Task 2 Contd & Task3 Coding part

Data Cleaning Process

The cleaning process addressed several data quality issues across both datasets:

Work Order Data:

- Replaced '0' values in 'Model Year' with 'Unknown' to properly handle missing information
- Removed the 'Cause' column due to excessive missing values (>50%)
- Identified German text in the 'Correction' column (translation attempted but not completed due to technical constraints)

Repair Data:

- Removed the 'Coverage' column with 84% missing values
- Standardized numeric formats:
  - Converted all quantities and revenues to absolute values
  - Cleaned monetary fields by removing '$' symbols and rounding to 2 decimals
  - Converted Excel serial dates to proper datetime format
- Ensured consistent decimal precision (2 places) across all numeric columns

Key challenges included handling mixed data types in monetary fields and managing technical issues with language translation.

---

2. Data Integration Approach

Primary Key Selection:
The integration used a composite key consisting of:

1. Primary Key (e.g., "SO0005588-1")
2. Order No
3. Segment Number

Rationale for Join Type (Inner Join):

- Ensures only complete records with matching entries in both datasets are included
- Appropriate for this analysis as we need repair details correlated with work order information
- Eliminates records that couldn't be matched, ensuring data consistency
- Preserves the one-to-many relationship where one work order may have multiple repair line items

The merged dataset contains 500 records with 23 columns combining information from both sources.

---

3. Key Trends and Findings

3.1 Failure Pattern Analysis:

1. **Failure Frequency Heatmap** revealed:
    - "Leak" failures are most common across equipment types
    - "APPL" category shows more "Oil Loss" failures
    - "SPRAYS" equipment has more "Error Code" issues
    - *Implication:* Different preventive maintenance strategies needed for different equipment types
2. **Cost vs. Repair Time Analysis** showed:
    - "Broken" components incur highest costs ($4,000+ avg)
    - Undocumented failures ("Not Mentioned") consume disproportionate repair time
    - *Implication:* Improved failure documentation could reduce labor costs

3.2 Seasonal Trends:

- Clear peaks in service demand during spring (April-May) and late summer
- Cost spikes exceed order volume increases, indicating some months have particularly expensive repairs
- *Recommendation:* Schedule preventive maintenance before peak seasons and increase parts inventory

Profitability Insights:

- The average markup on parts is approximately **34% (Revenue vs. Cost)**
- Labor costs show strong correlation with part costs (r=0.72)
- Highest-margin repairs involve electronic components (sensors, control units)

---

4. Recommended Actions

1. **Preventive Maintenance:**
    - Focus on leak prevention for all equipment
    - Specialized checks for "APPL" oil systems and "SPRAYS" electronics
2. **Process Improvements:**
    - Standardize failure documentation to reduce diagnostic time
    - Implement seasonal staffing adjustments
3. **Inventory Management:**
    - Stock high-cost failure components before peak seasons
    - Increase inventory of commonly replaced electronic parts

Coding part

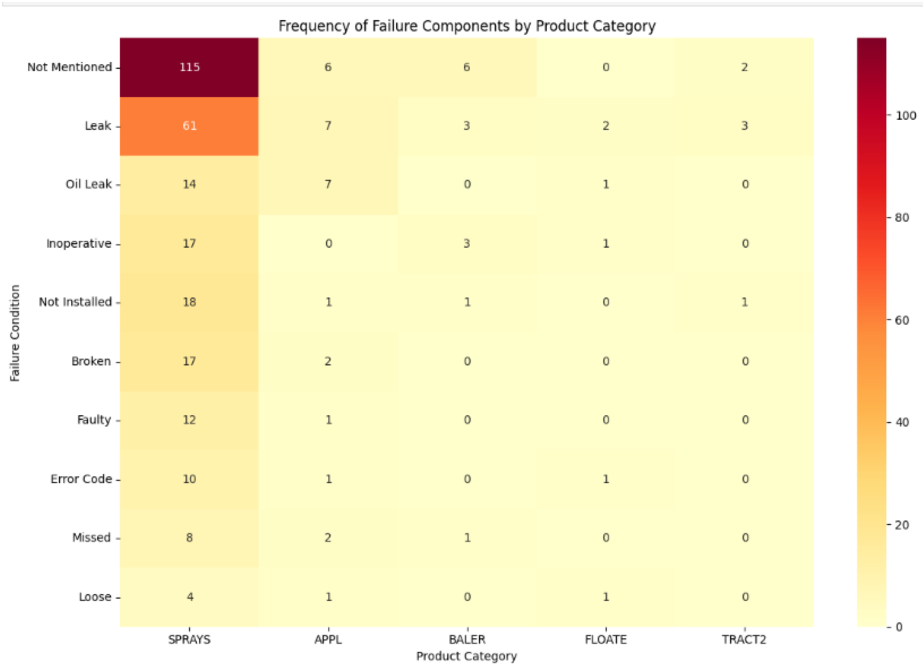
```
# 1. Primary Key Identification and Data Cleaning
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
excel_file = r"C:\Users\User\Downloads\Data_for_task_2.xlsx"
work_order_df = pd.read_excel(excel_file, sheet_name='Work Order Data')
repair_df = pd.read_excel(excel_file, sheet_name='Repair Data')

# Clean Work Order Data
work_order_df['Model Year'] = work_order_df['Model Year'].replace(0, 'Unknown')
work_order_df.drop(columns='Cause', inplace=True)

# Clean Repair Data
repair_df.drop(columns='Coverage', inplace=True)
repair_df['Qty'] = repair_df['Qty'].abs()
repair_df['Revenue'] = repair_df['Revenue'].abs().round(2)

# Clean Cost column
repair_df['Cost'] = (
    repair_df['Cost']
    .replace(r'[\$]', '', regex=True)
    .astype(float)
    .abs()
    .round(2)
repair_df.rename(columns={'Cost': 'Cost($)'}, inplace=True)

# Convert Invoice Date
```

```python
repair_df['Invoice Date'] = pd.to_datetime('1899-12-30') + pd.to_timedelta(repair_df['Invoice Date'],
unit='D')


# Clean Segment Total
repair_df['Segment Total $'] = (
    repair_df['Segment Total $']
    .astype(str)
    .str.replace('$', '', regex=False)
    .astype(float)
    .round(2))
repair_df.rename(columns={'Segment Total $': 'Segment Total($)'}, inplace=True)


# Round numeric columns
repair_df['Actual Hours'] = repair_df['Actual Hours'].round(2)


# Save cleaned data
output_file = "Cleaned_dataset_2.xlsx"
with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
    work_order_df.to_excel(writer, sheet_name="Work Order Cleaned", index=False)
    repair_df.to_excel(writer, sheet_name="Repair Cleaned", index=False)



# Merge datasets
merged_df = pd.merge(work_order_df, repair_df, on=('Primary Key', 'Order No', 'Segment Number'),
how='inner')
```

2. Exploratory Data Analysis

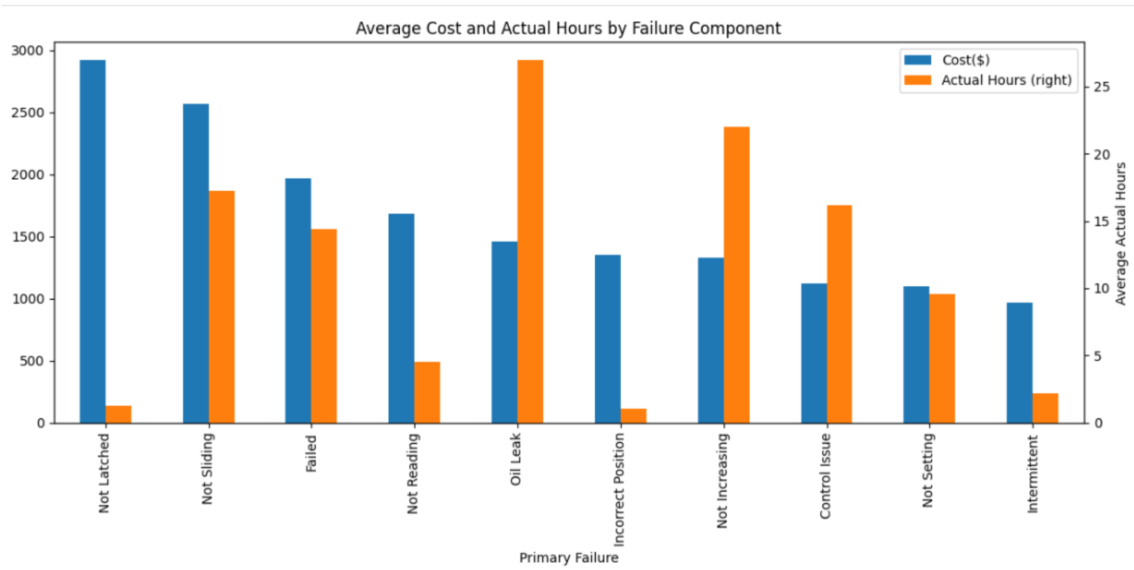
Visualization 1: Failure Component Frequency Heatmap

```
failure_components = merged_df['Failure Condition - Failure Component'].str.split(' - ').str[0].value_counts().head(10)

product_categories = merged_df['Product Category'].value_counts()

cross_tab = pd.crosstab(

    merged_df['Failure Condition - Failure Component'].str.split(' - ').str[0],

    merged_df['Product Category']

).loc[failure_components.index, product_categories.index]


plt.figure(figsize=(12, 8))

sns.heatmap(cross_tab, annot=True, fmt='d', cmap='YlOrRd')

plt.title('Frequency of Failure Components by Product Category')

plt.xlabel('Product Category')

plt.ylabel('Failure Condition')

plt.tight_layout()

plt.show()
```



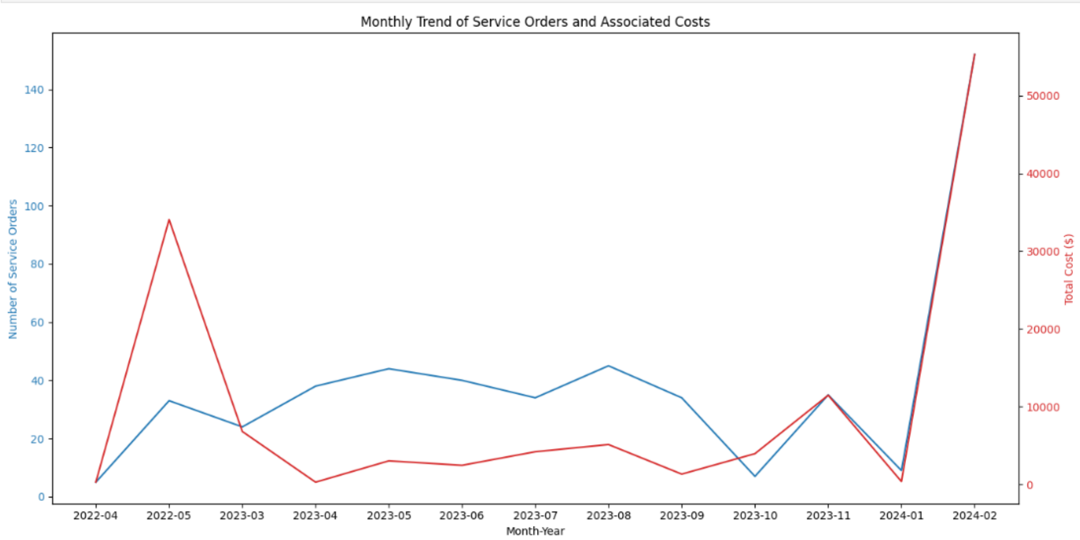
Visualization 2: Cost vs Actual Hours by Failure Component

```
merged_df['Primary Failure'] = merged_df['Failure Condition - Failure Component'].str.split(' - ').str[0]

failure_stats = merged_df.groupby('Primary Failure').agg({

    'Cost($)': 'mean',

    'Actual Hours': 'mean'

}).sort_values('Cost($)', ascending=False).head(10)


fig, ax = plt.subplots(figsize=(12, 6))

failure_stats.plot(kind='bar', ax=ax, secondary_y='Actual Hours')

plt.title('Average Cost and Actual Hours by Failure Component')

plt.ylabel('Average Cost ($)')

ax.right_ax.set_ylabel('Average Actual Hours')

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.show()
```



Visualization 3: Monthly Trend of Service Orders and Costs

```
merged_df['Order Date'] = pd.to_datetime(merged_df['Order Date'])
merged_df['Month-Year'] = merged_df['Order Date'].dt.to_period('M')
monthly_trends = merged_df.groupby('Month-Year').agg({
    'Primary Key': 'count',
    'Cost($)': 'sum',
    'Actual Hours': 'sum'
})

fig, ax1 = plt.subplots(figsize=(14, 7))
color = 'tab:blue'
ax1.set_xlabel('Month-Year')
ax1.set_ylabel('Number of Service Orders', color=color)
ax1.plot(monthly_trends.index.astype(str), monthly_trends['Primary Key'], color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Total Cost ($)', color=color)
ax2.plot(monthly_trends.index.astype(str), monthly_trends['Cost($)'], color=color)
ax2.tick_params(axis='y', labelcolor=color)

plt.title('Monthly Trend of Service Orders and Associated Costs')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Monthly Trend of Service Orders and Associated Costs
Number of Service Orders
Total Cost ($)
Month-Year
0
20
40
60
80
100
120
140
0
10000
20000
30000
40000
50000
2022-04
2022-05
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10
2023-11
2024-01
2024-02