Homework Grading Report

Student Name:	Deon Schoeman
Assignment:	2.2
Graded On:	September 22, 2025 at 05:12 PM
Final Score:	14.3 / 37.5 points (38.3%)

Score Summary

Overall Performance: Unsatisfactory (38.3%)

Component Scores:

- Working Directory: 2.0 points
- Package Loading: 4.0 points
- Data Import: 0.0 points
- Data Inspection: 4.0 points
- Reflection Questions: 4.3 points

Performance by Category

■ Excellent **Working Directory:** 2.0/2 points (100%)
■ Excellent **Package Loading:** 4.0/4 points (100%)
■ Needs Work **Data Import:** 0.0/5 points (0%)
■ Needs Work **Data Inspection:** 4.0/8 points (50%)
■ Needs Work **Reflection Questions:** 4.3/12.5 points (35%)

Detailed Analysis:

- Working Directory (2/2 points): Correctly used getwd() and showed output
- Package Loading (4/4 points): tidyverse loaded and executed successfully | readxl loaded and executed successfully
- ■ CSV Import (0/5 points): Missing sales_df <- read_csv() assignment
- ■ Excel Import (0/6 points): Missing ratings_df import | Missing comments_df import
- Data Inspection (4.0/8 points): head() used and executed | str() missing | summary() used and executed | sales_df not analyzed | ratings_df not analyzed | comments_df not analyzed | ■ Make sure to run ALL inspection functions (head, str, summary)
- Reflection Questions (4.3/12.5 points): Data Types Analysis (3.0/4 points) Good start - you mentioned data types, but try to discuss both Date and Amount columns | Good thinking about appropriateness - try connecting this to business needs | Good detail in your response

Data types matter more than you might think. If your dates are stored as text ("2023-01-15"), you can't calculate time differences or trends. If amounts have dollar signs ("$1,234.56"), you can't do math with them. When I see dates stored properly as date objects, I know you can calculate things like "days between orders" or "monthly sales patterns." When amounts are numeric (1234.56), you can sum, average, and analyze them. This isn't just technical nitpicking - it's about what analysis you can actually do with your data. Check this first, always. It'll save you headaches later. Data Quality Assessment (0.0/4 points) ■ Look at your data outputs - do you see any missing values (NA's) or unusual patterns? | ■ Think about this: how would missing data or errors affect your business conclusions?
Look for problems that will mess up your analysis. Missing values can throw off your totals. Inconsistent formatting (like "North" vs "NORTH" vs "north") will split your data when you try to group it. Watch for things that don't make business sense - negative sales amounts, future dates, or someone buying 999,999 keyboards (probably a data entry error). I also want to see you think about impact. If 5% of values are missing, that's different from 50% missing. If you have weird outliers, will they skew your averages? This isn't busy work - bad data leads to bad decisions. Spend time here and your analysis will be much more reliable. Analysis Readiness (1.4/4.5 points) You mentioned the datasets - now compare which is most ready for analysis | ■ Think about what you'd need to do to make the messiest dataset analysis-ready | You answered thoughtfully - could expand a bit more
Compare the datasets and tell me which one you'd start analyzing first. Think practically - which has fewer missing values? Which has cleaner, more consistent formatting? Which one can answer your most important business questions? For example, if your sales data is mostly complete but your feedback data has lots of gaps and messy text, you'd probably start with sales data to get quick insights, then clean up the feedback data later. In real work, you rarely get perfect data. You have to prioritize where to spend your time. Show me you can think strategically about this - it's a key skill. ■ Overall Reflection Quality: Needs Development The reflection questions are where you really develop your analytical thinking skills. Take more time with these - they're not just busy work! Look carefully at your data outputs, think about what you observe, and explain your reasoning. This kind of thinking is what separates good analysts from great ones.

Code Issues & Fixes

Issues Found:

- ERROR: Error: '/workspaces/assignment-2-DeonSchoeman/data/messy_sales_data.csv' does not exist.
- ERROR: Error: object 'messy_sales' not found
- ERROR: Error in parse(text = input): :1:1: unexpected '^' 1: ^
- ERROR: Error in parse(text = input): :2:2: unexpected ',' 1: "#### 2.2 Missing Value Treatment - Option A (Removal) 2: ", ^
- ERROR: Error: object 'sales_imputed' not found

Specific Code Solutions:

Data Import Fix

Common fixes: - Check file paths: make sure "data/" folder exists - Check sheet names: they're case-sensitive - Use forward slashes (/) not backslashes (\) in file paths **

```
# For CSV files
sales_df <- read_csv("data/sales_data.csv")
# For Excel files with multiple sheets
ratings_df <- read_excel("data/customer_feedback.xlsx", sheet = "ratings")
```

```
comments_df <- read_excel("data/customer_feedback.xlsx", sheet =
"customer_feedback")
```

Data Inspection Fix

Make sure to RUN each cell - you should see output below each command. **

```
# Run these for each dataset
head(sales_df) # First 6 rows
str(sales_df) # Structure and data types
summary(sales_df) # Statistical summary
# Do the same for other datasets
head(ratings_df)
str(ratings_df)
summary(ratings_df)
head(comments_df)
str(comments_df)
summary(comments_df)
```

Data Import Fix - CSV File Not Found

**

Working Directory Solutions:
Option 1: If your working directory is set to the data folder:
• Use: read_csv("sales_data.csv") - just the filename
Option 2: If your working directory is the project root:
• Use: read_csv("data/sales_data.csv") - include the data/ folder
Check your setup: Run getwd() to see where you are, then adjust your file paths accordingly.

```
# Check your working directory and file location
getwd() # See where R is currently looking
list.files() # See what files are in current directory
list.files("data/") # See what's in the data folder
# For CSV files, use:
sales_df <- read_csv("data/sales_data.csv")
# NOT: read_csv("../data/sales.csv") or read_csv("sales.csv")
# Make sure:
# 1. File is named exactly "sales_data.csv" (check spelling!)
# 2. File is in a "data" folder in your project
# 3. You're running from the correct working directory
```

Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```


Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Variable Fix - Object Not Found

**

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Variable Fix - Object Not Found

Study Tip: Go back through the lecture notebook and run the examples yourself. Practice is how you learn R. Come to office hours if you need help. We can work through any concepts that aren't clicking.

```
# This error means you're using a variable before creating it
# Common causes:
# 1. Typo in variable name (check spelling!)
# 2. Didn't run the cell that creates the variable
# 3. Variables are case-sensitive: sales_df ≠ Sales_df
# Solution: Run cells in order from top to bottom
```

Reflection Questions Feedback

Data Types: 3.0/4 points (Good)

Data Quality: 0.0/4 points (Needs Improvement)

Analysis Readiness: 1.4/4.5 points (Needs Improvement)

Next Steps

■ Let's Regroup (14.3/37.5 points - 38.3%) This is challenging material, so don't worry if it feels overwhelming. Focus on the basics and ask questions when you're stuck. Here's what to focus on for next time: Data Import: Make sure all three datasets (sales_df, ratings_df, comments_df) load successfully. Pay attention to file paths and sheet names for the Excel file. Data Inspection: Run `head()`, `str()`, and `summary()` on each dataset. Make sure you can see the outputs - this tells you what your data actually looks like. Reflection Questions: Take more time with these. Look at your data outputs and explain what you see. These aren't just busy work - they help you think analytically. Code Execution: Fix any error messages before submitting. Red error text means something went wrong - don't ignore it. Data Import Fix: ```r # For CSV files sales_df <- read_csv("data/sales_data.csv") # For Excel files with multiple sheets ratings_df <- read_excel("data/customer_feedback.xlsx", sheet = "ratings") comments_df <- read_excel("data/customer_feedback.xlsx", sheet = "customer_feedback") ``` Common fixes: - Check file paths: make sure "data/" folder exists - Check sheet names: they're

case-sensitive - Use forward slashes (/) not backslashes (\) in file paths Data Inspection Fix: ```r # Run these for each dataset head(sales_df) # First 6 rows str(sales_df) # Structure and data types summary(sales_df) # Statistical summary # Do the same for other datasets head(ratings_df) str(ratings_df) summary(ratings_df) head(comments_df) str(comments_df) summary(comments_df) ``` Make sure to RUN each cell - you should see output below each command. Data Import Fix - CSV File Not Found: ```r # Check your working directory and file location getwd() # See where R is currently looking list.files() # See what files are in current directory list.files("data/") # See what's in the data folder # For CSV files, use: sales_df <- read_csv("data/sales_data.csv") # NOT: read_csv("../data/sales.csv") or read_csv("sales.csv") # Make sure: # 1. File is named exactly "sales_data.csv" (check spelling!) # 2. File is in a "data" folder in your project # 3. You're running from the correct working directory ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Variable Fix - Object Not Found: ```r # This error means you're using a variable before creating it # Common causes: # 1. Typo in variable name (check spelling!) # 2. Didn't run the cell that creates the variable # 3. Variables are case-sensitive: sales_df ≠ Sales_df # Solution: Run cells in order from top to bottom ``` Study Tip: Go back through the lecture notebook and run the examples yourself. Practice is how you learn R. Come to office hours if you need help. We can work through any concepts that aren't clicking.

Study Tips:

- Review the lecture notebook and practice running the examples yourself
- Make sure to execute all code cells and check for outputs
- Focus on understanding the fundamental concepts before moving to advanced topics