Project Summary

This project implements an end-to-end workflow for analyzing customer purchase behavior using core Python data structures and control flow. The solution processes transactional data to classify customers by spending tier, analyze product categories, and generate business insights relevant to an e-commerce environment.

The final script computes per-customer spending totals and classifications, revenue by product category, unique products, product purchase frequencies, top-spending customers, and multiple set-based customer segments (e.g., multi-category buyers and customers who purchased both Electronics and Clothing). Output is formatted for clarity and readability to support business interpretation.

Role of GitHub Copilot

GitHub Copilot was used as an assistive tool throughout development to accelerate implementation and validate logic, not as a substitute for design or reasoning.

- **Sample data generation**
  Copilot generated realistic example customers, products, categories, and transactions. This enabled validation of logic against concrete data and made it easier to confirm that all algorithms behaved correctly across different scenarios (e.g., multi-category buyers, high-value customers).
- **Refactoring guidance**
  Recommended single-pass aggregation techniques (such as computing customer totals by iterating through transactions once), improving efficiency over nested-loop approaches.
- **Readable constructs**
  Proposed concise set-based logic and comprehensions to simplify category membership checks and customer segmentation.
- **Common idioms**
  Suggested standard patterns such as `dict.get()` for frequency counting and `sorted(..., key=..., reverse=True)[:N]` for top-N analysis.

Key Implementation Considerations

- **Data consistency**
  Product names and categories were kept consistent across all data structures to ensure reliable dictionary lookups and set operations.
- **Robust initialization**
  Customer totals were explicitly initialized so all customers are included in summaries, even if they have minimal or no transactions.
- **Efficiency and clarity**
  Where possible, the script favors single-pass aggregation for efficiency while maintaining clear, well-structured logic.
- **Set-based reasoning**
  Sets were used intentionally to model uniqueness and category relationships, enabling clean solutions for multi-category and cross-category customer queries.

Conclusion

This project demonstrates how fundamental Python constructs—lists, tuples, dictionaries, sets, loops, and conditionals—can be combined to answer practical business questions. By modeling transactions accurately, aggregating data efficiently, and applying set-based logic, the solution produces meaningful customer and product insights. The final implementation provides a strong foundation for more advanced data analysis workflows using tools such as pandas or SQL.