In [89]:
```
!pip install numpy pandas matplotlib seaborn
```

```
Requirement already satisfied: numpy in /Users/yaswanthkumarvejandla/yash/li
b/python3.12/site-packages (2.2.5)
Requirement already satisfied: pandas in /Users/yaswanthkumarvejandla/yash/l
ib/python3.12/site-packages (2.2.3)
Requirement already satisfied: matplotlib in /Users/yaswanthkumarvejandla/ya
sh/lib/python3.12/site-packages (3.10.1)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/yaswanthkuma
rvejandla/yash/lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Users/yaswanthkumarvejandla/
yash/lib/python3.12/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /Users/yaswanthkumarvejandl
a/yash/lib/python3.12/site-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/yaswanthkumarvejan
dla/yash/lib/python3.12/site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /Users/yaswanthkumarvejandla/
yash/lib/python3.12/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/yaswanthkumarveja
ndla/yash/lib/python3.12/site-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/yaswanthkumarveja
ndla/yash/lib/python3.12/site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /Users/yaswanthkumarvejand
la/yash/lib/python3.12/site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /Users/yaswanthkumarvejandla/yas
h/lib/python3.12/site-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/yaswanthkumarvejan
dla/yash/lib/python3.12/site-packages (from matplotlib) (3.2.3)
Requirement already satisfied: six>=1.5 in /Users/yaswanthkumarvejandla/yas
h/lib/python3.12/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.
0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
```

In [91]:
```
import pandas as pd
```

In [93]:
```
data = pd.read_csv("daily_sales_23_25.csv")
```

In [97]:
```
data.columns = [
    'period_no',
    'week_no',
    'date',
    'store_no',
    'store_name',
    'total_revenue',
    'projected_psa',
    'last_week_revenue',
    'last_week_pct_change',
    'last_year_revenue',
    'last_year_pct_change',
    'lunch_sales',
    'lunch_pct',
    'labor_pct',
    'scheduled_pct',
    'labor_cost',
    'scheduled_hours',
    'actual_hours',
    'hours_difference',
    'dpmh',
    'customer_count',
    'avg_ticket',
    'status'
]
```

In [101…

```
# Convert date column
data['date'] = pd.to_datetime(data['date'], errors='coerce')

# Convert numeric columns
numeric_cols = [
    'total_revenue', 'projected_psa', 'last_week_revenue',
    'last_week_pct_change', 'last_year_revenue', 'last_year_pct_change',
    'lunch_sales', 'lunch_pct', 'labor_pct', 'scheduled_pct', 'labor_cost',
    'scheduled_hours', 'actual_hours', 'hours_difference',
    'dpmh', 'customer_count', 'avg_ticket'
]

data[numeric_cols] = data[numeric_cols].apply(pd.to_numeric, errors='coerce'
```

In [103…

```
# Show count of missing values
print(data.isna().sum())

# Drop rows with missing dates or revenue
data = data.dropna(subset=['date', 'total_revenue'])

# Optionally, fill remaining missing numeric values with 0 or mean
data[numeric_cols] = data[numeric_cols].fillna(0)
```

```
period_no              0
week_no                0
date                   0
store_no               0
store_name             0
total_revenue          0
projected_psa          0
last_week_revenue      0
last_week_pct_change   0
last_year_revenue      0
last_year_pct_change   0
lunch_sales            0
lunch_pct              0
labor_pct              0
scheduled_pct          0
labor_cost             0
scheduled_hours        0
actual_hours           0
hours_difference       0
dpmh                   0
customer_count         0
avg_ticket             0
status                 0
dtype: int64
```

In [105…

```python
data = data.sort_values(by='date')
```

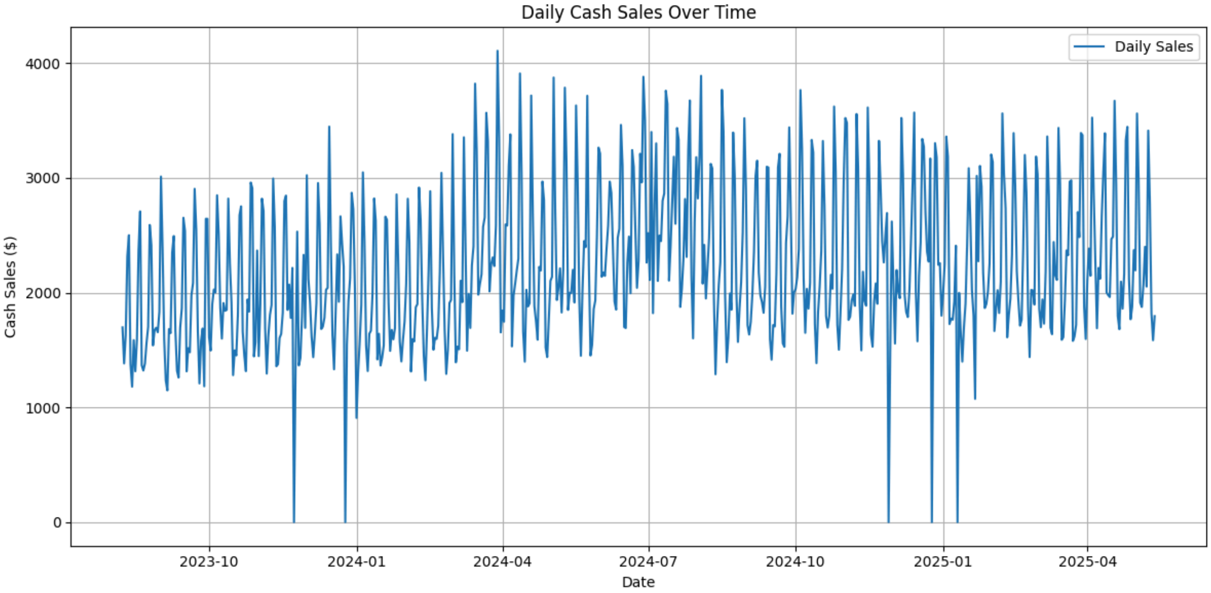
In [107…

```python
import numpy as np

sales = data['total_revenue'].to_numpy()
dates = data['date'].to_numpy()

print("Total revenue: $", np.sum(sales))
print("Average daily sales: $", np.mean(sales))
print("Max sale: $", sales.max(), "on", dates[np.argmax(sales)])
print("Min sale: $", sales.min(), "on", dates[np.argmin(sales)])
```

```
Total revenue: $ 1433697.25
Average daily sales: $ 2222.786434108527
Max sale: $ 4108.74 on 2024-03-29T00:00:00.000000000
Min sale: $ 0.0 on 2023-11-23T00:00:00.000000000
```

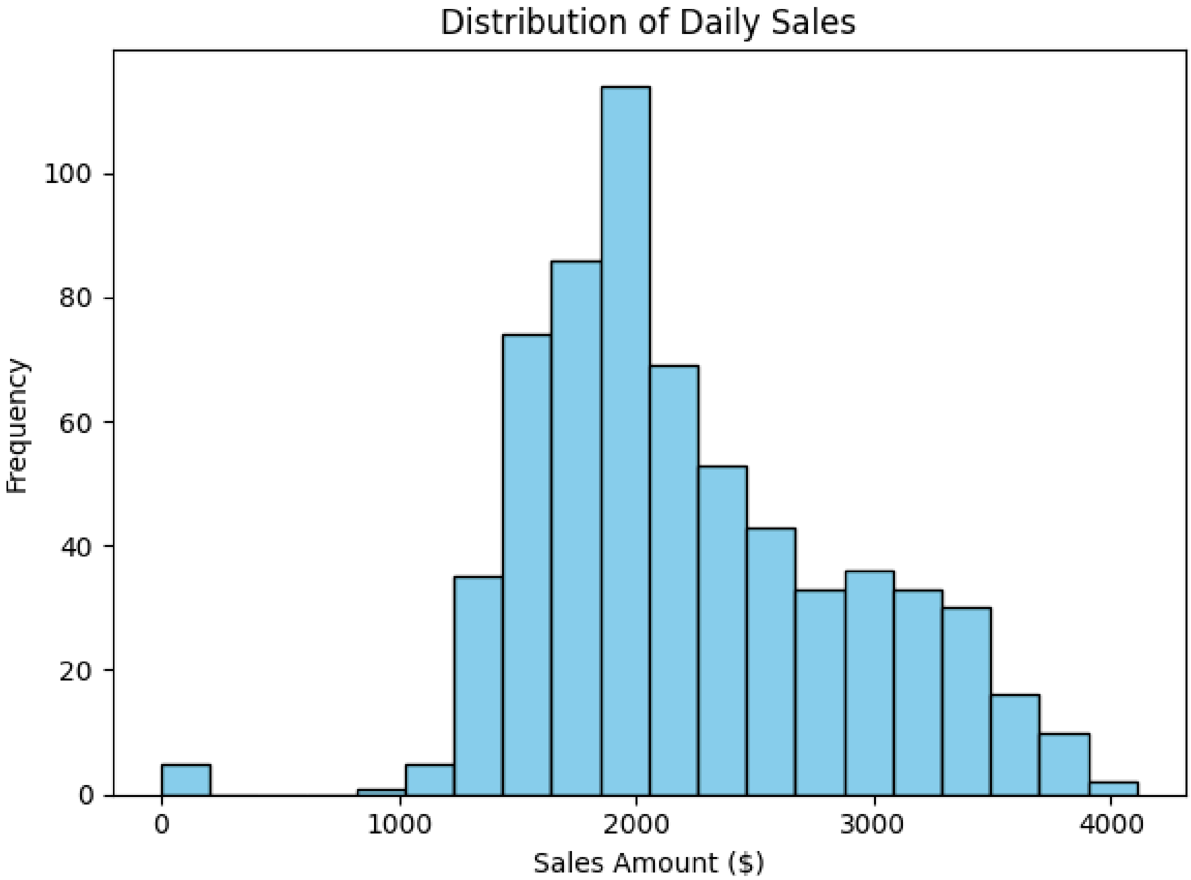
In [109…

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(data['date'], data['total_revenue'], label='Daily Sales')
plt.title("Daily Cash Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Cash Sales ($)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



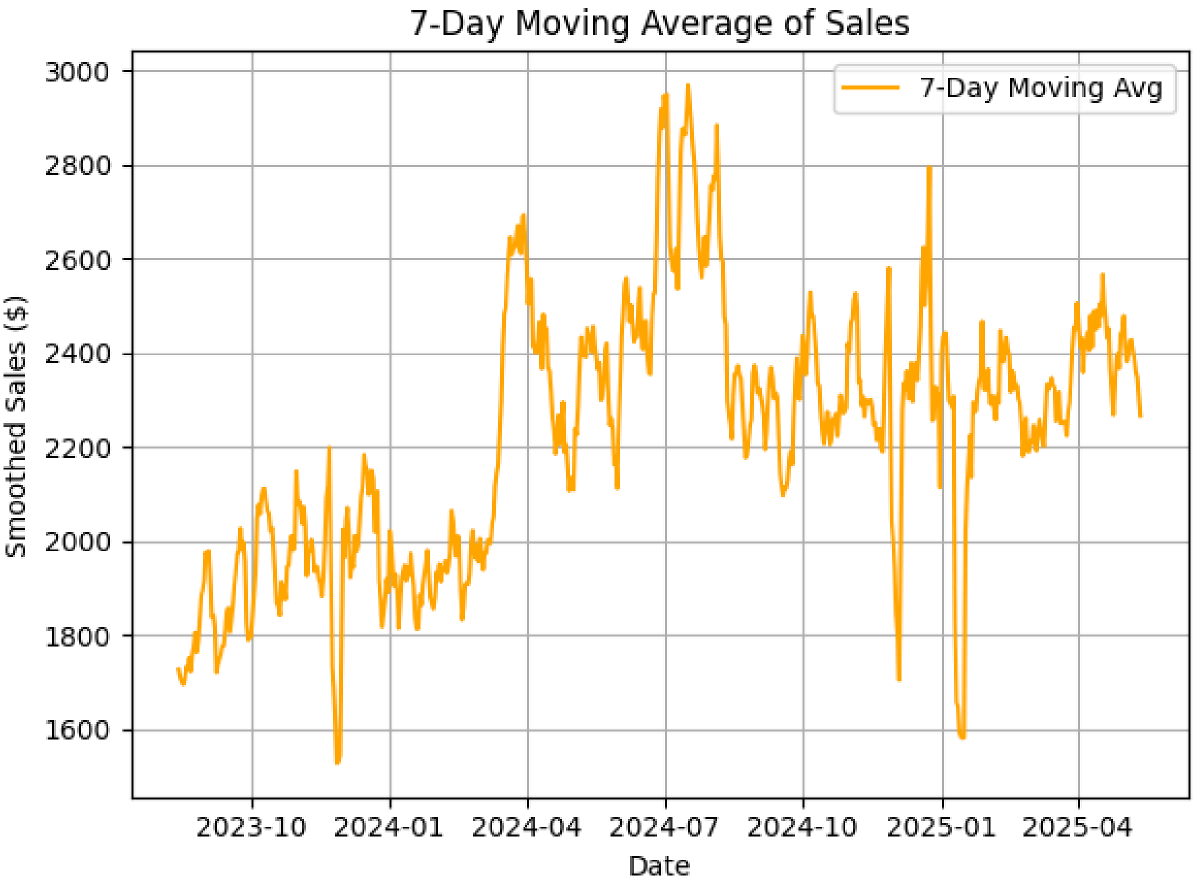
In [111…

```
plt.hist(sales, bins=20, color='skyblue', edgecolor='black')
plt.title("Distribution of Daily Sales")
plt.xlabel("Sales Amount ($)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



In [113…

```
moving_avg = np.convolve(sales, np.ones(7)/7, mode='valid')

plt.plot(data['date'][6:], moving_avg, color='orange', label='7-Day Moving A
plt.title("7-Day Moving Average of Sales")
plt.xlabel("Date")
plt.ylabel("Smoothed Sales ($)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



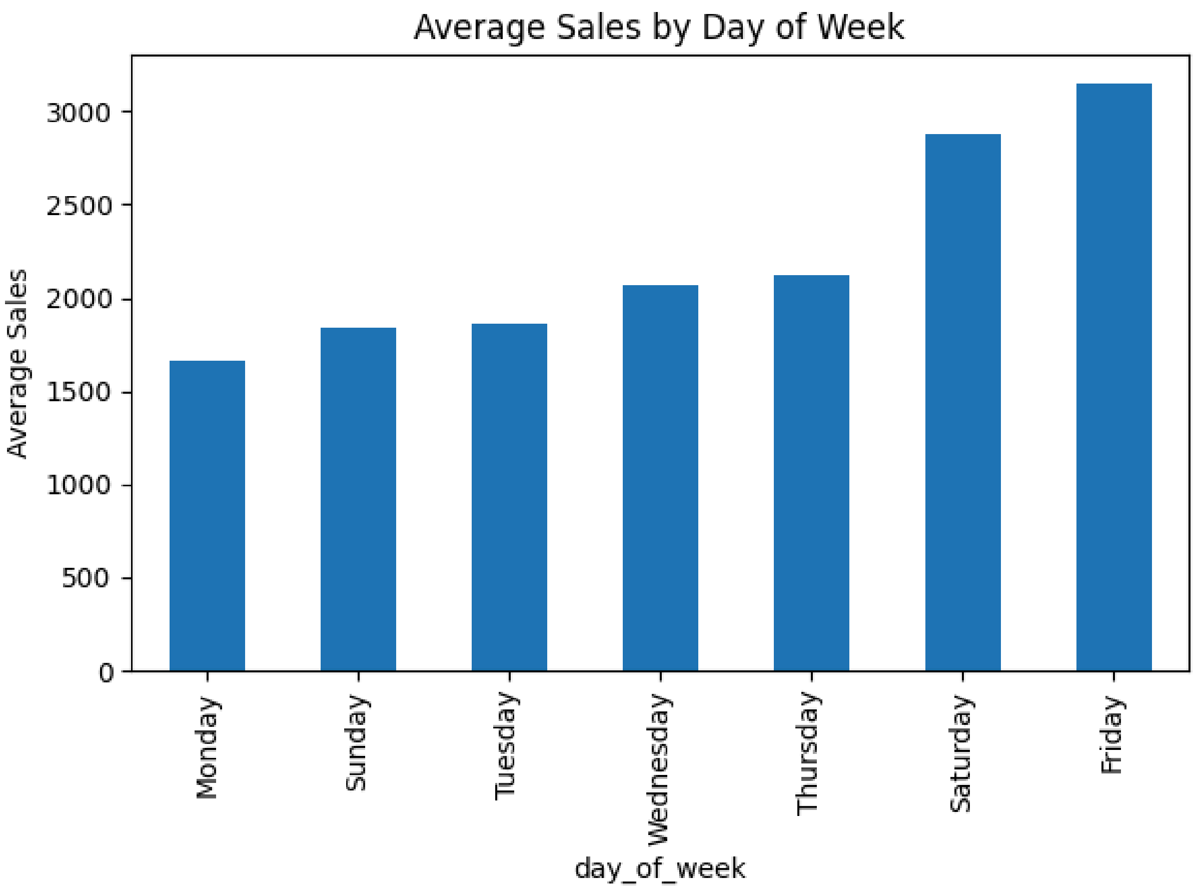
In [115…

```
top_days = data.sort_values(by='total_revenue', ascending=False).head(10)
print(top_days[['date', 'total_revenue']])
```

```
          date  total_revenue
410 2024-03-29        4108.74
396 2024-04-12        3911.43
283 2024-08-03        3890.77
319 2024-06-28        3882.43
375 2024-05-03        3875.77
424 2024-03-15        3822.61
368 2024-05-10        3787.33
270 2024-08-16        3768.02
221 2024-10-04        3766.21
305 2024-07-12        3761.20
```

In [117…

```
data['day_of_week'] = data['date'].dt.day_name()
avg_by_day = data.groupby('day_of_week')['total_revenue'].mean().sort_values

avg_by_day.plot(kind='bar', title="Average Sales by Day of Week")
plt.ylabel("Average Sales")
plt.tight_layout()
plt.show()
```



In [121…

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

# Grouped data
data['day_of_week'] = data['date'].dt.day_name()
data['week'] = data['date'].dt.isocalendar().week
data['month'] = data['date'].dt.to_period('M')
data['labor_per_revenue'] = data['labor_cost'] / data['total_revenue']
```

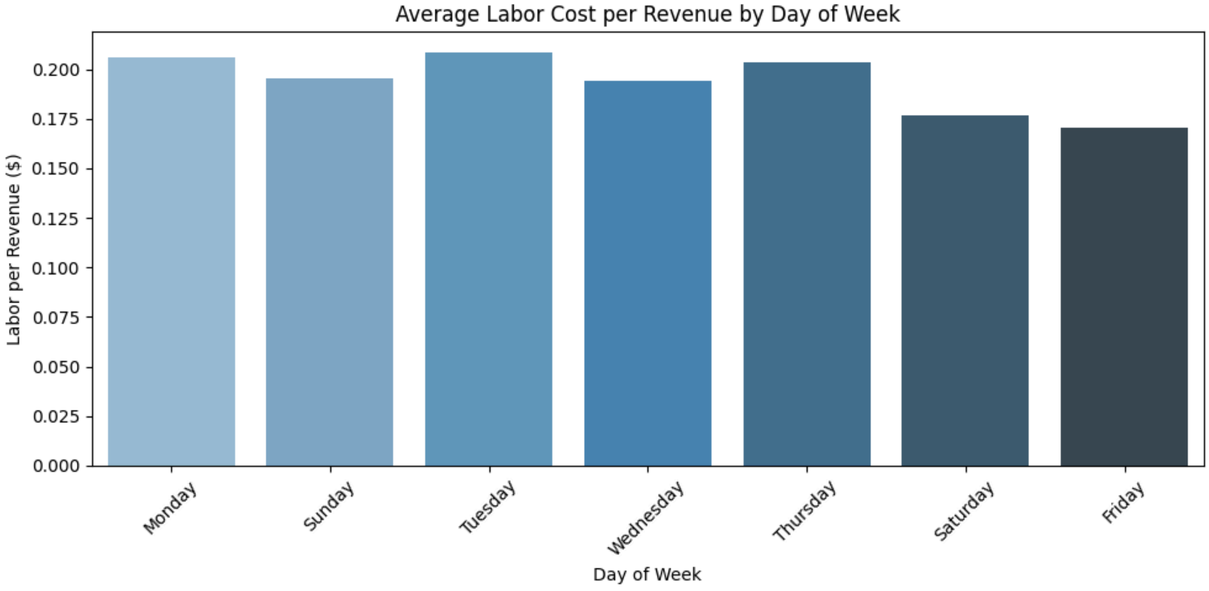
In [123…

```
day_avg = data.groupby('day_of_week')[['total_revenue', 'labor_cost', 'labor
week_avg = data.groupby('week')[['total_revenue', 'labor_cost', 'labor_per_r
month_avg = data.groupby('month')[['total_revenue', 'labor_cost', 'labor_per
```

In [125…

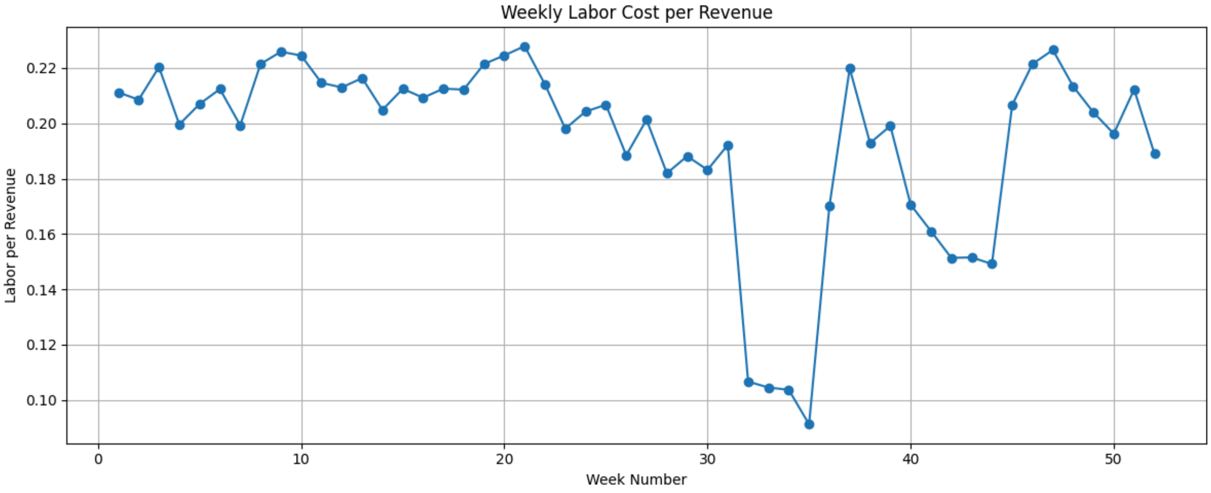
```
plt.figure(figsize=(10, 5))
sns.barplot(x=day_avg.index, y=day_avg['labor_per_revenue'], palette='Blues_
plt.title("Average Labor Cost per Revenue by Day of Week")
plt.ylabel("Labor per Revenue ($)")
plt.xlabel("Day of Week")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
/var/folders/md/yz4wns293812bb6m3pw1zpjm0000gn/T/ipykernel_17504/392813891.p
y:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.barplot(x=day_avg.index, y=day_avg['labor_per_revenue'], palette='Blue
s_d')
```



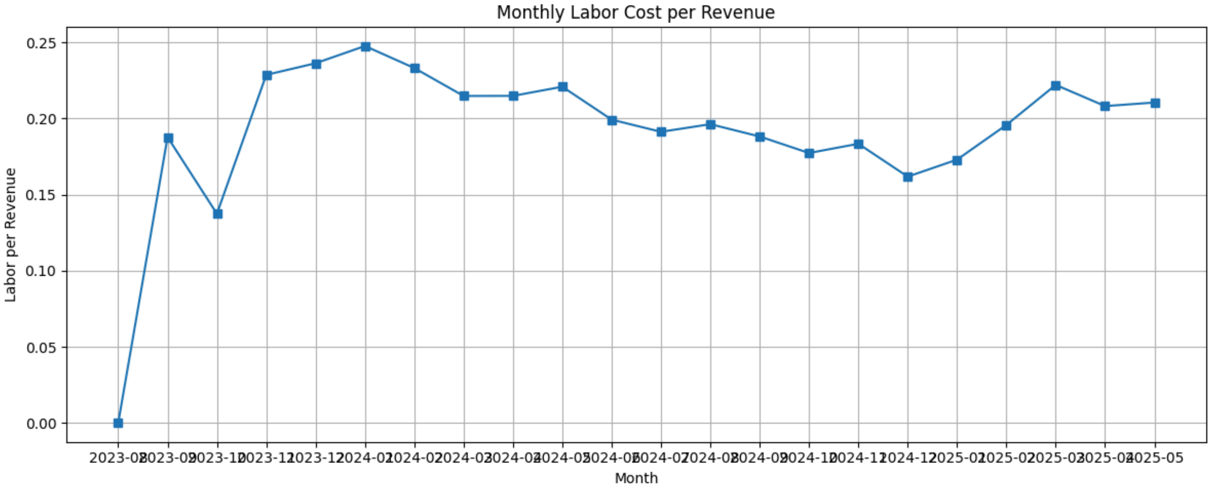
In [127…

```
plt.figure(figsize=(12, 5))
plt.plot(week_avg.index, week_avg['labor_per_revenue'], marker='o')
plt.title("Weekly Labor Cost per Revenue")
plt.xlabel("Week Number")
plt.ylabel("Labor per Revenue")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [129…

```
plt.figure(figsize=(12, 5))
plt.plot(month_avg.index.astype(str), month_avg['labor_per_revenue'], marker
plt.title("Monthly Labor Cost per Revenue")
plt.xlabel("Month")
plt.ylabel("Labor per Revenue")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [135…

```
import pandas as pd

# Make sure 'efficiency_ratio' exists
data['efficiency_ratio'] = data['total_revenue'] / data['labor_cost']

# Create a day of week column
data['day_of_week'] = data['date'].dt.day_name()

# Group by day and calculate average efficiency
day_efficiency = data.groupby('day_of_week')['efficiency_ratio'].mean().rese

# Classify each day
day_efficiency['category'] = day_efficiency['efficiency_ratio'].apply(
    lambda x: 'More Employees than Revenue' if x < 1 else 'More Revenue thar
)

# Sort for consistent weekday order
from pandas.api.types import CategoricalDtype
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Sa
day_efficiency['day_of_week'] = day_efficiency['day_of_week'].astype(Categor
day_efficiency = day_efficiency.sort_values('day_of_week')

print(day_efficiency)
```

```
  day_of_week  efficiency_ratio                     category
1      Monday               inf  More Revenue than Employees
5     Tuesday               inf  More Revenue than Employees
6   Wednesday               inf  More Revenue than Employees
4    Thursday               inf  More Revenue than Employees
0      Friday               inf  More Revenue than Employees
2    Saturday               inf  More Revenue than Employees
3      Sunday               inf  More Revenue than Employees
```

In [139…

```
import pandas as pd
```

```
# Ensure date column is in datetime format
data['date'] = pd.to_datetime(data['date'], errors='coerce')

# Extract year and month
data['year'] = data['date'].dt.year
data['month'] = data['date'].dt.month

# Group by year and month, summing revenue
monthly_revenue = data.groupby(['year', 'month'])['total_revenue'].sum().res

# Pivot the table so each year is a column, months are rows
monthly_pivot = monthly_revenue.pivot(index='month', columns='year', values=

# Compute year-over-year growth %
monthly_pivot['growth_23_to_24'] = ((monthly_pivot[2024] - monthly_pivot[202
monthly_pivot['growth_24_to_25'] = ((monthly_pivot[2025] - monthly_pivot[202

# Round values for presentation
monthly_pivot = monthly_pivot.round(2)

# Display the result
print(monthly_pivot)
```

```
year       2023      2024      2025  growth_23_to_24  growth_24_to_25
month
1           NaN  57295.18  66827.29              NaN            16.64
2           NaN  56717.97  64458.71              NaN            13.65
3           NaN  75526.00  71204.77              NaN            -5.72
4           NaN  67743.53  72477.54              NaN             6.99
5           NaN  73304.94  30456.61              NaN           -58.45
6           NaN  77405.65       NaN              NaN              NaN
7           NaN  83720.94       NaN              NaN              NaN
8      42397.56  75443.96       NaN            77.94              NaN
9      57001.42  66632.31       NaN            16.90              NaN
10     61404.85  71723.98       NaN            16.81              NaN
11     56636.08  68317.98       NaN            20.63              NaN
12     63949.58  73050.40       NaN            14.23              NaN
```

In [145…

```
non_zero_sales = sales[sales > 0]
min_sale = non_zero_sales.min()
max_sale = sales.max()

increase_pct = ((max_sale - min_sale) / min_sale) * 100
print(f"Sales increased by {increase_pct:.2f}% from the lowest (non-zero) to
```

```
Sales increased by 352.21% from the lowest (non-zero) to the highest day.
```

In [149…

```
import pandas as pd

# Load and prepare data
data = pd.read_csv("daily_sales_23_25.csv", header=None)
data.columns = [
    'period_no', 'week_no', 'date', 'store_no', 'store_name', 'total_revenue
    'projected_psa', 'last_week_revenue', 'last_week_pct_change',
    'last_year_revenue', 'last_year_pct_change', 'lunch_sales', 'lunch_pct',
    'labor_pct', 'scheduled_pct', 'labor_cost', 'scheduled_hours',
    'actual_hours', 'hours_difference', 'dpmh', 'customer_count',
    'avg_ticket', 'status'
]

# Clean data
data['date'] = pd.to_datetime(data['date'], errors='coerce')
data['total_revenue'] = pd.to_numeric(data['total_revenue'], errors='coerce'
data = data.dropna(subset=['date', 'total_revenue'])

# Add year and month
data['year'] = data['date'].dt.year
data['month'] = data['date'].dt.month

### PART 1: 2023 vs 2024 comparison (common months only)
months_2023 = set(data[data['year'] == 2023]['month'])
months_2024 = set(data[data['year'] == 2024]['month'])
common_23_24 = sorted(months_2023 & months_2024)

data_23_24 = data[data['month'].isin(common_23_24) & data['year'].isin([2023
summary_23_24 = data_23_24.groupby('year')['total_revenue'].agg(['mean', 'su
summary_23_24['Increase %'] = summary_23_24['mean'].pct_change().fillna(0) *
summary_23_24['Increase %'] = summary_23_24['Increase %'].round(2)
summary_23_24.columns = ['Year', 'Average Sale', 'Total Revenue', 'Increase

print("2023 vs 2024 Comparison (Same Months):")
print(summary_23_24)

### PART 2: 2024 vs 2025 comparison (up to latest month in 2025)
latest_month_2025 = data[data['year'] == 2025]['month'].max()
data_24_25 = data[(data['month'] <= latest_month_2025) & data['year'].isin([
summary_24_25 = data_24_25.groupby('year')['total_revenue'].agg(['mean', 'su
summary_24_25['Increase %'] = summary_24_25['mean'].pct_change().fillna(0) *
summary_24_25['Increase %'] = summary_24_25['Increase %'].round(2)
summary_24_25.columns = ['Year', 'Average Sale', 'Total Revenue', 'Increase

print("\n2024 vs 2025 Comparison (Up to Month", latest_month_2025, "):")
print(summary_24_25)
```

```
2023 vs 2024 Comparison (Same Months):
   Year  Average Sale  Total Revenue  Increase %
0  2023       1927.33      281389.49        0.00
1  2024       2321.36      355168.63       20.44

2024 vs 2025 Comparison (Up to Month 5 ):
   Year  Average Sale  Total Revenue  Increase %
0  2024       2174.92      330587.62        0.00
1  2025       2279.48      305449.88        4.81
```

In [ ]: