```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.graphics.gofplots import ProbPlot
import os

# Load the data
def load_data():
    # Load from CSV files
    df1 = pd.read_csv('/content/1.csv')  # Replace with actual file path
    df2 = pd.read_csv('/content/2.csv')  # Replace with actual file path

    # Combine the dataframes
    combined_df = pd.concat([df1, df2], ignore_index=True)

    # Clean and prepare the data
    combined_df['Debt Restructuring Announcement Date'] = pd.to_datetime(
        combined_df['Debt Restructuring Announcement Date'],
        format='%d/%m/%Y',
        errors='coerce'
    )

    # Create a flag for emerging markets
    combined_df['Is_Emerging_Market'] = combined_df['Country (Emerging Economy)'].fillna('No').apply(
        lambda x: 1 if x == 'Yes' else 0
    )

    # Fill any NaN values in numeric columns with median values
    numeric_cols = ['Closing Stock Price', 'Market Index Value', 'Trading Volume',
                    'Return on Assets (ROA)', 'Return on Equity (ROE)']
    for col in numeric_cols:
        combined_df[col] = pd.to_numeric(combined_df[col], errors='coerce')
        combined_df[col] = combined_df[col].fillna(combined_df[col].median())

    # Create dummy variables for restructuring types
    restructuring_dummies = pd.get_dummies(combined_df['Type of Debt Restructuring'],
                                           prefix='Restructuring')
    combined_df = pd.concat([combined_df, restructuring_dummies], axis=1)

    # Create dummy variables for industries
    industry_dummies = pd.get_dummies(combined_df['Industry'], prefix='Industry')
    combined_df = pd.concat([combined_df, industry_dummies], axis=1)

    # Save cleaned dataset
    combined_df.to_csv("cleaned_data.csv", index=False)

    # Save data summary
    with open("data_summary.txt", 'w') as f:
        f.write("Dataset Summary\n")
        f.write("==============\n\n")
        f.write(f"Total records: {len(combined_df)}\n")
        f.write(f"Columns: {', '.join(combined_df.columns)}\n\n")
        f.write("Data Types:\n")
        f.write(combined_df.dtypes.to_string())
        f.write("\n\nMissing Values:\n")
        f.write(combined_df.isnull().sum().to_string())

    return combined_df

# Exploratory Data Analysis
def exploratory_analysis(df):
    print("Basic Statistics:")
    stats_df = df[['Closing Stock Price', 'Return on Assets (ROA)', 'Return on Equity (ROE)']].describe()
    print(stats_df)

    # Save descriptive statistics
    stats_df.to_csv("descriptive_statistics.csv")

    # Save screenshot of descriptive statistics table
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.axis('tight')
    ax.axis('off')
    table = ax.table(cellText=stats_df.values,
                     rowLabels=stats_df.index,
```

```
                     rowLabels=stats_df.index,
                     colLabels=stats_df.columns,
                     cellLoc='center',
                     loc='center')
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.scale(1.2, 1.2)
    plt.title('Descriptive Statistics of Key Variables')
    plt.savefig('descriptive_statistics_table.png', dpi=300, bbox_inches='tight')
    plt.close()  # Close the figure to prevent display in notebooks if not needed

    # Distribution of share prices
    plt.figure(figsize=(10, 6))
    sns.histplot(df['Closing Stock Price'], kde=True)
    plt.title('Distribution of Closing Stock Prices')
    plt.xlabel('Stock Price')
    plt.savefig('stock_price_distribution.png', dpi=300)
    plt.close()

    # Box plot of share prices by restructuring type
    plt.figure(figsize=(12, 8))
    sns.boxplot(x='Type of Debt Restructuring', y='Closing Stock Price', data=df)
    plt.title('Share Prices by Type of Debt Restructuring')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig('share_prices_by_restructuring.png', dpi=300)
    plt.close()

    # Box plot of share prices by industry
    plt.figure(figsize=(14, 8))
    sns.boxplot(x='Industry', y='Closing Stock Price', data=df)
    plt.title('Share Prices by Industry')
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.savefig('share_prices_by_industry.png', dpi=300)
    plt.close()

    # Comparison between emerging and developed markets
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Is_Emerging_Market', y='Closing Stock Price', data=df)
    plt.title('Share Prices in Emerging vs Developed Markets')
    plt.xticks([0, 1], ['Developed', 'Emerging'])
    plt.savefig('share_prices_by_market_type.png', dpi=300)
    plt.close()

    # Distribution of ROA and ROE
    fig, axes = plt.subplots(1, 2, figsize=(16, 6))
    sns.histplot(df['Return on Assets (ROA)'], kde=True, ax=axes[0])
    axes[0].set_title('Distribution of Return on Assets (ROA)')
    sns.histplot(df['Return on Equity (ROE)'], kde=True, ax=axes[1])
    axes[1].set_title('Distribution of Return on Equity (ROE)')
    plt.tight_layout()
    plt.savefig('financial_metrics_distribution.png', dpi=300)
    plt.close()

    # ROA/ROE by restructuring type
    fig, axes = plt.subplots(1, 2, figsize=(18, 8))
    sns.boxplot(x='Type of Debt Restructuring', y='Return on Assets (ROA)', data=df, ax=axes[0])
    axes[0].set_title('ROA by Restructuring Type')
    axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=45)

    sns.boxplot(x='Type of Debt Restructuring', y='Return on Equity (ROE)', data=df, ax=axes[1])
    axes[1].set_title('ROE by Restructuring Type')
    axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation=45)

    plt.tight_layout()
    plt.savefig('financial_metrics_by_restructuring.png', dpi=300)
    plt.close()

    # Save restructuring type counts
    restructuring_counts = df['Type of Debt Restructuring'].value_counts()
    restructuring_counts.to_csv("restructuring_type_counts.csv")

    plt.figure(figsize=(10, 6))
    sns.countplot(y='Type of Debt Restructuring', data=df)
    plt.title('Count of Different Restructuring Types')
    plt.tight_layout()
    plt.savefig('restructuring_type_counts.png', dpi=300)
    plt.close()

# Correlation Analysis
```

```
def correlation_analysis(df):
    print("\nCorrelation Analysis:")

    # Select relevant numeric columns for correlation
    numeric_cols = ['Closing Stock Price', 'Market Index Value', 'Trading Volume',
                    'Return on Assets (ROA)', 'Return on Equity (ROE)', 'Is_Emerging_Market']

    # Add restructuring type columns
    restructuring_cols = [col for col in df.columns if 'Restructuring_' in col]

    # Add industry columns
    industry_cols = [col for col in df.columns if 'Industry_' in col]

    # Combine all columns for correlation
    corr_cols = numeric_cols + restructuring_cols + industry_cols

    # Calculate correlation matrix
    # Make sure all columns are numeric
    for col in corr_cols:
        if df[col].dtype == 'object':
            print(f"Warning: Column {col} is not numeric. Converting to numeric...")
            df[col] = pd.to_numeric(df[col], errors='coerce')

    correlation_matrix = df[corr_cols].corr()

    # Save full correlation matrix
    correlation_matrix.to_csv("full_correlation_matrix.csv")

    # Focus on correlations with stock price
    stock_price_corr = correlation_matrix['Closing Stock Price'].sort_values(ascending=False)
    print("Correlations with Closing Stock Price:")
    print(stock_price_corr)

    # Save correlations with stock price
    stock_price_corr.to_frame('Correlation with Stock Price').to_csv(
        "stock_price_correlations.csv")

    # Plot top 10 correlations with stock price
    plt.figure(figsize=(12, 8))
    # Make sure there are at least 10 correlations
    n_correlations = min(10, len(stock_price_corr))
    top_correlations = stock_price_corr.head(n_correlations)
    sns.barplot(x=top_correlations.values, y=top_correlations.index)
    plt.title('Top Correlations with Stock Price')
    plt.xlabel('Correlation Coefficient')
    plt.tight_layout()
    plt.savefig('top_correlations_with_stock_price.png', dpi=300)
    plt.close()

    # Visualize key correlations
    plt.figure(figsize=(14, 12))
    mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
    sns.heatmap(correlation_matrix, mask=mask, annot=True, cmap='coolwarm',
                linewidths=0.5, fmt='.2f', vmin=-1, vmax=1)
    plt.title('Correlation Matrix')
    plt.tight_layout()
    plt.savefig('correlation_matrix_heatmap.png', dpi=300)
    plt.close()

    # Create a smaller heatmap with just key variables
    key_vars = ['Closing Stock Price', 'Return on Assets (ROA)', 'Return on Equity (ROE)',
                'Is_Emerging_Market', 'Trading Volume']

    # Add top restructuring types - check if restructuring_cols exists and is not empty
    if restructuring_cols and len(restructuring_cols) >= 2:
        for col in restructuring_cols[:2]:  # Add top 2 restructuring columns
            if col in correlation_matrix.columns:
                key_vars.append(col)

    key_correlation = correlation_matrix.loc[key_vars, key_vars]

    plt.figure(figsize=(10, 8))
    sns.heatmap(key_correlation, annot=True, cmap='coolwarm', linewidths=0.5, fmt='.2f')
    plt.title('Key Variables Correlation Matrix')
    plt.tight_layout()
    plt.savefig('key_variables_correlation.png', dpi=300)
    plt.close()

    # Pairplot for key numeric variables
    pair_vars = ['Closing Stock Price', 'Return on Assets (ROA)',
```

```python
                 'Return on Equity (ROE)', 'Is_Emerging_Market']
    pair_df = df[pair_vars].copy()
    pair_df['Market Type'] = df['Is_Emerging_Market'].map({0: 'Developed', 1: 'Emerging'})

    g = sns.pairplot(pair_df, hue='Market Type', corner=True)
    g.fig.suptitle('Pairwise Relationships Between Key Variables', y=1.02, fontsize=16)
    plt.savefig('key_variables_pairplot.png', dpi=300)
    plt.close()

    # Scatterplots of ROA and ROE vs Stock Price
    fig, axes = plt.subplots(1, 2, figsize=(16, 6))

    sns.scatterplot(x='Return on Assets (ROA)', y='Closing Stock Price',
                    hue='Type of Debt Restructuring', data=df, ax=axes[0])
    axes[0].set_title('Stock Price vs ROA by Restructuring Type')

    sns.scatterplot(x='Return on Equity (ROE)', y='Closing Stock Price',
                    hue='Type of Debt Restructuring', data=df, ax=axes[1])
    axes[1].set_title('Stock Price vs ROE by Restructuring Type')

    plt.tight_layout()
    plt.savefig('financial_metrics_vs_stock_price.png', dpi=300)
    plt.close()

    return stock_price_corr

# Regression Analysis
def regression_analysis(df):
    print("\nRegression Analysis:")

    # Prepare the formula for regression
    # Include restructuring types, industry dummies, financial health metrics, and emerging market flag

    # Get lists of dummy columns
    restructuring_cols = [col for col in df.columns if 'Restructuring_' in col]
    industry_cols = [col for col in df.columns if 'Industry_' in col]

    # Avoid the dummy variable trap by dropping one category from each
    if restructuring_cols:
        restructuring_cols = restructuring_cols[:-1]
    if industry_cols:
        industry_cols = industry_cols[:-1]

    # Prepare X variables (independent variables)
    X_cols = ['Return on Assets (ROA)', 'Return on Equity (ROE)',
              'Trading Volume', 'Is_Emerging_Market'] + restructuring_cols + industry_cols

    # Remove any columns that might have all NaN values
    X_cols = [col for col in X_cols if not df[col].isna().all()]

    # Make sure all columns are numeric
    for col in X_cols + ['Closing Stock Price']:
        if df[col].dtype == 'object':
            df[col] = pd.to_numeric(df[col], errors='coerce')

    # Drop rows with NaN values in dependent or independent variables
    model_df = df.dropna(subset=['Closing Stock Price'] + X_cols)

    # Check if there are enough observations to run the regression
    if len(model_df) < len(X_cols) + 1:
        print("Error: Not enough observations to run regression with all variables.")
        print(f"Observations: {len(model_df)}, Variables: {len(X_cols)}")
        # Reduce the number of variables if necessary
        X_cols = X_cols[:len(model_df) - 2]  # Leave at least 1 degree of freedom
        print(f"Reducing to {len(X_cols)} variables.")

    # Create formula string for regression
    formula = 'Q("Closing Stock Price") ~ ' + ' + '.join([f'Q("{col}")' for col in X_cols])

    try:
        # Run regression
        model = ols(formula, data=model_df).fit()
        model_summary = model.summary()
        print(model_summary)

        # Save model summary to text file
        with open("full_regression_model_summary.txt", 'w') as f:
            f.write(str(model_summary))

        # Save model coefficients with p-values
```

```python
    coefs = pd.DataFrame({
        'Coefficient': model.params,
        'Std Error': model.bse,
        't-value': model.tvalues,
        'p-value': model.pvalues
    })
    coefs.to_csv("full_regression_coefficients.csv")

    # Plot of coefficients with p-values < 0.1
    significant_coefs = coefs[coefs['p-value'] < 0.1]
    if not significant_coefs.empty:
        significant_coefs = significant_coefs.sort_values('Coefficient')

        plt.figure(figsize=(12, len(significant_coefs) * 0.5 + 2))
        sns.barplot(x='Coefficient', y=significant_coefs.index, data=significant_coefs)
        plt.axvline(x=0, color='black', linestyle='-', alpha=0.7)
        plt.title('Significant Regression Coefficients (p < 0.1)')
        plt.tight_layout()
        plt.savefig('significant_coefficients.png', dpi=300)
        plt.close()

    # Run a simpler model focusing just on restructuring types and financial health
    simple_cols = ['Return on Assets (ROA)', 'Return on Equity (ROE)', 'Is_Emerging_Market']
    if restructuring_cols:
        simple_cols += restructuring_cols

    simple_formula = 'Q("Closing Stock Price") ~ ' + ' + '.join([f'Q("{col}")' for col in simple_cols])
    simple_model = ols(simple_formula, data=model_df).fit()
    simple_model_summary = simple_model.summary()
    print("\nSimplified Model:")
    print(simple_model_summary)

    # Save simple model summary
    with open("simple_regression_model_summary.txt", 'w') as f:
        f.write(str(simple_model_summary))

    # Save simple model coefficients
    simple_coefs = pd.DataFrame({
        'Coefficient': simple_model.params,
        'Std Error': simple_model.bse,
        't-value': simple_model.tvalues,
        'p-value': simple_model.pvalues
    })
    simple_coefs.to_csv("simple_regression_coefficients.csv")

    # Plot simple model coefficients
    simple_coefs = simple_coefs.sort_values('Coefficient')
    plt.figure(figsize=(12, len(simple_coefs) * 0.5 + 2))
    sns.barplot(x='Coefficient', y=simple_coefs.index, data=simple_coefs)
    plt.axvline(x=0, color='black', linestyle='-', alpha=0.7)
    plt.title('Simple Model Regression Coefficients')
    plt.tight_layout()
    plt.savefig('simple_model_coefficients.png', dpi=300)
    plt.close()

    # Compare observed vs predicted values
    predicted = model.predict(model_df)
    comparison_df = pd.DataFrame({
        'Observed': model_df['Closing Stock Price'],
        'Predicted': predicted,
        'Residual': model_df['Closing Stock Price'] - predicted,
        'Type of Debt Restructuring': model_df['Type of Debt Restructuring'],
        'Is_Emerging_Market': model_df['Is_Emerging_Market']
    })
    comparison_df.to_csv("observed_vs_predicted.csv")

    plt.figure(figsize=(10, 8))
    sns.scatterplot(x='Observed', y='Predicted', hue='Type of Debt Restructuring', data=comparison_df)
    plt.plot([0, comparison_df['Observed'].max()], [0, comparison_df['Observed'].max()], 'k--')
    plt.title('Observed vs Predicted Stock Prices')
    plt.tight_layout()
    plt.savefig('observed_vs_predicted.png', dpi=300)
    plt.close()

    return model, simple_model

except Exception as e:
    print(f"Error in regression analysis: {e}")
    return None, None
```

```
# Regression Diagnostics
def regression_diagnostics(model, df):
    if model is None:
        print("No model to perform diagnostics on.")
        return None

    print("\nRegression Diagnostics:")

    try:
        # Get model residuals
        residuals = model.resid
        fitted_values = model.fittedvalues

        # Save residuals to CSV
        residuals_df = pd.DataFrame({
            'Fitted_Values': fitted_values,
            'Residuals': residuals,
            'Standardized_Residuals': residuals / np.sqrt(np.var(residuals)),
            'Abs_Standardized_Residuals': np.abs(residuals / np.sqrt(np.var(residuals))),
            'Sqrt_Abs_Standardized_Residuals': np.sqrt(np.abs(residuals / np.sqrt(np.var(residuals))))
        })
        residuals_df.to_csv("regression_residuals.csv")

        # 1. Check for linearity and heteroscedasticity with residual plot
        plt.figure(figsize=(10, 6))
        sns.scatterplot(x=fitted_values, y=residuals)
        plt.axhline(y=0, color='r', linestyle='-')
        plt.title('Residuals vs Fitted Values')
        plt.xlabel('Fitted Values')
        plt.ylabel('Residuals')
        plt.savefig('residuals_vs_fitted.png', dpi=300)
        plt.close()

        # 2. Q-Q plot to check for normality of residuals
        plt.figure(figsize=(10, 6))
        qq = ProbPlot(residuals)
        qq.qqplot(line='45')
        plt.title('Q-Q Plot of Residuals')
        plt.savefig('qq_plot.png', dpi=300)
        plt.close()

        # 3. Scale-Location plot to check heteroscedasticity
        plt.figure(figsize=(10, 6))
        standardized_residuals = residuals / np.sqrt(np.var(residuals))
        sns.scatterplot(x=fitted_values, y=np.sqrt(np.abs(standardized_residuals)))
        plt.title('Scale-Location Plot')
        plt.xlabel('Fitted Values')
        plt.ylabel('√|Standardized Residuals|')
        plt.savefig('scale_location.png', dpi=300)
        plt.close()

        # 4. Check for multicollinearity using VIF
        # Get X variables from the model
        X_vars = model.model.exog
        X_vars_df = pd.DataFrame(X_vars, columns=model.model.exog_names)

        # Calculate VIF for each variable
        vif_data = pd.DataFrame()
        vif_data["Variable"] = model.model.exog_names
        vif_data["VIF"] = [variance_inflation_factor(X_vars, i) for i in range(X_vars.shape[1])]

        print("Variance Inflation Factors (VIF):")
        print(vif_data)

        # Save VIF values
        vif_data.to_csv("variance_inflation_factors.csv")

        # Plot VIF values
        plt.figure(figsize=(12, len(vif_data) * 0.5 + 2))
        vif_data = vif_data.sort_values('VIF', ascending=False)
        sns.barplot(x='VIF', y='Variable', data=vif_data)
        plt.axvline(x=5, color='r', linestyle='--', label='VIF=5 (Threshold)')
        plt.axvline(x=10, color='darkred', linestyle='--', label='VIF=10 (Critical)')
        plt.title('Variance Inflation Factors (VIF)')
        plt.legend()
        plt.tight_layout()
        plt.savefig('vif_values.png', dpi=300)
        plt.close()

        # 5. Durbin-Watson test for autocorrelation
```

```
        from statsmodels.stats.stattools import durbin_watson
        dw_stat = durbin_watson(residuals)
        print(f"Durbin-Watson statistic: {dw_stat:.4f}")

        # 6. Breusch-Pagan test for heteroscedasticity
        from statsmodels.stats.diagnostic import het_breuschpagan
        bp_test = het_breuschpagan(residuals, model.model.exog)
        print(f"Breusch-Pagan test p-value: {bp_test[1]:.4f}")

        # 7. Jarque-Bera test for normality
        from statsmodels.stats.diagnostic import jarque_bera
        jb_test = jarque_bera(residuals)
        print(f"Jarque-Bera test p-value: {jb_test[1]:.4f}")

        # Save diagnostic test results
        with open("regression_diagnostic_tests.txt", 'w') as f:
            f.write("Regression Diagnostic Tests Results\n")
            f.write("==================================\n\n")
            f.write(f"Durbin-Watson statistic: {dw_stat:.4f}\n")
            f.write("(Values close to 2 indicate no autocorrelation)\n\n")

            f.write(f"Breusch-Pagan test for heteroscedasticity:\n")
            f.write(f"  LM statistic: {bp_test[0]:.4f}\n")
            f.write(f"  p-value: {bp_test[1]:.4f}\n")
            f.write("  (p < 0.05 indicates heteroscedasticity)\n\n")

            f.write(f"Jarque-Bera test for normality of residuals:\n")
            f.write(f"  JB statistic: {jb_test[0]:.4f}\n")
            f.write(f"  p-value: {jb_test[1]:.4f}\n")
            f.write("  (p < 0.05 indicates non-normality of residuals)\n")

        # Create a leverage plot to identify influential observations
        from statsmodels.graphics.regressionplots import influence_plot

        fig, ax = plt.subplots(figsize=(12, 8))
        influence_plot(model, ax=ax)
        plt.title('Influence Plot (Cook\'s Distance)')
        plt.tight_layout()
        plt.savefig('influence_plot.png', dpi=300)
        plt.close()

        return vif_data

    except Exception as e:
        print(f"Error in regression diagnostics: {e}")
        return None

# Analysis by Restructuring Type
def analyze_by_restructuring_type(df):
    print("\nAnalysis by Restructuring Type:")

    try:
        # Check if we have necessary columns
        required_cols = ['Type of Debt Restructuring', 'Closing Stock Price',
                        'Return on Assets (ROA)', 'Return on Equity (ROE)',
                        'Is_Emerging_Market']

        for col in required_cols:
            if col not in df.columns:
                print(f"Error: Required column '{col}' not found in dataframe.")
                return None

        # Group by restructuring type and calculate mean statistics
        restructuring_analysis = df.groupby('Type of Debt Restructuring').agg({
            'Closing Stock Price': ['mean', 'median', 'std', 'count'],
            'Return on Assets (ROA)': ['mean', 'median'],
            'Return on Equity (ROE)': ['mean', 'median'],
            'Is_Emerging_Market': ['mean', 'sum']  # Proportion and count of emerging market companies
        }).reset_index()

        print(restructuring_analysis)

        # Save analysis to CSV
        restructuring_analysis.to_csv("restructuring_type_analysis.csv")

        # Visualize impact by restructuring type
        plt.figure(figsize=(12, 8))
        # Extract the mean values for Closing Stock Price
        mean_prices = restructuring_analysis[('Closing Stock Price', 'mean')].values
        types = restructuring_analysis['Type of Debt Restructuring'].values
```

```
        # Create a DataFrame for plotting
        plot_df = pd.DataFrame({
            'Restructuring Type': types,
            'Average Stock Price': mean_prices
        })

        # Sort by average stock price
        plot_df = plot_df.sort_values('Average Stock Price', ascending=False)

        sns.barplot(x='Average Stock Price', y='Restructuring Type', data=plot_df)
        plt.title('Average Share Price by Restructuring Type')
        plt.tight_layout()
        plt.savefig('avg_price_by_restructuring.png', dpi=300)
        plt.close()

        # Visualize ROA and ROE by restructuring type
        fig, axes = plt.subplots(1, 2, figsize=(16, 8))

        # Extract means for plotting
        roa_means = restructuring_analysis[('Return on Assets (ROA)', 'mean')].values
        roe_means = restructuring_analysis[('Return on Equity (ROE)', 'mean')].values

        # Create DataFrames for plotting
        roa_df = pd.DataFrame({
            'Restructuring Type': types,
            'Average ROA': roa_means
        }).sort_values('Average ROA', ascending=False)

        roe_df = pd.DataFrame({
            'Restructuring Type': types,
            'Average ROE': roe_means
        }).sort_values('Average ROE', ascending=False)

        # Check for empty DataFrames
        if not roa_df.empty and not roe_df.empty:
            sns.barplot(x='Average ROA', y='Restructuring Type', data=roa_df, ax=axes[0])
            axes[0].set_title('Average ROA by Restructuring Type')

            sns.barplot(x='Average ROE', y='Restructuring Type', data=roe_df, ax=axes[1])
            axes[1].set_title('Average ROE by Restructuring Type')

            plt.tight_layout()
            plt.savefig('financial_metrics_by_restructuring_type.png', dpi=300)
            plt.close()
        else:
            print("Warning: Empty dataframes for ROA/ROE plotting.")

        return restructuring_analysis

    except Exception as e:
        print(f"Error in restructuring type analysis: {e}")
        return None

# Analysis of Emerging vs Developed Markets
def analyze_emerging_markets(df):
    print("\nEmerging Markets Analysis:")

    try:
        # Create a dataframe with only valid market type indicators
        market_df = df.dropna(subset=['Is_Emerging_Market'])

        # Check if there's enough data in each category
        em_count = market_df[market_df['Is_Emerging_Market'] == 1].shape[0]
        dev_count = market_df[market_df['Is_Emerging_Market'] == 0].shape[0]

        print(f"Emerging Markets: {em_count} companies")
        print(f"Developed Markets: {dev_count} companies")

        if em_count < 5 or dev_count < 5:
            print("Warning: Sample size is small for reliable comparison.")

        # Compare key metrics between emerging and developed markets
        market_analysis = market_df.groupby('Is_Emerging_Market').agg({
            'Closing Stock Price': ['mean', 'median', 'std', 'count'],
            'Return on Assets (ROA)': ['mean', 'median'],
            'Return on Equity (ROE)': ['mean', 'median'],
            'Type of Debt Restructuring': 'count'  # Count of restructurings
        }).reset_index()

        # Rename for clarity
```

```
    # Rename for clarity
    market_analysis['Market Type'] = market_analysis['Is_Emerging_Market'].map({
        0: 'Developed', 1: 'Emerging'
    })

    print(market_analysis)

    # Save analysis to CSV
    market_analysis.to_csv("emerging_vs_developed_analysis.csv")

    # Visualize differences in stock prices
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Is_Emerging_Market', y='Closing Stock Price', data=market_df)
    plt.title('Stock Prices: Emerging vs Developed Markets')
    plt.xticks([0, 1], ['Developed', 'Emerging'])
    plt.savefig('stock_prices_by_market_type_boxplot.png', dpi=300)
    plt.close()

    # Visualize differences in ROA and ROE
    fig, axes = plt.subplots(1, 2, figsize=(16, 6))

    sns.boxplot(x='Is_Emerging_Market', y='Return on Assets (ROA)', data=market_df, ax=axes[0])
    axes[0].set_title('ROA: Emerging vs Developed Markets')
    axes[0].set_xticks([0, 1])
    axes[0].set_xticklabels(['Developed', 'Emerging'])

    sns.boxplot(x='Is_Emerging_Market', y='Return on Equity (ROE)', data=market_df, ax=axes[1])
    axes[1].set_title('ROE: Emerging vs Developed Markets')
    axes[1].set_xticks([0, 1])
    axes[1].set_xticklabels(['Developed', 'Emerging'])

    plt.tight_layout()
    plt.savefig('financial_metrics_by_market_type.png', dpi=300)
    plt.close()

    # Compare restructuring types by market
    restructuring_by_market = pd.crosstab(
        market_df['Is_Emerging_Market'],
        market_df['Type of Debt Restructuring'],
        normalize='index'
    ) * 100  # Convert to percentages

    restructuring_by_market.index = ['Developed', 'Emerging']
    restructuring_by_market.to_csv("restructuring_types_by_market.csv")

    # Visualize restructuring types by market
    plt.figure(figsize=(14, 8))
    restructuring_by_market.plot(kind='bar', stacked=True)
    plt.title('Restructuring Types by Market (%)')
    plt.ylabel('Percentage')
    plt.xlabel('Market Type')
    plt.xticks(rotation=0)
    plt.legend(title='Restructuring Type')
    plt.tight_layout()
    plt.savefig('restructuring_types_by_market.png', dpi=300)
    plt.close()

    # T-test for difference in means of stock prices
    t_stat, p_val = stats.ttest_ind(
        market_df[market_df['Is_Emerging_Market'] == 0]['Closing Stock Price'],
        market_df[market_df['Is_Emerging_Market'] == 1]['Closing Stock Price'],
        equal_var=False  # Welch's t-test assuming unequal variances
    )

    print(f"\nT-test for difference in stock prices between developed and emerging markets:")
    print(f"t-statistic: {t_stat:.4f}")
    print(f"p-value: {p_val:.4f}")
    print(f"{'Significant difference' if p_val < 0.05 else 'No significant difference'} at 5% level.")

    # Save statistical test results
    with open("market_comparison_tests.txt", 'w') as f:
        f.write("Statistical Tests for Emerging vs Developed Markets\n")
        f.write("==================================================\n\n")
        f.write("T-test for difference in stock prices:\n")
        f.write(f"t-statistic: {t_stat:.4f}\n")
        f.write(f"p-value: {p_val:.4f}\n")
        f.write(f"{'Significant difference' if p_val < 0.05 else 'No significant difference'} at 5% level.\n\n")

    return market_analysis

except Exception as e:
```

```python
    except Exception as e:
        print(f"Error in emerging markets analysis: {e}")
        return None

# Main function to run all analyses
def main():
    print("Starting debt restructuring analysis...")

    # Create output directory if it doesn't exist
    if not os.path.exists("output"):
        os.makedirs("output")
        print("Created output directory.")

    # Change to output directory
    os.chdir("output")

    # Load and prepare data
    df = load_data()

    if df is not None and not df.empty:
        # Run all analyses
        exploratory_analysis(df)
        stock_price_corr = correlation_analysis(df)
        model, simple_model = regression_analysis(df)
        if model is not None:
            vif_data = regression_diagnostics(model, df)
        restructuring_analysis = analyze_by_restructuring_type(df)
        market_analysis = analyze_emerging_markets(df)

        # Create summary report
        create_summary_report(df, stock_price_corr, model, simple_model,
                              restructuring_analysis, market_analysis)

        print("\nAnalysis completed successfully. Results saved in the output directory.")
    else:
        print("Error: Failed to load or prepare data properly.")

# Create a summary report
def create_summary_report(df, stock_price_corr, model, simple_model,
                         restructuring_analysis, market_analysis):
    print("\nCreating Summary Report...")

    try:
        with open("summary_report.md", 'w') as f:
            f.write("# Debt Restructuring Impact Analysis Report\n\n")
            f.write(f"*Report generated on {pd.Timestamp.now().strftime('%Y-%m-%d')}*\n\n")

            # Dataset overview
            f.write("## Dataset Overview\n\n")
            f.write(f"* Total observations: {len(df)}\n")
            f.write(f"* Number of restructuring types: {df['Type of Debt Restructuring'].nunique()}\n")
            f.write(f"* Number of industries: {df['Industry'].nunique()}\n")
            f.write(f"* Emerging market companies: {df['Is_Emerging_Market'].sum()} ")
            f.write(f"({df['Is_Emerging_Market'].sum() / len(df) * 100:.1f}%)\n\n")

            # Summary statistics
            f.write("## Summary Statistics\n\n")
            f.write("### Key Financial Metrics\n\n")
            f.write(df[['Closing Stock Price', 'Return on Assets (ROA)', 'Return on Equity (ROE)']]
                   .describe().round(2).to_markdown())
            f.write("\n\n")

            # Key findings
            f.write("## Key Findings\n\n")

            # 1. Correlations
            f.write("### 1. Key Correlations with Stock Price\n\n")
            if stock_price_corr is not None:
                top_corr = stock_price_corr.drop('Closing Stock Price').nlargest(5)
                bottom_corr = stock_price_corr.drop('Closing Stock Price').nsmallest(5)

                f.write("**Strongest Positive Correlations:**\n\n")
                for var, corr in top_corr.items():
                    f.write(f"* {var}: {corr:.3f}\n")

                f.write("\n**Strongest Negative Correlations:**\n\n")
                for var, corr in bottom_corr.items():
                    f.write(f"* {var}: {corr:.3f}\n")
            else:
                f.write("Correlation analysis not available.\n")
```

```
        f.write("\n")

        # 2. Regression results
        f.write("### 2. Regression Model Results\n\n")
        if simple_model is not None:
            f.write(f"* R-squared: {simple_model.rsquared:.3f}\n")
            f.write(f"* Adjusted R-squared: {simple_model.rsquared_adj:.3f}\n")
            f.write(f"* F-statistic p-value: {simple_model.f_pvalue:.6f}")
            f.write(" (significant)\n" if simple_model.f_pvalue < 0.05 else "\n")

            f.write("\n**Significant Factors (p < 0.05):**\n\n")
            sig_coefs = pd.DataFrame({
                'Coefficient': simple_model.params,
                'p-value': simple_model.pvalues
            })
            sig_coefs = sig_coefs[sig_coefs['p-value'] < 0.05]

            if not sig_coefs.empty:
                for var, row in sig_coefs.iterrows():
                    f.write(f"* {var}: {row['Coefficient']:.3f} (p={row['p-value']:.4f})\n")
            else:
                f.write("No coefficients were significant at the 5% level.\n")
        else:
            f.write("Regression analysis not available.\n")

        f.write("\n")

        # 3. Restructuring type impact
        f.write("### 3. Impact by Restructuring Type\n\n")
        if restructuring_analysis is not None:
            # Create a simpler version of the restructuring analysis for reporting
            types = restructuring_analysis['Type of Debt Restructuring'].values
            prices = restructuring_analysis[('Closing Stock Price', 'mean')].values
            counts = restructuring_analysis[('Closing Stock Price', 'count')].values

            # Create a table
            f.write("| Restructuring Type | Avg. Stock Price | Count |\n")
            f.write("|-------------------|------------------|-------|\n")
            for t, p, c in zip(types, prices, counts):
                f.write(f"| {t} | {p:.2f} | {int(c)} |\n")
        else:
            f.write("Restructuring type analysis not available.\n")

        f.write("\n")

        # 4. Emerging vs Developed markets
        f.write("### 4. Emerging vs. Developed Markets\n\n")
        if market_analysis is not None:
            f.write("**Average Key Metrics by Market Type:**\n\n")
            f.write("| Market Type | Avg. Stock Price | Avg. ROA | Avg. ROE | Count |\n")
            f.write("|------------|------------------|----------|----------|-------|\n")

            for _, row in market_analysis.iterrows():
                market = row['Market Type']
                price = row[('Closing Stock Price', 'mean')]
                roa = row[('Return on Assets (ROA)', 'mean')]
                roe = row[('Return on Equity (ROE)', 'mean')]
                count = row[('Closing Stock Price', 'count')]

                f.write(f"| {market} | {price:.2f} | {roa:.2f} | {roe:.2f} | {int(count)} |\n")
        else:
            f.write("Market comparison analysis not available.\n")

        f.write("\n")

        # 5. Conclusion
        f.write("## Conclusion\n\n")
        f.write("Based on the analysis, the following conclusions can be drawn:\n\n")

        # Add placeholders for conclusions that would be filled in based on actual results
        f.write("1. [Add key finding about most impactful restructuring types]\n")
        f.write("2. [Add key finding about financial metrics relationship with stock prices]\n")
        f.write("3. [Add key finding about emerging vs. developed markets differences]\n")
        f.write("4. [Add key finding about industry effects, if significant]\n\n")

        f.write("## Recommendations\n\n")
        f.write("Based on the analysis, we recommend:\n\n")
        f.write("1. [Add recommendation based on findings]\n")
        f.write("2. [Add recommendation based on findings]\n")
        f.write("3. [Add recommendation based on findings]\n\n")
```

```
            f.write("## Limitations and Further Research\n\n")
            f.write("This analysis has several limitations that should be considered:\n\n")
            f.write("1. [Add limitation of the analysis]\n")
            f.write("2. [Add limitation of the analysis]\n\n")

            f.write("Future research should:\n\n")
            f.write("1. [Add suggestion for future research]\n")
            f.write("2. [Add suggestion for future research]\n")

        print("Summary report created successfully.")

    except Exception as e:
        print(f"Error creating summary report: {e}")

# Run the analysis if this file is executed directly
if __name__ == "__main__":
    main()
```

```
Starting debt restructuring analysis...
Created output directory.
Basic Statistics:
       Closing Stock Price  Return on Assets (ROA)  Return on Equity (ROE)
count            44.000000               44.000000               44.000000
mean              5.816364                0.954545                2.422727
std               7.305819                2.102319                5.218877
min               0.000000               -5.300000              -12.100000
25%               2.062500                0.250000                1.150000
50%               3.800000                1.500000                3.800000
75%               6.975000                2.225000                5.225000
max              42.300000                4.200000               12.300000
<ipython-input-6-6479c1bbbb97>:138: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_tic
  axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=45)
<ipython-input-6-6479c1bbbb97>:142: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_tic
  axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation=45)

Correlation Analysis:
Correlations with Closing Stock Price:
Closing Stock Price                          1.000000
Restructuring_Government Bailout             0.770348
Industry_Automobile                          0.520703
Return on Equity (ROE)                       0.466946
Return on Assets (ROA)                       0.330389
Market Index Value                           0.266329
Industry_Entertainment                       0.188653
Industry_Steel                               0.136901
Restructuring_Bankruptcy Reorganization      0.063971
Industry_Airlines                            0.062200
Industry_Fashion                             0.058776
Restructuring_Debt Refinancing               0.052381
Industry_Energy                              0.049325
Industry_Hospitality                         0.022166
Trading Volume                               0.014738
Industry_Metals & Mining                     0.013102
Industry_Telecom                            -0.011825
Industry_Photography                        -0.021460
Industry_Utilities                          -0.042575
Industry_Aerospace                          -0.042575
Industry_Automotive                         -0.042575
Restructuring_Chapter 11                    -0.044309
Industry_Aviation                           -0.048678
Industry_Health                             -0.070025
Restructuring_Debt-for-Equity Swap          -0.070237
Industry_Music                              -0.084805
Industry_Construction                       -0.085861
Industry_Consumer Goods                     -0.091140
Restructuring_Refinancing                   -0.105518
Industry_Travel                             -0.113310
Restructuring_Bankruptcy                    -0.146557
Is_Emerging_Market                          -0.174626
Industry_Retail                             -0.175063
Industry_Real Estate                        -0.190127
Name: Closing Stock Price, dtype: float64

Regression Analysis:
                              OLS Regression Results
==================================================================================
Dep. Variable:     Q("Closing Stock Price")   R-squared:                       0.945
Model:                                  OLS   Adj. R-squared:                  0.817
Method:                       Least Squares   F-statistic:                     7.381
Date:                      Sun, 06 Apr 2025   Prob (F-statistic):           0.000234
Time:                              20:42:56   Log-Likelihood:                -85.801
No. Observations:                        44   AIC:                             233.6
Df Residuals:                            13   BIC:                             288.9
Df Model:                                30
Covariance Type:                  nonrobust
====================================================================================================================
                                                          coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------
Intercept                                               1.2333      3.803      0.324      0.751      -6.982       9.449
Q("Restructuring_Bankruptcy")[T.True]                   3.8481      3.195      1.204      0.250      -3.054      10.750
Q("Restructuring_Bankruptcy Reorganization")[T.True]    3.8069      1.932      1.971      0.070      -0.367       7.981
Q("Restructuring_Chapter 11")[T.True]                   6.5209      2.957      2.206      0.046       0.134      12.908
Q("Restructuring_Debt Refinancing")[T.True]             6.6087      2.827      2.337      0.036       0.501      12.717
Q("Restructuring_Debt-for-Equity Swap")[T.True]        -1.0050      2.720     -0.369      0.718      -6.882       4.872
Q("Restructuring_Government Bailout")[T.True]          23.3011      5.399      4.316      0.001      11.637      34.965
Q("Industry_Aerospace")[T.True]                      1.038e-13      4.425   2.35e-14      1.000      -9.560       9.560
Q("Industry_Airlines")[T.True]                         -1.5740      4.699     -0.335      0.743     -11.726       8.578
Q("Industry_Automobile")[T.True]                      1.99e-13      4.425    4.5e-14      1.000      -9.560       9.560
Q("Industry_Automotive")[T.True]                        1.9163      4.478      0.428      0.676      -7.758      11.591
Q("Industry_Aviation")[T.True]                         -1.6338      3.844     -0.425      0.678      -9.938       6.670
Q("Industry_Construction")[T.True]                      5.1656      5.673      0.911      0.379      -7.090      17.421
Q("Industry_Consumer Goods")[T.True]                   -5.0988      5.274     -0.967      0.351     -16.492       6.294
Q("Industry_Energy")[T.True]                            4.9334      4.774      1.033      0.320      -5.380      15.247
Q("Industry_Entertainment")[T.True]                     7.4921      3.826      1.958      0.072      -0.773      15.757
Q("Industry_Fashion")[T.True]                           8.1164      5.117      1.586      0.137      -2.939      19.172
```