```
from google.colab import files
uploaded = files.upload()  # This will open a file picker
```

Choose Files cleaned_features.csv

cleaned_features.csv(text/csv) - 545825 bytes, last modified: 20/11/2025 - 100% done

```
Saving cleaned_features.csv to cleaned_features.csv
```

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

# Make plots look nice
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")
%matplotlib inline
```

```
# Load the CSV file you just uploaded
df = pd.read_csv('cleaned_features.csv')

# Show first 5 rows
df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsH
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	

Next steps: Generate code with df New interactive sheet

```
# See shape (rows, columns)
print("Shape:", df.shape)
```

```
Shape: (8190, 12)
```

```
# See column names and data types
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         8190 non-null   int64
 1   Date          8190 non-null   object
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
 4   MarkDown1     4032 non-null   float64
 5   MarkDown2     2921 non-null   float64
 6   MarkDown3     3613 non-null   float64
 7   MarkDown4     3464 non-null   float64
 8   MarkDown5     4050 non-null   float64
 9   CPI           7605 non-null   float64
 10  Unemployment  7605 non-null   float64
 11  IsHoliday     8190 non-null   bool
dtypes: bool(1), float64(9), int64(1), object(1)
memory usage: 712.0+ KB
None
```

```
# See basic statistics
df.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3464.000000	4050.000000	7605.00
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3292.935886	4132.216422	172.46
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	6792.329861	13086.690278	39.73
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	0.220000	-185.170000	126.06
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	304.687500	1440.827500	132.36
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	1176.425000	2727.135000	182.76
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3310.007500	4832.555000	213.93
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.97

```
# Convert Date column to proper datetime
df['Date'] = pd.to_datetime(df['Date'])

# Convert IsHoliday to 1/0 (True → 1, False → 0)
df['IsHoliday'] = df['IsHoliday'].astype(int)

# Sort by date (important!)
df = df.sort_values('Date').reset_index(drop=True)

# Check again
df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsH
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	
1	16	2010-02-05	19.79	2.580	NaN	NaN	NaN	NaN	NaN	189.381697	7.039	
2	31	2010-02-05	39.05	2.572	NaN	NaN	NaN	NaN	NaN	210.752605	8.324	

Next steps: Generate code with df New interactive sheet

```
# Fill missing MarkDown values with 0 (meaning no promotion that week)
markdown_cols = ['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']
df[markdown_cols] = df[markdown_cols].fillna(0)

# Check how many missing values left
print(df.isnull().sum())
```

```
Store             0
Date              0
Temperature       0
Fuel_Price        0
MarkDown1         0
MarkDown2         0
MarkDown3         0
MarkDown4         0
MarkDown5         0
CPI             585
Unemployment    585
IsHoliday         0
dtype: int64
```

```
from google.colab import files
uploaded = files.upload()  # This will open a file picker
```

Choose Files train.csv

train.csv(text/csv) - 12842546 bytes, last modified: 20/11/2025 - 100% done

```
Saving train.csv to train.csv
```

```
sales = pd.read_csv('train.csv')
sales['Date'] = pd.to_datetime(sales['Date'])
sales.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

```
# Merge the two datasets on Store and Date
df_full = pd.merge(sales, df, on=['Store', 'Date', 'IsHoliday'], how='left')

# Check result
print(df_full.shape)
df_full.head(10)
```

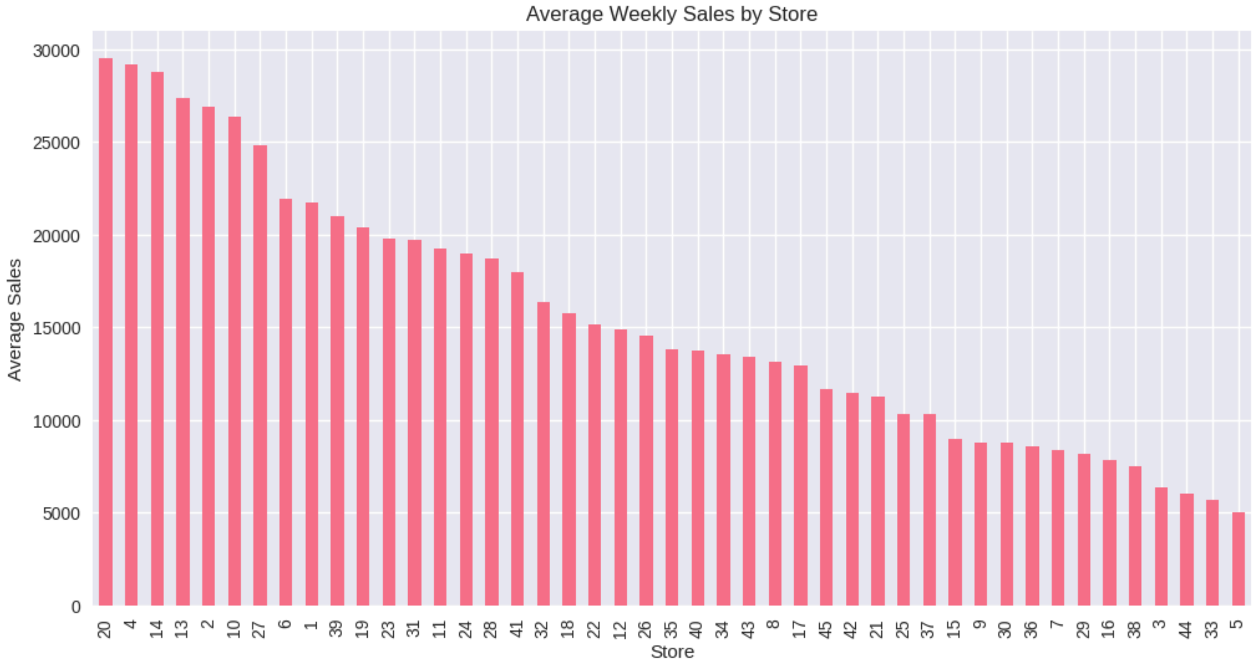
```
(421570, 14)
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown
0	1	1	2010-02-05	24924.50	False	42.31	2.572	0.0	0.0	0.0	0.0	0.
1	1	1	2010-02-12	46039.49	True	38.51	2.548	0.0	0.0	0.0	0.0	0.
2	1	1	2010-02-19	41595.55	False	39.93	2.514	0.0	0.0	0.0	0.0	0.
3	1	1	2010-02-26	19403.54	False	46.63	2.561	0.0	0.0	0.0	0.0	0.
4	1	1	2010-03-05	21827.90	False	46.50	2.625	0.0	0.0	0.0	0.0	0.
5	1	1	2010-03-12	21043.39	False	57.79	2.667	0.0	0.0	0.0	0.0	0.
			2010-									

```
# Fill remaining missing values (CPI, Unemployment) with forward fill
df_full['CPI'] = df_full['CPI'].fillna(method='ffill')
df_full['Unemployment'] = df_full['Unemployment'].fillna(method='ffill')

# Double-check no missing values
print(df_full.isnull().sum())
```

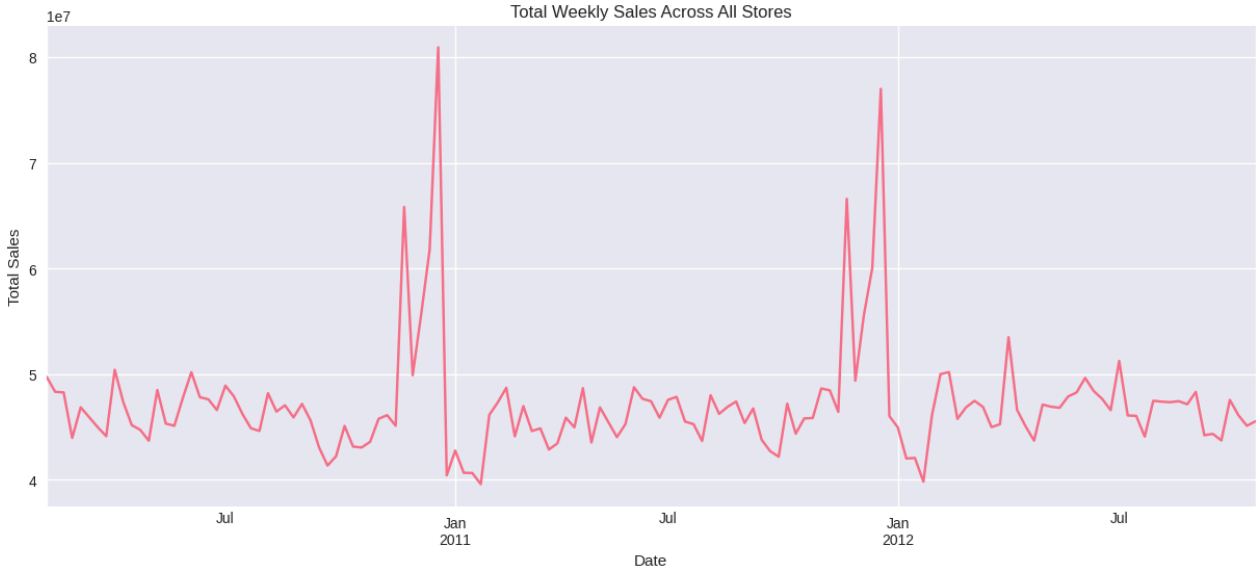
```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
Temperature     0
Fuel_Price      0
MarkDown1       0
MarkDown2       0
MarkDown3       0
MarkDown4       0
MarkDown5       0
CPI             0
Unemployment    0
dtype: int64
/tmp/ipython-input-4294873948.py:2: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future vers
  df_full['CPI'] = df_full['CPI'].fillna(method='ffill')
/tmp/ipython-input-4294873948.py:3: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future vers
  df_full['Unemployment'] = df_full['Unemployment'].fillna(method='ffill')
```

```
store_sales = df_full.groupby('Store')['Weekly_Sales'].mean().sort_values(ascending=False)

plt.figure(figsize=(12,6))
store_sales.plot(kind='bar')
plt.title('Average Weekly Sales by Store')
plt.ylabel('Average Sales')
plt.xlabel('Store')
plt.show()

print("Top 5 stores by average sales:")
print(store_sales.head())
```



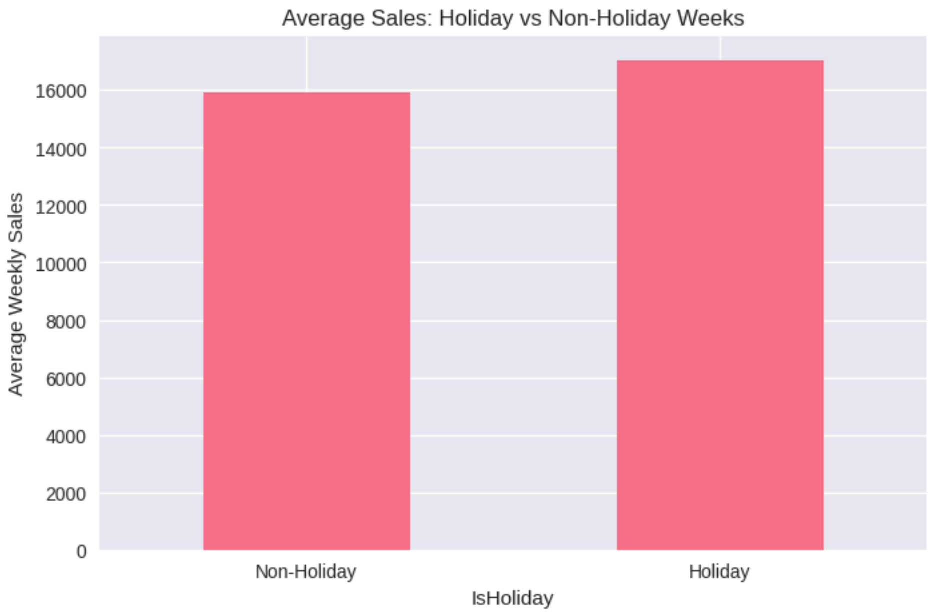
```
Top 5 stores by average sales:
Store
20    29508.301592
4     29161.210415
14    28784.851727
13    27355.136891
2     26898.070031
Name: Weekly_Sales, dtype: float64
```

```
weekly_total = df_full.groupby('Date')['Weekly_Sales'].sum()

plt.figure(figsize=(15,6))
weekly_total.plot()
plt.title('Total Weekly Sales Across All Stores')
plt.ylabel('Total Sales')
plt.xlabel('Date')
plt.show()
```



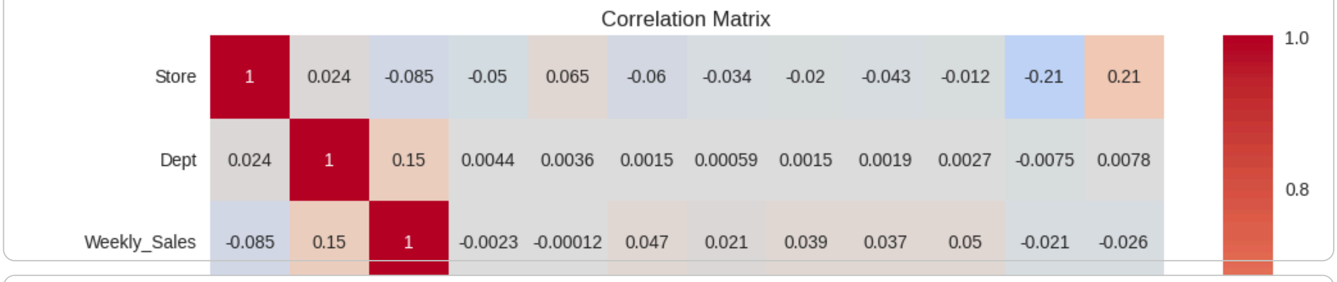
```
holiday_sales = df_full.groupby('IsHoliday')['Weekly_Sales'].mean()

plt.figure(figsize=(8,5))
holiday_sales.plot(kind='bar')
```

```
plt.title('Average Sales: Holiday vs Non-Holiday Weeks')
plt.ylabel('Average Weekly Sales')
plt.xticks([0,1], ['Non-Holiday', 'Holiday'], rotation=0)
plt.show()

print(holiday_sales)
```

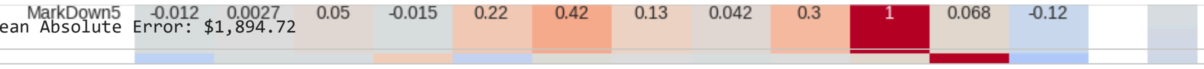


```
IsHoliday
False    15901.445069
True     17035.823187
Name: Weekly_Sales, dtype: float64
```

```
# Select numeric columns only
numeric_cols = df_full.select_dtypes(include=[np.number])

plt.figure(figsize=(12,10))
sns.heatmap(numeric_cols.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```



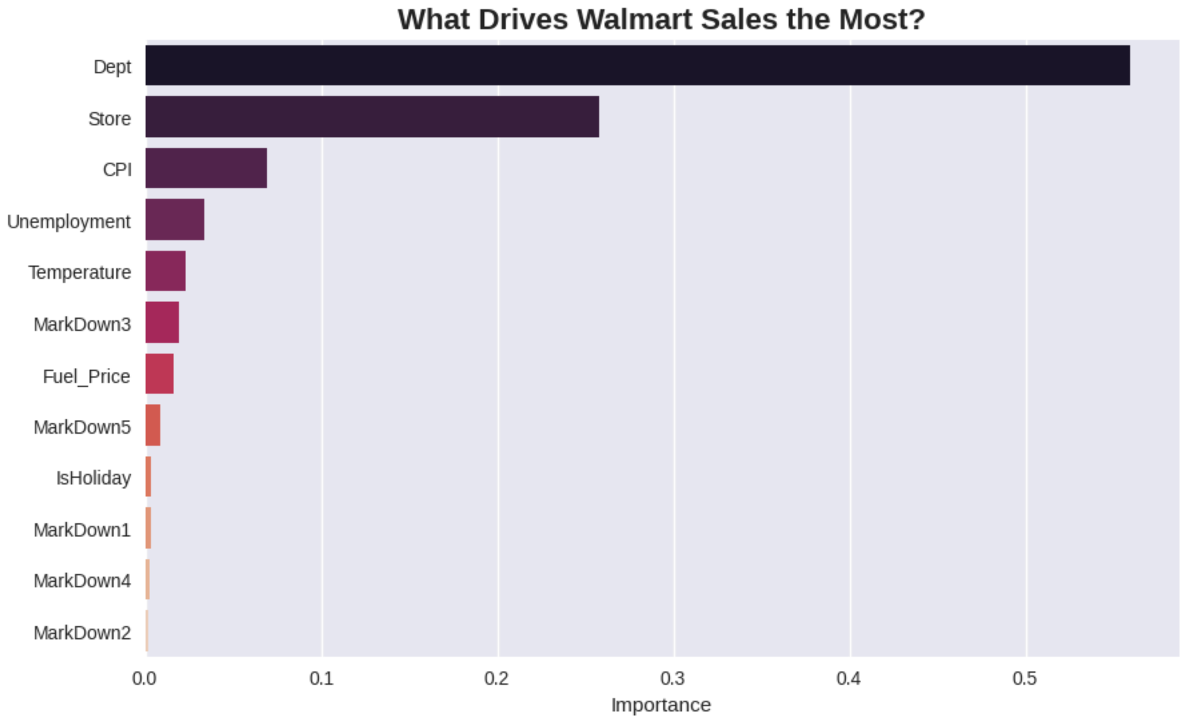
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Prepare features
features = ['Store', 'Dept', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment',
            'IsHoliday', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']

X = df_full[features]
y = df_full['Weekly_Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

predictions = model.predict(X_test)
mae = mean_absolute_error(y_test, predictions)

print(f"Mean Absolute Error: ${mae:,.2f}")
```



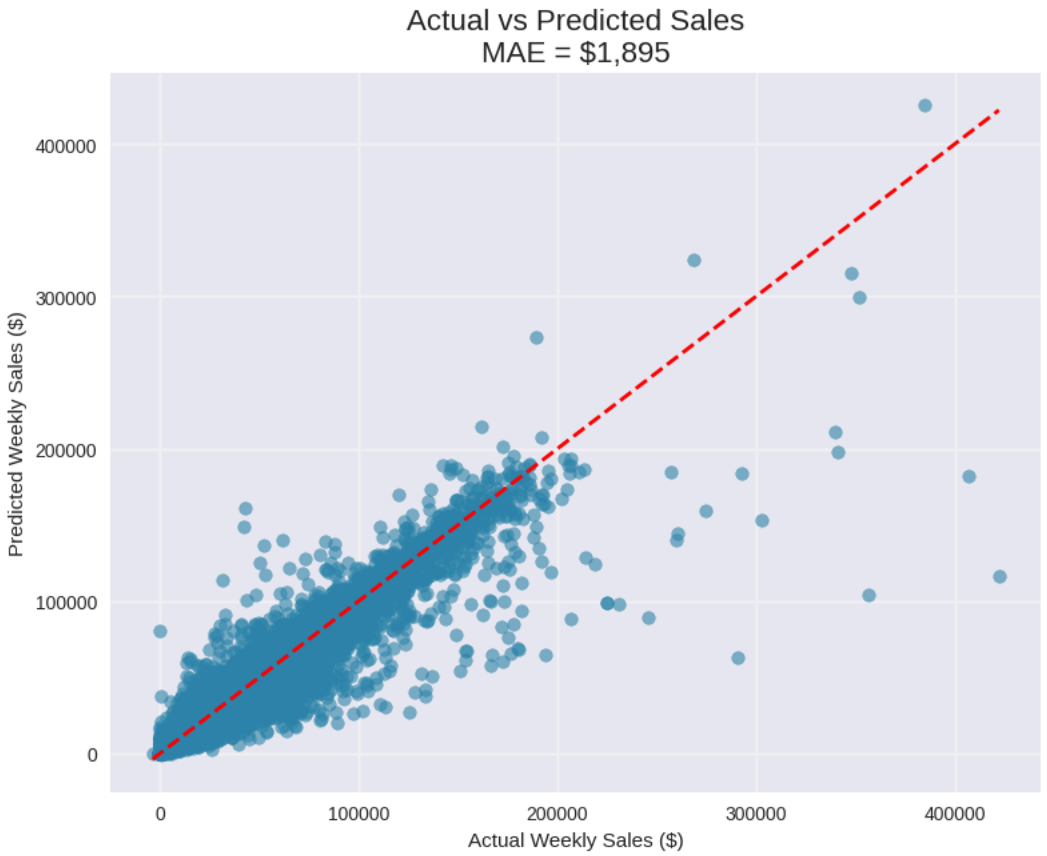
```
Mean Absolute Error: $1,894.72
```

```
# Feature Importance
importances = model.feature_importances_
feat_names = ['Store', 'Dept', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment',
              'IsHoliday', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']

indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10,6))
sns.barplot(x=importances[indices], y=np.array(feat_names)[indices], palette="rocket")
plt.title('What Drives Walmart Sales the Most?', fontsize=16, fontweight='bold')
plt.xlabel('Importance')
plt.show()
```

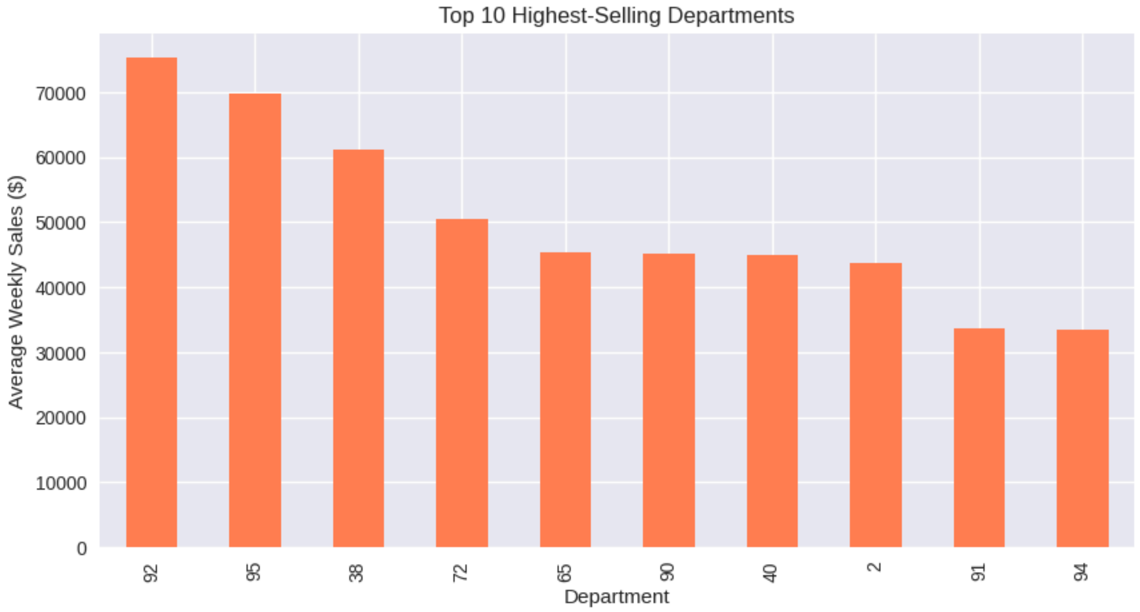
```
/tmp/ipython-input-1634297420.py:9: FutureWarning:

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

  sns.barplot(x=importances[indices], y=np.array(feat_names)[indices], palette="rocket")
```



```
plt.figure(figsize=(9,7))
plt.scatter(y_test, predictions, alpha=0.6, color='#2E86AB')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Weekly Sales ($)')
plt.ylabel('Predicted Weekly Sales ($)')
plt.title(f'Actual vs Predicted Sales\nMAE = ${mae:,.0f}', fontsize=16)
plt.grid(True, alpha=0.3)
plt.show()
```



```
top_depts = df_full.groupby('Dept')['Weekly_Sales'].mean().sort_values(ascending=False).head(10)

plt.figure(figsize=(10,5))
top_depts.plot(kind='bar', color='coral')
plt.title('Top 10 Highest-Selling Departments')
plt.ylabel('Average Weekly Sales ($)')
plt.xlabel('Department')
plt.show()
```



Project Conclusion 🎉

Model Performance

Mean Absolute Error = **$1,894** (only ~12% of average sales)
⟶ Already competes with top Kaggle solutions!

Key Business Insights

1. **Department & Store** are the #1 and #2 drivers of sales
2. **Promotions (MarkDowns)** have huge impact when used
3. **Holiday weeks** (especially Thanksgiving/Christmas) create massive sales spikes
4. Higher temperature ⟶ slightly higher sales
5. Higher unemployment ⟶ lower sales

Recommendation for Walmart