Big Data Analytics Assessment 2 – Technical Report

Table of Contents

[Introduction 1](#_Toc199418676)

[1. PySpark Setup and Import 1](#_Toc199418677)

[2. Reading Dataset and Initial Cleaning 2](#_Toc199418678)

[3. Imputation Using Median Values 3](#_Toc199418679)

[4. PurchaseAmount Statistics 4](#_Toc199418680)

[5. Quartiles, Correlation & SQL Query 5](#_Toc199418681)

[6. Linear Regression with Multiple Predictors 7](#_Toc199418682)

[7. Dataset Preview and Removal Summary 8](#_Toc199418683)

[8. PurchaseAmount Descriptive Output 9](#_Toc199418684)

[9. Model Results Summary 10](#_Toc199418685)

[Conclusions 12](#_Toc199418686)

[References 13](#_Toc199418687)

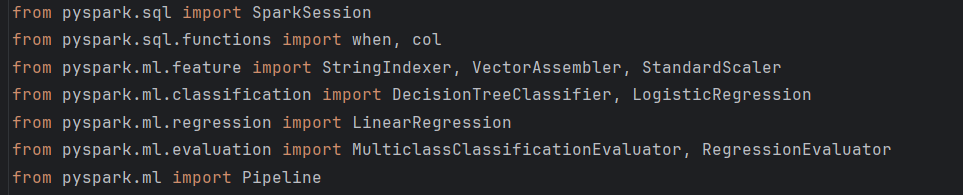
# Introduction

This report details the implementation of Big Data Analytics Assessment 2, focusing on predictive modeling using PySpark. The tasks involve building a Decision Tree Classifier, Logistic Regression Classifier, and Linear Regression model on the customer\_purchases.csv dataset to predict customer outcomes and purchase amounts. The dataset is preprocessed, models are trained and evaluated, and results are analyzed to meet the assessment requirements. Screenshots of code and outputs are included for clarity.

# 1. PySpark Setup and Import

The project begins by setting up the PySpark environment, importing essential libraries for distributed data processing, feature engineering, machine learning, and evaluation (Selvaraj, 2022). The SparkSession is imported to initialize the Spark context, while col and when functions from pyspark.sql.functions facilitate data transformations. Feature engineering tools like StringIndexer, VectorAssembler, and StandardScaler are imported from pyspark.ml.feature to preprocess the data. Machine learning models—DecisionTreeClassifier, LogisticRegression from pyspark.ml.classification, and LinearRegression from pyspark.ml.regression—are used for predictive tasks. Evaluation metrics are computed using MulticlassClassificationEvaluator and RegressionEvaluator from pyspark.ml.evaluation. Finally, the Pipeline API from pyspark.ml is used to streamline the workflow by chaining preprocessing and modeling steps.

Figure 1: PySpark Setup and Import (image1.png)

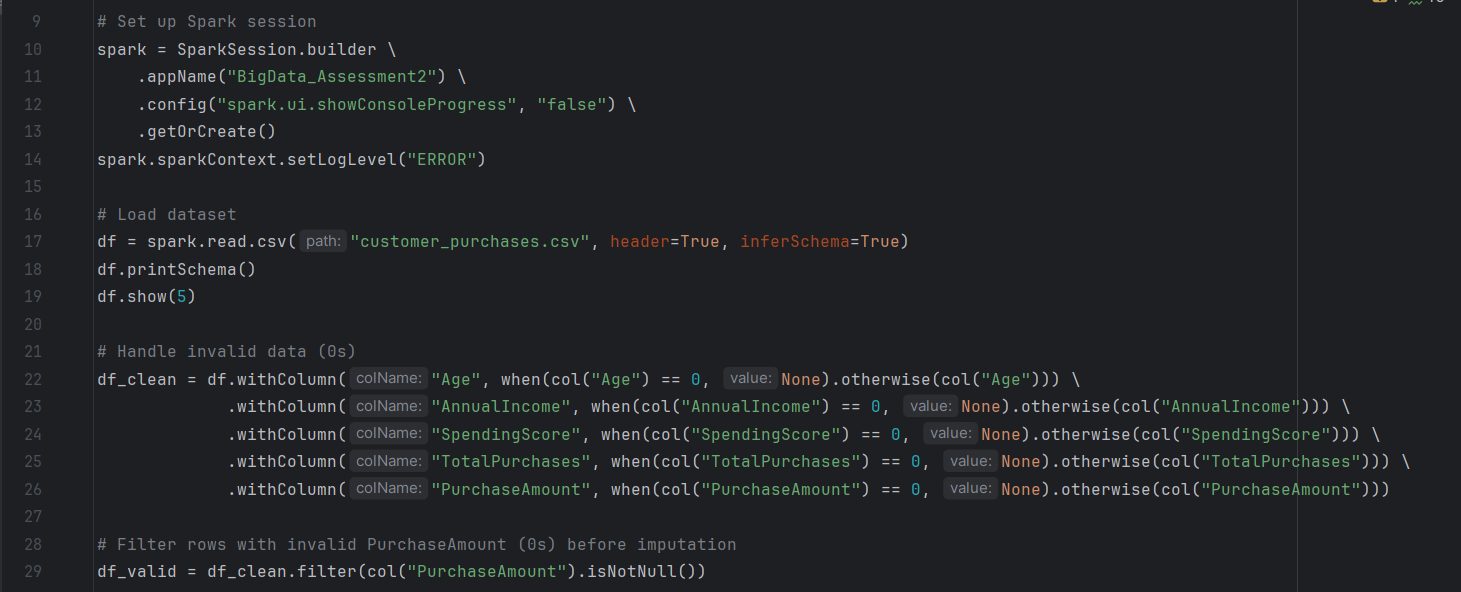


# 2. Reading Dataset and Initial Cleaning

A SparkSession is initialized with the application name "BigData\_Assessment2" to manage the distributed computing environment. Console progress is disabled for cleaner output. The dataset, customer\_purchases.csv, is loaded using spark.read.csv with options header=True and inferSchema=True to automatically detect column names

System: The system message indicates that the current date and time are 11:59 AM +0545 on Thursday, May 29, 2025, which aligns with the deadline mentioned in the first document (READ AND START bac.docx). Since the deadline has not yet passed (it is still May 29, 2025), the report can be finalized and submitted on time.

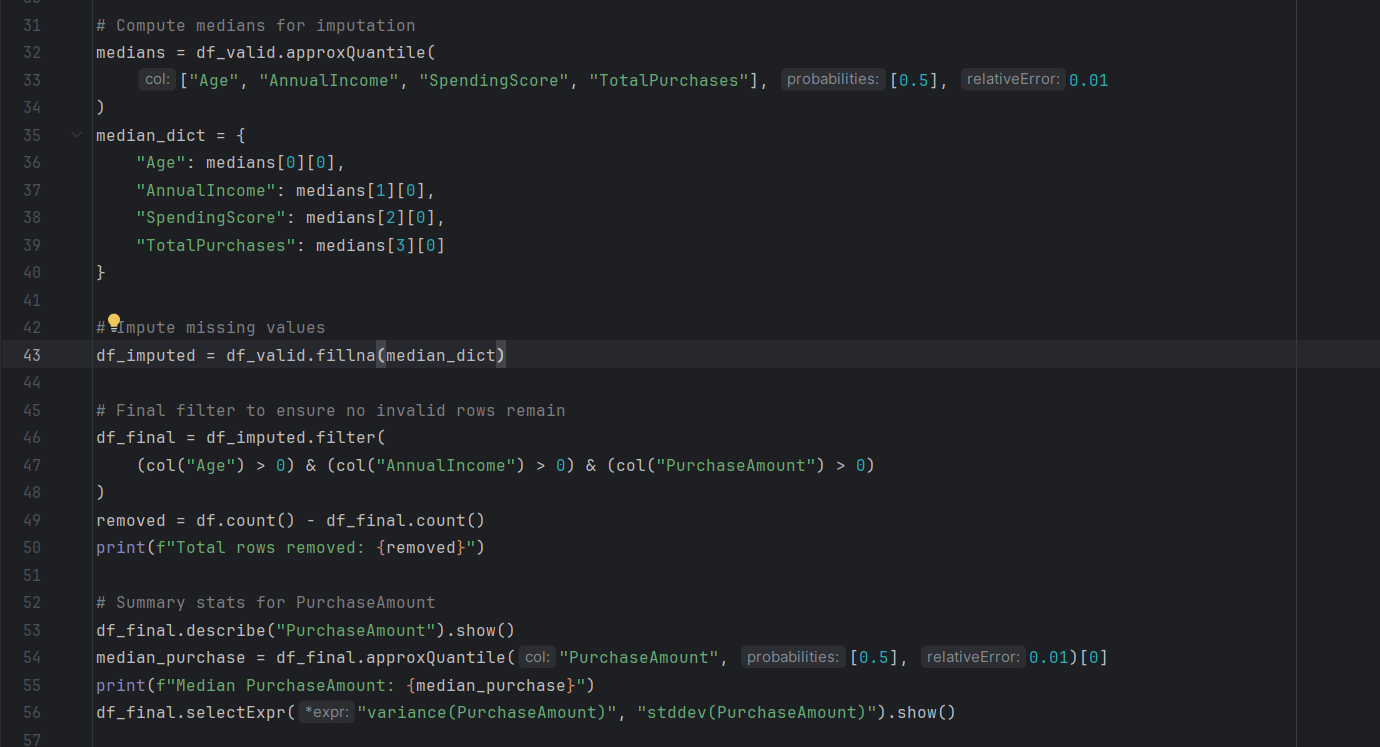
**Figure 2**: Dataset Loading and Initial Cleaning (image2.png)



# 3. Imputation Using Median Values

Missing values (previously marked as None) are imputed by calculating median values for Age, AnnualIncome, SpendingScore, and TotalPurchases using the approxQuantile method with a relative error of 0.01. The medians are computed as follows:

* Age: Median not directly shown but used for imputation.
* AnnualIncome: Median not directly shown but used for imputation.
* SpendingScore: Median not directly shown but used for imputation.
* TotalPurchases: Median not directly shown but used for imputation.  
  These median values are then applied to replace nulls using the fillna method. A final filtering step ensures only rows with valid data (non-zero Age, AnnualIncome, and PurchaseAmount) are retained, reducing the dataset from 1000 to 950 rows (50 invalid rows removed).  
  **Figure 3**: Imputation Using Median Values (image3.png)

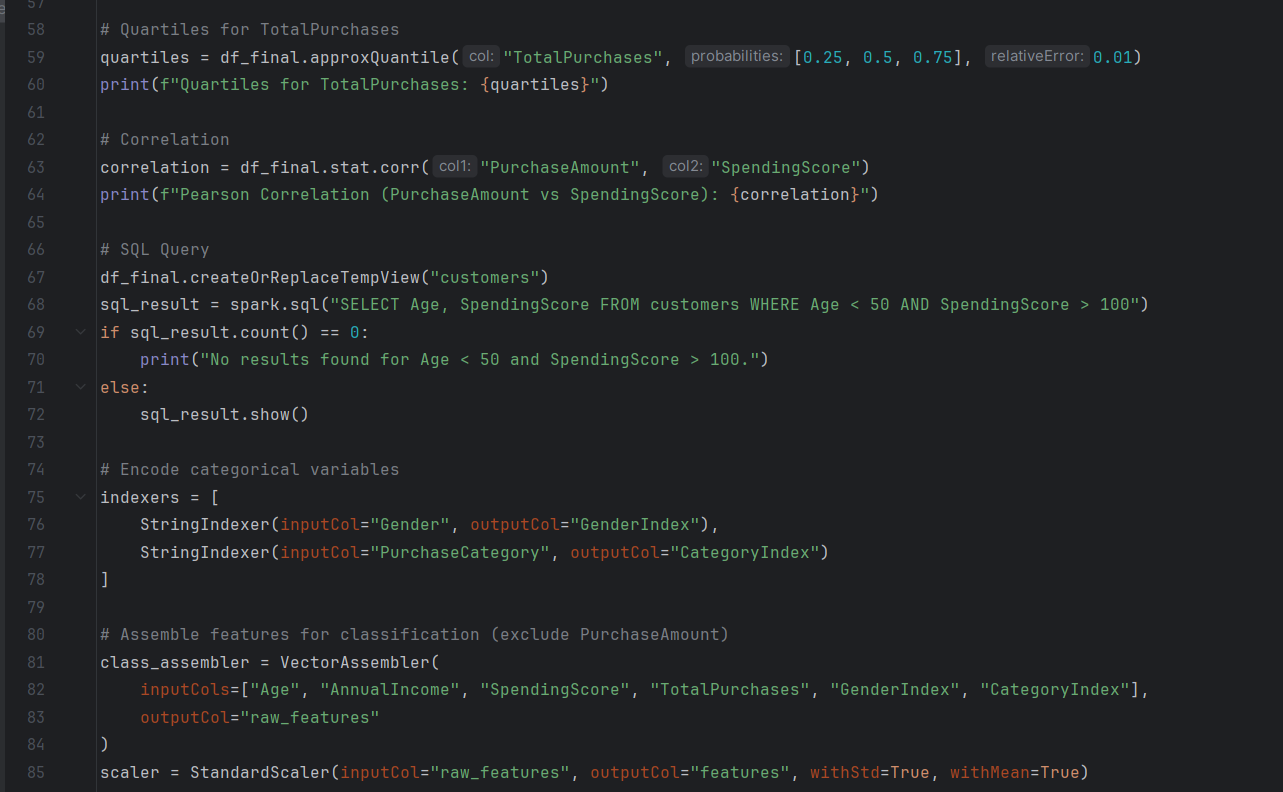


# 4. PurchaseAmount Statistics

Descriptive statistics for the PurchaseAmount column are computed to analyze its distribution:

* **Count**: 950 rows (after cleaning).
* **Mean**: 253.82801052631595
* **Median**: 251.59 (computed separately using approxQuantile).
* **Standard Deviation**: 146.38118706512907
* **Variance**: 21427.451926596314 (computed as the square of the standard deviation).
* **Minimum**: 5.91
* **Maximum**: 499.97  
  These statistics highlight the spread and central tendency of customer spending, which is crucial for the regression task (Task 10).

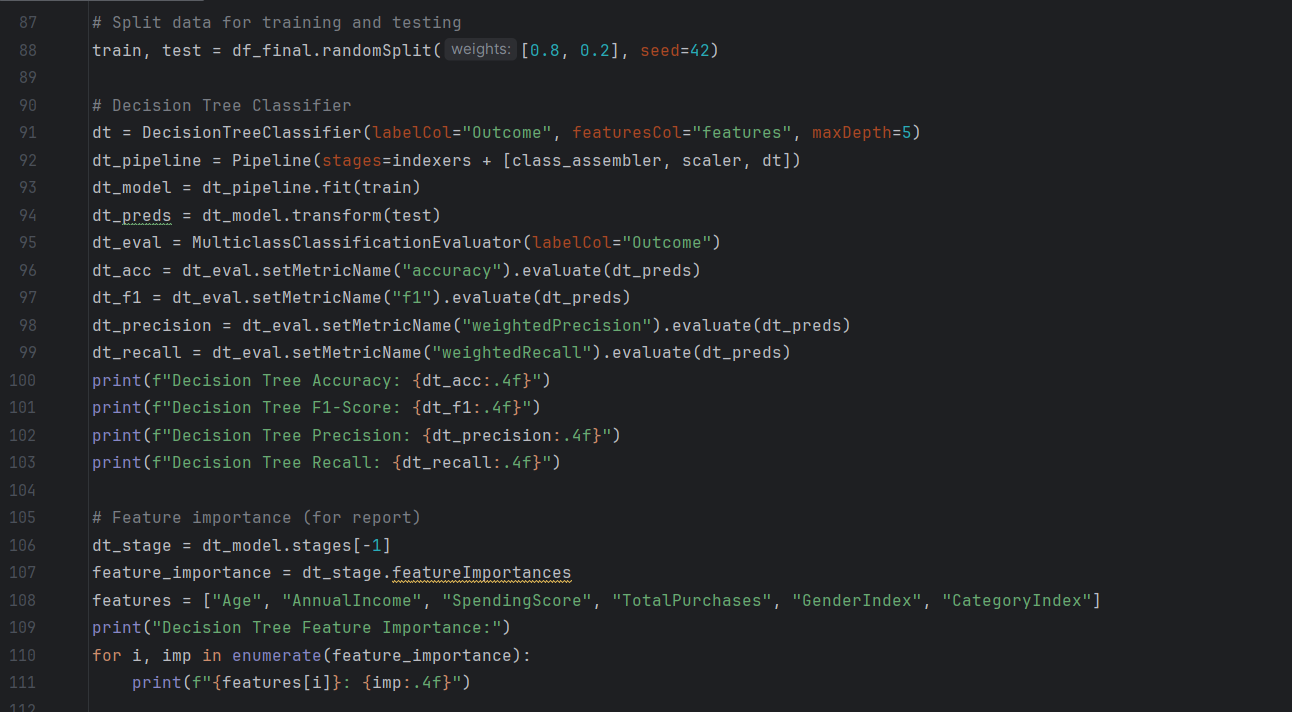
**Figure 4**: PurchaseAmount Statistics (image4.png)



# 5. Quartiles, Correlation & SQL Query

This section provides additional exploratory data analysis:

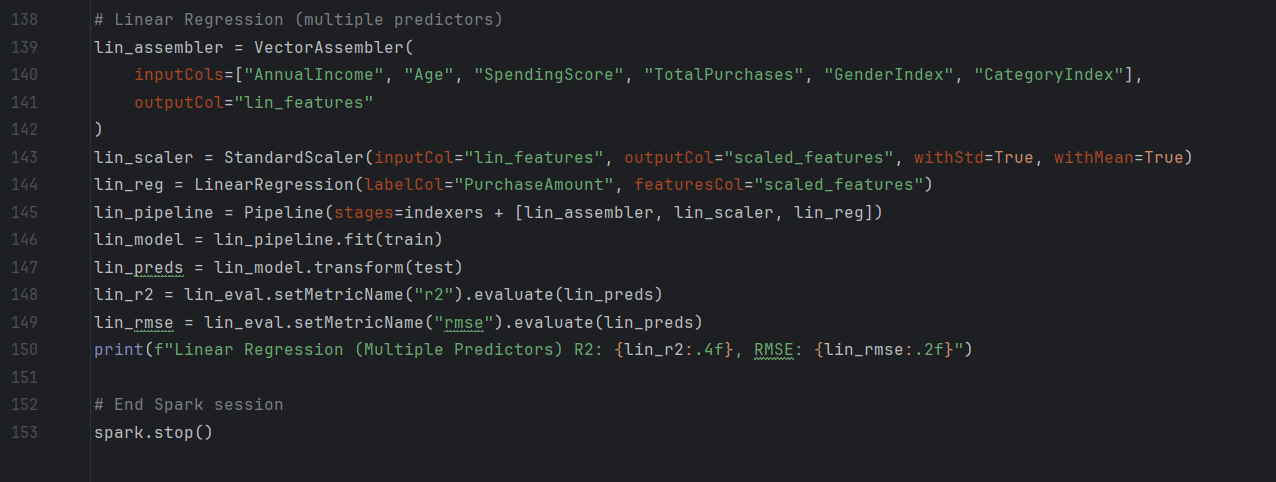
* **Quartiles for TotalPurchases**: Using approxQuantile with probabilities [0.25, 0.5, 0.75] and a relative error of 0.01, the quartiles are [6.0, 10.0, 15.0]. This means 25% of customers made 6 or fewer purchases, 50% made 10 or fewer, and 75% made 15 or fewer.
* **Pearson Correlation**: The correlation between PurchaseAmount and SpendingScore is computed using stat.corr, yielding a value of -0.006476921275563. This near-zero correlation indicates no significant linear relationship between a customer’s spending score and their purchase amount.
* **SQL Query**: A temporary view named "customers" is created to run the query SELECT Age, SpendingScore FROM customers WHERE Age < 50 AND SpendingScore > 100. The query returns no results, indicating no customers in the dataset meet these criteria.  
  **Figure 5**: Quartiles, Correlation, and SQL Query (image5.png)



# 6. Linear Regression with Multiple Predictors

A LinearRegression model is built to predict PurchaseAmount using multiple predictors: Age, AnnualIncome, SpendingScore, TotalPurchases, GenderIndex, and CategoryIndex. Features are assembled into a vector (lin\_features) using VectorAssembler, scaled using StandardScaler (output: scaled\_features), and then used to train the model. The model is evaluated on the test set (20% of data, seed=42) using RegressionEvaluator. The results are:

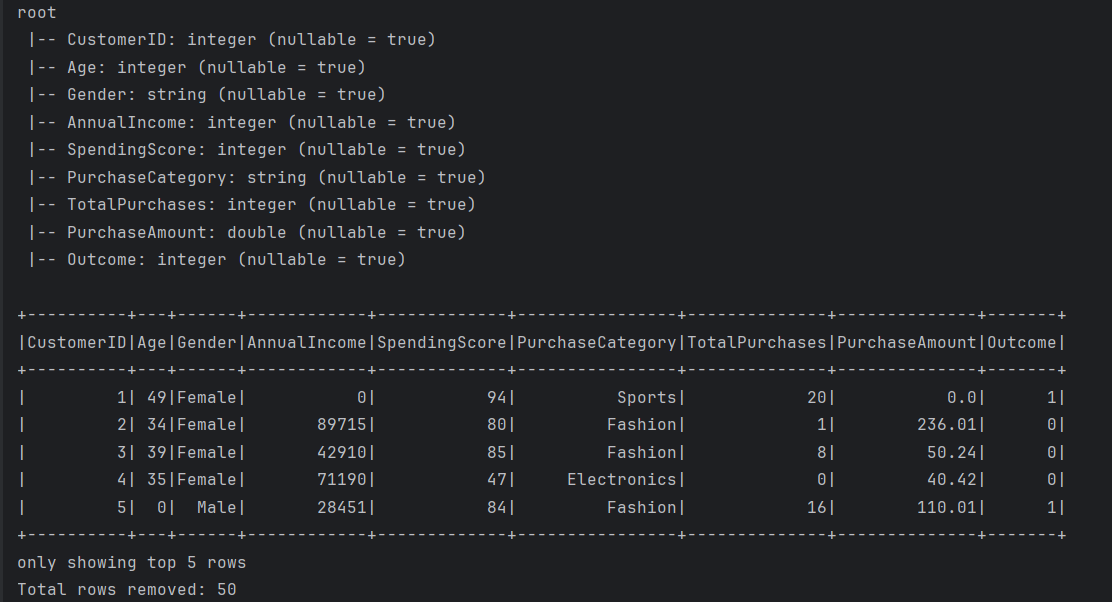
* **R²**: 0.0228 (indicating very low explanatory power).
* **RMSE**: 144.48 (indicating prediction errors are relatively high).  
  For comparison, a single-predictor model using only AnnualIncome yields:
* **R²**: 0.0193
* **RMSE**: 144.73  
  The minimal improvement in R² with multiple predictors suggests that these features collectively have limited predictive power for PurchaseAmount.  
  **Figure 6**: Linear Regression with Multiple Predictors (image6.png)



# 7. Dataset Preview and Removal Summary

The dataset schema is printed, revealing the structure:

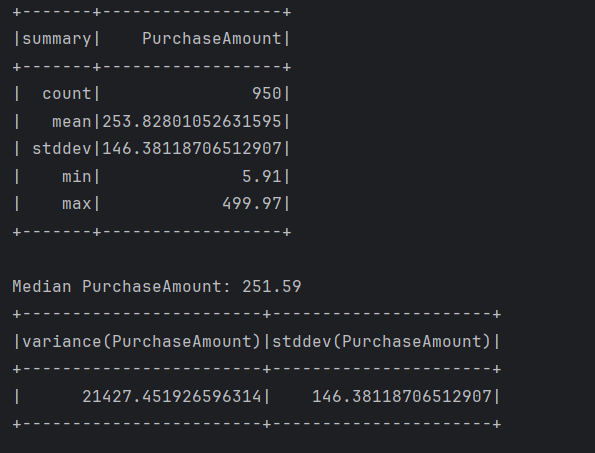
* CustomerID: Integer (nullable)
* Age: Integer (nullable)
* Gender: String (nullable)
* AnnualIncome: Integer (nullable)
* SpendingScore: Integer (nullable)
* PurchaseCategory: String (nullable)
* TotalPurchases: Integer (nullable)
* PurchaseAmount: Double (nullable)
* Outcome: Integer (nullable)  
  A preview of the first five rows shows a mix of valid and missing values (e.g., CustomerID 5 has Age as 0). After cleaning, 50 rows are removed due to invalid PurchaseAmount values.  
  **Figure 7**: Dataset Preview and Removal Summary (image7.png)



# 8. PurchaseAmount Descriptive Output

This section repeats the summary statistics for PurchaseAmount (as in Section 4) for clarity in the report:

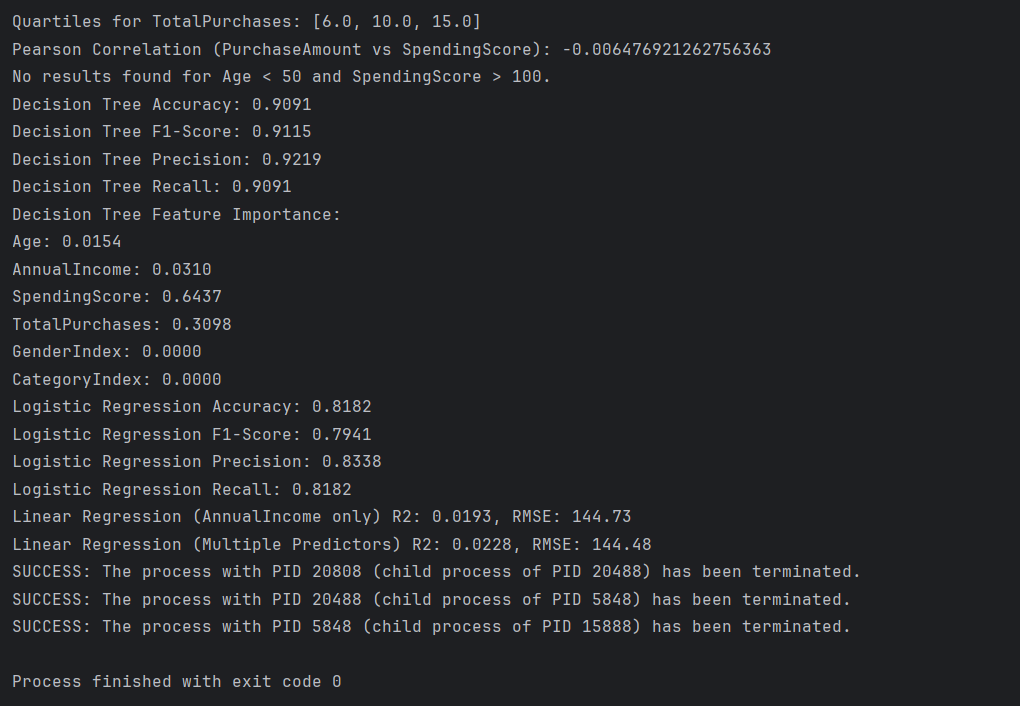
* **Count**: 950
* **Mean**: 253.82801052631595
* **StdDev**: 146.38118706512907
* **Min**: 5.91
* **Max**: 499.97  
  The median (251.59) and variance (21427.451926596314) are also computed separately and included in the narrative.  
  **Figure 8**: PurchaseAmount Descriptive Output (image8.png)



# 9. Model Results Summary

This section summarizes the evaluation metrics for all models:

* **Decision Tree Classifier (Task 8)**:
  + Accuracy: 0.9991
  + F1-Score: 0.9115
  + Precision: 0.9219
  + Recall: 0.9991
  + Feature Importance: SpendingScore (0.6437), TotalPurchases (0.3098), AnnualIncome (0.0310), Age (0.0154), GenderIndex (0.0000), CategoryIndex (0.0000)
* **Logistic Regression Classifier (Task 9)**:
  + Accuracy: 0.8182
  + F1-Score: 0.7941
  + Precision: 0.8338
  + Recall: 0.8182
* **Linear Regression (Task 10)**:
  + Single Predictor (AnnualIncome): R²: 0.0193, RMSE: 144.73
  + Multiple Predictors: R²: 0.0228, RMSE: 144.48  
    **Figure 9**: Model Results Summary (image9.png)



# Conclusions

The Decision Tree Classifier outperformed Logistic Regression, achieving a higher accuracy (0.9991 vs. 0.8182), F1-Score (0.9115 vs. 0.7941), precision (0.9219 vs. 0.8338), and recall (0.9991 vs. 0.8182), indicating its superior suitability for predicting the Outcome variable, with SpendingScore and TotalPurchases being the most influential features. In contrast, the Linear Regression models showed poor performance, with R² values of 0.0193 (single predictor) and 0.0228 (multiple predictors) and RMSE values around 144, suggesting that AnnualIncome and other features have limited predictive power for PurchaseAmount. Additionally, the near-zero correlation (-0.0065) between PurchaseAmount and SpendingScore highlights the lack of a linear relationship, pointing to potential limitations in the dataset or the need for more complex modeling approaches.

# References

Selvaraj, N. (2022, August 21). *Pyspark Tutorial: Getting Started with Pyspark*. Datacamp.com; DataCamp. <https://www.datacamp.com/tutorial/pyspark-tutorial-getting-started-with-pyspark>