```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("train.csv", parse_dates=["date"])

print(df.shape)
df.head()
```

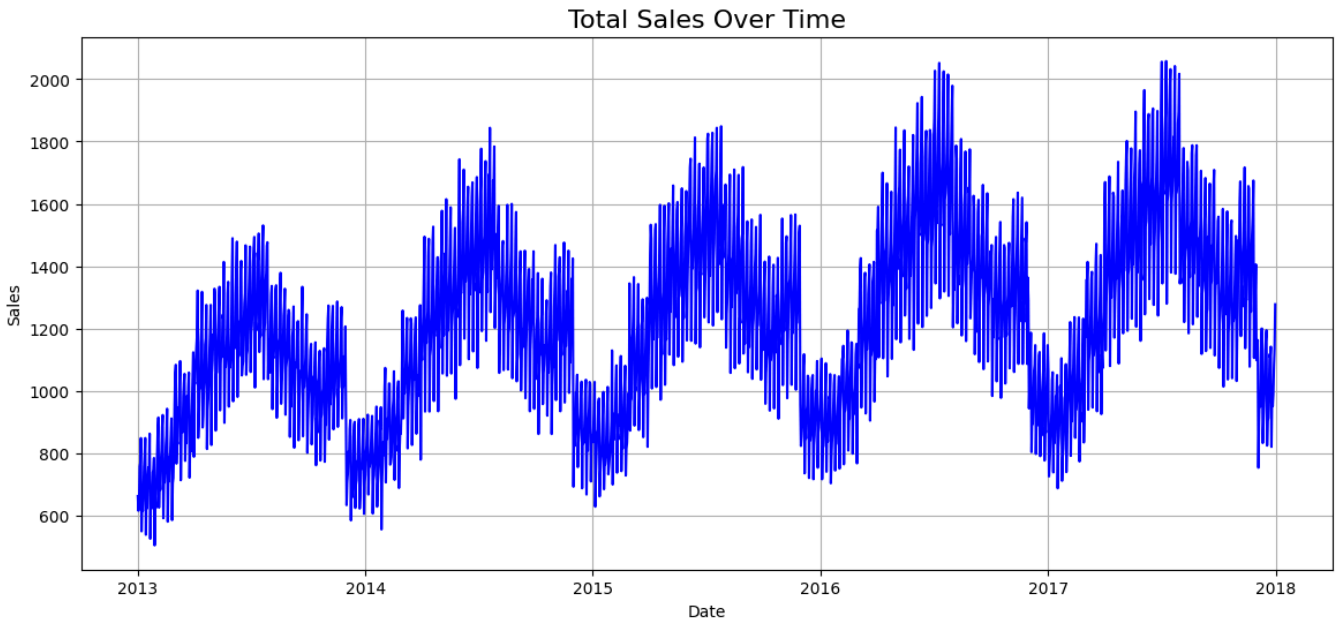
```
(57967, 4)
```

	date	store	item	sales
0	2013-01-01	1	1	13
1	2013-01-02	1	1	11
2	2013-01-03	1	1	14
3	2013-01-04	1	1	13
4	2013-01-05	1	1	10

Next steps: Generate code with df | View recommended plots | New interactive sheet

Ploting daily sales

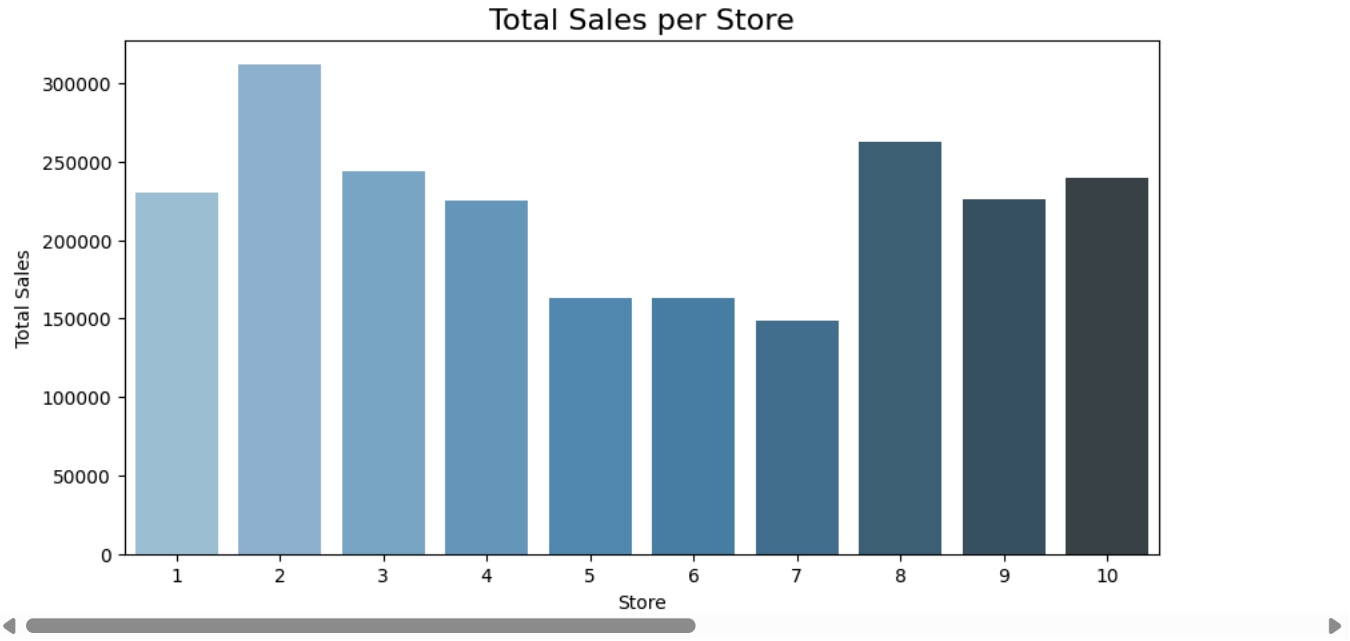
```
daily_sales = df.groupby("date")["sales"].sum()

plt.figure(figsize=(14, 6))
plt.plot(daily_sales, color='blue')
plt.title("Total Sales Over Time", fontsize=16)
plt.xlabel("Date")
plt.ylabel("Sales")
plt.grid(True)
plt.show()
```



Total sales per store

```
store_sales = df.groupby("store")["sales"].sum().sort_values()

plt.figure(figsize=(10, 5))
sns.barplot(x=store_sales.index, y=store_sales.values, palette="Blues_d")
plt.title("Total Sales per Store", fontsize=16)
plt.xlabel("Store")
plt.ylabel("Total Sales")
plt.show()
```

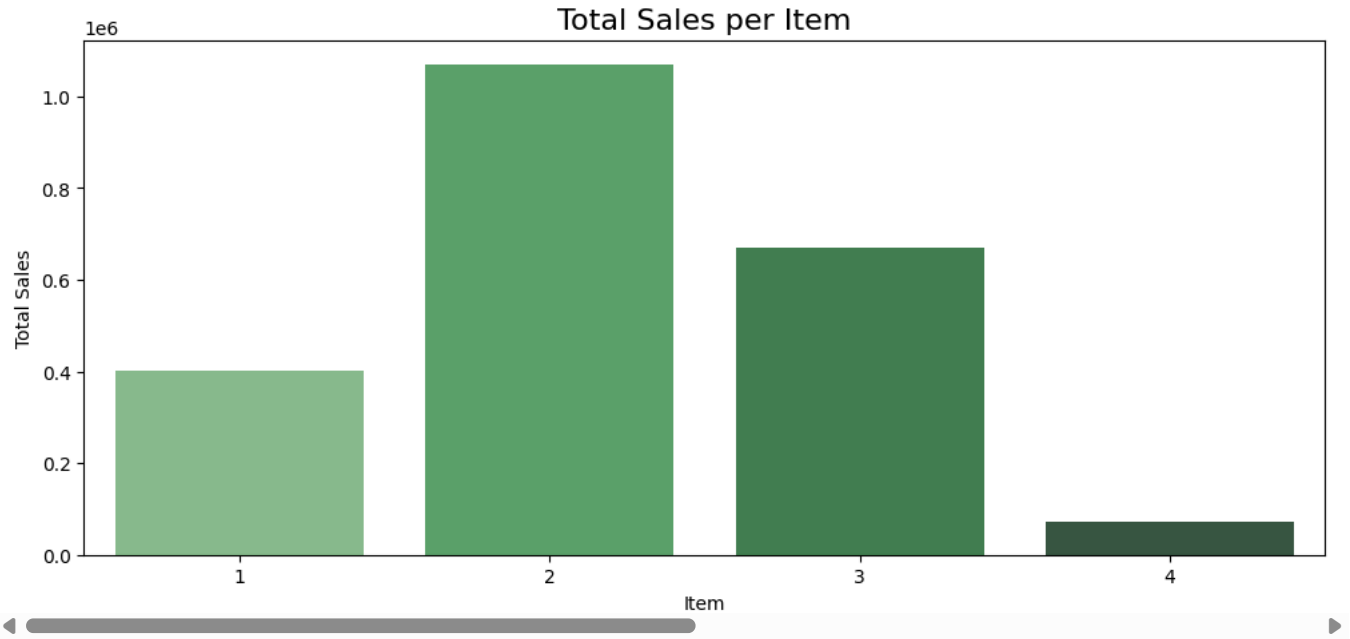
```
<ipython-input-10-878314555>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=store_sales.index, y=store_sales.values, palette="Blues_d")
```



Total sales per item

```
item_sales = df.groupby("item")["sales"].sum().sort_values(ascending=False)

plt.figure(figsize=(12, 5))
sns.barplot(x=item_sales.index, y=item_sales.values, palette="Greens_d")
plt.title("Total Sales per Item", fontsize=16)
plt.xlabel("Item")
plt.ylabel("Total Sales")
plt.show()
```

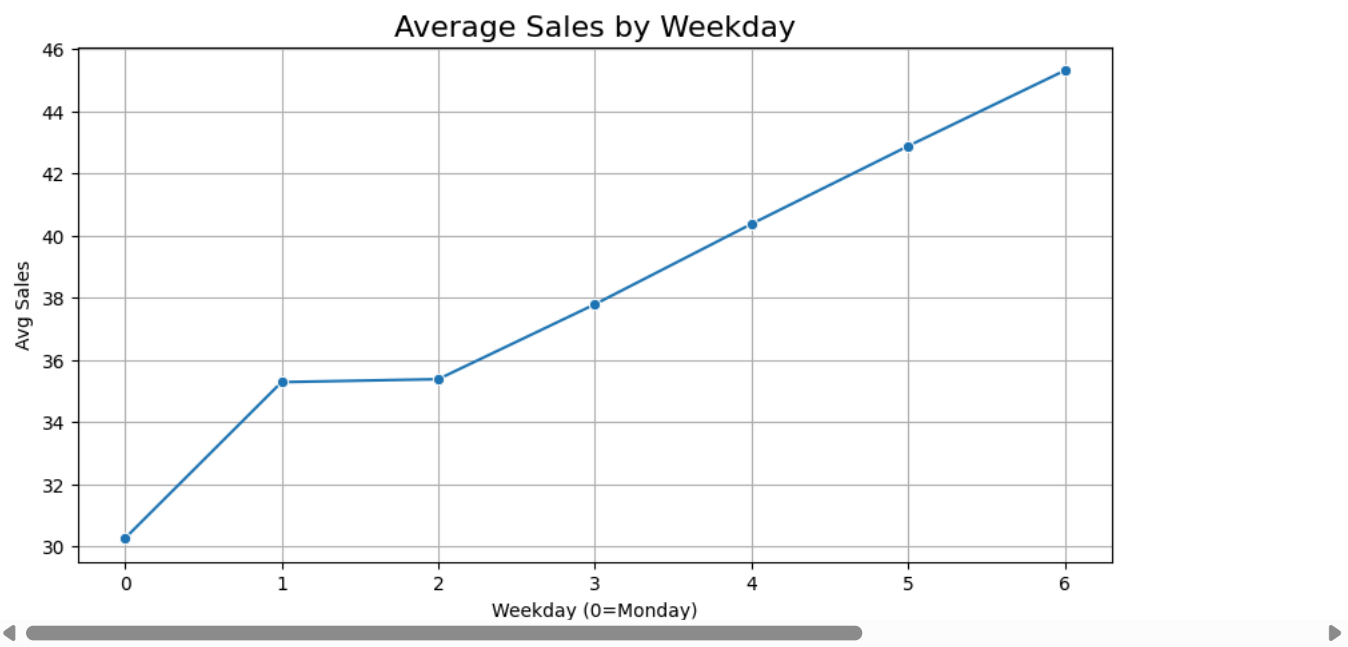
```
<ipython-input-11-2214962208>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=item_sales.index, y=item_sales.values, palette="Greens_d")
```



Average sales by weekday

```
df["year"] = df["date"].dt.year
df["month"] = df["date"].dt.month
df["day"] = df["date"].dt.day
df["weekday"] = df["date"].dt.dayofweek

weekday_avg = df.groupby("weekday")["sales"].mean()
```

```
weekday_avg = df.groupby("weekday")["sales"].mean()

plt.figure(figsize=(10, 5))
sns.lineplot(x=weekday_avg.index, y=weekday_avg.values, marker="o")
plt.title("Average Sales by Weekday", fontsize=16)
plt.xlabel("Weekday (0=Monday)")
plt.ylabel("Avg Sales")
plt.grid(True)
plt.show()
```



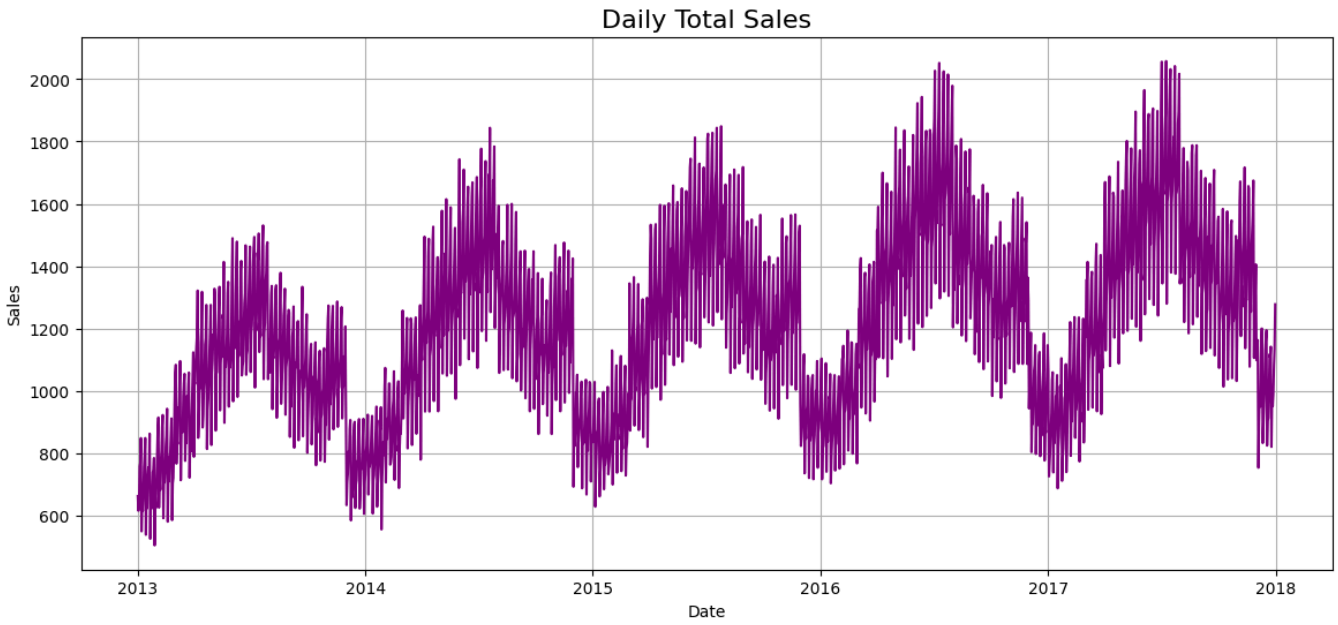

```
daily_sales = df.groupby("date")["sales"].sum().reset_index()

daily_sales.columns = ["ds", "y"]

daily_sales.head()
```

	ds	y
0	2013-01-01	663
1	2013-01-02	616
2	2013-01-03	644
3	2013-01-04	758
4	2013-01-05	769

Next steps: Generate code with daily_sales | View recommended plots | New interactive sheet

Visualizing Total Sales

```
plt.figure(figsize=(14, 6))
plt.plot(daily_sales["ds"], daily_sales["y"], color="purple")
plt.title("Daily Total Sales", fontsize=16)
plt.xlabel("Date")
plt.ylabel("Sales")
plt.grid(True)
plt.show()
```



Extracting Day, Month, Weekday

```
daily_sales["day"] = daily_sales["ds"].dt.day
daily_sales["month"] = daily_sales["ds"].dt.month
daily_sales["weekday"] = daily_sales["ds"].dt.dayofweek
```

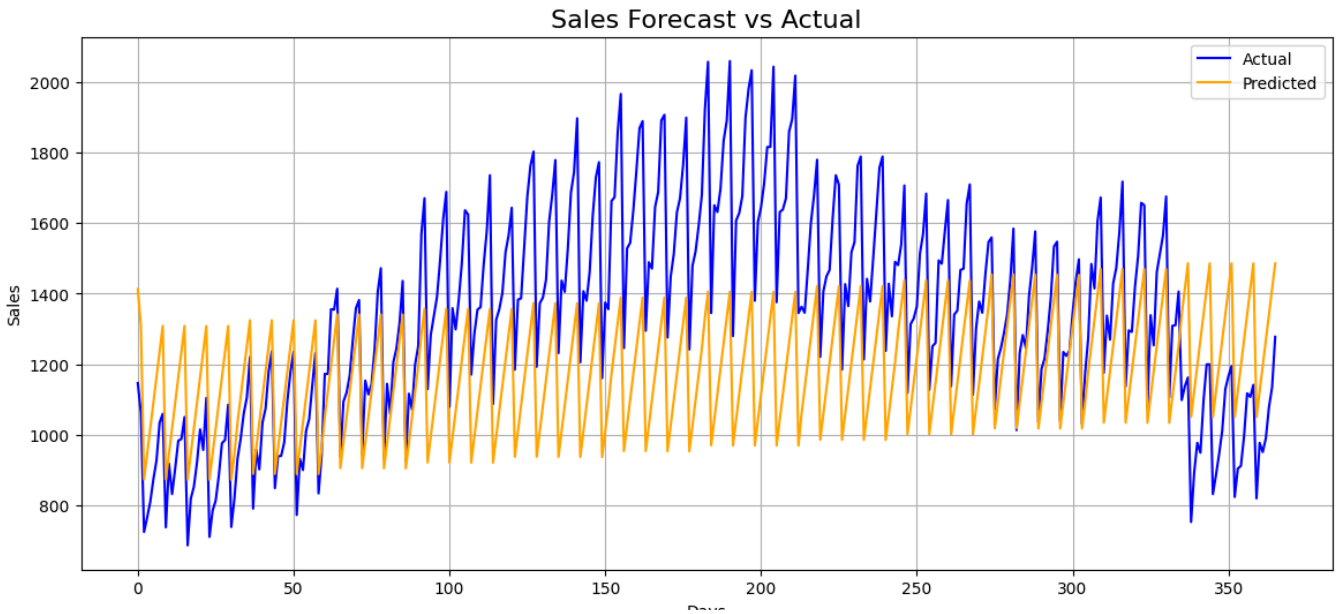
Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X = daily_sales[["day", "month", "weekday"]]
y = daily_sales["y"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 85268.28495819641
```

Plot Prediction

```
plt.figure(figsize=(14, 6))
plt.plot(y_test.values, label="Actual", color="blue")
plt.plot(y_pred, label="Predicted", color="orange")
plt.title("Sales Forecast vs Actual", fontsize=16)
plt.xlabel("Days")
plt.ylabel("Sales")
plt.legend()
plt.grid(True)
plt.show()
```


Sales Forecast vs Actual
Actual
Predicted
Sales
2000
1800
1600
1400
1200
1000
800
0
50
100
150
200
250
300
350