv
```

```
spark.conf.get("spark.sql.files.maxPartitionBytes")
```

```
'134217728b'
```

```
df.rdd.getNumPartitions()
```

```
1
```

Wide Transformation

```
df_new_withoutAQE = df.groupBy("Item_Fat_Content").count()
df_new_withoutAQE.show()
```

```
+----------------+-----+
|Item_Fat_Content|count|
+----------------+-----+
|         low fat|  112|
|         Low Fat| 5089|
|              LF|  316|
|         Regular| 2889|
|             reg|  117|
+----------------+-----+
```

```
df_new_withoutAQE.explain(mode="formatted")
```

```
== Physical Plan ==
AdaptiveSparkPlan (5)
+- HashAggregate (4)
   +- Exchange (3)
      +- HashAggregate (2)
         +- Scan csv  (1)


(1) Scan csv
Output [1]: [Item_Fat_Content#289]
Batched: false
Location: InMemoryFileIndex [file:/content/BigMart_Sales.csv]
ReadSchema: struct<Item_Fat_Content:string>

(2) HashAggregate
Input [1]: [Item_Fat_Content#289]
Keys [1]: [Item_Fat_Content#289]
Functions [1]: [partial_count(1)]
Aggregate Attributes [1]: [count#395L]
Results [2]: [Item_Fat_Content#289, count#396L]

(3) Exchange
Input [2]: [Item_Fat_Content#289, count#396L]
Arguments: hashpartitioning(Item_Fat_Content#289, 200), ENSURE_REQUIREMENTS, [plan_id=205]

(4) HashAggregate
Input [2]: [Item_Fat_Content#289, count#396L]
Keys [1]: [Item_Fat_Content#289]
Functions [1]: [count(1)]
Aggregate Attributes [1]: [count(1)#385L]
Results [2]: [Item_Fat_Content#289, count(1)#385L AS count#386L]

(5) AdaptiveSparkPlan
Output [2]: [Item_Fat_Content#289, count#386L]
Arguments: isFinalPlan=false
```

```
Start coding or generate with AI.
```

With AQE

```
spark.conf.set("spark.sql.adaptive.enabled","true")
spark.conf.get("spark.sql.adaptive.enabled")
```

```
'true'
```

```
df = spark.read.format("csv") \
    .option("inferSchema", True) \
    .option("header", True) \
    .load("BigMart_Sales.csv")

df.show(2)
```

```
+---------------+-----------+---------------+---------------+-----------+--------+----------------+------------------------+---------
|Item_Identifier|Item_Weight|Item_Fat_Content|Item_Visibility|  Item_Type|Item_MRP|Outlet_Identifier|Outlet_Establishment_Year|Outlet_Si
+---------------+-----------+---------------+---------------+-----------+--------+----------------+------------------------+---------
|          FDA15|        9.3|        Low Fat|    0.016047301|      Dairy|249.8092|           OUT049|                    1999|     Medi
|          DRC01|       5.92|        Regular|    0.019278216|Soft Drinks| 48.2692|           OUT018|                    2009|     Medi
+---------------+-----------+---------------+---------------+-----------+--------+----------------+------------------------+---------
only showing top 2 rows
```

```
df.rdd.getNumPartitions()
```

```
1
```

```
df_new_with_AQE = df.groupBy("Item_Fat_Content").count()
df_new_with_AQE.show()
```

```
+----------------+-----+
|Item_Fat_Content|count|
+----------------+-----+
|         low fat|  112|
|         Low Fat| 5089|
|              LF|  316|
|         Regular| 2889|
|             reg|  117|
+----------------+-----+
```

```
df_new_with_AQE.explain(mode="formatted")
```

```
== Physical Plan ==
AdaptiveSparkPlan (5)
+- HashAggregate (4)
   +- Exchange (3)
      +- HashAggregate (2)
         +- Scan csv  (1)


(1) Scan csv
Output [1]: [Item_Fat_Content#289]
Batched: false
Location: InMemoryFileIndex [file:/content/BigMart_Sales.csv]
ReadSchema: struct<Item_Fat_Content:string>

(2) HashAggregate
Input [1]: [Item_Fat_Content#289]
Keys [1]: [Item_Fat_Content#289]
Functions [1]: [partial_count(1)]
Aggregate Attributes [1]: [count#435L]
Results [2]: [Item_Fat_Content#289, count#436L]

(3) Exchange
Input [2]: [Item_Fat_Content#289, count#436L]
Arguments: hashpartitioning(Item_Fat_Content#289, 200), ENSURE_REQUIREMENTS, [plan_id=262]

(4) HashAggregate
Input [2]: [Item_Fat_Content#289, count#436L]
Keys [1]: [Item_Fat_Content#289]
Functions [1]: [count(1)]
Aggregate Attributes [1]: [count(1)#425L]
Results [2]: [Item_Fat_Content#289, count(1)#425L AS count#426L]
```

```
(5) AdaptiveSparkPlan
Output [2]: [Item_Fat_Content#289, count#426L]
Arguments: isFinalPlan=false
```

```
Start coding or generate with AI.
```