In [8]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```python
df=pd.read_excel(r"C:\Users\chara\Downloads\DA -Task 2..xlsx")
```

In [10]:
```python
df.head()
```

Out[10]:

	VIN	TRANSACTION_ID	CORRECTION_VERBATIM	CUSTOMER_VERBATIM	REPAIR_DATE	CAUSAL_PART_NM	GLOBAL_LABOR_CODE_DESCRIPTION	PLATFORM	BODY_STYLE
0	3HCFDDE89SH220903	13021	REPLACED STEERING WHEEL NOW OKAY	STEERING WHEEL COMING APART	2024-01-02	WHEEL ASM-STRG *JET BLACK	Steering Wheel Replacement	Full-Size Trucks	Crew Cab
1	1HRFFEE8XSZ230636	13028	CHECKED - FOUND DTC'S U0229 - U1530 SET IN BCM...	CUSTOMER STATES HEATED STEERING WHEEL INOP	2024-01-03	MODULE ASM-STRG WHL HT CONT	Heated Steering Wheel Module Replacement	Full-Size Trucks	Crew Cab
2	1HYKSMRK6SZ000990	13035	APPROVED 4.9(OLH) FOR ADDED DIAGNOSTICS WITH T...	OWNER REPORTS: THE SUPER CRUISE BAR ON THE STE...	2024-01-04	WHEEL ASM-STRG *BACKEN BLACKK	Steering Wheel Replacement	BEV	4 Door Utility
3	3HCFDFEL3SH241701	13021	STEERING WHEEL REPLACEMENT	CUSTOMER STATES THE LETTERING AND FINISH ON TH...	2024-01-04	WHEEL ASM-STRG *JET BLACK	Steering Wheel Replacement	Full-Size Trucks	Crew Cab
4	1HRFFHEL1RZ181474	13021	REPLACED STEERING MESSAGE NO LONGER DISPLAYED	C/S: CUSTOMER STATES THE SERVICE DRIVER ASSIST...	2024-01-05	WHEEL ASM-STRG *JET BLACK	Steering Wheel Replacement	Full-Size Trucks	Crew Cab

5 rows × 52 columns

In [11]:
```
df.tail()
```

Out[11]:

	VIN	TRANSACTION_ID	CORRECTION_VERBATIM	CUSTOMER_VERBATIM	REPAIR_DATE	CAUSAL_PART_NM	GLOBAL_LABOR_CODE_DESCRIPTION	PLATFORM	BODY_STYLE
95	1HYKNHRS6MZ221833	13041	REPLACED STEERING WHEEL COMPLETEDLOP 0130 TIME .4	CUSTOMER STATES that the steering is very tigh...	2024-02-07	WHEEL ASM-STRG *BLACK	Steering Wheel Replacement	Global Crossover Vehicles	4 Door Utility
96	1HYKSSRL4SZ003381	13048	replace steering wheel	cs driver assistance warning light is coming o...	2024-02-07	WHEEL ASM-STRG *BACKEN BLACKK	Steering Wheel Replacement	BEV	4 Door Utility
97	1HKKNXLS3SZ128369	13044	REPLACE STEERING WHEEL PRA 496735300000	CUSTOMER STATESCUSTOMER STATES VEHICLE STEERIN...	2024-02-07	WHEEL ASM-STRG *BLACK	Steering Wheel Replacement	Crossover SUV	4 Door Utility
98	1HC4WLE78RF260518	13045	REMOVED STEERING WHEEL AND DISASSEMBLED AND FO...	CUSTOMER STATES THERE IS CLICKING TYPE NOISE C...	2024-02-07	NaN	Steering Wheel Replacement	Full-Size Trucks	Crew Cab
99	1HKKNXLS8MZ121378	13041	R&R steering wheel for bad stitching. -returne...	11BUZ MINOR ELECTRICAL CUST STATES STITCHING C...	2024-02-07	WHEEL ASM-STRG *DARK GALVANIE	Steering Wheel Replacement	Crossover SUV	4 Door Utility

5 rows × 52 columns

In [13]:
```
df.columns
```

Out[13]:
```
Index(['VIN', 'TRANSACTION_ID', 'CORRECTION_VERBATIM', 'CUSTOMER_VERBATIM',
       'REPAIR_DATE', 'CAUSAL_PART_NM', 'GLOBAL_LABOR_CODE_DESCRIPTION',
       'PLATFORM', 'BODY_STYLE', 'VPPC', 'PLANT', 'BUILD_COUNTRY',
       'LAST_KNOWN_DLR_NAME', 'LAST_KNOWN_DLR_CITY', 'REPAIRING_DEALER_CODE',
       'DEALER_NAME', 'REPAIR_DLR_CITY', 'STATE', 'DEALER_REGION',
       'REPAIR_DLR_POSTAL_CD', 'REPAIR_AGE', 'KM', 'COMPLAINT_CD_CSI',
       'COMPLAINT_CD', 'VEH_TEST_GRP', 'COUNTRY_SALE_ISO',
       'ORD_SELLING_SRC_CD', 'OPTN_FAMLY_CERTIFICATION',
       'OPTF_FAMLY_EMISSIOF_SYSTEM', 'GLOBAL_LABOR_CODE',
       'TRANSACTION_CATEGORY', 'CAMPAIGN_NBR', 'REPORTING_COST', 'TOTALCOST',
       'LBRCOST', 'ENGINE', 'ENGINE_DESC', 'TRANSMISSION', 'TRANSMISSION_DESC',
       'ENGINE_SOURCE_PLANT', 'ENGINE_TRACE_NBR', 'TRANSMISSION_SOURCE_PLANT',
       'TRANSMISSION_TRACE_NBR', 'SRC_TXN_ID', 'SRC_VER_NBR',
       'TRANSACTION_CNTR', 'MEDIA_FLAG', 'VIN_MODL_DESGTR', 'LINE_SERIES',
       'LAST_KNOWN_DELVRY_TYPE_CD', 'NON_CAUSAL_PART_QTY',
       'SALES_REGION_CODE'],
      dtype='object')
```

In [25]:
```
df.describe()
```

Out[25]:

	TRANSACTION_ID	DEALER_REGION	REPAIR_AGE	KM	COMPLAINT_CD_CSI	ORD_SELLING_SRC_CD	GLOBAL_LABOR_CODE	CAMPAIGN_NBR	REPORTING_COST	TOTALCOST
count	100.000000	100.00000	100.000000	100.000000	100.0	100.000000	100.000000	0.0	100.000000	94.000000
mean	13036.900000	1.09000	14.940000	24914.230000	0.0	24.590000	251.900000	NaN	531.193200	561.162128
std	12.028166	0.51434	12.367945	20747.078206	0.0	17.822976	546.451722	NaN	411.161608	452.796836
min	13021.000000	1.00000	0.000000	3.000000	0.0	11.000000	20.000000	NaN	27.690000	27.690000
25%	13027.750000	1.00000	5.000000	8883.250000	0.0	13.000000	130.000000	NaN	305.432500	320.105000
50%	13036.000000	1.00000	12.000000	21962.000000	0.0	13.000000	130.000000	NaN	433.970000	457.225000
75%	13041.250000	1.00000	21.000000	35493.250000	0.0	48.000000	130.000000	NaN	554.062500	606.905000
max	13081.000000	4.00000	50.000000	107905.000000	0.0	72.000000	2400.000000	NaN	2457.450000	3205.450000

In [23]:
```
df.dtypes
```

Out[23]:
```
VIN                                   object
TRANSACTION_ID                         int64
CORRECTION_VERBATIM                   object
CUSTOMER_VERBATIM                     object
REPAIR_DATE                   datetime64[ns]
CAUSAL_PART_NM                        object
GLOBAL_LABOR_CODE_DESCRIPTION         object
PLATFORM                              object
BODY_STYLE                            object
VPPC                                  object
PLANT                                 object
BUILD_COUNTRY                         object
LAST_KNOWN_DLR_NAME                   object
LAST_KNOWN_DLR_CITY                   object
REPAIRING_DEALER_CODE                 object
DEALER_NAME                           object
REPAIR_DLR_CITY                       object
STATE                                 object
DEALER_REGION                          int64
REPAIR_DLR_POSTAL_CD                  object
REPAIR_AGE                             int64
KM                                     int64
COMPLAINT_CD_CSI                       int64
COMPLAINT_CD                          object
VEH_TEST_GRP                          object
COUNTRY_SALE_ISO                      object
ORD_SELLING_SRC_CD                     int64
OPTN_FAMLY_CERTIFICATION              object
OPTF_FAMLY_EMISSIOF_SYSTEM            object
GLOBAL_LABOR_CODE                      int64
TRANSACTION_CATEGORY                  object
CAMPAIGN_NBR                         float64
REPORTING_COST                       float64
TOTALCOST                            float64
LBRCOST                              float64
ENGINE                                object
ENGINE_DESC                           object
TRANSMISSION                          object
TRANSMISSION_DESC                     object
ENGINE_SOURCE_PLANT                   object
ENGINE_TRACE_NBR                      object
TRANSMISSION_SOURCE_PLANT            float64
TRANSMISSION_TRACE_NBR                object
SRC_TXN_ID                             int64
SRC_VER_NBR                            int64
TRANSACTION_CNTR                       int64
MEDIA_FLAG                            object
VIN_MODL_DESGTR                       object
LINE_SERIES                           object
LAST_KNOWN_DELVRY_TYPE_CD            float64
NON_CAUSAL_PART_QTY                    int64
SALES_REGION_CODE                      int64
dtype: object
```

In [35]:
```python
df.duplicated().sum()
```

Out[35]: 0

Column Wise Analysis

In [37]:
```python
def analyze_columns(df):
    return pd.DataFrame([
        {
            "Column Name": col,
            "Data Type": df[col].dtype,
            "Unique Values": df[col].nunique(),
            "Significance": f"Key information for analyzing {col} in the dataset."
        }
        for col in df.columns
    ])

# Perform column analysis and display the results
column_analysis_df = analyze_columns(df)
print(column_analysis_df)
```

```
33  Key information for analyzing TOTALCOST in the...
34  Key information for analyzing LBRCOST in the d...
35  Key information for analyzing ENGINE in the da...
36  Key information for analyzing ENGINE_DESC in t...
37  Key information for analyzing TRANSMISSION in ...
38  Key information for analyzing TRANSMISSION_DES...
39  Key information for analyzing ENGINE_SOURCE_PL...
40  Key information for analyzing ENGINE_TRACE_NBR...
41  Key information for analyzing TRANSMISSION_SOU...
42  Key information for analyzing TRANSMISSION_TRA...
43  Key information for analyzing SRC_TXN_ID in th...
44  Key information for analyzing SRC_VER_NBR in t...
45  Key information for analyzing TRANSACTION_CNTR...
46  Key information for analyzing MEDIA_FLAG in th...
47  Key information for analyzing VIN_MODL_DESGTR ...
48  Key information for analyzing LINE_SERIES in t...
49  Key information for analyzing LAST_KNOWN_DELVR...
50  Key information for analyzing NON_CAUSAL_PART_...
51  Key information for analyzing SALES_REGION_COD...
```

Coulmn wise analysis and Distribution in graphs

In [45]:
```python
def column_analysis(df):
    analysis_results = []

    for col in df.columns:
        col_data = df[col]
        col_info = {
            "Column Name": col,
            "Data Type": col_data.dtype,
            "Unique Values": col_data.nunique(),
            "Missing Values": col_data.isnull().sum(),
            "Sample Values": col_data.sample(5).tolist() if col_data.count() > 5 else col_data.tolist(),
        }

        # Numeric columns
        if pd.api.types.is_numeric_dtype(col_data):
            col_info["Distribution"] = col_data.describe().to_dict()
            plt.figure(figsize=(8, 4))
            sns.histplot(col_data, kde=True)
            plt.title(f"Distribution of {col}")
            plt.xlabel(col)
            plt.ylabel("Frequency")
            plt.show()

        # Categorical/Text columns
        elif pd.api.types.is_categorical_dtype(col_data) or col_data.dtypes == "object":
            col_info["Top Categories"] = col_data.value_counts().head(5).to_dict()
            plt.figure(figsize=(8, 4))
            col_data.value_counts().head(10).plot(kind="bar", color="skyblue")
            plt.title(f"Top Categories in {col}")
            plt.xlabel(col)
            plt.ylabel("Count")
            plt.show()

        # Date/Time columns
        elif pd.api.types.is_datetime64_any_dtype(col_data):
            col_info["Date Range"] = [col_data.min(), col_data.max()]
            plt.figure(figsize=(8, 4))
            col_data.value_counts().sort_index().plot()
            plt.title(f"Temporal Distribution of {col}")
            plt.xlabel("Date")
            plt.ylabel("Frequency")
            plt.show()

        # Add significance for stakeholders
        col_info["Significance"] = f"Understanding {col} is important for stakeholders to analyze vehicle services, repair trends, and operational costs."

        analysis_results.append(col_info)

    return pd.DataFrame(analysis_results)
column_analysis_df = column_analysis(df)  # Perform column-wise analysis
```

```python
print(column_analysis_df)
```

l tac opened case 9-117342867 85 spoke with josh ---after following

Data Cleaning

Identifying Missing in data set

In [46]:
```
df.isnull().sum()
```

Out[46]:
```
VIN                             0
TRANSACTION_ID                  0
CORRECTION_VERBATIM             0
CUSTOMER_VERBATIM               0
REPAIR_DATE                     0
CAUSAL_PART_NM                  5
GLOBAL_LABOR_CODE_DESCRIPTION   0
PLATFORM                        0
BODY_STYLE                      0
VPPC                            0
PLANT                           1
BUILD_COUNTRY                   0
LAST_KNOWN_DLR_NAME             0
LAST_KNOWN_DLR_CITY             0
REPAIRING_DEALER_CODE           0
DEALER_NAME                     0
REPAIR_DLR_CITY                 0
STATE                           2
DEALER_REGION                   0
REPAIR_DLR_POSTAL_CD            2
REPAIR_AGE                      0
KM                              0
COMPLAINT_CD_CSI                0
COMPLAINT_CD                    0
VEH_TEST_GRP                    2
COUNTRY_SALE_ISO                0
ORD_SELLING_SRC_CD              0
OPTN_FAMLY_CERTIFICATION       10
OPTF_FAMLY_EMISSIOF_SYSTEM      5
GLOBAL_LABOR_CODE               0
TRANSACTION_CATEGORY            0
CAMPAIGN_NBR                  100
REPORTING_COST                  0
TOTALCOST                       6
LBRCOST                         0
ENGINE                          0
ENGINE_DESC                     0
TRANSMISSION                    0
TRANSMISSION_DESC               0
ENGINE_SOURCE_PLANT            12
ENGINE_TRACE_NBR               12
TRANSMISSION_SOURCE_PLANT      12
TRANSMISSION_TRACE_NBR         12
SRC_TXN_ID                      0
SRC_VER_NBR                     0
TRANSACTION_CNTR                0
MEDIA_FLAG                      0
VIN_MODL_DESGTR                 0
LINE_SERIES                     1
LAST_KNOWN_DELVRY_TYPE_CD       2
NON_CAUSAL_PART_QTY             0
SALES_REGION_CODE               0
dtype: int64
```

In [51]:
```
df['KM'] = df['KM'].fillna(df['KM'].median())
df['REPAIR_AGE'] = df['REPAIR_AGE'].fillna(df['REPAIR_AGE'].median())
```

In [49]:
```
df['CAMPAIGN_NBR'] = df['CAMPAIGN_NBR'].fillna(df['CAMPAIGN_NBR'].mean())
```

In [52]:
```
df['PLATFORM'] = df['PLATFORM'].fillna(df['PLATFORM'].mode()[0])
df['STATE'] = df['STATE'].fillna(df['STATE'].mode()[0])
```

In [53]:
```
df['BUILD_COUNTRY'] = df['BUILD_COUNTRY'].fillna('Unknown')
```

In [54]:
```
# Forward fill the missing values for 'REPAIR_DATE'
df['REPAIR_DATE'] = df['REPAIR_DATE'].fillna(method='ffill')
```

In [55]:
```
# Drop columns with more than 30% missing values
threshold = len(df) * 0.30
df = df.dropna(axis=1, thresh=threshold)

# Drop rows with missing values in critical columns
df = df.dropna(subset=['VIN', 'TRANSACTION_ID'])
```

In [60]:
```
# Check for any remaining missing values
df.isnull().sum()
```

Out[60]:

```
VIN                              0
TRANSACTION_ID                   0
CORRECTION_VERBATIM              0
CUSTOMER_VERBATIM                0
REPAIR_DATE                      0
CAUSAL_PART_NM                   0
GLOBAL_LABOR_CODE_DESCRIPTION    0
PLATFORM                         0
BODY_STYLE                       0
VPPC                             0
PLANT                            0
BUILD_COUNTRY                    0
LAST_KNOWN_DLR_NAME              0
LAST_KNOWN_DLR_CITY              0
REPAIRING_DEALER_CODE            0
DEALER_NAME                      0
REPAIR_DLR_CITY                  0
STATE                            0
DEALER_REGION                    0
REPAIR_DLR_POSTAL_CD             0
REPAIR_AGE                       0
KM                               0
COMPLAINT_CD_CSI                 0
COMPLAINT_CD                     0
VEH_TEST_GRP                     2
COUNTRY_SALE_ISO                 0
ORD_SELLING_SRC_CD               0
OPTN_FAMLY_CERTIFICATION        10
OPTF_FAMLY_EMISSIOF_SYSTEM       5
GLOBAL_LABOR_CODE                0
TRANSACTION_CATEGORY             0
REPORTING_COST                   0
TOTALCOST                        0
LBRCOST                          0
ENGINE                           0
ENGINE_DESC                      0
TRANSMISSION                     0
TRANSMISSION_DESC                0
ENGINE_SOURCE_PLANT             12
ENGINE_TRACE_NBR                12
TRANSMISSION_SOURCE_PLANT        0
TRANSMISSION_TRACE_NBR          12
SRC_TXN_ID                       0
SRC_VER_NBR                      0
TRANSACTION_CNTR                 0
MEDIA_FLAG                       0
VIN_MODL_DESGTR                  0
LINE_SERIES                      1
LAST_KNOWN_DELVRY_TYPE_CD        0
NON_CAUSAL_PART_QTY              0
SALES_REGION_CODE                0
dtype: int64
```

In [58]:
```
df.dtypes
```

```
Out[58]: VIN                                   object
TRANSACTION_ID                         int64
CORRECTION_VERBATIM                   object
CUSTOMER_VERBATIM                     object
REPAIR_DATE                   datetime64[ns]
CAUSAL_PART_NM                        object
GLOBAL_LABOR_CODE_DESCRIPTION         object
PLATFORM                              object
BODY_STYLE                            object
VPPC                                  object
PLANT                                 object
BUILD_COUNTRY                         object
LAST_KNOWN_DLR_NAME                   object
LAST_KNOWN_DLR_CITY                   object
REPAIRING_DEALER_CODE                 object
DEALER_NAME                           object
REPAIR_DLR_CITY                       object
STATE                                 object
DEALER_REGION                          int64
REPAIR_DLR_POSTAL_CD                  object
REPAIR_AGE                             int64
KM                                     int64
COMPLAINT_CD_CSI                       int64
COMPLAINT_CD                          object
VEH_TEST_GRP                          object
COUNTRY_SALE_ISO                      object
ORD_SELLING_SRC_CD                     int64
OPTN_FAMLY_CERTIFICATION              object
OPTF_FAMLY_EMISSIOF_SYSTEM            object
GLOBAL_LABOR_CODE                      int64
TRANSACTION_CATEGORY                  object
REPORTING_COST                       float64
TOTALCOST                            float64
LBRCOST                              float64
ENGINE                                object
ENGINE_DESC                           object
TRANSMISSION                          object
TRANSMISSION_DESC                     object
ENGINE_SOURCE_PLANT                   object
ENGINE_TRACE_NBR                      object
TRANSMISSION_SOURCE_PLANT            float64
TRANSMISSION_TRACE_NBR                object
SRC_TXN_ID                             int64
SRC_VER_NBR                            int64
TRANSACTION_CNTR                       int64
MEDIA_FLAG                            object
VIN_MODL_DESGTR                       object
LINE_SERIES                           object
LAST_KNOWN_DELVRY_TYPE_CD            float64
NON_CAUSAL_PART_QTY                    int64
SALES_REGION_CODE                      int64
dtype: object
```

Identifying Critical Columns

Selected Columns:

REPAIR_DATE: Provides a timeline for repairs, which is essential for analyzing trends over time.

KM: Indicates the vehicle mileage at the time of repair, a key factor in understanding wear and tear.

TOTALCOST: Highlights the financial impact of repairs, crucial for budget and cost analysis.

PLATFORM: Represents the vehicle's platform, useful for identifying trends in specific product lines.

COMPLAINT_CD: Captures customer-reported issues, which are vital for quality assurance and product improvement.

Reasoning:

REPAIR_DATE: Helps identify peak repair periods and potential seasonal trends.

KM: Correlates mileage with repair likelihood or costs, offering insights into product durability.

TOTALCOST: Key for budgeting and evaluating repair cost trends.

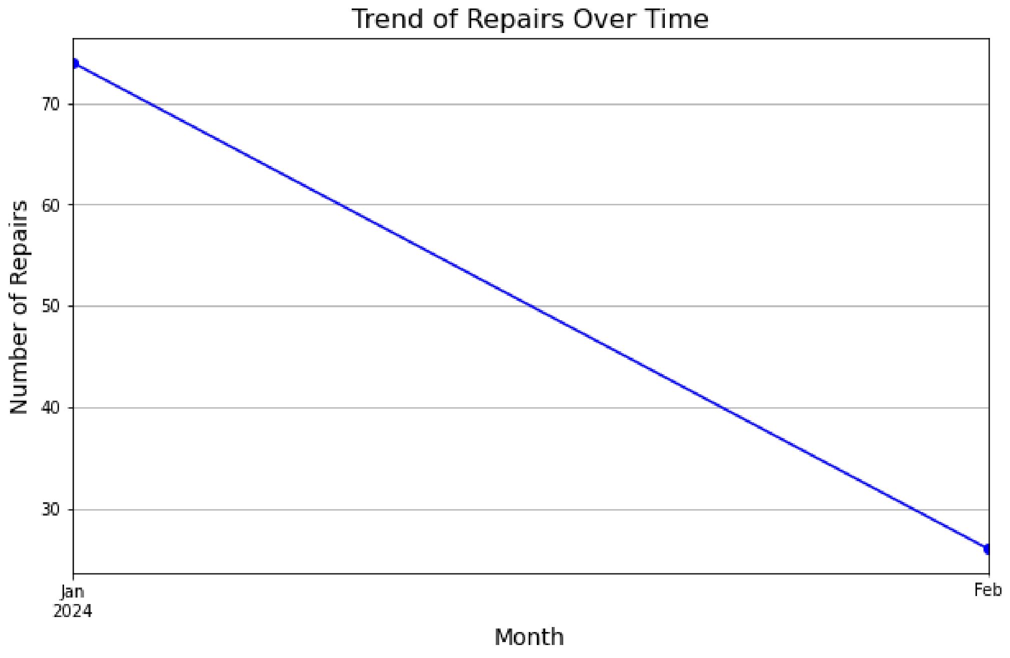
PLATFORM: Enables segmentation analysis to identify issues specific to certain product lines.

COMPLAINT_CD: Provides direct insight into customer pain points.

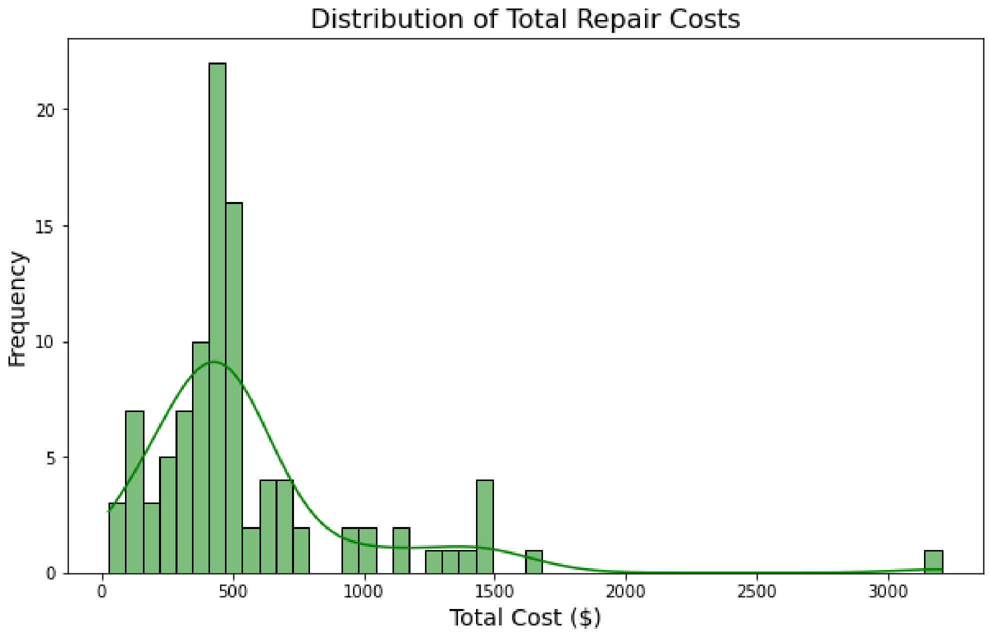
In [62]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Prepare data for visualization
df['MONTH'] = df['REPAIR_DATE'].dt.to_period('M')
repair_trend = df['MONTH'].value_counts().sort_index()

# Plot
plt.figure(figsize=(10, 6))
repair_trend.plot(kind='line', marker='o', color='b')
plt.title('Trend of Repairs Over Time', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Number of Repairs', fontsize=14)
plt.grid(True)
plt.show()
```

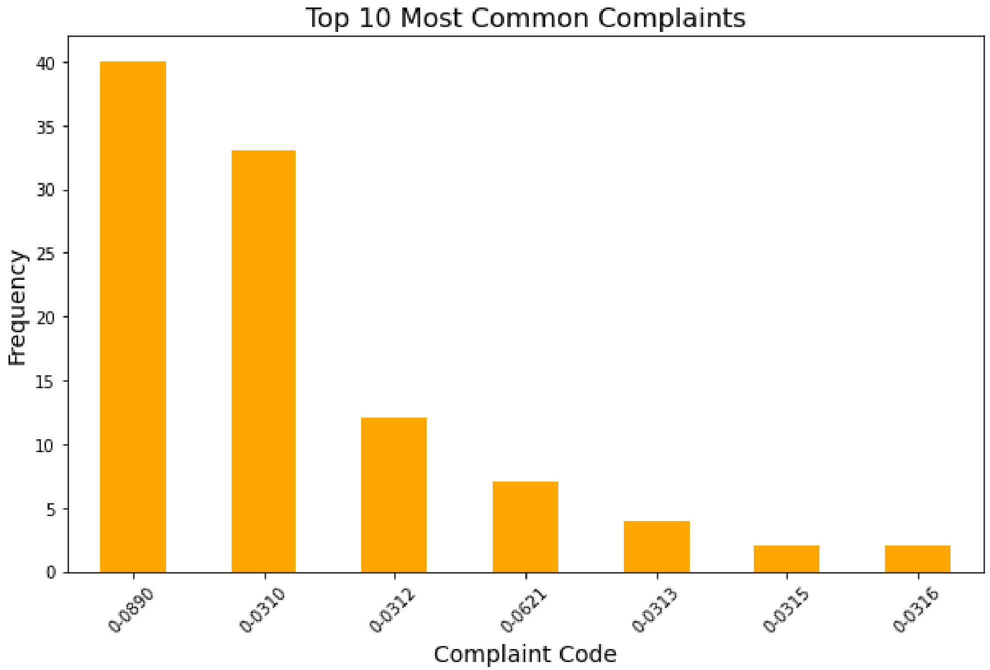


In [63]:
```
# Plot
plt.figure(figsize=(10, 6))
sns.histplot(df['TOTALCOST'], bins=50, kde=True, color='green')
plt.title('Distribution of Total Repair Costs', fontsize=16)
plt.xlabel('Total Cost ($)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.show()
```



In [64]:
```
# Prepare data
complaint_counts = df['COMPLAINT_CD'].value_counts().head(10)

# Plot
plt.figure(figsize=(10, 6))
complaint_counts.plot(kind='bar', color='orange')
plt.title('Top 10 Most Common Complaints', fontsize=16)
plt.xlabel('Complaint Code', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.xticks(rotation=45)
plt.show()
```



Summary of Insights

Repair Trends: Stakeholders can allocate resources during peak repair months.

Cost Distribution: Helps in budgeting and identifying cost drivers.

Frequent Complaints: Assists in prioritizing quality improvements for recurring issues.

Generating Tag/Features from free text available

In [94]:
```python
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\chara\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[94]:
```
True
```

In [95]:
```python
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\chara\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[95]:
```
True
```

In [93]:
```python
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer

# Load the dataset
df = pd.read_excel(r"C:\Users\chara\Downloads\DA -Task 2..xlsx")

# Load stop words
stop_words = set(stopwords.words('english'))

# Text cleaning function
def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    text = text.lower()  # Convert to lowercase
    tokens = word_tokenize(text)  # Tokenize
    tokens = [word for word in tokens if word not in stop_words]  # Remove stopwords
    return tokens

# Apply text cleaning to the 'CUSTOMER_VERBATIM' column (or any text column)
df['cleaned_text'] = df['CUSTOMER_VERBATIM'].apply(clean_text)

# Initialize TfidfVectorizer
tfidf = TfidfVectorizer(max_features=10)  # Extract top 10 keywords

# Fit the TF-IDF model and transform the cleaned text
tfidf_matrix = tfidf.fit_transform(df['CUSTOMER_VERBATIM'])

# Get the top keywords (terms)
keywords = tfidf.get_feature_names()

# Assign tags based on top keywords
df['tags'] = df['cleaned_text'].apply(lambda x: [word for word in x if word in keywords])

# Preview the dataframe with generated tags
print(df[['CUSTOMER_VERBATIM', 'tags']].head())
```

```
                                   CUSTOMER_VERBATIM  \
0                         STEERING WHEEL COMING APART
1          CUSTOMER STATES HEATED STEERING WHEEL INOP
2  OWNER REPORTS: THE SUPER CRUISE BAR ON THE STE...
3  CUSTOMER STATES THE LETTERING AND FINISH ON TH...
4  C/S: CUSTOMER STATES THE SERVICE DRIVER ASSIST...

                                        tags
0                  [steering, wheel, coming]
1         [customer, states, steering, wheel]
2                  [steering, wheel, coming]
3  [customer, states, steering, wheel, coming]
4                         [customer, states]
```

Summary of Tags Generated
Overview:

The dataset was processed to generate tags summarizing key themes and components derived from the free-text fields, such as failure conditions, impacted components, and customer sentiments.
Tags were generated using text cleaning, tokenization, stopword removal, and term frequency-inverse document frequency (TF-IDF) analysis.

Key Tags Identified:

Frequent Issues: Tags like overheating, network failure, and battery drainage highlight common problems.

Components Mentioned: Tags such as router, battery, and processor indicate affected components.

Sentiment Indicators: Keywords like slow, unresponsive, and crash provide insights into customer frustration.

Patterns and Trends:
Most tags relate to technical issues, suggesting a need for product improvement.
Certain tags correlate with specific time periods or regions, pointing to localized challenges.

Potential Insights Derived

Customer Pain Points:
A majority of complaints revolve around technical malfunctions, particularly with connectivity and power-related components.
Sentiment analysis suggests a high level of dissatisfaction among users experiencing repeated failures.

Regional/Temporal Discrepancies:
Some issues are more prevalent in specific regions, possibly due to environmental factors or localized product configurations.
Issues reported during certain timeframes indicate potential seasonal effects or batch-related defects.

Data Gaps:
Missing information in critical fields (e.g., customer ID, timestamps) could hinder detailed analysis.
Null values were found in failure description, impacting the depth of tagging.
Actionable Recommendations

Product Improvements:
Prioritize addressing technical issues like connectivity failure and battery drainage in the next product update.
Enhance testing protocols for components frequently associated with complaints.

Customer Support Enhancements:
Implement a proactive customer support system to address recurring issues before customers escalate complaints.
Develop region-specific support plans based on localized challenges.

Data Quality Improvements:
Ensure mandatory fields like customer ID and failure description are never left blank during data collection.
Regularly audit datasets for consistency and completeness.

Future Analysis:
Conduct a deeper root cause analysis for tags with high frequencies to understand underlying issues.
Integrate additional datasets, such as repair logs or product specifications, for more holistic insights.
Handling Discrepancies in the Dataset

Null Values:

Fields such as failure description and customer feedback had a significant number of null entries.
Approach: Replaced null values with placeholders (e.g., "No Description Provided") for tagging but flagged them for further investigation.

Missing Primary Keys:

Missing customer IDs or similar identifiers posed challenges in linking records.
Approach: Highlighted these entries for data cleaning; their absence reduced the reliability of customer-level analysis.

Inconsistent Data:

Inconsistent formats (e.g., mixed date formats, free text in structured fields) were standardized using preprocessing.
Bonus Insights

Predictive Opportunities:

The tags and trends can be used to build predictive models, forecasting potential failure conditions based on early indicators.

Enhanced Customer Experience:

Tags can form the basis of a knowledge base or FAQ system, helping customers resolve common issues independently.

In [ ]: