|  |  |
| --- | --- |
| Blue letters on a white background  Description automatically generated | Text  Description automatically generated with medium confidence |

# Student Assessment Submission and Declaration

When submitting evidence for assessment, you must sign a declaration confirming that the work is your own.

|  |  |  |  |
| --- | --- | --- | --- |
| Student name: | Rajakokila Muralitharan | ESL ID No | 122854 |
| BSU ID No | 664333 |
| Submission date: | | | **05.08.2025** |
|  | | |  |
| Programme: | Software foundation | | |
| Module name and code: | **CPUF001-20** | | |
| Title: | SOFTWARE FOUNDATION T-SHIRT SALE ANALYS | | |
| Assessor name: | AYODEJI ARIMIYAU TORIOLA | | |

**Plagiarism**

Plagiarism is a form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student declaration**  I certify that the assignment submission is entirely my own work. I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
| Student signature: | Rajakokila Muralitharan | Date: | 05.08.2025 |

Contents

[Student Assessment Submission and Declaration 1](#_Toc205234977)

[System for T-Shirt Sales Analysis: Detailed Code Description 3](#_Toc205234978)

[Overview of the System Architecture 3](#_Toc205234979)

[Module Dissection 4](#_Toc205234980)

[1. The primary controller, tshirt\_main.py 4](#_Toc205234981)

[2. T-shirt\_data\_reader.py, the Data Reader 5](#_Toc205234982)

[3. Core Analyser is another name for tshirt\_analyzer.py. 5](#_Toc205234983)

[4. Generation of Insight 6](#_Toc205234984)

[5. The Report Writer, tshirt\_report\_writer.py 6](#_Toc205234985)

[6. Automation of Batch Files (run\_tshirt\_analyzer.bat) 7](#_Toc205234986)

[Flow of Data Processing 7](#_Toc205234987)

[Appendix: 8](#_Toc205234988)

[1.Tshirt\_main 8](#_Toc205234989)

[2. T-shirt\_data\_reader.py 9](#_Toc205234990)

[3. Tshirt\_analyzer.py 10](#_Toc205234991)

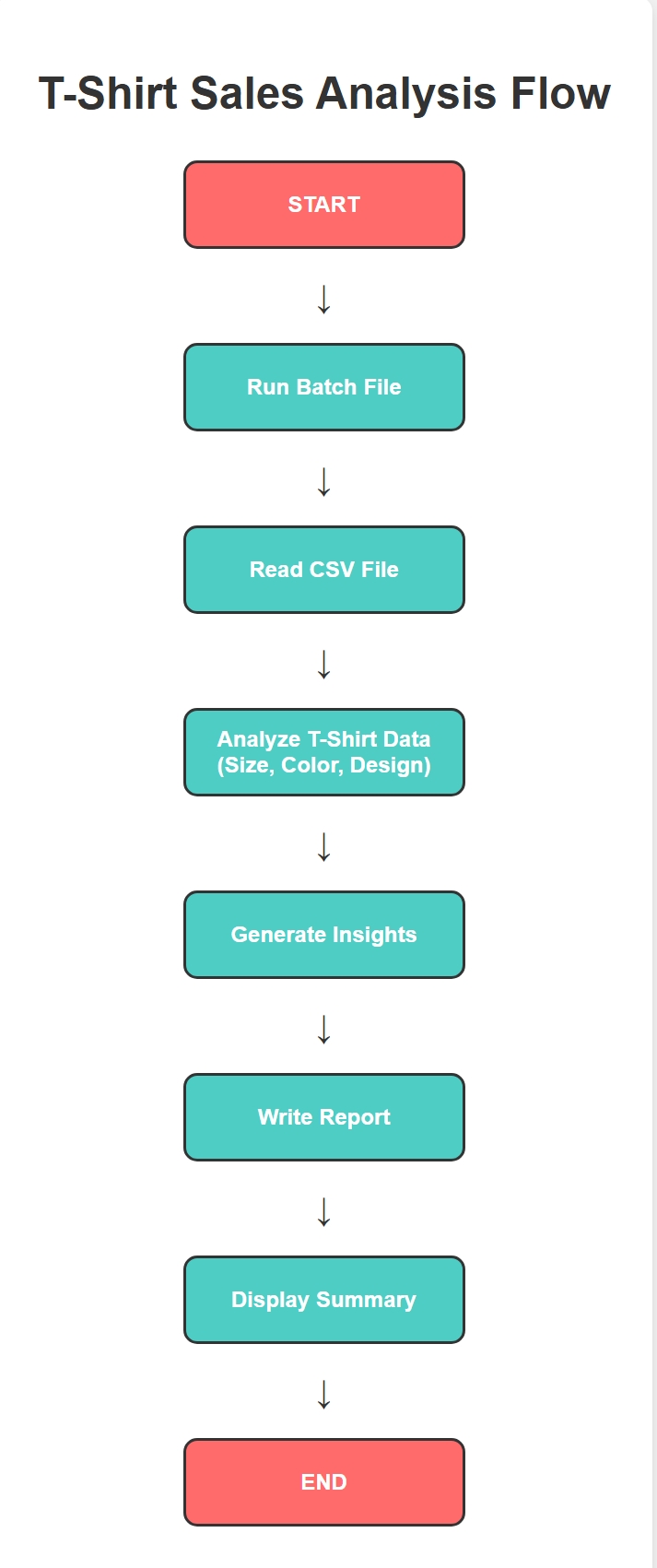
[4. Tshirt\_report\_writer.py 12](#_Toc205234992)

# System for T-Shirt Sales Analysis: Detailed Code Description

## Overview of the System Architecture

Four essential Python modules and a Windows batch file for simple execution make up the modular architecture used to build the t-shirt sales analysis system. Each module in this design manages a distinct component of the data processing pipeline, adhering to the single responsibility concept. The system uses only the built-in Python libraries and is lightweight with no external dependencies.

**Here is a flow chart:**

****

# Module Dissection

## 1. The primary controller, tshirt\_main.py

The entire analytic process is orchestrated by the main module. It manages the validation of command-line arguments, making sure that users supply the input and output file directories. By verifying the file's existence before processing and offering explicit usage instructions in the event that parameters are missing, the module demonstrates reliable error handling.

Data reading → analysis → insight generation → report writing → console summary is the linear pipeline that the execution flow follows. This method guarantees that data moves methodically through every processing step. Additionally, the module informs the user about the current operation and shows important data, such as the total number of records processed, at each stage.

The system gently shuts down when files are missing or when no data is detected in the input file, demonstrating its thorough error handling. In addition to preventing downstream mistakes, this gives customers unambiguous feedback about what went wrong.

## 2. T-shirt\_data\_reader.py, the Data Reader

Initial data validation and CSV file parsing are handled by the data reader module. Each row is automatically transformed into a dictionary with column headings as keys using Python's built-in csv. DictReader. In later processing stages, this method makes the data structure simple to use and intuitive.

Error handling for common file problems, such as malformed CSV structures, missing files, or encoding issues, is included in the module. When CSV files don't match expected formats, it helps debug by showing users the accessible columns. The main module can gracefully manage failures because the function returns None on errors.

To handle international characters in t-shirt descriptions, colours, or design names, the encoding is specifically set to UTF-8. This guarantees that the system is compatible with a variety of product catalogues, some of which may contain non-ASCII characters.

## 3. Core Analyser is another name for tshirt\_analyzer.py.

The business logic for handling T-shirt sales data is contained in the analyser module. It eliminates the requirement for manual key existence checks by automatically initialising counters using Python's defaultdict from the collection’s module. As a result, the code is more effective and cleaner.

The study concurrently monitors several dimensions:

* **Size metrics: Quantity and revenue by size of t-shirts (S, M, L, XL, etc.)**
* **Colour performance: Sales information categorised by preferred colour**
* **Design analysis: Revenue distribution according to design types**
* **Financial indicators: average price per unit, quantity sold, and total revenue**

The module logs warnings and skips invalid rows to gently resolve data quality issues. It gives default values for missing data and translates string values to the proper numeric types (float for amounts, int for quantities). Even with incomplete data, the analysis is certain to proceed because of this strong methodology.

Finding top achievers is made simple by the sorting feature, which arranges items by revenue in descending order. In addition to absolute values, the module computes percentage distributions and provides relative performance indicators.

## 4. Generation of Insight

Advanced analytics are carried out on the processed data by the generate\_insights function. Key business metrics are identified, such as the most popular size by amount sold versus the most lucrative size by revenue. For inventory management, this distinction is essential since low revenue, high-quantity items may be a sign of pricing problems.

By calculating size distribution percentages, the function aids businesses in comprehending the makeup of their clientele. Planning an inventory and creating a marketing strategy both benefit from this information. Top-performing colours and designs are also identified by the insight’s module, which offers useful information for marketing and product development.

## 5. The Report Writer, tshirt\_report\_writer.py

The report writer creates thorough business reports in an organised, easily understandable style. For every dimension that is examined (size, colour, and design), it generates comprehensive breakdowns that include both absolute values and percentage contributions. Stakeholders can better grasp the size and relative significance of various parts thanks to this dual presentation.

Key financial figures, in-depth analyses with rankings, and practical business recommendations are all included in the report's summary section. The proposals, which span marketing strategy, product bundling, and inventory management, are well suited to the retail T-shirt industry.

Without requiring users to open the complete report file, the console summary function shows top performers and offers instant feedback while it is running. When the system is being used in automated environments or for rapid analysis, this functionality is quite helpful.

## 6. Automation of Batch Files (run\_tshirt\_analyzer.bat)

For non-technical users, the Windows batch file offers an intuitive user interface. When Python is unavailable, it prevents ambiguous error messages by including Python installation verification. Both interactive mode (which prompts for file names) and command-line inputs for automatic execution are supported by the script.

Comprehensive error handling is included in the batch file, which verifies the file's existence before starting any analysis. It pauses execution to allow users to view feedback before the window closes and displays unambiguous error messages. Users can view findings instantly without having to open the output file by using the optional report preview feature.

# Flow of Data Processing

The system uses a transparent pipeline to process data:

1. **Verify the existence of files and the accuracy of arguments using input validation.**
2. **Parse CSV data into structured dictionaries for data loading.**
3. **Data Cleaning: Address incorrect entries and missing values.**
4. **Analysis: Determine metrics in a variety of dimensions.**
5. **Finding patterns and business intelligence is the first step in creating insights.**
6. **Report Generation: Produce output that is formatted and includes suggestions.**
7. **User input: save the comprehensive report and show the summary**

**Important attributes and advantages**

**Modularity:** The system is tested and maintainable since each component has a single duty. It is possible to change one component without impacting the others.

**Error Resilience:** Thorough error handling makes sure the system keeps running even when the data isn't flawless, and it logs into problems for the user to see.

**Business Intelligence:** The system offers actionable information unique to t-shirt retail, such as size distribution analysis and colour preference trends, going beyond simple reporting.

**User Experience:** The system may be used by non-programmers thanks to the batch file interface, and automation and system integration are made possible via command-line support.

**Scalability:** Additional metrics or new kinds of new product can be easily added thanks to the modular design**.** It is possible to add new analysis dimensions without changing the current code.

**No Dependencies:** Simple deployment without complicated dependency management is ensured by using only the Python standard library.

This design produces a strong, intuitive t-shirt sales analysis system that offers comprehensive reporting and strategic business insights, which makes it useful for inventory control and retail decision-making

# Appendix:

## 1.Tshirt\_main

1. """

2. Main module for t-shirt sales analysis application.

3. No external dependencies required.

4. """

5. import sys

6. import os

7. from tshirt\_data\_reader import read\_data

8. from tshirt\_analyzer import analyze\_tshirts, generate\_insights

9. from tshirt\_report\_writer import write\_report, print\_summary

10.

11. def main():

12. """Main function to run the t-shirt sales analysis."""

13. # Check command line arguments

14. if len(sys.argv) != 3:

15. print("Usage: python tshirt\_main.py <input\_file> <output\_file>")

16. print("Example: python tshirt\_main.py tshirt\_sales.csv tshirt\_report.txt")

17. sys.exit(1)

18.

19. input\_file = sys.argv[1]

20. output\_file = sys.argv[2]

21.

22. # Verify input file exists

23. if not os.path.exists(input\_file):

24. print(f"Error: Input file '{input\_file}' not found.")

25. sys.exit(1)

26.

27. # Read and process data

28. print(f"Reading t-shirt sales data from {input\_file}...")

29. data = read\_data(input\_file)

30.

31. if not data:

32. print("No data found in input file.")

33. sys.exit(1)

34.

35. print(f"\nTotal t-shirt sales records: {len(data)}")

36.

37. # Analyze t-shirts

38. print("\nAnalyzing t-shirt performance...")

39. analysis\_results = analyze\_tshirts(data)

40.

41. # Generate business insights

42. insights = generate\_insights(analysis\_results)

43.

44. # Write report

45. print(f"\nWriting t-shirt report to {output\_file}...")

46. write\_report(analysis\_results, insights, output\_file)

47.

48. # Print summary to console

49. print\_summary(analysis\_results)

50.

51. print("\nT-shirt analysis complete!")

52. print(f"Report generated: {output\_file}")

53.

54. if \_\_name\_\_ == "\_\_main\_\_":

55. main()

56.

## 2. T-shirt\_data\_reader.py

1. """

2. Module for reading t-shirt sales data from CSV files.

3. No external dependencies required.

4. """

5. import csv

6.

7. def read\_data(input\_file):

8.     """

9.     Read t-shirt sales data from CSV file and return as a list of dictionaries.

10.

11.     Args:

12.         input\_file (str): Path to the CSV file

13.

14.     Returns:

15.         list: List of dictionaries containing the t-shirt sales data, or None if error

16.     """

17.     try:

18.         with open(input\_file, 'r', encoding='utf-8') as file:

19.             reader = csv.DictReader(file)

20.             print("Available columns:", reader.fieldnames)

21.             data = list(reader)

22.             if not data:

23.                 return None

24.             return data

25.     except Exception as e:

26.         print(f"Error reading file: {str(e)}")

27.         return None

28.

## 3. Tshirt\_analyzer.py

1. """

2. Module for analyzing t-shirt sales data.

3. No external dependencies required.

4. """

5. from collections import defaultdict

6.

7. def analyze\_tshirts(data):

8.     """

9.     Analyze t-shirt sales data with t-shirt specific metrics.

10.

11.     Args:

12.         data (list): List of dictionaries containing t-shirt sales data

13.

14.     Returns:

15.         dict: Analysis results containing various metrics

16.     """

17.     # Initialize tracking dictionaries

18.     size\_sales = defaultdict(float)

19.     color\_sales = defaultdict(float)

20.     design\_sales = defaultdict(float)

21.     size\_quantity = defaultdict(int)

22.     color\_quantity = defaultdict(int)

23.

24.     total\_revenue = 0

25.     total\_quantity = 0

26.

27.     # Process each row

28.     for row in data:

29.         try:

30.             # Convert amount and quantity, handling empty values

31.             amount = float(row.get('amount', 0)) if row.get('amount') else 0

32.             quantity = int(row.get('quantity', 1)) if row.get('quantity') else 1

33.

34.             # Get t-shirt attributes

35.             size = row.get('size', 'Unknown').upper()

36.             color = row.get('color', 'Unknown').title()

37.             design = row.get('design', 'Unknown')

38.

39.             # Update totals

40.             total\_revenue += amount

41.             total\_quantity += quantity

42.

43.             # Update size metrics

44.             size\_sales[size] += amount

45.             size\_quantity[size] += quantity

46.

47.             # Update color metrics

48.             color\_sales[color] += amount

49.             color\_quantity[color] += quantity

50.

51.             # Update design metrics

52.             design\_sales[design] += amount

53.

54.         except (ValueError, KeyError) as e:

55.             print(f"Warning: Skipping invalid row: {row} - Error: {e}")

56.             continue

57.

58.     # Sort results

59.     sorted\_sizes = sorted(size\_sales.items(), key=lambda x: x[1], reverse=True)

60.     sorted\_colors = sorted(color\_sales.items(), key=lambda x: x[1], reverse=True)

61.     sorted\_designs = sorted(design\_sales.items(), key=lambda x: x[1], reverse=True)

62.

63.     # Calculate average price per unit

64.     avg\_price = total\_revenue / total\_quantity if total\_quantity > 0 else 0

65.

66.     return {

67.         'total\_revenue': total\_revenue,

68.         'total\_quantity': total\_quantity,

69.         'avg\_price': avg\_price,

70.         'sorted\_sizes': sorted\_sizes,

71.         'sorted\_colors': sorted\_colors,

72.         'sorted\_designs': sorted\_designs,

73.         'size\_quantity': dict(size\_quantity),

74.         'color\_quantity': dict(color\_quantity)

75.     }

76.

77. def generate\_insights(analysis\_results):

78.     """

79.     Generate t-shirt specific business insights.

80.

81.     Args:

82.         analysis\_results (dict): Results from analyze\_tshirts function

83.

84.     Returns:

85.         dict: Dictionary containing business insights

86.     """

87.     insights = {}

88.

89.     # Most popular size by revenue and quantity

90.     if analysis\_results['sorted\_sizes']:

91.         top\_size\_revenue = analysis\_results['sorted\_sizes'][0]

92.         insights['top\_size\_revenue'] = top\_size\_revenue

93.

94.         # Find most popular size by quantity

95.         size\_qty = analysis\_results['size\_quantity']

96.         top\_size\_qty = max(size\_qty.items(), key=lambda x: x[1])

97.         insights['top\_size\_quantity'] = top\_size\_qty

98.

99.     # Most popular color

100.     if analysis\_results['sorted\_colors']:

101.         top\_color = analysis\_results['sorted\_colors'][0]

102.         insights['top\_color'] = top\_color

103.

104.     # Best performing design

105.     if analysis\_results['sorted\_designs']:

106.         top\_design = analysis\_results['sorted\_designs'][0]

107.         insights['top\_design'] = top\_design

108.

109.     # Size distribution analysis

110.     total\_revenue = analysis\_results['total\_revenue']

111.     if total\_revenue > 0:

112.         size\_percentages = []

113.         for size, revenue in analysis\_results['sorted\_sizes']:

114.             percentage = (revenue / total\_revenue) \* 100

115.             size\_percentages.append((size, percentage))

116.         insights['size\_distribution'] = size\_percentages

117.

118.     return insights

119.

## 4. Tshirt\_report\_writer.py

1. """

2. Module for generating t-shirt sales reports.

3. No external dependencies required.

4. """

5.

6. def write\_report(analysis\_results, insights, output\_file):

7.     """

8.     Write t-shirt analysis report to file.

9.

10.     Args:

11.         analysis\_results (dict): Results from t-shirt analysis

12.         insights (dict): Dictionary containing business insights

13.         output\_file (str): Path to the output report file

14.     """

15.     try:

16.         with open(output\_file, 'w', encoding='utf-8') as file:

17.             # Write header

18.             file.write("T-Shirt Sales Analysis Report\n")

19.             file.write("============================\n\n")

20.

21.             # Write summary

22.             file.write(f"Total Revenue: ${analysis\_results['total\_revenue']:.2f}\n")

23.             file.write(f"Total T-Shirts Sold: {analysis\_results['total\_quantity']}\n")

24.             file.write(f"Average Price per T-Shirt: ${analysis\_results['avg\_price']:.2f}\n\n")

25.

26.             # Write size analysis

27.             file.write("T-Shirt Sizes by Revenue\n")

28.             file.write("----------------------\n")

29.             for i, (size, revenue) in enumerate(analysis\_results['sorted\_sizes'], 1):

30.                 percentage = (revenue / analysis\_results['total\_revenue']) \* 100

31.                 quantity = analysis\_results['size\_quantity'].get(size, 0)

32.                 file.write(f"{i}. Size {size}: ${revenue:.2f} ({percentage:.1f}%) - {quantity} units\n")

33.

34.             file.write("\n")

35.

36.             # Write color analysis

37.             file.write("T-Shirt Colors by Revenue\n")

38.             file.write("-----------------------\n")

39.             for i, (color, revenue) in enumerate(analysis\_results['sorted\_colors'], 1):

40.                 percentage = (revenue / analysis\_results['total\_revenue']) \* 100

41.                 quantity = analysis\_results['color\_quantity'].get(color, 0)

42.                 file.write(f"{i}. {color}: ${revenue:.2f} ({percentage:.1f}%) - {quantity} units\n")

43.

44.             file.write("\n")

45.

46.             # Write design analysis

47.             file.write("T-Shirt Designs by Revenue\n")

48.             file.write("------------------------\n")

49.             for i, (design, revenue) in enumerate(analysis\_results['sorted\_designs'], 1):

50.                 percentage = (revenue / analysis\_results['total\_revenue']) \* 100

51.                 file.write(f"{i}. {design}: ${revenue:.2f} ({percentage:.1f}%)\n")

52.

53.             file.write("\n")

54.

55.             # Write insights

56.             file.write("Key Insights\n")

57.             file.write("-----------\n")

58.

59.             if 'top\_size\_revenue' in insights:

60.                 size, revenue = insights['top\_size\_revenue']

61.                 file.write(f"Best selling size by revenue: {size} (${revenue:.2f})\n")

62.

63.             if 'top\_size\_quantity' in insights:

64.                 size, quantity = insights['top\_size\_quantity']

65.                 file.write(f"Most popular size by quantity: {size} ({quantity} units)\n")

66.

67.             if 'top\_color' in insights:

68.                 color, revenue = insights['top\_color']

69.                 file.write(f"Best selling color: {color} (${revenue:.2f})\n")

70.

71.             if 'top\_design' in insights:

72.                 design, revenue = insights['top\_design']

73.                 file.write(f"Top performing design: {design} (${revenue:.2f})\n")

74.

75.             file.write("\n")

76.

77.             # Write recommendations

78.             file.write("Recommendations\n")

79.             file.write("--------------\n")

80.             file.write("1. Stock more inventory of top-selling sizes and colors\n")

81.             file.write("2. Consider promoting less popular designs with successful colors/sizes\n")

82.             file.write("3. Analyze seasonal trends for color preferences\n")

83.             file.write("4. Bundle slow-moving designs with popular ones\n")

84.             file.write("5. Focus marketing on best-performing size/color combinations\n")

85.

86.     except Exception as e:

87.         print(f"Error writing report: {str(e)}")

88.

89. def print\_summary(analysis\_results, count=3):

90.     """

91.     Print a summary of top t-shirt metrics to the console.

92.

93.     Args:

94.         analysis\_results (dict): Results from t-shirt analysis

95.         count (int): Number of top items to display

96.     """

97.     print(f"\nT-Shirt Sales Summary:")

98.     print(f"Total Revenue: ${analysis\_results['total\_revenue']:.2f}")

99.     print(f"Total Units Sold: {analysis\_results['total\_quantity']}")

100.     print(f"Average Price: ${analysis\_results['avg\_price']:.2f}")

101.

102.     print(f"\nTop {count} Sizes by Revenue:")

103.     for i, (size, revenue) in enumerate(analysis\_results['sorted\_sizes'][:count], 1):

104.         percentage = (revenue / analysis\_results['total\_revenue']) \* 100

105.         print(f"{i}. Size {size}: ${revenue:.2f} ({percentage:.1f}%)")

106.

107.     print(f"\nTop {count} Colors by Revenue:")

108.     for i, (color, revenue) in enumerate(analysis\_results['sorted\_colors'][:count], 1):

109.         percentage = (revenue / analysis\_results['total\_revenue']) \* 100

110.         print(f"{i}. {color}: ${revenue:.2f} ({percentage:.1f}%)")

111.

112.

113.

**Module for analyzing t-shirt sales data.**

**No external dependencies required.**

**"""**

**from collections import defaultdict**

**def analyze\_tshirts(data):**

**"""**

**Analyze t-shirt sales data with t-shirt specific metrics.**

**Args:**

**data (list): List of dictionaries containing t-shirt sales data**

**Returns:**

**dict: Analysis results containing various metrics**

**"""**

**# Initialize tracking dictionaries**

**size\_sales = defaultdict(float)**

**color\_sales = defaultdict(float)**

**design\_sales = defaultdict(float)**

**size\_quantity = defaultdict(int)**

**color\_quantity = defaultdict(int)**

**total\_revenue = 0**

**total\_quantity = 0**

**# Process each row**

**for row in data:**

**try:**

**# Convert amount and quantity, handling empty values**

**amount = float(row.get('amount', 0)) if row.get('amount') else 0**

**quantity = int(row.get('quantity', 1)) if row.get('quantity') else 1**

**# Get t-shirt attributes**

**size = row.get('size', 'Unknown').upper()**

**color = row.get('color', 'Unknown').title()**

**design = row.get('design', 'Unknown')**

**# Update totals**

**total\_revenue += amount**

**total\_quantity += quantity**

**# Update size metrics**

**size\_sales[size] += amount**

**size\_quantity[size] += quantity**

**# Update color metrics**

**color\_sales[color] += amount**

**color\_quantity[color] += quantity**

**# Update design metrics**

**design\_sales[design] += amount**

**except (ValueError, KeyError) as e:**

**print(f"Warning: Skipping invalid row: {row} - Error: {e}")**

**continue**

**# Sort results**

**sorted\_sizes = sorted(size\_sales.items(), key=lambda x: x[1], reverse=True)**

**sorted\_colors = sorted(color\_sales.items(), key=lambda x: x[1], reverse=True)**

**sorted\_designs = sorted(design\_sales.items(), key=lambda x: x[1], reverse=True)**

**# Calculate average price per unit**

**avg\_price = total\_revenue / total\_quantity if total\_quantity > 0 else 0**

**return {**

**'total\_revenue': total\_revenue,**

**'total\_quantity': total\_quantity,**

**'avg\_price': avg\_price,**

**'sorted\_sizes': sorted\_sizes,**

**'sorted\_colors': sorted\_colors,**

**'sorted\_designs': sorted\_designs,**

**'size\_quantity': dict(size\_quantity),**

**'color\_quantity': dict(color\_quantity)**

**}**

**def generate\_insights(analysis\_results):**

**"""**

**Generate t-shirt specific business insights.**

**Args:**

**analysis\_results (dict): Results from analyze\_tshirts function**

**Returns:**

**dict: Dictionary containing business insights**

**"""**

**insights = {}**

**# Most popular size by revenue and quantity**

**if analysis\_results['sorted\_sizes']:**

**top\_size\_revenue = analysis\_results['sorted\_sizes'][0]**

**insights['top\_size\_revenue'] = top\_size\_revenue**

**# Find most popular size by quantity**

**size\_qty = analysis\_results['size\_quantity']**

**top\_size\_qty = max(size\_qty.items(), key=lambda x: x[1])**

**insights['top\_size\_quantity'] = top\_size\_qty**

**# Most popular color**

**if analysis\_results['sorted\_colors']:**

**top\_color = analysis\_results['sorted\_colors'][0]**

**insights['top\_color'] = top\_color**

**# Best performing design**

**if analysis\_results['sorted\_designs']:**

**top\_design = analysis\_results['sorted\_designs'][0]**

**insights['top\_design'] = top\_design**

**# Size distribution analysis**

**total\_revenue = analysis\_results['total\_revenue']**

**if total\_revenue > 0:**

**size\_percentages = []**

**for size, revenue in analysis\_results['sorted\_sizes']:**

**percentage = (revenue / total\_revenue) \* 100**

**size\_percentages.append((size, percentage))**

**insights['size\_distribution'] = size\_percentages**

**return insights**

**Module for reading t-shirt sales data from CSV files.**

**No external dependencies required.**

**"""**

**import csv**

**def read\_data(input\_file):**

**"""**

**Read t-shirt sales data from CSV file and return as a list of dictionaries.**

**Args:**

**input\_file (str): Path to the CSV file**

**Returns:**

**list: List of dictionaries containing the t-shirt sales data, or None if error**

**"""**

**try:**

**with open(input\_file, 'r', encoding='utf-8') as file:**

**reader = csv.DictReader(file)**

**print("Available columns:", reader.fieldnames)**

**data = list(reader)**

**if not data:**

**return None**

**return data**

**except Exception as e:**

**print(f"Error reading file: {str(e)}")**

**return None**

**Main module for t-shirt sales analysis application.**

**No external dependencies required.**

**"""**

**import sys**

**import os**

**from tshirt\_data\_reader import read\_data**

**from tshirt\_analyzer import analyze\_tshirts, generate\_insights**

**from tshirt\_report\_writer import write\_report, print\_summary**

**def main():**

**"""Main function to run the t-shirt sales analysis."""**

**# Check command line arguments**

**if len(sys.argv) != 3:**

**print("Usage: python tshirt\_main.py <input\_file> <output\_file>")**

**print("Example: python tshirt\_main.py tshirt\_sales.csv tshirt\_report.txt")**

**sys.exit(1)**

**input\_file = sys.argv[1]**

**output\_file = sys.argv[2]**

**# Verify input file exists**

**if not os.path.exists(input\_file):**

**print(f"Error: Input file '{input\_file}' not found.")**

**sys.exit(1)**

**# Read and process data**

**print(f"Reading t-shirt sales data from {input\_file}...")**

**data = read\_data(input\_file)**

**if not data:**

**print("No data found in input file.")**

**sys.exit(1)**

**print(f"\nTotal t-shirt sales records: {len(data)}")**

**# Analyze t-shirts**

**print("\nAnalyzing t-shirt performance...")**

**analysis\_results = analyze\_tshirts(data)**

**# Generate business insights**

**insights = generate\_insights(analysis\_results)**

**# Write report**

**print(f"\nWriting t-shirt report to {output\_file}...")**

**write\_report(analysis\_results, insights, output\_file)**

**# Print summary to console**

**print\_summary(analysis\_results)**

**print("\nT-shirt analysis complete!")**

**print(f"Report generated: {output\_file}")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**"""**

**Module for generating t-shirt sales reports.**

**No external dependencies required.**

**"""**

**def write\_report(analysis\_results, insights, output\_file):**

**"""**

**Write t-shirt analysis report to file.**

**Args:**

**analysis\_results (dict): Results from t-shirt analysis**

**insights (dict): Dictionary containing business insights**

**output\_file (str): Path to the output report file**

**"""**

**try:**

**with open(output\_file, 'w', encoding='utf-8') as file:**

**# Write header**

**file.write("T-Shirt Sales Analysis Report\n")**

**file.write("============================\n\n")**

**# Write summary**

**file.write(f"Total Revenue: ${analysis\_results['total\_revenue']:.2f}\n")**

**file.write(f"Total T-Shirts Sold: {analysis\_results['total\_quantity']}\n")**

**file.write(f"Average Price per T-Shirt: ${analysis\_results['avg\_price']:.2f}\n\n")**

**# Write size analysis**

**file.write("T-Shirt Sizes by Revenue\n")**

**file.write("----------------------\n")**

**for i, (size, revenue) in enumerate(analysis\_results['sorted\_sizes'], 1):**

**percentage = (revenue / analysis\_results['total\_revenue']) \* 100**

**quantity = analysis\_results['size\_quantity'].get(size, 0)**

**file.write(f"{i}. Size {size}: ${revenue:.2f} ({percentage:.1f}%) - {quantity} units\n")**

**file.write("\n")**

**# Write color analysis**

**file.write("T-Shirt Colors by Revenue\n")**

**file.write("-----------------------\n")**

**for i, (color, revenue) in enumerate(analysis\_results['sorted\_colors'], 1):**

**percentage = (revenue / analysis\_results['total\_revenue']) \* 100**

**quantity = analysis\_results['color\_quantity'].get(color, 0)**

**file.write(f"{i}. {color}: ${revenue:.2f} ({percentage:.1f}%) - {quantity} units\n")**

**file.write("\n")**

**# Write design analysis**

**file.write("T-Shirt Designs by Revenue\n")**

**file.write("------------------------\n")**

**for i, (design, revenue) in enumerate(analysis\_results['sorted\_designs'], 1):**

**percentage = (revenue / analysis\_results['total\_revenue']) \* 100**

**file.write(f"{i}. {design}: ${revenue:.2f} ({percentage:.1f}%)\n")**

**file.write("\n")**

**# Write insights**

**file.write("Key Insights\n")**

**file.write("-----------\n")**

**if 'top\_size\_revenue' in insights:**

**size, revenue = insights['top\_size\_revenue']**

**file.write(f"Best selling size by revenue: {size} (${revenue:.2f})\n")**

**if 'top\_size\_quantity' in insights:**

**size, quantity = insights['top\_size\_quantity']**

**file.write(f"Most popular size by quantity: {size} ({quantity} units)\n")**

**if 'top\_color' in insights:**

**color, revenue = insights['top\_color']**

**file.write(f"Best selling color: {color} (${revenue:.2f})\n")**

**if 'top\_design' in insights:**

**design, revenue = insights['top\_design']**

**file.write(f"Top performing design: {design} (${revenue:.2f})\n")**

**file.write("\n")**

**# Write recommendations**

**file.write("Recommendations\n")**

**file.write("--------------\n")**

**file.write("1. Stock more inventory of top-selling sizes and colors\n")**

**file.write("2. Consider promoting less popular designs with successful colors/sizes\n")**

**file.write("3. Analyze seasonal trends for color preferences\n")**

**file.write("4. Bundle slow-moving designs with popular ones\n")**

**file.write("5. Focus marketing on best-performing size/color combinations\n")**

**except Exception as e:**

**print(f"Error writing report: {str(e)}")**

**def print\_summary(analysis\_results, count=3):**

**"""**

**Print a summary of top t-shirt metrics to the console.**

**Args:**

**analysis\_results (dict): Results from t-shirt analysis**

**count (int): Number of top items to display**

**"""**

**print(f"\nT-Shirt Sales Summary:")**

**print(f"Total Revenue: ${analysis\_results['total\_revenue']:.2f}")**

**print(f"Total Units Sold: {analysis\_results['total\_quantity']}")**

**print(f"Average Price: ${analysis\_results['avg\_price']:.2f}")**

**print(f"\nTop {count} Sizes by Revenue:")**

**for i, (size, revenue) in enumerate(analysis\_results['sorted\_sizes'][:count], 1):**

**percentage = (revenue / analysis\_results['total\_revenue']) \* 100**

**print(f"{i}. Size {size}: ${revenue:.2f} ({percentage:.1f}%)")**

**print(f"\nTop {count} Colors by Revenue:")**

**for i, (color, revenue) in enumerate(analysis\_results['sorted\_colors'][:count], 1):**

**percentage = (revenue / analysis\_results['total\_revenue']) \* 100**

**print(f"{i}. {color}: ${revenue:.2f} ({percentage:.1f}%)")**