Amazon_And_Walmart_Aggreg...

```
// /user/rr4577_nyu_edu/amazon-clean.parquet
```
READY

```
val schema = "rating DOUBLE, category STRING, date DATE"
```
FINISHED

```
schema: String = rating DOUBLE, category STRING, date DATE
```
Took 31 sec. Last updated by sc10670_nyu_edu at December 13 2024, 7:33:46 PM.

```
// Reading the First Parquet files:
val cleanDF = spark.read.schema(schema).parquet("/user/rr4577_nyu_edu/amazon-clean.parquet")
```
FINISHED

```
cleanDF: org.apache.spark.sql.DataFrame = [rating: double, category: string ... 1 more field]
```
Took 2 sec. Last updated by sc10670_nyu_edu at December 13 2024, 7:33:48 PM.

```
val mycleanedDF = spark.read.schema(schema).parquet("/user/sc10670_nyu_edu/project/amazon-clean.parquet")
```
FINISHED

```
mycleanedDF: org.apache.spark.sql.DataFrame = [rating: double, category: string ... 1 more field]
```
Took 0 sec. Last updated by sc10670_nyu_edu at December 13 2024, 7:33:48 PM.

```
val combinedDF = cleanDF.union(mycleanedDF)
```
FINISHED

```
combinedDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [rating: double, category: string ... 1 more field]
```
Took 0 sec. Last updated by sc10670_nyu_edu at December 13 2024, 7:33:48 PM.

```
// Remove rows where the category is "Unknown"
val filteredDF = combinedDF.filter(col("category") =!= "Unknown")
```
FINISHED

```
filteredDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [rating: double, category: string ... 1 more field]
```
Took 0 sec. Last updated by sc10670_nyu_edu at December 13 2024, 7:33:48 PM.

```
z.show(filteredDF)
```
SPARK JOB (http://nyu-dataproc-sw-cfv5.c.hpc-dataproc-19b8.internal:39707/jobs/job?id=45) FINISHED

settings

rating	category
1.0	Movies and TV
1.0	Movies and TV
5.0	Movies and TV
1.0	Movies and TV
5.0	Movies and TV
2.0	Movies and TV
4.0	Movies and TV
5.0	Movies and TV

Output is truncated to 1000 rows. Learn more about **zeppelin.spark.maxResult**

Took 0 sec. Last updated by anonymous at December 10 2024, 11:21:07 AM. (outdated)

READY

```
z.show(filteredDF.summary())
```
SPARK JOB FINISHED

settings

summary	rating

count	174289703
mean	4.325330280699371
stddev	1.171246260818209
min	0.0
25%	4.0
50%	5.0
75%	5.0
max	5.0

Took 16 min 46 sec. Last updated by anonymous at December 09 2024, 9:36:53 PM. (outdated)

SPARK JOB FINISHED

```
import org.apache.spark.sql.functions._

// Step 1: Filter the DataFrame for "Gift Cards" and convert the date column
val giftCardsDF = filteredDF.filter(col("category") === "Gift Cards")
  .withColumn("review_date", to_date(col("date")))

// Step 2: Filter for reviews from the last 5 years
val last5YearsDF = giftCardsDF.filter(col("review_date").between(date_add(current_date(), -365 * 5), current_date()))

// Step 3: Add year and week columns
val weeklyReviewsDF = last5YearsDF
  .withColumn("year", year(col("review_date")))
  .withColumn("week", weekofyear(col("review_date")))

// Step 4: Group by year and week, and count the reviews
val weeklyReviewCounts = weeklyReviewsDF.groupBy("year", "week")
  .agg(count("*").alias("review_count"))
  .orderBy("year", "week")

// Step 5: Display the results
z.show(weeklyReviewCounts)
```

settings

Available Fields

year week review_count

keys

week

groups

values

review_count **SUM**

☐ **force Y to 0**

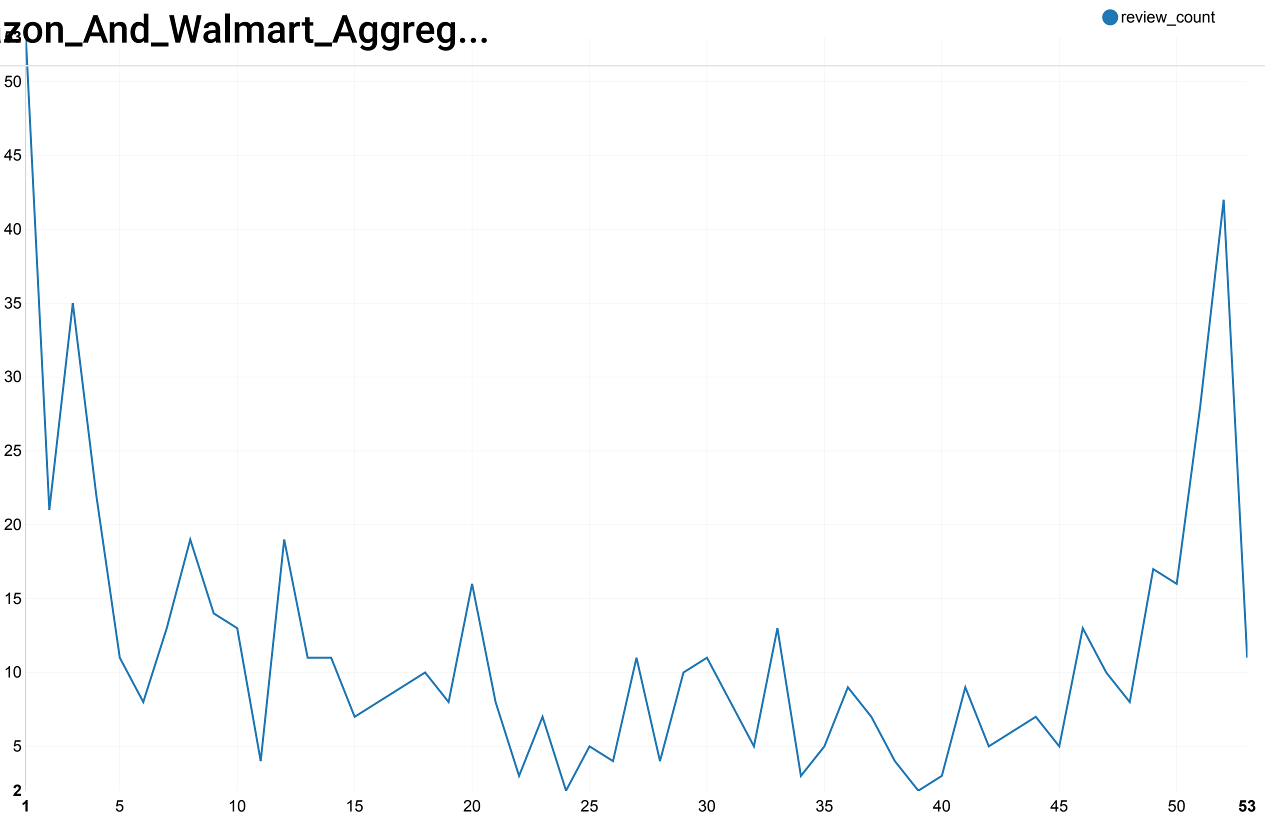
☐ **zoom**

☐ **Date format**

xAxis :

Default Rotate Hide

Amazon_And_Walmart_Aggreg...



```
import org.apache.spark.sql.functions._
giftCardsDF: org.apache.spark.sql.DataFrame = [rating: double, category: string ... 2 more fields]
last5YearsDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [rating: double, category: string ... 2 more fields]
weeklyReviewsDF: org.apache.spark.sql.DataFrame = [rating: double, category: string ... 4 more fields]
weeklyReviewCounts: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [year: int, week: int ... 1 more field]
```

Took 18 sec. Last updated by sc10670_nyu_edu at December 13 2024, 7:38:56 PM. (outdated)

```
val dowDF = filteredDF.withColumn("rating_dow", dayofweek(col("date")))
```

FINISHED

```
dowDF: org.apache.spark.sql.DataFrame = [rating: double, category: string ... 2 more fields]
```

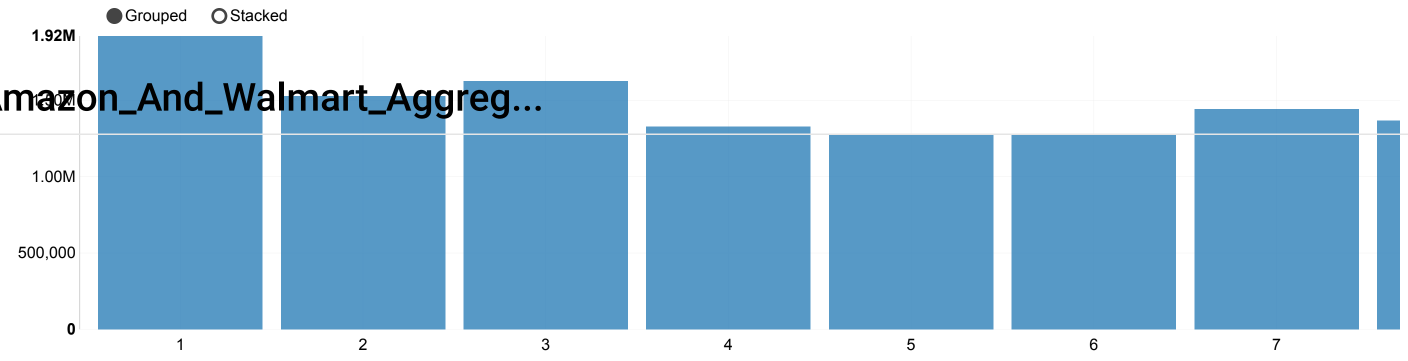
Took 0 sec. Last updated by anonymous at December 10 2024, 11:01:27 AM.

SPARK JOB FINISHED

```
import org.apache.spark.sql.functions._

// Filter data for years after 2018
val filteredDF = combinedDF.filter(year(to_date(col("date"), "yyyy-MM-dd")) > 2018)

// Extract year and month
val dfWithYearMonth = filteredDF
  .withColumn("year", year(to_date(col("date"), "yyyy-MM-dd")))
  .withColumn("month", month(to_date(col("date"), "yyyy-MM-dd")))

// Group by year and month, and count the number of ratings
val ratingsByMonth = dfWithYearMonth.groupBy("year", "month")
  .agg(count("rating").alias("total_ratings"))

// Calculate the average number of ratings for each month across all years
val avgRatingsByMonth = ratingsByMonth.groupBy("month")
  .agg(avg("total_ratings").alias("average_ratings"))
  .orderBy("month")

// Show the result
z.show(avgRatingsByMonth)
```

settings ▾

Amazon_And_Walmart_Aggreg...



```
import org.apache.spark.sql.functions._
filteredDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [rating: double, category: string ... 1 more field]
dfWithYearMonth: org.apache.spark.sql.DataFrame = [rating: double, category: string ... 3 more fields]
ratingsByMonth: org.apache.spark.sql.DataFrame = [year: int, month: int ... 1 more field]
avgRatingsByMonth: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [month: int, average_ratings: double]
```

Took 30 sec. Last updated by anonymous at December 10 2024, 11:28:47 AM. (outdated)

READY

```
z.show(dowDF.limit(10))
```

SPARK JOB (http://nyu-dataproc-sw-cfv5.c.hpc-dataproc-19b8.internal:39707/jobs/job?id=0) FINISHED

settings

rating	category	date
1.0	Movies and TV	2015-05-13
1.0	Movies and TV	2001-03-06
5.0	Movies and TV	2009-10-19
1.0	Movies and TV	2000-12-08
5.0	Movies and TV	2016-03-01
2.0	Movies and TV	2000-11-28
4.0	Movies and TV	2005-12-06
5.0	Movies and TV	2015-02-09

Took 6 sec. Last updated by anonymous at December 10 2024, 11:01:33 AM.

```
val ratingDayDF = dowDF.groupBy("rating_dow").agg(count("rating_dow") as "rating_count")
```

FINISHED

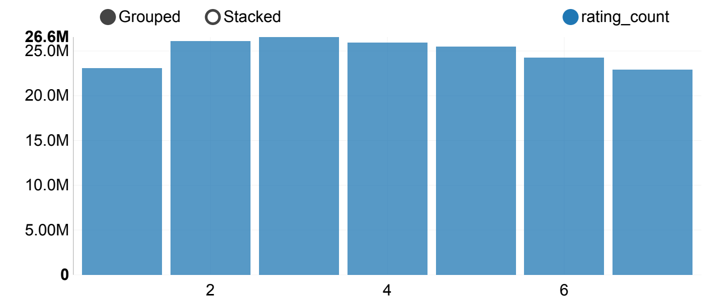
```
ratingDayDF: org.apache.spark.sql.DataFrame = [rating_dow: int, rating_count: bigint]
```

Took 0 sec. Last updated by anonymous at December 10 2024, 11:01:33 AM.

```
z.show(ratingDayDF)
```

SPARK JOB FINISHED

settings



Took 24 sec. Last updated by anonymous at December 09 2024, 9:17:02 PM. (outdated)

```
val categoryRatingsDF = dowDF.groupBy("rating_dow", "category").agg(count("rating_dow") as "rating_count")
```

FINISHED

```
categoryRatingsDF: org.apache.spark.sql.DataFrame = [rating_dow: int, category: string ... 1 more field]
```

Amazon_And_Walmart_Aggreg...

Took 0 sec. Last updated by anonymous at December 10 2024, 11:01:33 AM.

```
z.show(categoryRatingsDF)
```

SPARK JOB FINISHED

settings

rating_dow	category
4	Kindle Store
4	Sports and Outdoors
3	Magazine Subscriptions
7	Video Games
3	Video Games
2	Sports and Outdoors
6	Industrial and Scientific
3	Patio Lawn and Garden

Took 34 sec. Last updated by anonymous at December 10 2024, 11:02:07 AM.

```
filteredDF.show()
```

SPARK JOB (http://nyu-dataproc-sw-cfv5.c.hpc-dataproc-19b8.internal:39707/jobs/job?id=3) FINISHED

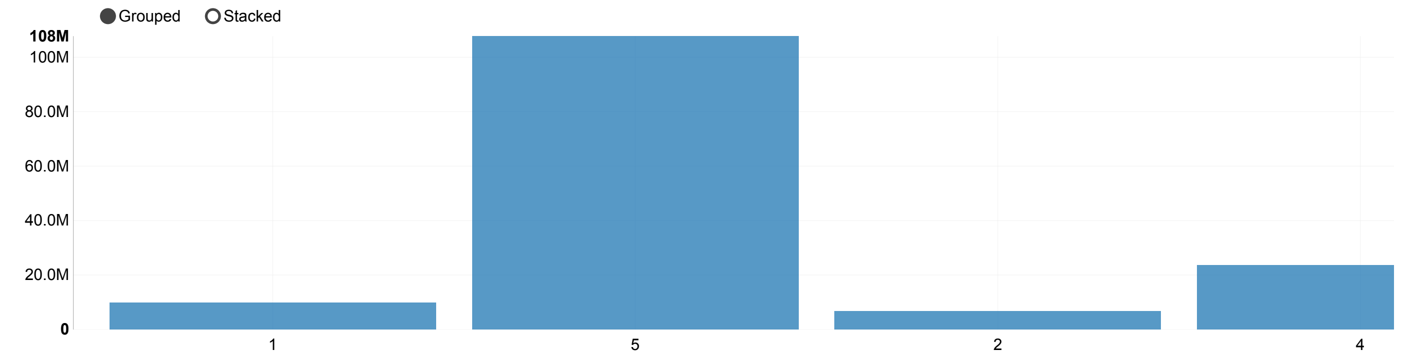
```
+------+------------+----------+
|rating|    category|      date|
+------+------------+----------+
|   1.0|Movies and TV|2015-05-13|
|   1.0|Movies and TV|2001-03-06|
|   5.0|Movies and TV|2009-10-19|
|   1.0|Movies and TV|2000-12-08|
|   5.0|Movies and TV|2016-03-01|
|   2.0|Movies and TV|2000-11-28|
|   4.0|Movies and TV|2005-12-06|
|   5.0|Movies and TV|2015-02-09|
|   5.0|Movies and TV|2014-01-25|
|   5.0|Movies and TV|2023-03-09|
|   4.0|Movies and TV|2014-04-19|
|   5.0|Movies and TV|2015-09-29|
|   5.0|Movies and TV|2014-01-11|
|   5.0|Movies and TV|2014-02-28|
```

Took 0 sec. Last updated by anonymous at December 10 2024, 11:02:07 AM.

```
val ratingDistributionDF = filteredDF.groupBy("rating").count()
z.show(ratingDistributionDF)
```

SPARK JOB FINISHED

settings



```
ratingDistributionDF: org.apache.spark.sql.DataFrame = [rating: double, count: bigint]
```

Took 18 sec. Last updated by anonymous at December 10 2024, 11:02:25 AM.

```
//Top Categories by rating count
```

```
val topCategoriesDF = filteredDF.groupBy("category").agg(count("rating") as "rating_count").orderBy(desc("rating_count"))
z.show(topCategoriesDF)
```

SPARK JOB FINISHED

settings

Amazon_And_Walmart_Aggreg...

category	rating_count
Home and Kitchen	28202600
Clothing Shoes and Jewelry	23102537
Kindle Store	16070782
Electronics	15473536
Books	9488297
Tools and Home Improvement	7891024
Movies and TV	7441129
Health and Household	7176552

```
topCategoriesDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [category: string, rating_count: bigint]
```

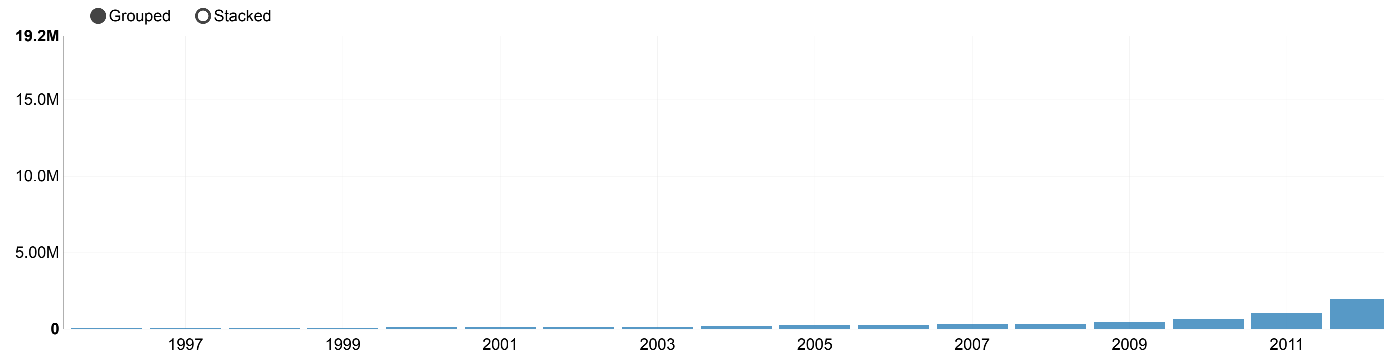
Took 22 sec. Last updated by anonymous at December 10 2024, 11:02:47 AM.

READY

```
// Trend of Ratings Over Time in Reverse Order of Year
val ratingsTrendDF = filteredDF.withColumn("year", year(col("date")))
                               .groupBy("year")
                               .agg(count("rating").alias("rating_count"))
                               .orderBy(desc("year")) // Order by year in descending order

z.show(ratingsTrendDF)
```

SPARK JOB FINISHED

settings



```
ratingsTrendDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [year: int, rating_count: bigint]
```

Took 16 sec. Last updated by anonymous at December 10 2024, 1:57:25 PM. (outdated)

```
//Average Ratings per category
val avgRatingDF = filteredDF.groupBy("category").agg(avg("rating") as "avg_rating")
z.show(avgRatingDF)
```

SPARK JOB FINISHED

settings

category	avg_rating
CDs and Vinyl	4.482091290112342

All Beauty	4.314792899408284

Amazon_And_Walmart_Aggreg...

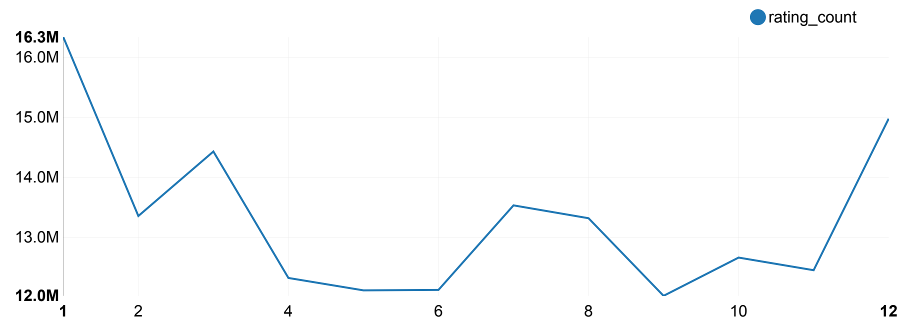
```
avgRatingDF: org.apache.spark.sql.DataFrame = [category: string, avg_rating: double]
```

Took 23 sec. Last updated by anonymous at December 10 2024, 11:03:28 AM. (outdated)

SPARK JOB FINISHED

```
//Seasonal Ratings(Month wise analysis)
val monthWiseDF = filteredDF.withColumn("month", month(col("date")))
                            .groupBy("month")
                            .agg(count("rating") as "rating_count")
z.show(monthWiseDF)
```

settings



```
monthWiseDF: org.apache.spark.sql.DataFrame = [month: int, rating_count: bigint]
```

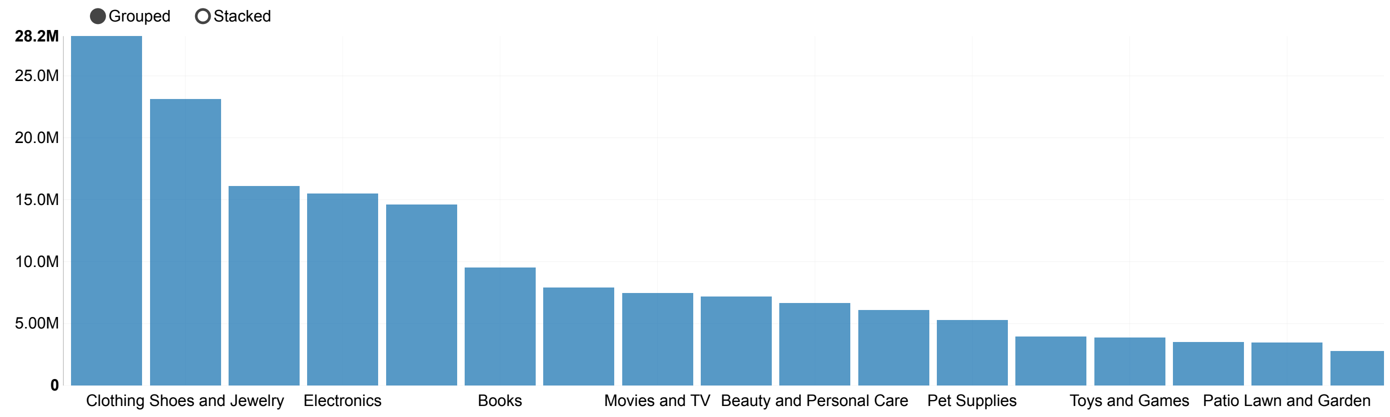
Took 18 sec. Last updated by anonymous at December 10 2024, 11:03:46 AM. (outdated)

READY

SPARK JOB FINISHED

```
import org.apache.spark.sql.functions._

// Group by category and count the total number of ratings
val totalRatingsByCategory = combinedDF.groupBy("category")
  .agg(count("rating").alias("total_ratings"))
  .orderBy(desc("total_ratings")) // Order by total ratings in descending order

// Show the result
z.show(totalRatingsByCategory)
```

settings



```
import org.apache.spark.sql.functions._
totalRatingsByCategory: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [category: string, total_ratings: bigint]
```

Took 12 sec. Last updated by anonymous at December 10 2024, 11:52:01 AM. (outdated)

SPARK JOB FINISHED

```
//HeatMaps of Ratings by Day of week and Month
val dowMonthDF = filteredDF.withColumn("day_of_week", dayofweek(col("date")))
                           .withColumn("month", month(col("date")))
                           .groupBy("day_of_week", "month")
                           .agg(count("rating") as "rating_count")
z.show(dowMonthDF)
```

settings

day_of_week	month
3	1
1	9
1	1
3	5
2	5
2	9
7	12
7	11

```
dowMonthDF: org.apache.spark.sql.DataFrame = [day_of_week: int, month: int ... 1 more field]
```

Took 21 sec. Last updated by anonymous at December 10 2024, 11:04:07 AM.

SPARK JOB FINISHED

```
// NOW WORKING ON THE WALMART DATA HERE ON AS WELL
///user/sc10670_nyu_edu/project/cleaned-walmart-data.parquet
val walmartDF = spark.read.parquet("project/cleaned-walmart-data.parquet")
walmartDF.show()
```

```
+----------------+------------+----------------+--------+----------+--------------+----------------+--------------------+--------------------+---
---------+-------+------------------+-------------+--------+
|            city|customer_age|customer_segment|discount|order_date|order_quantity|product_category|        product_name|product_sub_category|
profit| region|     shipping_cost|         state|zip_code|
+----------------+------------+----------------+--------+----------+--------------+----------------+--------------------+--------------------+---
---------+-------+------------------+-------------+--------+
|        Parkland|          22|       Corporate|     0.1|2020-08-10|            27| Office Supplies|GBC Imprintable C...|Binders and Binde...|  5
493.93774|   West|122.50176357542627|   Washington|   98444|
|West Valley City|          29|     Home Office|    0.16|2022-12-11|            13| Office Supplies|           Avery 501|              Labels| 27
098.13955|   West| 17.54262454889772|         Utah|   84120|
|       Claremont|          78|        Consumer|     0.2|2019-07-10|            34| Office Supplies|Prang Drawing Pen...| Pens & Art Supplies|  3
916.31334|   West|12.185698591528354|   California|   91711|
|       Claremont|          78|        Consumer|     0.2|2019-07-10|            34| Office Supplies|Prang Drawing Pen...| Pens & Art Supplies|  3
916.31334|   West| 47.86808107116942|   California|   91711|
|        Ferguson|          82|     Home Office|    0.25|2022-05-07|            29| Office Supplies|Avery Poly Binder...|Binders and Binde...| -5
937.97782|Central|100.27971116167916|           MO|   63135|
|        Savannah|          53|  Small Business|    0.24|2020-11-09|            26| Office Supplies|GBC Poly Designer...|Binders and Binde...|
```

Took 43 sec. Last updated by anonymous at December 13 2024, 6:00:33 PM.

SPARK JOB FINISHED

```
// Day Wise Statistics:
// Add a new column for day of the week extracted from 'order_date'
val walmartDFWithDay = walmartDF.withColumn("day_of_week", date_format(to_date(col("order_date"), "yyyy-MM-dd"), "EEEE"))

// Group data by day of the week and calculate trends
// For example, calculate total orders, total profit, and average shipping cost by day of the week
val trendsByDay = walmartDFWithDay.groupBy("day_of_week")
  .agg(
    count("*").alias("total_orders"),
    sum("profit").alias("total_profit"),
    avg("shipping_cost").alias("average_shipping_cost"),
    sum("order_quantity").alias("total_quantity")
  )
  .orderBy("day_of_week") // Sort by day of the week

// Show the results
z.show(trendsByDay)
```

settings

day_of_week	total_orders	total_profit
Friday	145257	9.486676153688476E8
Monday	144357	9.5079782742482E8
Saturday	145008	9.560626784944531E8
Sunday	145135	9.48903087026384E8
Thursday	145380	9.517193007878091E8
Tuesday	145421	9.610082306486115E8

Wednesday	145544	9.499971769005625E8

Amazon_And_Walmart_Aggreg...

```
walmartDFWithDay: org.apache.spark.sql.DataFrame = [city: string, customer_age: string ... 13 more fields]
trendsByDay: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [day_of_week: string, total_orders: bigint ... 3 more fields]
```

Took 7 sec. Last updated by anonymous at December 13 2024, 6:00:40 PM. (outdated)

```
// Number of Unique Cities per State
val stateCitiesDF = walmartDF.groupBy("state")
  .agg(countDistinct("city").alias("number_of_unique_cities"))
  .orderBy(desc("number_of_unique_cities"))
  .limit(10)
```

FINISHED

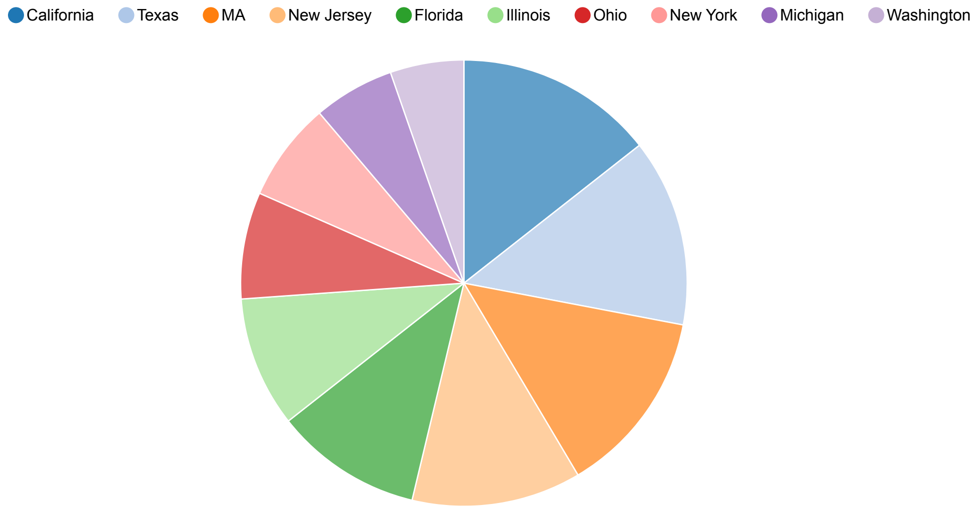
```
stateCitiesDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [state: string, number_of_unique_cities: bigint]
```

Took 0 sec. Last updated by anonymous at December 13 2024, 6:02:18 PM.

```
z.show(stateCitiesDF)
```

SPARK JOB FINISHED

settings



Took 5 sec. Last updated by anonymous at December 13 2024, 6:02:25 PM. (outdated)

```
//1. Customer Insights
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._
```

FINISHED

```
import org.apache.spark.sql.types._
import org.apache.spark.sql.functions._
```

Took 0 sec. Last updated by anonymous at December 10 2024, 11:04:08 AM.

```
//Age Group Analysis
import org.apache.spark.sql.functions._

val ageGroupDF = walmartDF
  .withColumn("age_group", when(col("customer_age").cast("int").between(0,
      20), "0-20")
    .when(col("customer_age").cast("int").between(21, 40), "21-40")
    .when(col("customer_age").cast("int").between(41, 60), "41-60")
    .otherwise("60+"))
  .groupBy("age_group")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit" // Corrected
        casting
  )
z.show(ageGroupDF)
```

SPARK JOB FINISHED

settings

Available Fields

age_group total_order_quantity total_profit

Amazon_And_Walmart_Aggreg...

keys

age_group ✖

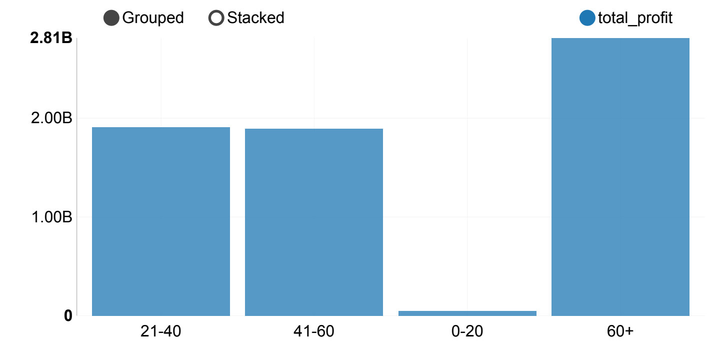
groups

values

total_profit SUM ✖

xAxis :

Default Rotate Hide



```
import org.apache.spark.sql.functions._
ageGroupDF: org.apache.spark.sql.DataFrame = [age_group: string, total_order_quantity: bigint ... 1 more field]
```

Took 2 sec. Last updated by anonymous at December 10 2024, 11:04:10 AM. (outdated)

```
val segmentTrendsDF = walmartDF
  .groupBy("customer_segment")
  .agg(sum("order_quantity") as "total_order_quantity", sum(col("profit").cast("double")) as "total_profit")
z.show(segmentTrendsDF)
```

SPARK JOB FINISHED

settings

customer_segment	total_order_quantity
Corporate	6496091
Home Office	6488038
Consumer	6430325
Small Business	6486496

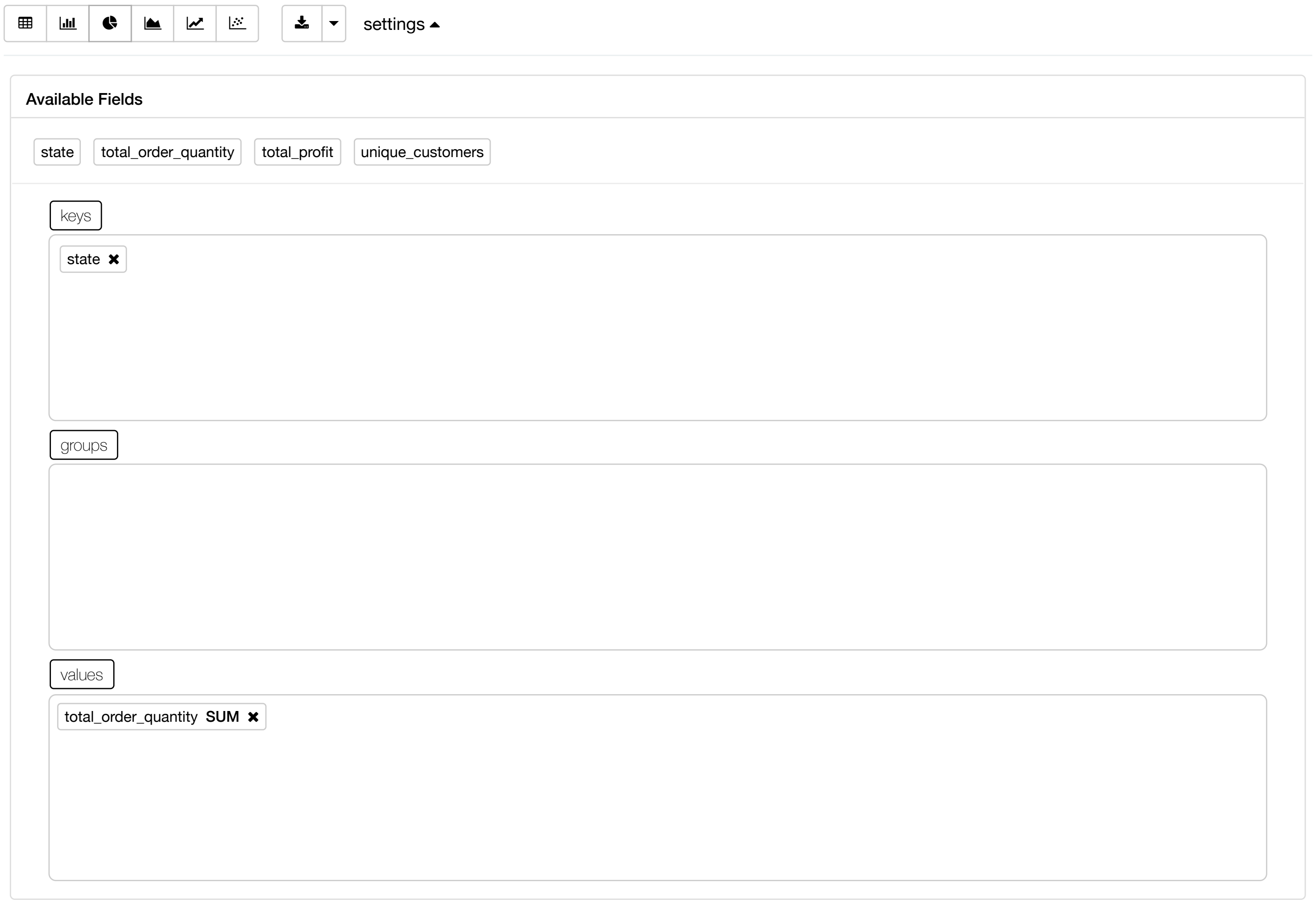
```
segmentTrendsDF: org.apache.spark.sql.DataFrame = [customer_segment: string, total_order_quantity: bigint ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:11 AM.

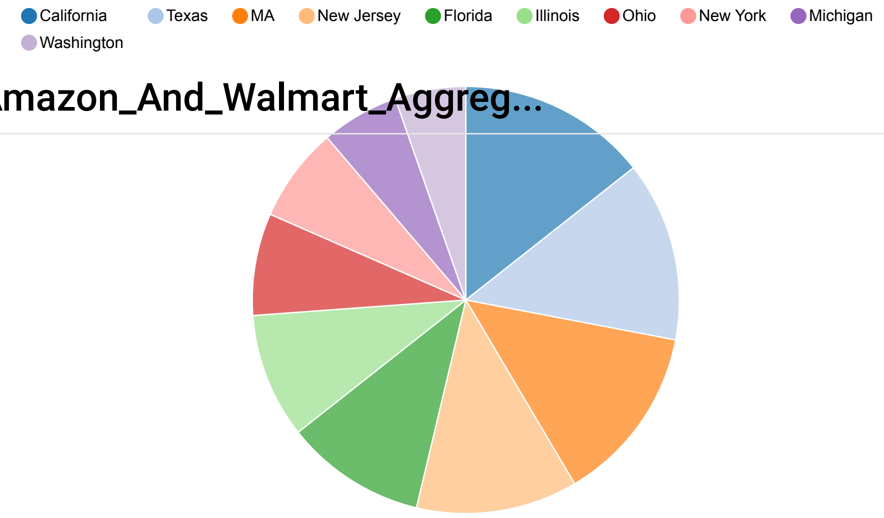
Amazon_And_Walmart_Aggreg...

```
// GEOGRAPHIC INSIGHTS
// State-Wise Performance Ordered by Total Order Quantity (Top 10)
val statePerformanceDF = walmartDF
  .groupBy("state")
  .agg(
    sum("order_quantity").alias("total_order_quantity"),
    sum(col("profit").cast("double")).alias("total_profit"),
    countDistinct("city").alias("unique_customers")
  )
  .orderBy(desc("total_order_quantity")) // Order by total order quantity in descending order
  .limit(10) // Show only the top 10 states

z.show(statePerformanceDF)
```

SPARK JOB FINISHED



Amazon_And_Walmart_Aggreg...



```
statePerformanceDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [state: string, total_order_quantity: bigint ... 2 more fields]
```

Took 5 sec. Last updated by anonymous at December 13 2024, 6:06:31 PM. (outdated)

```
val cityTrendsDF = walmartDF
  .groupBy("city")
  .agg(
    sum(col("profit").cast("double")) as "total_profit"
  )
  .orderBy(desc("total_profit"))
  .limit(100)

z.show(cityTrendsDF)
```

SPARK JOB FINISHED

settings

city	total_profit
Ozark	5984938.296080001
Hillside	5848526.820260001
Ferguson	5837432.217740001
Cheshire	5772963.262820002
Beckley	5772184.0782200005
Wentzville	5769941.050080003
Meriden	5723116.733729998
Lemon Grove	5667371.879629998

```
cityTrendsDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [city: string, total_profit: double]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:14 AM.

```
val regionPerformanceDF = walmartDF
  .groupBy("region")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(regionPerformanceDF)
```

SPARK JOB FINISHED

settings

Available Fields

region total_order_quantity total_profit

keys

region ✖

Amazon_And_Walmart_Aggreg...

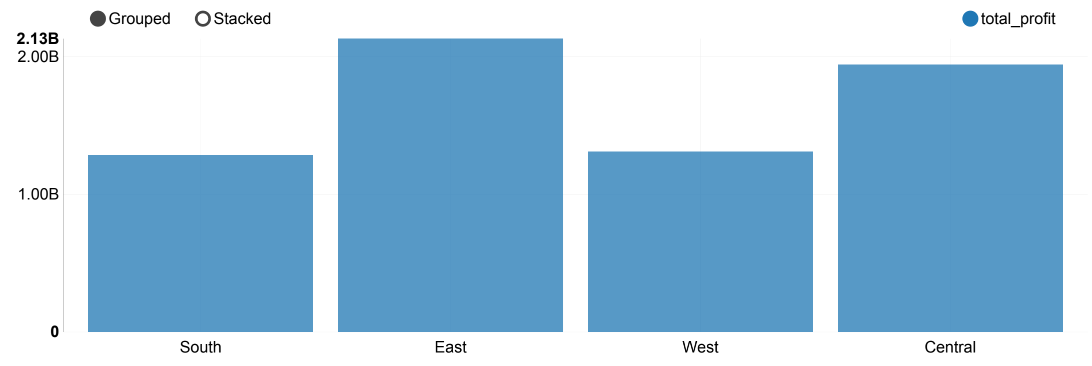
groups

values

total_profit SUM ✖

xAxis :

Default Rotate Hide



```
regionPerformanceDF: org.apache.spark.sql.DataFrame = [region: string, total_order_quantity: bigint ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:15 AM. (outdated)

```
//3. Sales and Profitability
```

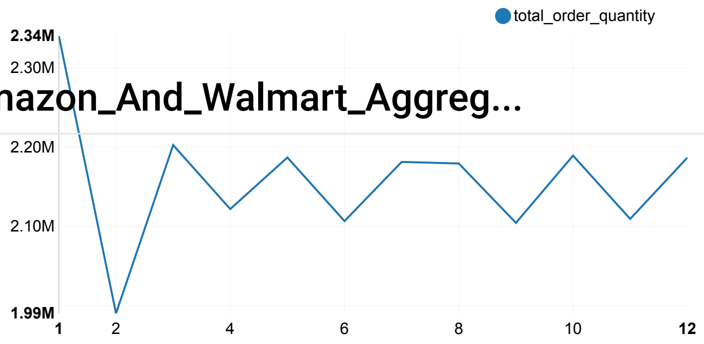
FINISHED

Took 0 sec. Last updated by anonymous at December 09 2024, 11:54:01 PM.

```
//Monthly Trends
val monthlyTrendsDF = walmartDF
  .withColumn("month", month(col("order_date")))
  .groupBy("month")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(monthlyTrendsDF)
```

SPARK JOB FINISHED

settings ▾



Amazon_And_Walmart_Aggreg...

```
monthlyTrendsDF: org.apache.spark.sql.DataFrame = [month: int, total_order_quantity: bigint ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:16 AM. (outdated)

```
//Discounts Impact on Profit
val discountProfitCorrelationDF = walmartDF
  .select(
    corr(col("discount").cast("double"), col("profit").cast("double")) as "correlation_discount_profit"
  )

z.show(discountProfitCorrelationDF)
```

SPARK JOB FINISHED

settings

correlation_discount_profit
-0.001620777132428078

```
discountProfitCorrelationDF: org.apache.spark.sql.DataFrame = [correlation_discount_profit: double]
```

Took 3 sec. Last updated by anonymous at December 10 2024, 11:04:19 AM.

```
//YoY Growth
val yoyGrowthDF = walmartDF
  .withColumn("year", year(col("order_date")))
  .groupBy("year")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(yoyGrowthDF)
```

SPARK JOB FINISHED

settings

year	total_order_quantity
2020	6446777
2023	147515
2021	6415929
2022	6446571
2019	6444158

```
yoyGrowthDF: org.apache.spark.sql.DataFrame = [year: int, total_order_quantity: bigint ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:20 AM.

Amazon_And_Walmart_Aggreg...

SPARK JOB FINISHED

```
//Seasonal Trends
val quarterlyTrendsDF = walmartDF
  .withColumn("quarter", quarter(col("order_date")))
  .groupBy("quarter")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(quarterlyTrendsDF)
```

settings

quarter	total_order_quantity
1	6533554
3	6465616
4	6485833
2	6415947

```
quarterlyTrendsDF: org.apache.spark.sql.DataFrame = [quarter: int, total_order_quantity: bigint ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:21 AM.

SPARK JOB FINISHED

```
//Weekday vs Weekend
val dayOfWeekDF = walmartDF
  .withColumn("day_of_week", date_format(col("order_date"), "E"))
  .withColumn("is_weekend", when(col("day_of_week").isin("Sat", "Sun"), "Weekend").otherwise("Weekday"))
  .groupBy("is_weekend")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(dayOfWeekDF)
```

settings

is_weekend	total_order_quantity
Weekday	18509345
Weekend	7391605

```
dayOfWeekDF: org.apache.spark.sql.DataFrame = [is_weekend: string, total_order_quantity: bigint ... 1 more field]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:22 AM.

SPARK JOB FINISHED

```
//Cross Dimensional Insights
//Region Segment Trends
val regionSegmentTrendsDF = walmartDF
  .groupBy("region", "customer_segment")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(regionSegmentTrendsDF)
```

Amazon_And_Walmart_Aggreg...

settings

region	customer_segment	total_order_quantity
Central	Home Office	1883319
West	Home Office	1281051
South	Consumer	1244152
Central	Corporate	1874548
South	Small Business	1255689
South	Home Office	1248935
West	Small Business	1275144
East	Corporate	2075442

```
regionSegmentTrendsDF: org.apache.spark.sql.DataFrame = [region: string, customer_segment: string ... 2 more fields]
```

Took 1 sec. Last updated by anonymous at December 10 2024, 11:04:23 AM.

SPARK JOB FINISHED

```
// Age vs Region Trends
val ageRegionTrendsDF = walmartDF
  .withColumn("age_group", when(col("customer_age").cast("int").between(0, 20), "0-20")
    .when(col("customer_age").cast("int").between(21, 40), "21-40")
    .when(col("customer_age").cast("int").between(41, 60), "41-60")
    .otherwise("60+"))
  .groupBy("region", "age_group")
  .agg(
    sum("order_quantity") as "total_order_quantity",
    sum(col("profit").cast("double")) as "total_profit"
  )

z.show(ageRegionTrendsDF)
```

settings

region ▲	age_group	total_order_quantity
Central	41-60	2146706
Central	60+	3163420
Central	21-40	2135494
Central	0-20	55254
East	60+	3495246
East	0-20	58558
East	21-40	2360676
East	41-60	2361095

```
ageRegionTrendsDF: org.apache.spark.sql.DataFrame = [region: string, age_group: string ... 2 more fields]
```

Took 2 sec. Last updated by anonymous at December 10 2024, 11:04:25 AM.

READY