Colorado Motor Vehicle Sales at County Level Analysis

Informatics Practices (IP) Project

Project Title: Time Series Analysis and Forecasting of Colorado Motor Vehicle Sales at County Level

Data Source: Colorado Motor Vehicle Sales dataset (quarterly county-level sales in USD, provided as colorado_motor_vehicle_sales.csv)

Submitted By:

Adarsh N

TABLE OF CONTENTS

1. Introduction

This project analyzes historical motor vehicle sales in Colorado at the county and quarterly level. The goal is to understand temporal trends, regional patterns, and potential risks in the automotive supply chain using data-driven methods.

2. Project Objectives

- Identify yearly and quarterly trends in Colorado motor vehicle sales.
- Compare sales performance between major counties.
- Examine seasonality and growth patterns over time.
- Build simple predictive models for future sales.
- Discuss supply chain risks and practical applications of the insights.

3. Scope of the Project

The analysis focuses on Colorado motor vehicle sales aggregated by year, quarter, and county. It covers descriptive statistics, visualization, and basic forecasting but does not model individual consumer behavior or vehicle types.

4. Tools & Technologies Used

- Programming language: Python (Jupyter Notebook / any IDE).
- Data analysis: pandas, NumPy.
- Visualization: Matplotlib, Seaborn.

- Machine learning: scikit-learn (LinearRegression, metrics).

5. System Requirements

- Operating System: Windows / macOS / Linux.
- Python: version 3.10+ (you used Python 3.13).
- RAM: at least 4 GB (8 GB recommended).
- Installed libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, jupyter (optional).

6. Dataset Description

The dataset contains quarterly motor vehicle sales at the county level for Colorado.

- File name: colorado_motor_vehicle_sales.csv.
- Main columns:
    - year: Calendar year (e.g., 2008, 2009, …).
    - quarter: Quarter number (1–4).
    - county: County or aggregated region name (e.g., Adams, Denver, Rest of State).
    - sales: Total motor vehicle sales amount in dollars for that county and quarter.

You also engineered:

- year_quarter: Combined period label (e.g., 2010Q1).
- time_index: Integer index counting quarters from the start of the series.

7. Data Analysis Methodology

- Data loading and cleaning: Import CSV, standardize column names, ensure correct data types, and handle missing values.
- Feature engineering: Create year_quarter and time_index for time-series analysis.
- Aggregation: Compute state-level yearly and quarterly totals, and total sales by county.
- Exploratory data analysis: Use groupby operations and visualizations to study trends, seasonality, and top-performing counties.
- Modeling: Fit simple linear regression models on aggregated quarterly sales to capture trend and generate forecasts.

8. Python Libraries Used

- pandas for data loading, cleaning, and aggregation.
- numpy for numerical operations (e.g., computing RMSE).
- matplotlib.pyplot and seaborn for line and bar charts.
- sklearn.linear_model.LinearRegression for regression modeling.
- sklearn.metrics (mean_squared_error, r2_score) for error and performance evaluation.

9. Code Implementation

The implementation is structured into clear sections:

1. Setup and data loading (read_csv, initial head, info, describe).

2. Data cleaning (lowercasing column names, type casting, handling missing values).
3. Feature engineering (creating year_quarter, time_index).
4. Aggregations (yearly totals, quarterly totals, county totals).
5. Time-series plots for state and key counties.
6. Linear regression models for state-wide quarterly sales and for a selected county.

(Full source code is included in Section 18.)

10. Analysis & Outputs

Key analytical steps and outputs:

- Yearly state-wide sales table showing total sales per year.
- Quarterly aggregated sales table showing how sales vary across quarters.
- Ranked list of counties by total sales, highlighting top-performing counties like Adams, Arapahoe, Denver, and El Paso over the period.
- Year-over-year growth percentages for each county, revealing periods of expansion and decline.
- Model performance metrics such as MSE, RMSE, and R^2 for both state-wide and county-level models.

11. Graphical Representations

You can include the following plots in the report:

- Line chart: Total Colorado motor vehicle sales by year.

- Bar chart: Total sales by quarter (Q1–Q4) across all years to show seasonality.
- Horizontal bar chart: Top 10 counties by cumulative sales.
- Multi-line chart: Quarterly sales over time for top counties (e.g., Denver, Arapahoe, Adams, Jefferson, El Paso).
- Bar chart: Year-over-year growth for a specific county such as Denver.

Each figure should have a clear title, labeled axes, and a brief caption interpreting the pattern.

12. Supply Chain Risk Assessment

Motor vehicle sales depend on a multi-stage supply chain: component suppliers, vehicle manufacturers, logistics, dealerships, and local demand. Sales fluctuations by quarter and year can signal different types of risks.

- Demand risk: Sharp declines in sales for specific years or quarters may indicate economic downturns, reduced consumer confidence, or credit tightening that can leave inventory unsold.
- Geographic risk: Strong concentration of sales in a few counties (e.g., large urban areas) exposes manufacturers and dealers to localized demand or regulatory shocks (policy changes, environmental regulations).
- Seasonality risk: If sales peak in specific quarters, supply chain operations must handle capacity surges; poor planning can lead to stock-outs or excessive inventory.
- Disruption sensitivity: Sudden drops or spikes in particular years might be linked to external events (e.g., macroeconomic

crises); similar future events could again disrupt production and distribution.

Using the time-series models and historical variability, stakeholders can stress-test scenarios (e.g., 10–20% demand shocks) and plan inventory, logistics, and sourcing diversification accordingly.

13. Key Findings & Observations

- Total sales trend: State-wide sales generally show changes over the years, with periods of decline followed by recovery and growth, reflecting broader economic cycles.
- County dominance: A small set of populous counties (Adams, Arapahoe, Denver, El Paso, Jefferson) account for a major share of total sales, indicating spatial concentration of demand.
- Seasonal pattern: Aggregated quarterly data typically shows some quarters outperforming others (e.g., Q2 or Q3), highlighting seasonality in vehicle purchases.
- Predictability: Linear trend models capture long-term direction reasonably but may miss short-term volatility, indicating that more advanced time-series models could improve accuracy.

14. Applications

- Demand forecasting for dealerships and manufacturers at state and county levels.
- Inventory management and allocation based on historical seasonality and location-wise demand.
- Capacity planning for logistics providers, finance companies, and service centers.

- Policy analysis for state and local governments to understand how infrastructure or tax changes may affect vehicle sales.

15. Limitations

- Limited features: The dataset only includes sales amounts, year, quarter, and county; it lacks information on vehicle type, price range, demographic factors, and macroeconomic indicators.
- Aggregation level: County-level aggregation hides individual dealership or consumer-level behavior.
- Simple models: Linear regression on a time index ignores non-linear trends, shocks, and complex seasonality.
- External events: The dataset alone does not encode events like policy changes, pandemics, or fuel price spikes, which may explain some fluctuations.

16. Future Scope

- Incorporate additional explanatory variables such as fuel prices, interest rates, population, and income levels.
- Use more advanced forecasting methods (ARIMA, Prophet, gradient boosting, or LSTM models) for improved accuracy.
- Build interactive dashboards (Power BI, Tableau, Plotly Dash) for real-time exploration by region and time period.
- Extend supply chain analysis by linking sales data with production, inventory, and logistics datasets.

17. Conclusion

The Colorado motor vehicle sales analysis provides a structured view of how vehicle demand evolves over time and across counties. By combining descriptive analytics, visualization, and simple forecasting, the project offers useful insights for demand planning, supply chain risk assessment, and policy considerations.

18. Python Source Code

Below is the core Python code used in the project (you can adjust paths and county names as needed).

```
# ========================

# 1. Setup and data loading

# ========================

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


# For forecasting / modeling

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score


# Display options

pd.set_option('display.float_format', lambda x: f'{x:,.0f}')


# Load the data

file_path = "colorado_motor_vehicle_sales.csv"  # update if needed

df = pd.read_csv(file_path)


print("First 5 rows:")

print(df.head())
```

```
print("\nInfo:")
print(df.info())

print("\nSummary stats:")
print(df.describe(include='all'))
```

```
First 5 rows:
   year  quarter                 county       sales
0  2008        1                  Adams   231609000
1  2008        1              Arapahoe   550378000
2  2008        1  Boulder/Broomfield   176771000
3  2008        1                 Denver   200103000
4  2008        1               Douglas    93259000

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 501 entries, 0 to 500
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   year      501 non-null    int64
 1   quarter   501 non-null    int64
 2   county    501 non-null    object
 3   sales     501 non-null    int64
dtypes: int64(3), object(1)
memory usage: 15.8+ KB
None

Summary stats:
          year  quarter county        sales
count      501      501     501          501
unique     NaN      NaN      17          NaN
top        NaN      NaN  Adams           NaN
freq       NaN      NaN      32          NaN
mean     2,012        3     NaN  176,058,483
std          2        1     NaN  164,205,510
min      2,008        1     NaN    6,274,000
25%      2,010        2     NaN   61,482,000
50%      2,012        3     NaN  138,582,000
75%      2,014        4     NaN  224,158,000
max      2,015        4     NaN  916,910,000
```

```
# ======================
# 2. Basic data cleaning
# ======================
# Standardize column names
df.columns = df.columns.str.lower().str.strip()

# Check for missing values
print("\nMissing values per column:")
print(df.isna().sum())

# Drop completely empty rows if any
df = df.dropna(how="all")

# If there are missing sales, you can decide to drop or impute
# Here we simply drop rows with missing 'sales'
df = df.dropna(subset=['sales'])

# Ensure dtypes are correct
df['year'] = df['year'].astype(int)
df['quarter'] = df['quarter'].astype(int)
df['sales'] = df['sales'].astype(float)

# Strip whitespace in county names
df['county'] = df['county'].str.strip()
```

```
print("\nCleaned data preview:")
print(df.head())
```

```
Missing values per column:
year       0
quarter    0
county     0
sales      0
dtype: int64

Cleaned data preview:
   year  quarter                county       sales
0  2008        1                 Adams 231,609,000
1  2008        1               Arapahoe 550,378,000
2  2008        1  Boulder/Broomfield 176,771,000
3  2008        1                Denver 200,103,000
4  2008        1               Douglas  93,259,000
```

```
# ====================================
# 3. Feature engineering for analysis
# ====================================
# Create a period index if useful: e.g., year-quarter as a string
df['year_quarter'] = df['year'].astype(str) + "Q" +
df['quarter'].astype(str)


# Optionally create a time index starting at 0 for the earliest quarter
df = df.sort_values(['year', 'quarter'])
df['time_index'] = (df['year'] - df['year'].min()) * 4 + (df['quarter'] -
df['quarter'].min())
```

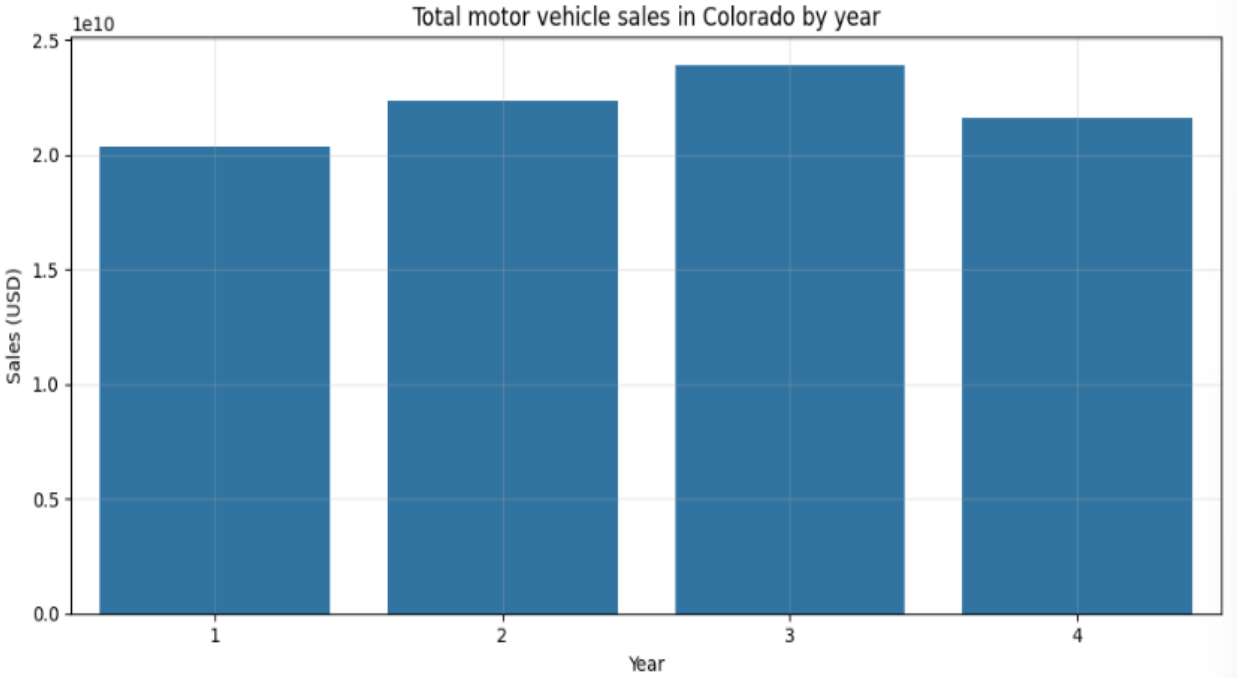
```
print("\nWith engineered features:")
print(df.head())
```

```
With engineered features:
   year  quarter              county       sales year_quarter  time_index
0  2008        1               Adams 231,609,000       2008Q1           0
1  2008        1            Arapahoe 550,378,000       2008Q1           0
2  2008        1  Boulder/Broomfield 176,771,000       2008Q1           0
3  2008        1              Denver 200,103,000       2008Q1           0
4  2008        1             Douglas  93,259,000       2008Q1           0
```

```
# ==========================
# 4. Exploratory data analysis
# ==========================

# 4.1 Overall sales by year (sum across counties)
yearly_sales = df.groupby('year')['sales'].sum().reset_index()
print("\nYearly total sales:")
print(yearly_sales)

plt.figure(figsize=(10, 5))
sns.barplot(
    data=quarterly_sales,
    x='quarter',
    y='sales'
```

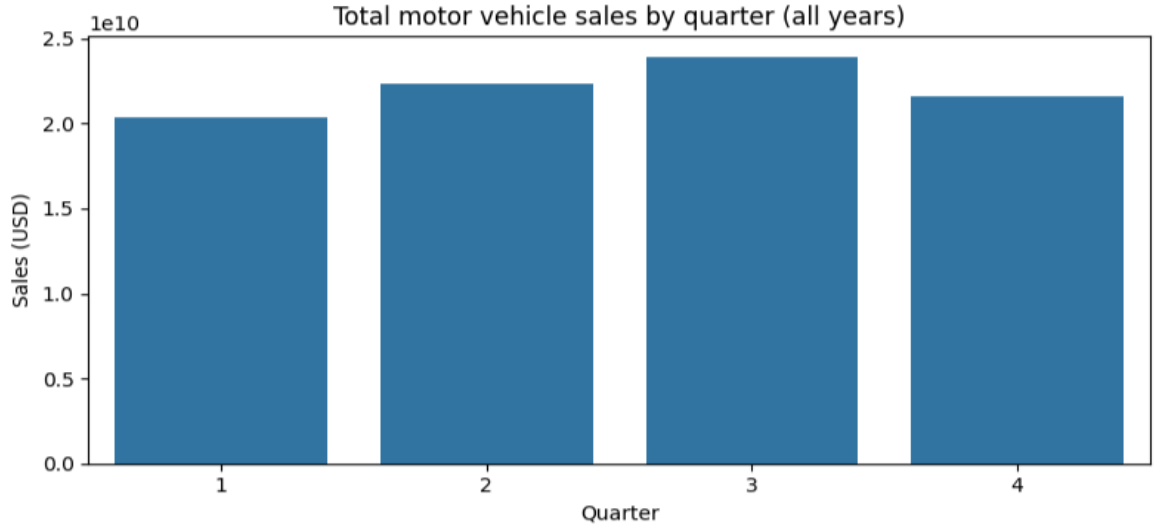
```
)
plt.title("Total motor vehicle sales in Colorado by year")
plt.xlabel("Year")
plt.ylabel("Sales (USD)")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Yearly total sales:
   year          sales
0  2008  8,965,561,000
1  2009  7,652,500,000
2  2010  8,556,088,000
3  2011 12,170,441,000
4  2012 10,960,876,000
5  2013 12,000,615,000
6  2014 13,392,487,000
7  2015 14,506,732,000
```



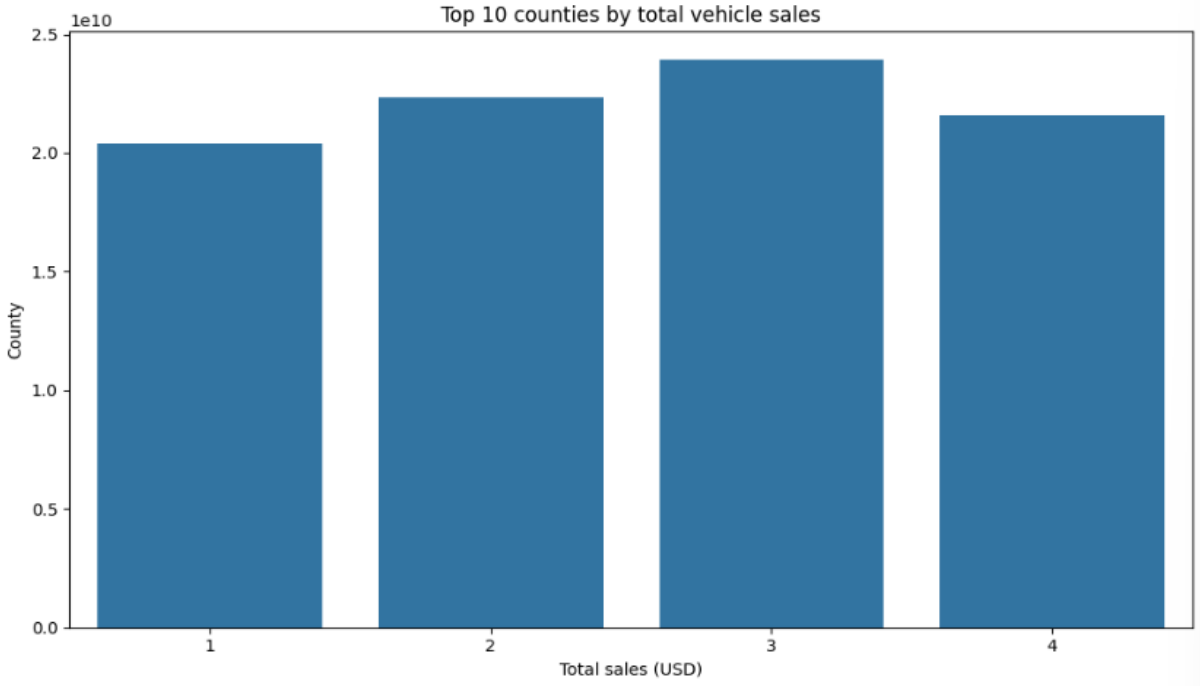
```
# 4.2 Sales by quarter (aggregated over all years)
quarterly_sales = df.groupby('quarter')['sales'].sum().reset_index()
print("\nTotal sales by quarter (all years combined):")
print(quarterly_sales)
plt.figure(figsize=(8, 4))
sns.barplot(
    data=quarterly_sales,
    x='quarter',
    y='sales'
)
plt.title("Total motor vehicle sales by quarter (all years)")
plt.xlabel("Quarter")
plt.ylabel("Sales (USD)")
plt.tight_layout()
plt.show()
```

```
Total sales by quarter (all years combined):
   quarter          sales
0        1 20,376,846,000
1        2 22,343,294,000
2        3 23,914,934,000
3        4 21,570,226,000
```



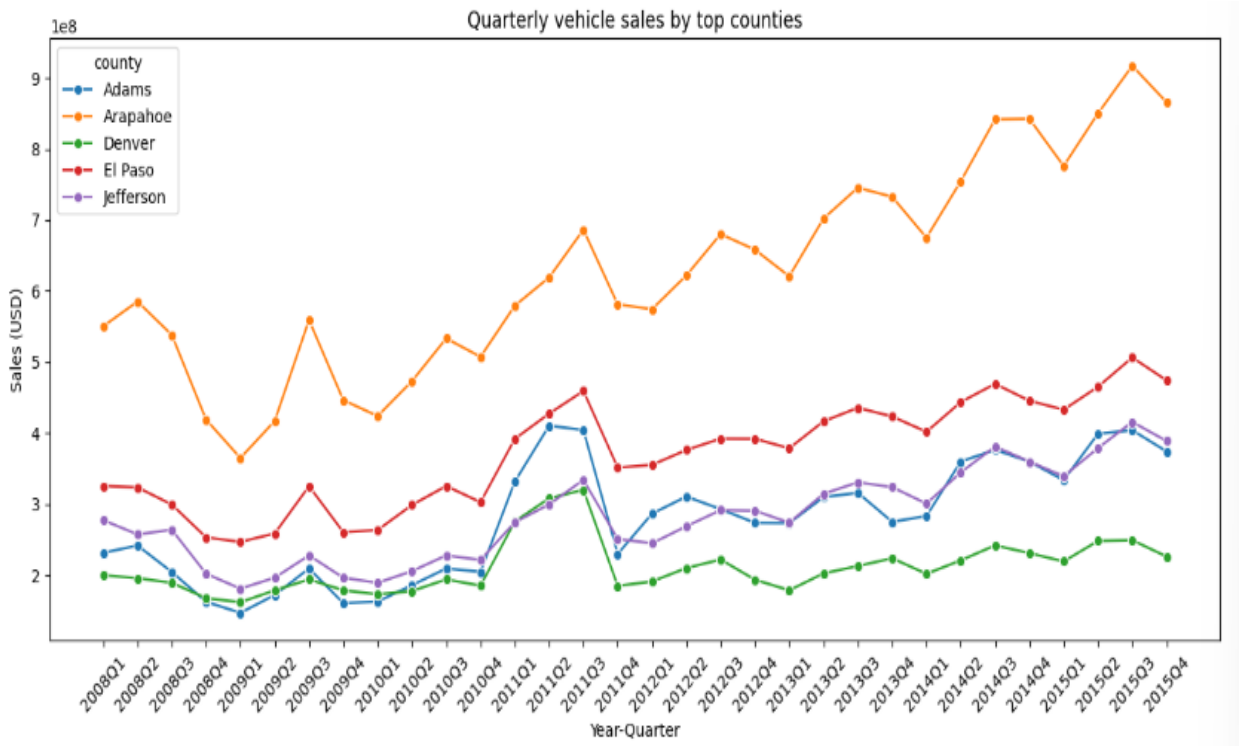
```
# 4.3 Top counties by total sales
county_sales =
df.groupby('county')['sales'].sum().reset_index().sort_values('sales',
ascending=False)
print("\nTop 10 counties by total sales:")
print(county_sales.head(10))


plt.figure(figsize=(10, 6))
sns.barplot(
    data=quarterly_sales,
    x='quarter',
    y='sales'
)
plt.title("Top 10 counties by total vehicle sales")
plt.xlabel("Total sales (USD)")
plt.ylabel("County")
plt.tight_layout()
plt.show()
```

```
Top 10 counties by total sales:
           county          sales
1        Arapahoe 20,142,323,000
7         El Paso 11,926,044,000
10      Jefferson  9,058,407,000
0           Adams  8,902,115,000
5          Denver  6,763,613,000
12        Larimer  5,344,367,000
16           Weld  5,086,889,000
2         Boulder  4,742,532,000
15  Rest of State  3,582,170,000
6         Douglas  3,236,493,000
```




```
# ==========================
# 5. Time series by county
# ==========================


# Example: pick a few big counties to visualize over time
top_counties = county_sales.head(5)['county'].tolist()
df_top = df[df['county'].isin(top_counties)].copy()


plt.figure(figsize=(12, 6))
sns.lineplot(
```

```
    data=df_top.sort_values(['year', 'quarter']),
    x='year_quarter',
    y='sales',
    hue='county',
    marker='o'
)
plt.title("Quarterly vehicle sales by top counties")
plt.xlabel("Year-Quarter")
plt.ylabel("Sales (USD)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



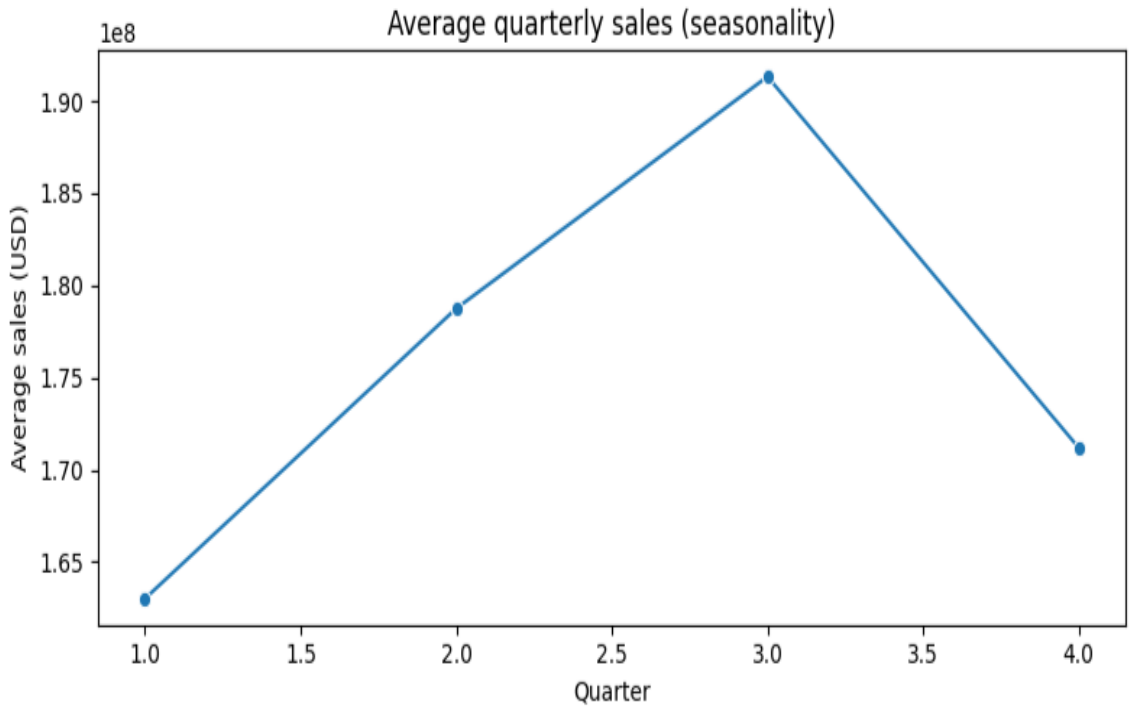
```
# ========================================

# 6. Seasonality and growth pattern analysis

# ========================================


# 6.1 Year-over-year growth by county

county_yearly = df.groupby(['county',
'year'])['sales'].sum().reset_index()

county_yearly['yoy_growth'] =
county_yearly.groupby('county')['sales'].pct_change() * 100


print("\nYear-over-year growth sample:")

print(county_yearly.head(15))
```

```
Year-over-year growth sample:
      county  year         sales  yoy_growth
0      Adams  2008   840,958,000         NaN
1      Adams  2009   689,914,000         -18
2      Adams  2010   763,549,000          11
3      Adams  2011 1,376,918,000          80
4      Adams  2012 1,164,347,000         -15
5      Adams  2013 1,174,812,000           1
6      Adams  2014 1,380,114,000          17
7      Adams  2015 1,511,503,000          10
8   Arapahoe  2008 2,092,912,000         NaN
9   Arapahoe  2009 1,787,961,000         -15
10  Arapahoe  2010 1,937,961,000           8
11  Arapahoe  2011 2,466,086,000          27
12  Arapahoe  2012 2,534,655,000           3
13  Arapahoe  2013 2,800,854,000          11
14  Arapahoe  2014 3,113,457,000          11
```

```
# Visualize YoY growth for a specific county (e.g., Denver)
county_name = "Denver"  # change to any county in the dataset
denver_yoy = county_yearly[county_yearly['county'] == county_name]

plt.figure(figsize=(8, 4))
sns.barplot(data=denver_yoy, x='year', y='yoy_growth', color='steelblue')
plt.title(f"Year-over-year sales growth in {county_name}")
plt.xlabel("Year")
plt.ylabel("YoY growth (%)")
plt.axhline(0, color='black', linewidth=1)
plt.tight_layout()
plt.show()
```

```
# 6.2 Seasonal pattern: average sales per quarter across years
seasonality = df.groupby('quarter')['sales'].mean().reset_index()
print("\nAverage sales per quarter:")
print(seasonality)

plt.figure(figsize=(8, 4))
sns.lineplot(data=seasonality, x='quarter', y='sales', marker='o')
plt.title("Average quarterly sales (seasonality)")
plt.xlabel("Quarter")
plt.ylabel("Average sales (USD)")
plt.tight_layout()
plt.show()
```

```
Average sales per quarter:
   quarter       sales
0        1 163,014,768
1        2 178,746,352
2        3 191,319,472
3        4 171,192,270
```



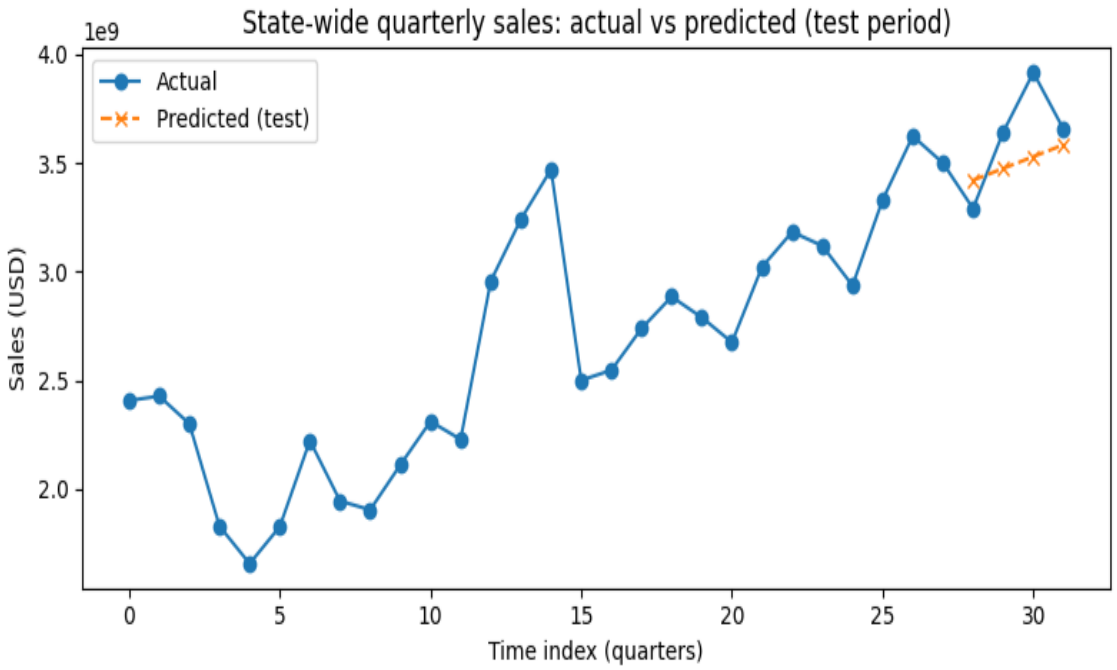
```
# ====================================
# 7. Simple predictive modeling example
# ====================================

# Goal: Predict total state-wide sales per quarter using time_index

state_quarter = df.groupby(['year',
'quarter'])['sales'].sum().reset_index()
state_quarter = state_quarter.sort_values(['year',
'quarter']).reset_index(drop=True)

# Create time index
state_quarter['time_index'] = np.arange(len(state_quarter))

X = state_quarter[['time_index']]
y = state_quarter['sales']

# Train-test split (e.g., last 4 quarters as test)
test_size = 4
X_train, X_test = X.iloc[:-test_size], X.iloc[-test_size:]
y_train, y_test = y.iloc[:-test_size], y.iloc[-test_size:]

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

```
y_pred = model.predict(X_test)

print("\nTest performance (simple linear trend model):")
mse = mean_squared_error(y_test, y_pred)  # no squared argument
rmse = np.sqrt(mse)

print("MSE:", mse)
print("RMSE:", rmse)
print("R^2:", r2_score(y_test, y_pred))
```

```
Test performance (simple linear trend model):
MSE: 4.984197040594626e+16
RMSE: 223253153.18253908
R^2: -0.008996759860718129
```

```
# Plot actual vs predicted for test period
plt.figure(figsize=(8, 4))
plt.plot(state_quarter['time_index'], state_quarter['sales'],
label='Actual', marker='o')
plt.plot(X_test['time_index'], y_pred, label='Predicted (test)',
marker='x', linestyle='--')
plt.title("State-wide quarterly sales: actual vs predicted (test
period)")
plt.xlabel("Time index (quarters)")
```

```
plt.ylabel("Sales (USD)")
plt.legend()
plt.tight_layout()
plt.show()
```

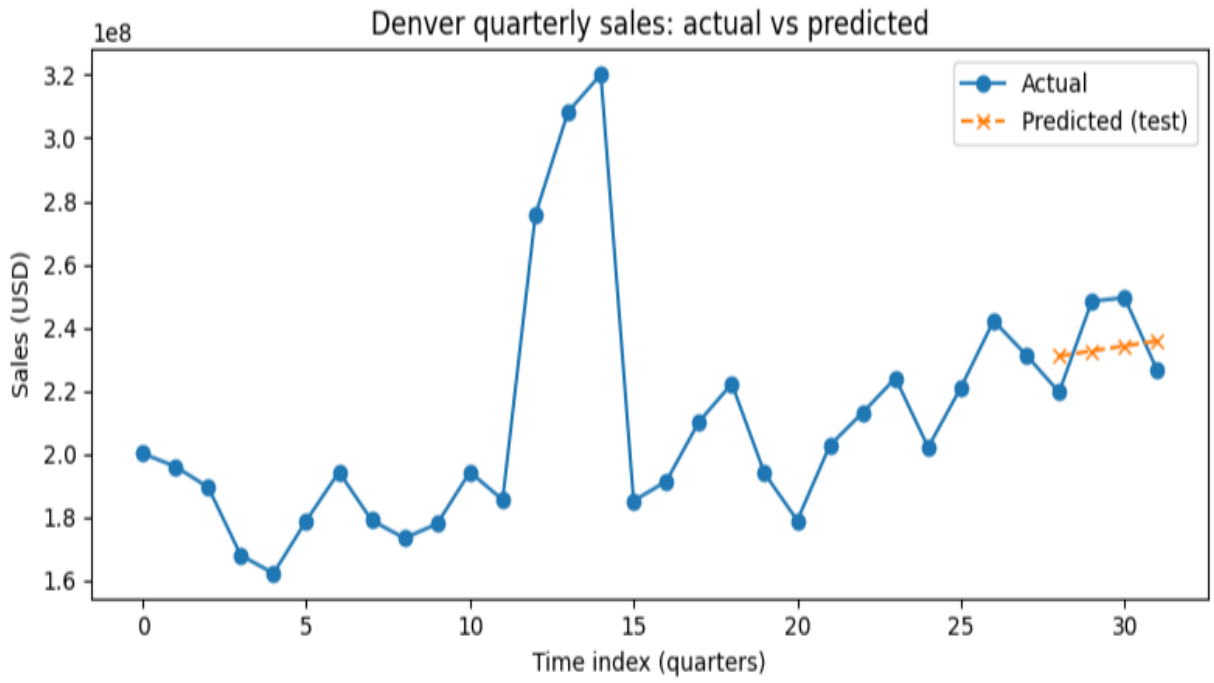



```
# 7.1 Forecast next 4 quarters
future_periods = 4
last_index = state_quarter['time_index'].max()
future_time_index = np.arange(last_index + 1, last_index + 1 +
future_periods)

# Use a DataFrame with the same column name as training
future_df = pd.DataFrame({'time_index': future_time_index})
future_pred = model.predict(future_df)
```

```
forecast_df = pd.DataFrame({
    'time_index': future_time_index,
    'forecast_sales': future_pred
})

# ============================================
# 8. County-level model
# ============================================

# Example: fit a separate linear trend for a single county

county_name = "Denver"  # change as needed
county_data = df[df['county'] == county_name].copy()
county_data = county_data.sort_values(['year', 'quarter'])

county_data['time_index'] = np.arange(len(county_data))

X_c = county_data[['time_index']]
y_c = county_data['sales']

test_size_c = 4
Xc_train, Xc_test = X_c.iloc[:-test_size_c], X_c.iloc[-test_size_c:]
yc_train, yc_test = y_c.iloc[:-test_size_c], y_c.iloc[-test_size_c:]
```

```
model_c = LinearRegression()
model_c.fit(Xc_train, yc_train)

yc_pred = model_c.predict(Xc_test)

print(f"\n{county_name} model test performance:")
mse_c = mean_squared_error(yc_test, yc_pred)   # no squared argument
rmse_c = np.sqrt(mse_c)

print("MSE:", mse_c)
print("RMSE:", rmse_c)
print("R^2:", r2_score(yc_test, yc_pred))
plt.figure(figsize=(8, 4))
plt.plot(county_data['time_index'], county_data['sales'], label='Actual', marker='o')
plt.plot(Xc_test['time_index'], yc_pred, label='Predicted (test)', marker='x', linestyle='--')
plt.title(f"{county_name} quarterly sales: actual vs predicted")
plt.xlabel("Time index (quarters)")
plt.ylabel("Sales (USD)")
plt.legend()
plt.tight_layout()
plt.show()
```

```
Denver model test performance:
MSE: 175602027763003.75
RMSE: 13251491.529748783
R^2: -0.005572912713094569
```



================================

9. Saving processed datasets

================================

Save yearly and county-level aggregates for reporting / dashboarding

yearly_sales.to_csv("yearly_state_sales.csv", index=False)

county_sales.to_csv("total_sales_by_county.csv", index=False)

county_yearly.to_csv("county_yearly_sales_with_yoy.csv", index=False)

state_quarter.to_csv("state_quarterly_sales_with_time_index.csv", index=False)

```
print("\nDerived CSV files saved:")
print("- yearly_state_sales.csv")
print("- total_sales_by_county.csv")
print("- county_yearly_sales_with_yoy.csv")
print("- state_quarterly_sales_with_time_index.csv")
```

```
Derived CSV files saved:
- yearly_state_sales.csv
- total_sales_by_county.csv
- county_yearly_sales_with_yoy.csv
- state_quarterly_sales_with_time_index.csv
```