Loading the Dataset and Initial Review

In [2]:

```
import pandas as pd

# Load the dataset
df = pd.read_csv('merged_dataset.csv')

# Display the first few rows to understand its structure and what kind of data it co
df.head()
```

Out[2]:

	WorkOrder	Item No.	Item	Shipment	DateClosed	Order Qty_x	Completed Qty	FOH	VO
0	15887.0	80-015-11582716	ETWD BATTERY LINER, MACH	MS	2024-03-08	110.0	110.0	516.91	2072.7
1	15854.0	80-585-7103926-9MW	ENCLOSURE+HDWT INSTALL ASSEM	WELD	2024-03-13	1.0	1.0	2037.73	7523.9
2	15808.0	80-535-7104257-2M	COVER, BOTTOM, MACHINED	MS	2024-03-13	1.0	1.0	171.27	646.5
3	15807.0	80-535-7104257-2M	COVER, BOTTOM, MACHINED	MS	2024-03-13	1.0	1.0	231.31	867.2
4	15806.0	80-535-7104257-2M	COVER, BOTTOM, MACHINED	MS	2024-02-29	1.0	1.0	259.70	973.0

5 rows × 30 columns

Assessing Data Cleanliness and Preprocessing Needs

In [3]:

```
# Check for missing values in each column
missing_values = df.isnull().sum()

# Check for duplicate rows
duplicate_rows = df.duplicated().sum()

# Checking data types for potential inconsistencies
data_types = df.dtypes

missing_values, duplicate_rows, data_types
```

Out[3]:
```
(WorkOrder                0
 Item No.                 0
 Item                     1
 Shipment               164
 DateClosed               0
 Order Qty_x              0
 Completed Qty            0
 FOH                      0
 VOH                      0
 ActualMaterialCost       0
 ActualLaborCost          0
 MaterialPerUnit          0
 LaborPerUnit             0
 OHPerUnit                0
 CostPerUnit              0
 Type                     0
 lot                      0
 Program                  3
 Work Order               0
 Routing Number           2
 Item Bill Number         0
 Order Qty_y              0
 Hours in date range      0
 Up to date hours         0
 Machines                 0
 Cycle Time              27
 Predicted Hours         27
 remainHours             27
 Employee                 0
 WorkOrderID              0
 dtype: int64,
 0,
 WorkOrder              float64
 Item No.                object
 Item                    object
 Shipment                object
 DateClosed              object
 Order Qty_x            float64
 Completed Qty          float64
 FOH                    float64
 VOH                    float64
 ActualMaterialCost     float64
 ActualLaborCost        float64
 MaterialPerUnit         object
 LaborPerUnit            object
 OHPerUnit               object
 CostPerUnit             object
 Type                    object
 lot                     object
 Program                 object
 Work Order              object
 Routing Number          object
 Item Bill Number        object
 Order Qty_y             object
 Hours in date range     object
 Up to date hours        object
 Machines                object
 Cycle Time              object
 Predicted Hours         object
 remainHours             object
 Employee                object
 WorkOrderID            float64
 dtype: object)
```

Data Cleaning Process

In [6]:

```
# Handling Missing Values and Correcting Data Types
df['Cycle Time'] = pd.to_numeric(df['Cycle Time'], errors='coerce').fillna(0)
df['Predicted Hours'] = pd.to_numeric(df['Predicted Hours'], errors='coerce').fillna
df['remainHours'] = pd.to_numeric(df['remainHours'], errors='coerce').fillna(0)
df['Shipment'].fillna('Unknown', inplace=True)
df['Program'].fillna('Unknown', inplace=True)
df['DateClosed'] = pd.to_datetime(df['DateClosed'])

# Confirm the changes by re-checking data types and missing values.
df.dtypes, df.isnull().sum()
```

Out[6]:
```
(WorkOrder                     float64
 Item No.                       object
 Item                           object
 Shipment                       object
 DateClosed             datetime64[ns]
 Order Qty_x                   float64
 Completed Qty                 float64
 FOH                           float64
 VOH                           float64
 ActualMaterialCost            float64
 ActualLaborCost               float64
 MaterialPerUnit                object
 LaborPerUnit                   object
 OHPerUnit                      object
 CostPerUnit                    object
 Type                           object
 lot                            object
 Program                        object
 Work Order                     object
 Routing Number                 object
 Item Bill Number               object
 Order Qty_y                    object
 Hours in date range            object
 Up to date hours               object
 Machines                       object
 Cycle Time                    float64
 Predicted Hours               float64
 remainHours                   float64
 Employee                       object
 WorkOrderID                   float64
 dtype: object,
 WorkOrder              0
 Item No.               0
 Item                   1
 Shipment               0
 DateClosed             0
 Order Qty_x            0
 Completed Qty          0
 FOH                    0
 VOH                    0
 ActualMaterialCost     0
 ActualLaborCost        0
 MaterialPerUnit        0
 LaborPerUnit           0
 OHPerUnit              0
 CostPerUnit            0
 Type                   0
 lot                    0
 Program                0
 Work Order             0
 Routing Number         2
 Item Bill Number       0
 Order Qty_y            0
 Hours in date range    0
 Up to date hours       0
 Machines               0
 Cycle Time             0
 Predicted Hours        0
 remainHours            0
 Employee               0
 WorkOrderID            0
 dtype: int64)
```

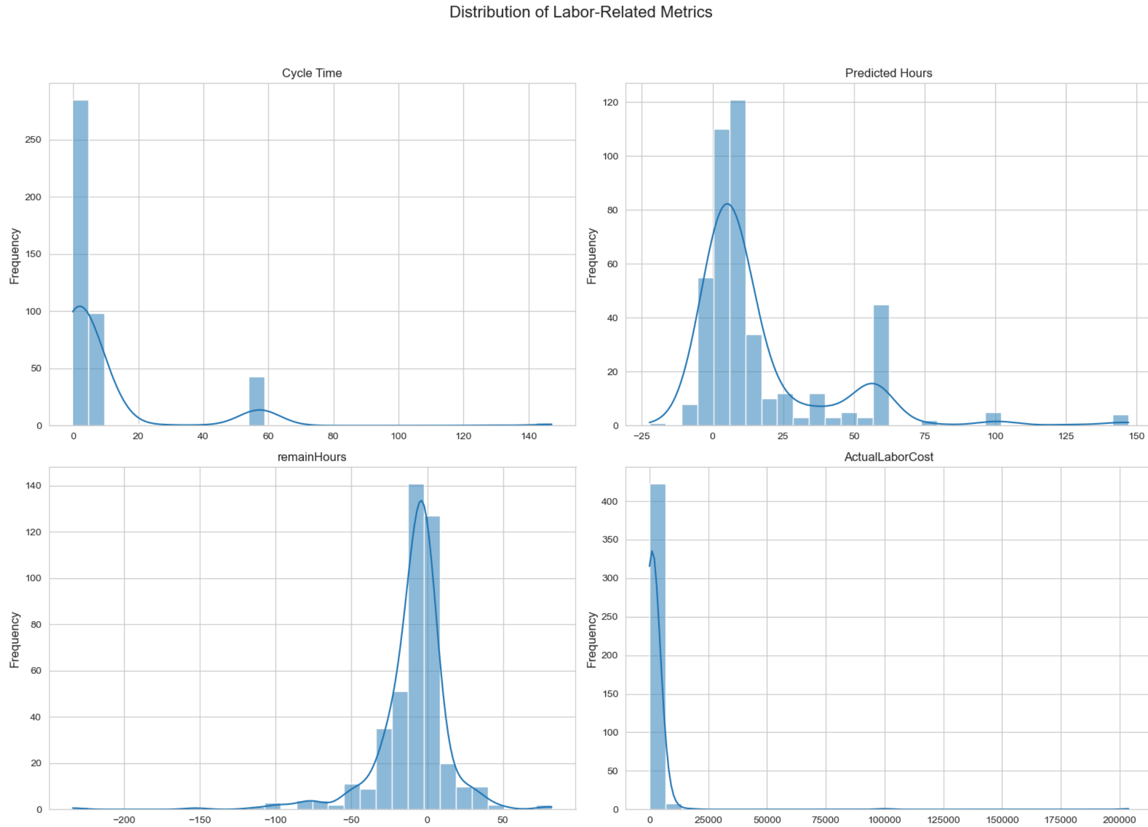
Exploratory Data Analysis (EDA)

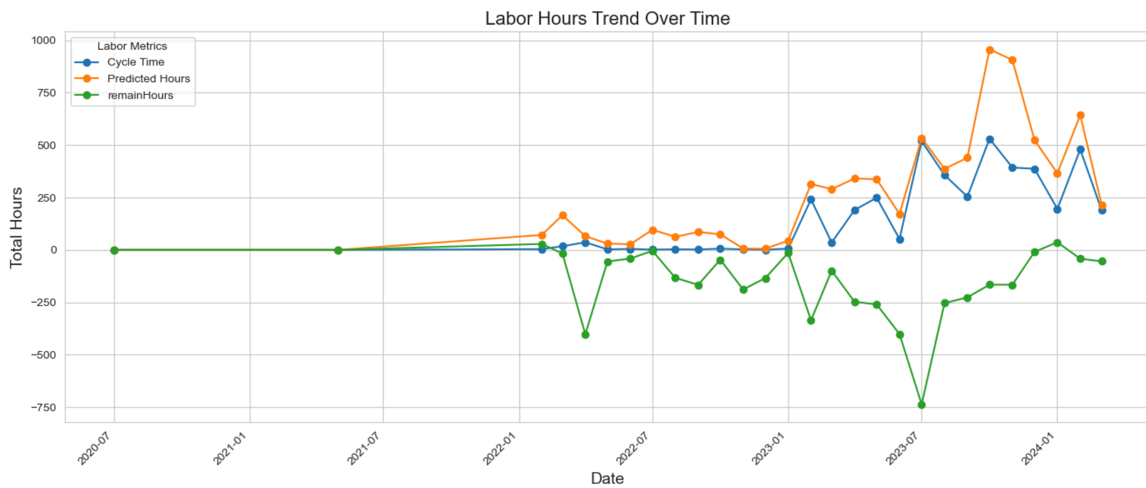
In [7]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Summary statistics for labor-related columns
labor_statistics = df[['Cycle Time', 'Predicted Hours', 'remainHours', 'ActualLaborC

# Distribution plots for labor-related columns
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
fig.suptitle('Distribution of Labor-Related Metrics', fontsize=16)

for ax, column in zip(axes.flatten(), ['Cycle Time', 'Predicted Hours', 'remainHours
    sns.histplot(df[column], kde=True, bins=30, ax=ax)
    ax.set_title(column)
    ax.set_xlabel('')
    ax.set_ylabel('Frequency')
    ax.xaxis.get_label().set_fontsize(12)
    ax.yaxis.get_label().set_fontsize(12)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])

# Trends over time
labor_trend_data = df.groupby(df['DateClosed'].dt.to_period('M'))[['Cycle Time', 'Pr
labor_trend_data['DateClosed'] = labor_trend_data['DateClosed'].dt.to_timestamp()

# Plotting labor hours trend over time
fig, ax = plt.subplots(figsize=(14, 6))
labor_trend_data.plot(x='DateClosed', y=['Cycle Time', 'Predicted Hours', 'remainHou
ax.set_title('Labor Hours Trend Over Time', fontsize=16)
ax.set_xlabel('Date', fontsize=14)
ax.set_ylabel('Total Hours', fontsize=14)
ax.legend(title='Labor Metrics')

plt.xticks(rotation=45)
plt.tight_layout()

labor_statistics
```

Out[7]:

	Cycle Time	Predicted Hours	remainHours	ActualLaborCost
count	434.000000	434.000000	434.000000	434.000000
mean	9.584032	16.494977	-9.546313	2159.752419
std	21.227471	24.857878	24.779028	10922.555809
min	-0.220000	-22.290000	-234.000000	0.000000
25%	0.112500	1.445000	-14.305000	556.227500
50%	0.950000	8.310000	-5.980000	943.590000
75%	8.310000	16.222500	0.000000	1779.657500
max	147.170000	147.170000	82.150000	203835.680000


Distribution of Labor-Related Metrics
Cycle Time
Predicted Hours
remainHours
ActualLaborCost
Frequency

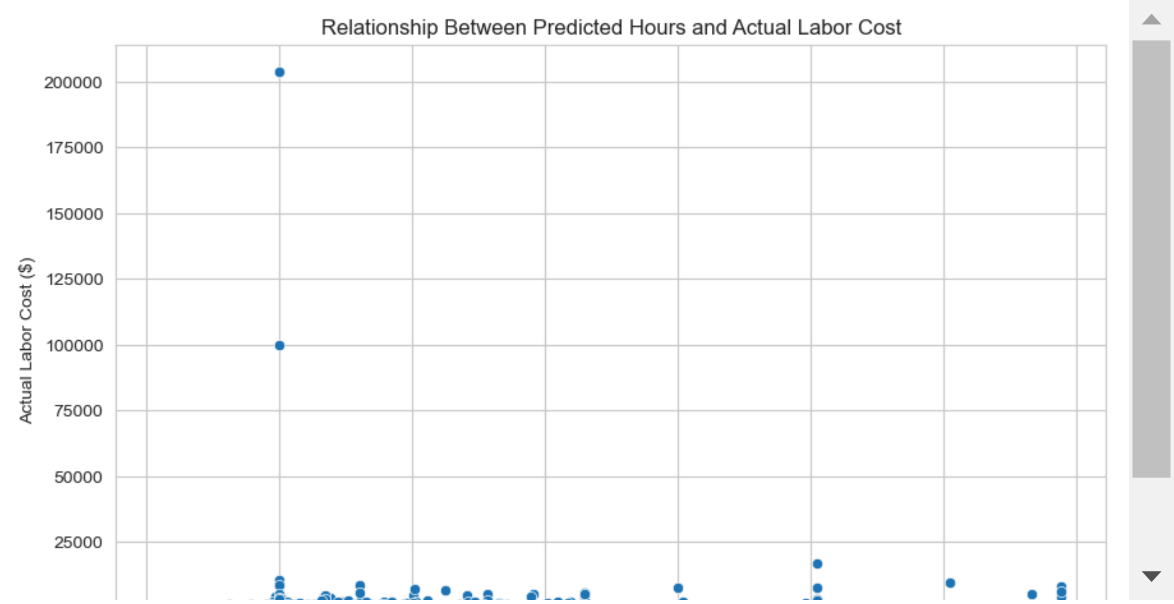

Labor Hours Trend Over Time
Labor Metrics
Cycle Time
Predicted Hours
remainHours
Total Hours
Date

Deep Dive Analysis

Relationship Between Predicted Hours and Actual Labor Cost

In [8]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Predicted Hours', y='ActualLaborCost', data=df)
plt.title('Relationship Between Predicted Hours and Actual Labor Cost')
plt.xlabel('Predicted Hours')
plt.ylabel('Actual Labor Cost ($)')
plt.grid(True)
plt.show()
```



Identifying High Labor Cost Work Orders

In [9]:

```
top_labor_cost_wo = df.sort_values(by='ActualLaborCost', ascending=False)[['WorkOrde
top_labor_cost_wo
```

Out[9]:

	WorkOrder	ActualLaborCost	DateClosed
433	5966.0	203835.68	2022-06-23
283	12785.0	99903.04	2024-01-24
147	13976.0	17084.07	2023-09-05
180	13616.0	10385.50	2024-01-24
287	12726.0	9569.11	2023-07-25

Correlation Analysis

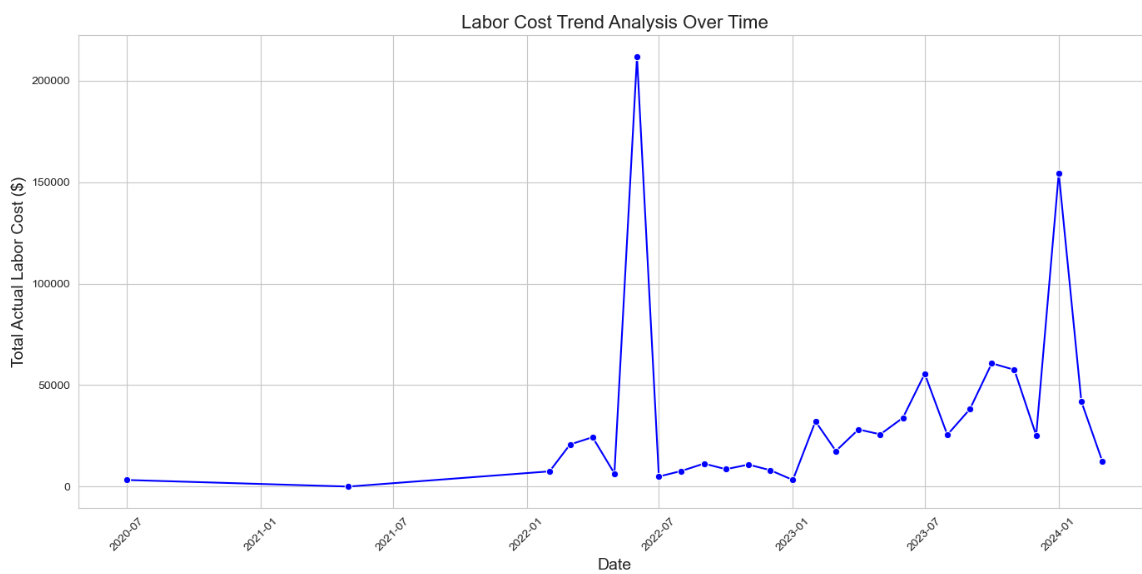
In [10]:
```
labor_efficiency_correlation = df[['Cycle Time', 'Predicted Hours', 'remainHours', '/
labor_efficiency_correlation
```

Out[10]:

	Cycle Time	Predicted Hours	remainHours	ActualLaborCost
Cycle Time	1.000000	0.753286	-0.015541	0.031626
Predicted Hours	0.753286	1.000000	0.065792	0.045396
remainHours	-0.015541	0.065792	1.000000	-0.052120
ActualLaborCost	0.031626	0.045396	-0.052120	1.000000

Labor Cost Trend Analysis

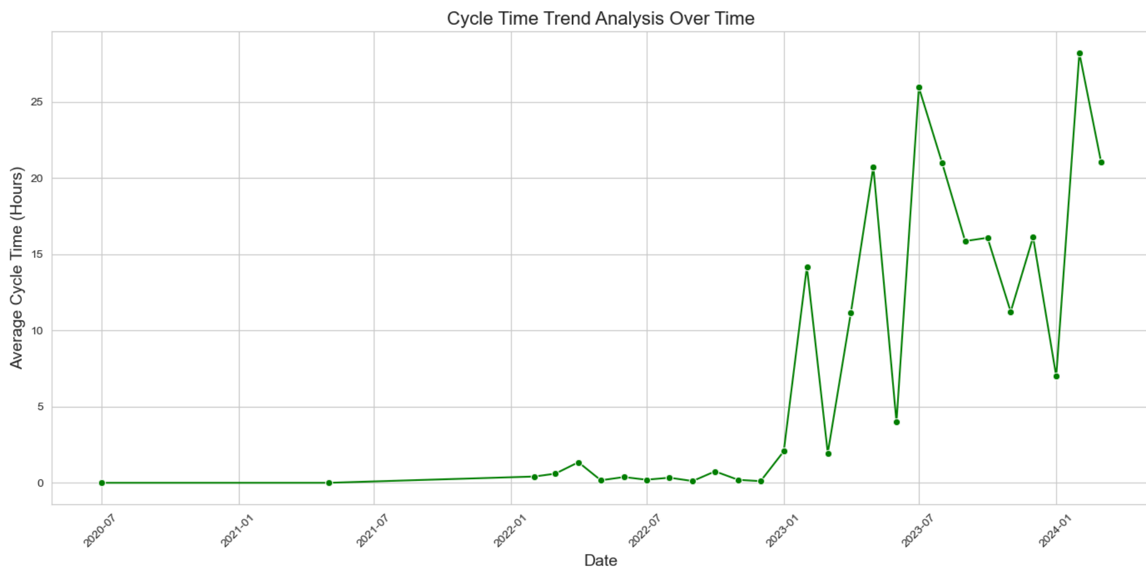
In [11]:
```
# Aggregating actual labor costs by month
labor_cost_trend = df.groupby(df['DateClosed'].dt.to_period('M'))['ActualLaborCost']
labor_cost_trend['DateClosed'] = labor_cost_trend['DateClosed'].dt.to_timestamp()

# Plotting the labor cost trend over time
plt.figure(figsize=(14, 7))
sns.lineplot(x='DateClosed', y='ActualLaborCost', data=labor_cost_trend, marker='o',
plt.title('Labor Cost Trend Analysis Over Time', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Total Actual Labor Cost ($)', fontsize=14)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Cycle Time Trend Analysis

In [12]:

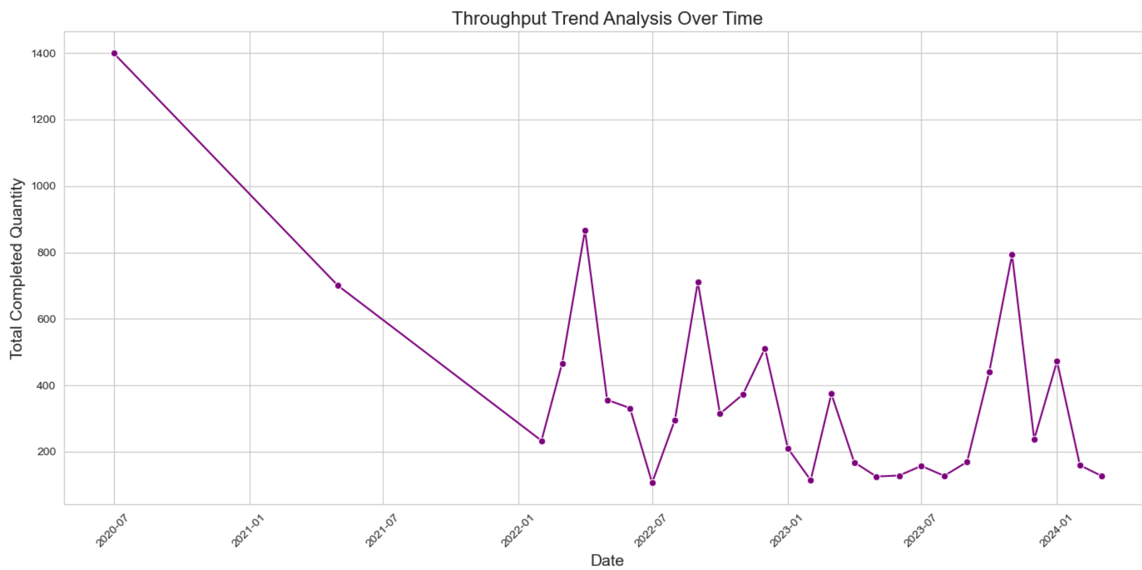
```
# Aggregating cycle times by month
cycle_time_trend = df.groupby(df['DateClosed'].dt.to_period('M'))['Cycle Time'].mean
cycle_time_trend['DateClosed'] = cycle_time_trend['DateClosed'].dt.to_timestamp()

# Plotting the cycle time trend over time
plt.figure(figsize=(14, 7))
sns.lineplot(x='DateClosed', y='Cycle Time', data=cycle_time_trend, marker='o', colo
plt.title('Cycle Time Trend Analysis Over Time', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Average Cycle Time (Hours)', fontsize=14)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Throughput Analysis

In [13]:

```
# Aggregating completed quantities by month to represent throughput
throughput_trend = df.groupby(df['DateClosed'].dt.to_period('M'))['Completed Qty'].su
throughput_trend['DateClosed'] = throughput_trend['DateClosed'].dt.to_timestamp()

# Plotting the throughput trend over time
plt.figure(figsize=(14, 7))
sns.lineplot(x='DateClosed', y='Completed Qty', data=throughput_trend, marker='o', co
plt.title('Throughput Trend Analysis Over Time', fontsize=16)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Total Completed Quantity', fontsize=14)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [ ]: