1. Data Collection:

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# load the dataset in dataframe
df = pd.read_csv(r"C:\Users\Hp\Documents\colorado_motor_vehicle_sales2.csv")
df
```

2. Data Preparation:

```
# Create a new month column
def quarter_to_month(year, quarter):
if quarter == 1:
return f"{year}-01"
elif quarter == 2:
return f"{year}-04"
elif quarter == 3:
return f"{year}-07"
elif quarter == 4:
return f"{year}-10"
df['month'] = df.apply(lambda row: quarter_to_month(row['year'], row['quarter']), axis=1)
cols = list(df.columns)
cols.insert(0, cols.pop(cols.index('month')))
df = df.reindex(columns=cols)
df.to_csv(r'C:\Users\Hp\Documents\colorado_motor_vehicle_sales2.csv', index=False)
df.head()
```

```
# Convert month column to datetime format
df['month'] = pd.to_datetime(df['month'], format='%Y-%m')
print(df.dtypes)
# Check for missing values
df.isnull().sum()
```

3. Exploratory Data Analysis (EDA):

```
# Monthly Sales Trend
df["month"] = pd.to_datetime(df["month"])
monthly_sales = df.groupby("month")["sales"].sum()
plt.figure(figsize=(8, 5))
plt.plot(monthly_sales.index, monthly_sales.values, marker="o", linestyle="-", color="Red",
label="Month ")
plt.grid(True, linestyle="--", alpha=0.6)
plt.xlabel("month", fontsize=12)
plt.ylabel("sales ($)", fontsize=12)
plt.title("Monthly Sales Trend", fontsize=14)
plt.xticks(rotation=45)
plt.legend(["Month"], loc="upper left")
plt.show()
#Quarterly Sales Trend
quarterly_sales = df.groupby("quarter")["sales"].sum()
colors = [ '#ffcc99', 'lightgreen'] plt.figure(figsize=(8, 5))
plt.bar(quarterly_sales.index, quarterly_sales.values, color=[colors[i % 2]
for i in range(len(quarterly_sales))], edgecolor="black", alpha=0.8)
plt.xlabel("Quarter")
plt.ylabel("Total Sales ($)")plt.title("Quarterly Sales")
plt.xticks(rotation=0)
plt.show()
#Vehicle Sales Across Counties
county_sales = df.groupby("county")["sales"].sum().reset_index()
```

```
plt.figure(figsize=(8, 4))
colors = sns.color_palette("husl", len(county_sales))
plt.bar(county_sales["county"], county_sales["sales"], color=colors)
plt.xticks(rotation=45, ha="right")
plt.xlabel("county", fontsize=12)
plt.ylabel("sales ($)", fontsize=12)
plt.title("Vehicle Sales By Counties", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
#Sales Frequency
plt.figure(figsize=(7, 5))
sns.histplot(df['sales'], bins=10, kde=True, color="green", edgecolor='black')
plt.xlabel('Sales($)')
plt.ylabel('Frequency')
plt.title('Sales Distribution ')
plt.show()
```

4. Statistical Analysis:

```
# df.shape
#df.describe()
# Correlation Between Monthly Sales(Top 10 Counties)
top_counties = df.groupby('county')['sales'].sum().nlargest(10).index
df_top = df[df['county'].isin(top_counties)]
monthly_sales = df_top.groupby(['month', 'county'])['sales'].sum().reset_index()
sales_by_county = monthly_sales.pivot(index='month', columns='county', values='sales')
correlation_matrix = sales_by_county.corr()
plt.figure(figsize=(9, 4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Between Monthly Sales (Top 10 Counties)')
plt.xticks(rotation=45)
plt.show()
```

5. Predictive Modeling:

```
# Quarterly Sales Trends
def quarter_to_month(year, quarter):
if quarter == 1:
return f"{year}-01"
elif quarter == 2:
return f"{year}-04"
elif quarter == 3:
elif quarter == 3:
return f"{year}-07"
elif quarter == 4:
return f"{year}-10"
df['month'] = df.apply(lambda row: quarter_to_month(row['year'], row['quarter']), axis=1)
df['month'] = pd.to_datetime(df['month'])
quarterly_sales = df.groupby(['year', 'quarter'])['sales'].sum().reset_index()
plt.figure(figsize=(7,5))
sns.lineplot(data=quarterly_sales, x='year', y='sales', hue='quarter', marker='o')
plt.title('Quarterly Sales Trends')
plt.xlabel('Year')
plt.ylabel('Total Sales ($)')
plt.grid(True)
plt.show()
# Top 10 Counties by sales
county_sales = df.groupby('county')['sales'].sum().reset_index()
county_sales = county_sales.sort_values(by='sales', ascending=False)
plt.figure(figsize=(7,5))
sns.barplot(data=county_sales.head(10), x='sales', y='county', palette='viridis')
plt.title('Top 10 County by Sales')
plt.xlabel('Total Sales ($)')
plt.ylabel('County')
```

```
plt.show()
# Forecasted Sales for Next 12 Months
df_monthly = df.groupby('month')['sales'].sum()
df_monthly = df_monthly.sort_index()
model = ARIMA(df_monthly_diff, order=(5, 0, 0))
model_fit = model.fit()
forecast_steps = 12
forecast = model_fit.forecast(steps=forecast_steps)
forecast_cumsum = forecast.cumsum() + df_monthly.iloc[-1]
plt.figure(figsize=(7,5))
plt.plot(df_monthly, label='Historical Sales')
plt.plot(pd.date_range(df_monthly.index[-1], periods=forecast_steps+1, freq='M')[1:],
forecast_cumsum, label='Forecast', linestyle='--', color='red')
plt.title('Forecasted Motor Vehicle Sales for Next 12 Months')
plt.xlabel('Month')
plt.ylabel('Sales ($)')
plt.legend()
plt.grid(True)
plt.show()
#Random Forest sales forecast
df['month_int'] = df['month'].dt.month
df = pd.get_dummies(df, columns=['county'], drop_first=True)
X = df[['month_int'] + [col for col in df.columns if 'county' in col]]
y = df['sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)
y_pred = model_rf.predict(X_test)
plt.figure(figsize=(7,5))
plt.plot(y_test.index, y_test, label='Actual Sales')
```

```
plt.plot(y_test.index, y_pred, label='Predicted Sales', linestyle='--')
plt.title('Random Forest Sales Forecast')
plt.xlabel('Month')
plt.ylabel('Sales ($)')
plt.legend()
plt.grid(True)
plt.show()
# Identify Factors Influencing Sales Trends
X = df[['month_int'] + [col for col in df.columns if 'county' in col]]
y = df['sales']
model_lr = LinearRegression()
model_lr.fit(X, y)
plt.figure(figsize=(9,3))
plt.bar(coefficients.index, coefficients['Coefficient'], color=['#F5DEB3', '#452B1F', '#FFD700', '#964B00'])
plt.xlabel('Features')
plt.ylabel('Coefficient')
plt.title('Linear Regression Coefficients')
plt.xticks(rotation=45)
plt.show()
```