```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt


data = pd.read_csv("dataset.csv", encoding='cp1252')
data.fillna(data.mean(), inplace=True)
```

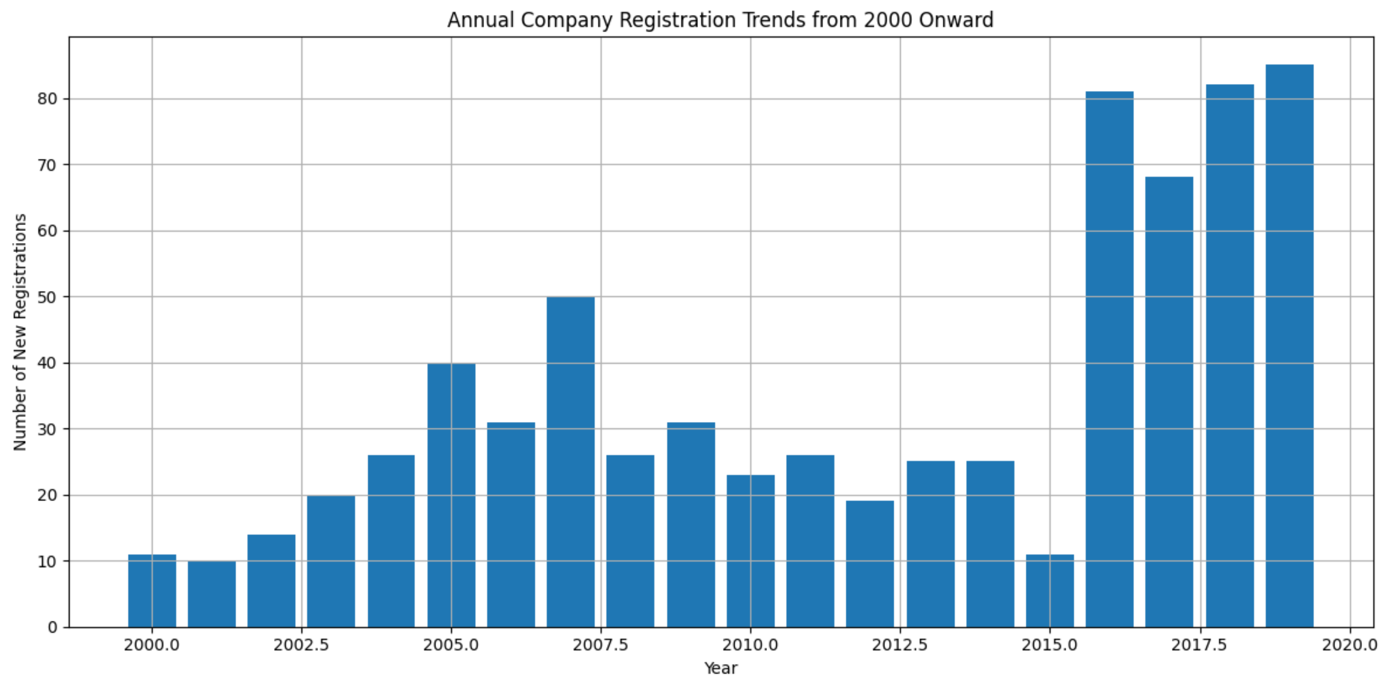
```
<ipython-input-52-e5efee82830c>:2: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future versio
  data.fillna(data.mean(), inplace=True)
```

```
data['DATE_OF_REGISTRATION'] = pd.to_datetime(data['DATE_OF_REGISTRATION'])
start_year =2000
data = data[data['DATE_OF_REGISTRATION'].dt.year >= start_year]
```

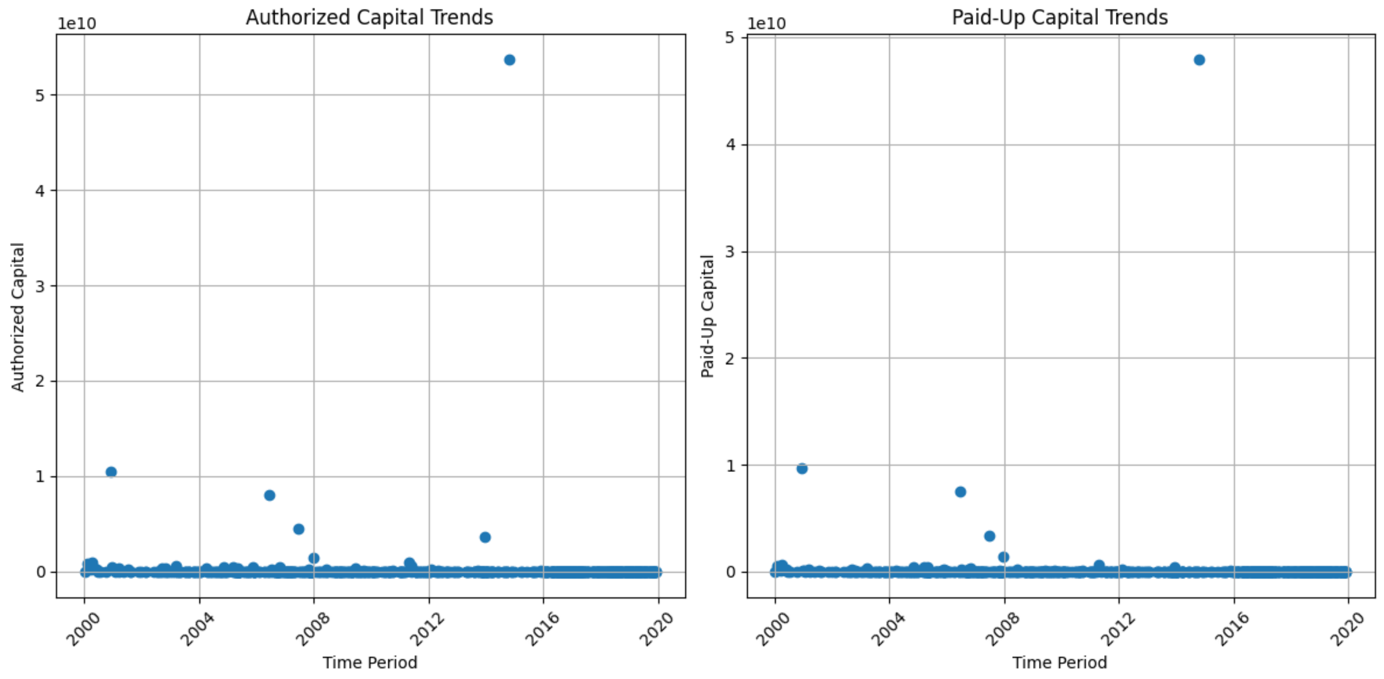
```
<ipython-input-53-1c05663a0baa>:1: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. Thi
  data['DATE_OF_REGISTRATION'] = pd.to_datetime(data['DATE_OF_REGISTRATION'])
```

```
monthly_registrations = data.groupby(data['DATE_OF_REGISTRATION'].dt.to_period("M")).size()
subsampling_factor = 3# Adjust as needed to control the number of data points
subsetted_data = monthly_registrations.iloc[::subsampling_factor]
```

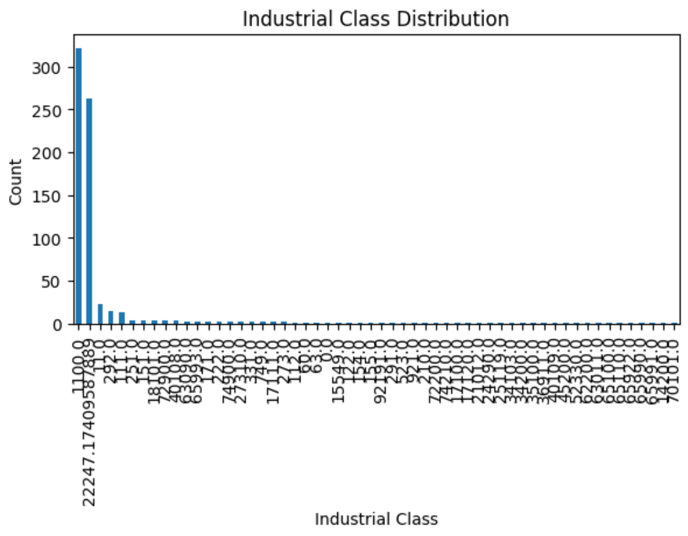
```
yearly_registrations = data.groupby(data['DATE_OF_REGISTRATION'].dt.year).size()

# Create a bar plot to visualize the annual registration trends
plt.figure(figsize=(12, 6))
plt.bar(yearly_registrations.index, yearly_registrations.values, align='center')
plt.title('Annual Company Registration Trends from 2000 Onward')
plt.xlabel('Year')
plt.ylabel('Number of New Registrations')
plt.grid(True)

# Display the plot
plt.tight_layout()
plt.show()
```

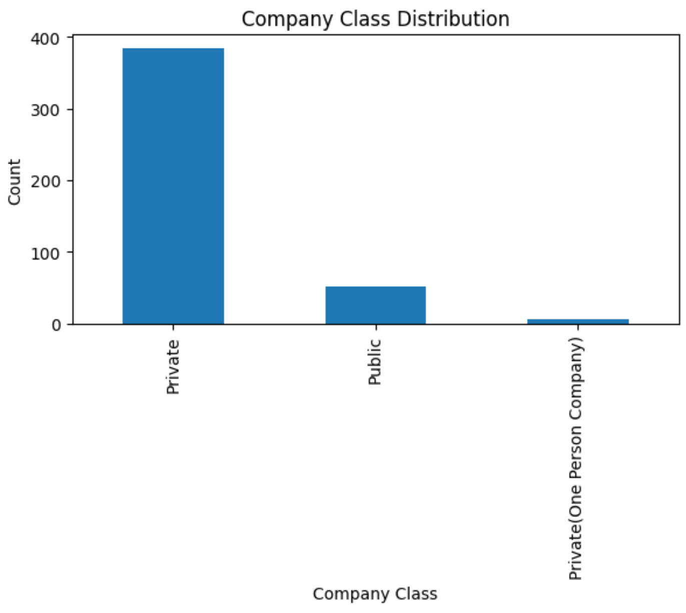


```
# Analyze trends in other columns, for example, 'AUTHORIZED_CAP' and 'PAIDUP_CAPITAL'
plt.figure(figsize=(12, 6))

# Plot Authorized Capital trends
plt.subplot(1, 2, 1)
plt.scatter(data['DATE_OF_REGISTRATION'], data['AUTHORIZED_CAP'], marker='o')
plt.title('Authorized Capital Trends')
plt.xlabel('Time Period')
plt.ylabel('Authorized Capital')
plt.grid(True)
plt.xticks(rotation=45)

# Plot Paid-Up Capital trends
plt.subplot(1, 2, 2)
plt.scatter(data['DATE_OF_REGISTRATION'], data['PAIDUP_CAPITAL'], marker='o')
plt.title('Paid-Up Capital Trends')
plt.xlabel('Time Period')
plt.ylabel('Paid-Up Capital')
plt.grid(True)
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



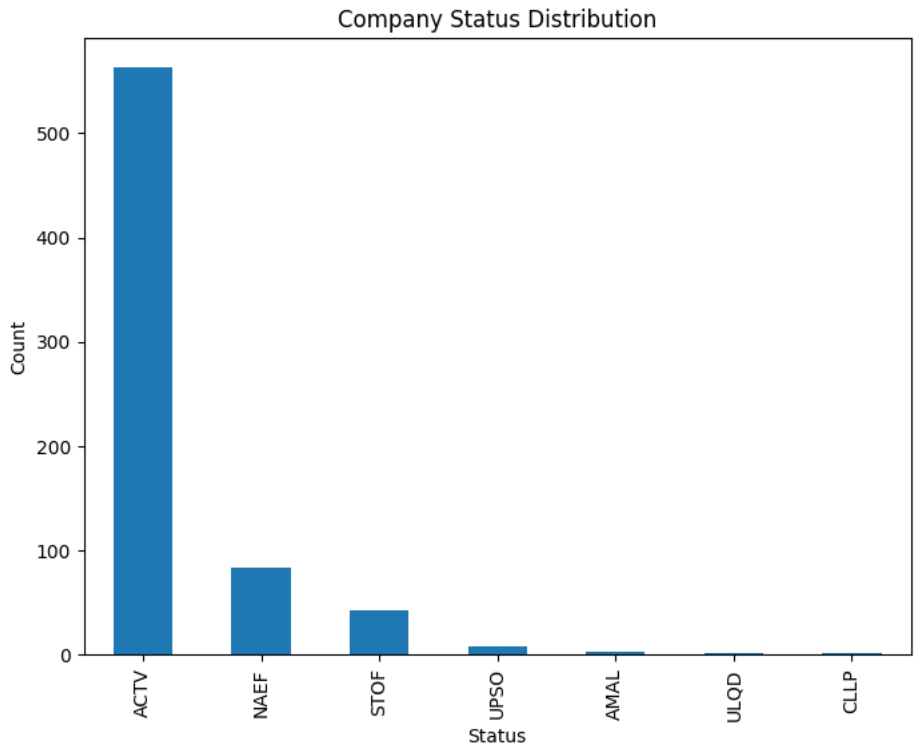
```
plt.figure(figsize=(12, 8))

# Plot Industrial Class trends
plt.subplot(2, 2, 1)
data['INDUSTRIAL_CLASS'].value_counts().plot(kind='bar')
plt.title('Industrial Class Distribution')
plt.xlabel('Industrial Class')
plt.ylabel('Count')

# Plot Company Class trends
plt.subplot(2, 2, 2)
data['COMPANY_CLASS'].value_counts().plot(kind='bar')
plt.title('Company Class Distribution')
plt.xlabel('Company Class')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```





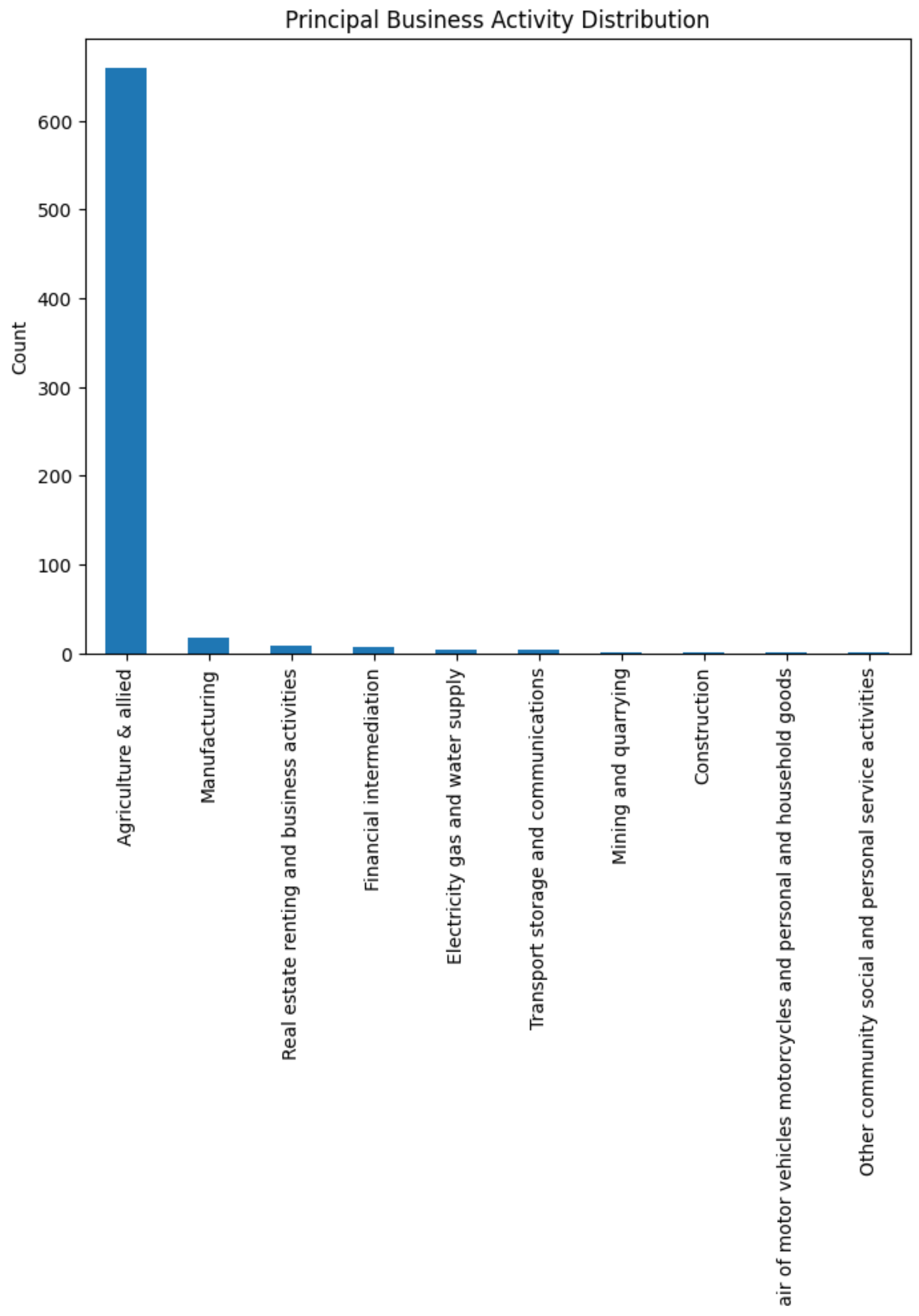
```
# Analyze trends in two additional columns, for example, 'COMPANY_STATUS' and 'COMPANY_CLASS'
plt.figure(figsize=(12, 8))

# Plot Company Status trends

data['COMPANY_STATUS'].value_counts().plot(kind='bar', figsize=(8, 6))
plt.title('Company Status Distribution')
plt.xlabel('Status')
plt.ylabel('Count')
```

```
    Text(0, 0.5, 'Count')
```



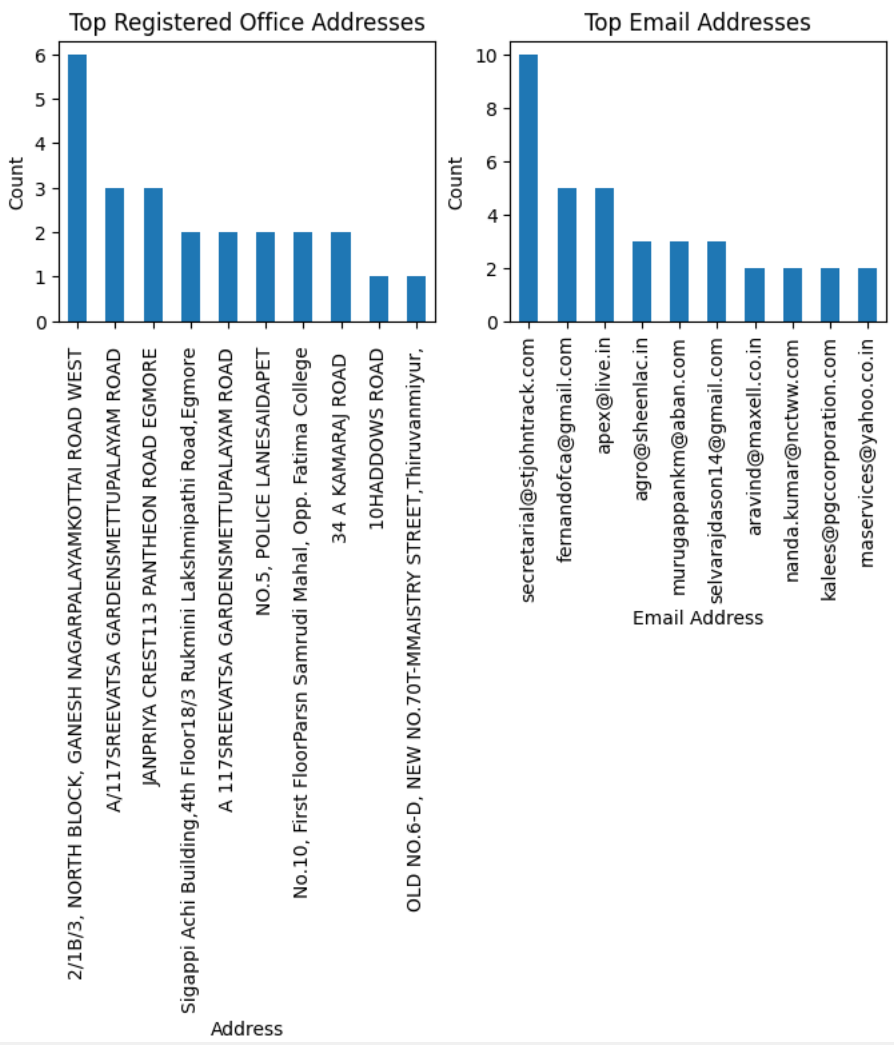
```
plt.figure(figsize=(12, 8))

# Plot Principal Business Activity Distribution

data['PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN'].value_counts().plot(kind='bar', figsize=(8, 6))
plt.title('Principal Business Activity Distribution')
plt.xlabel('Business Activity')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



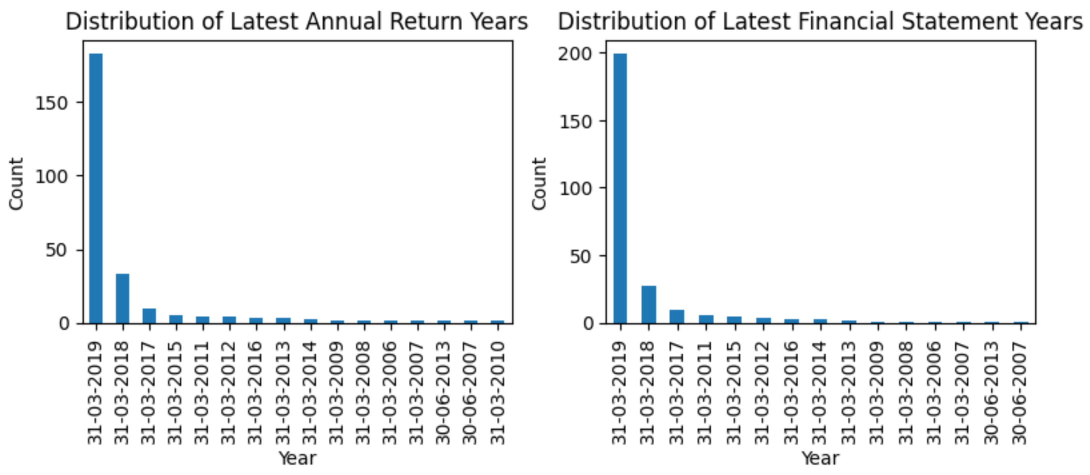
```
plt.figure(figsize=(12, 8))

# Plot Registered Office Address trends
plt.subplot(2, 2, 1)
data['REGISTERED_OFFICE_ADDRESS'].value_counts().head(10).plot(kind='bar', figsize=(8, 6))
plt.title('Top Registered Office Addresses')
plt.xlabel('Address')
plt.ylabel('Count')

# Plot Email Address trends
plt.subplot(2, 2, 2)
data['EMAIL_ADDR'].value_counts().head(10).plot(kind='bar', figsize=(8, 6))
plt.title('Top Email Addresses')
plt.xlabel('Email Address')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

```
<ipython-input-42-1db337c18803>:17: UserWarning: Tight layout not applied. tight_layout cannot make axes height small enough to accommo
  plt.tight_layout()
```

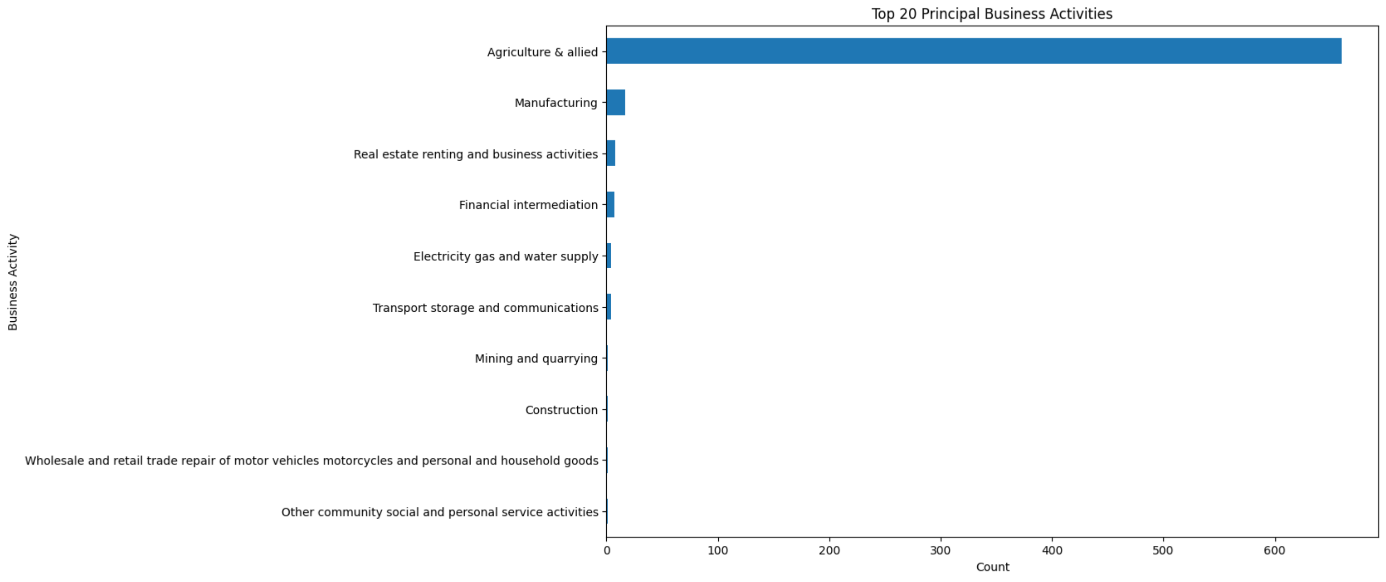


```
plt.figure(figsize=(12, 8))

# Plot the distribution of the latest annual return years
plt.subplot(2, 2, 1)
data['LATEST_YEAR_ANNUAL_RETURN'].value_counts().plot(kind='bar', figsize=(8, 6))
plt.title('Distribution of Latest Annual Return Years')
plt.xlabel('Year')
plt.ylabel('Count')

# Plot the distribution of the latest financial statement years
plt.subplot(2, 2, 2)
data['LATEST_YEAR_FINANCIAL_STATEMENT'].value_counts().plot(kind='bar', figsize=(8, 6))
```

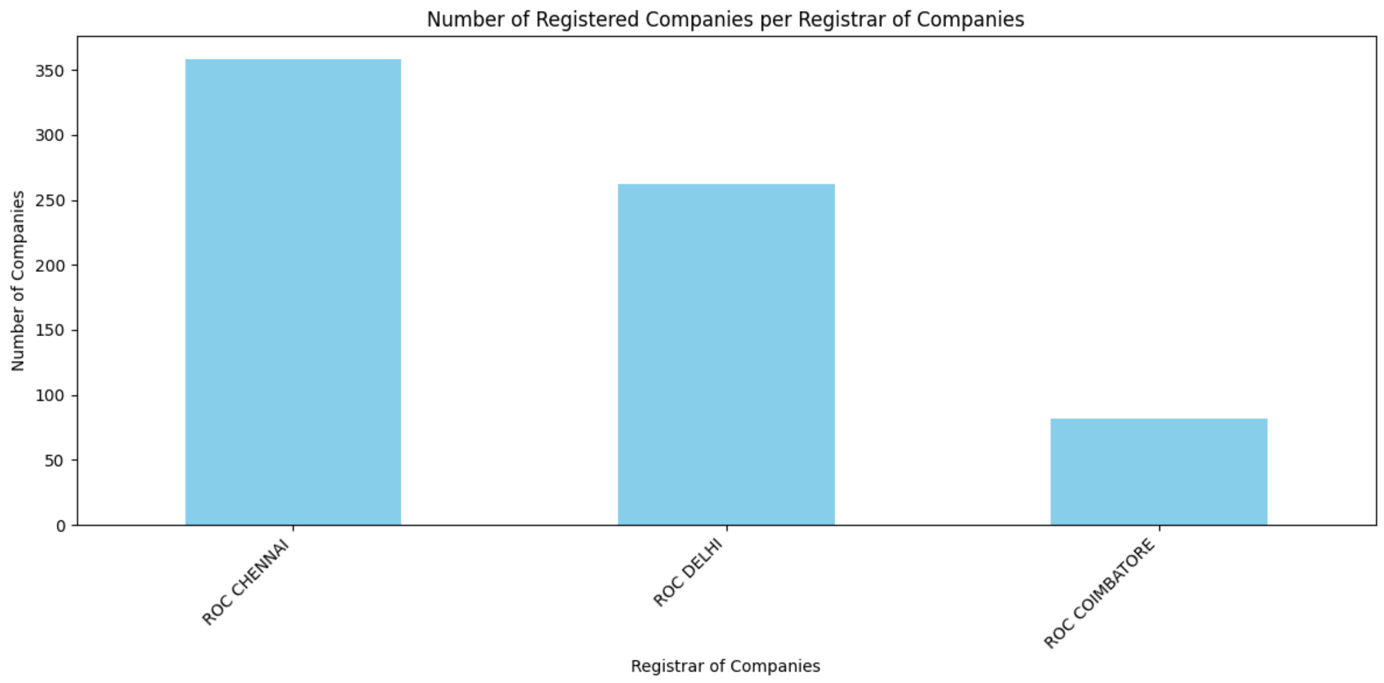
```
plt.title('Distribution of Latest Financial Statement Years')
plt.xlabel('Year')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```



```
business_activities = data['PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN'].value_counts()

# Visualize the distribution of business activities
plt.figure(figsize=(12, 8))
business_activities.head(20).plot(kind='barh')
plt.title('Top 20 Principal Business Activities')
plt.xlabel('Count')
plt.ylabel('Business Activity')
plt.gca().invert_yaxis()  # Invert the y-axis to show the most common activities at the top
plt.show()
```



```
roc_counts = data['REGISTRAR_OF_COMPANIES'].value_counts()

# Create a bar chart to visualize the number of registered companies per Registrar of Companies
plt.figure(figsize=(12, 6))
roc_counts.plot(kind='bar', color='skyblue')
plt.title('Number of Registered Companies per Registrar of Companies')
plt.xlabel('Registrar of Companies')
plt.ylabel('Number of Companies')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()

# Show the plot
plt.show()
```



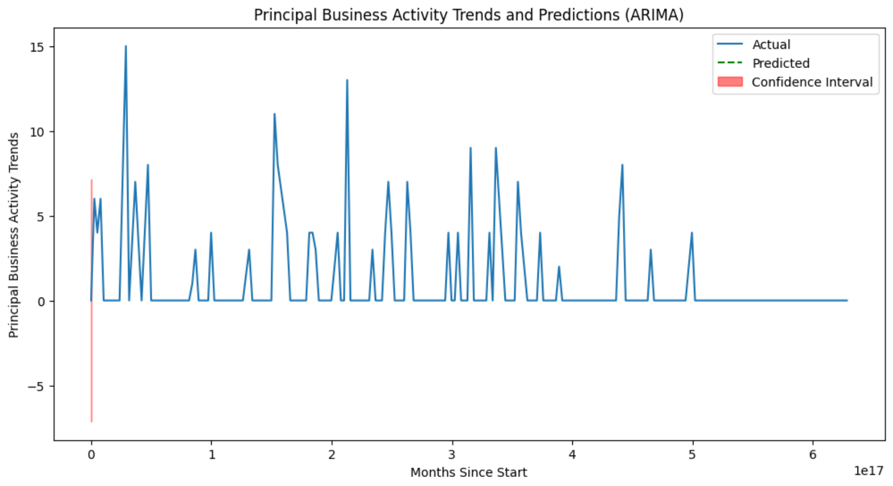
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data['COMPANY_STATUS'] = label_encoder.fit_transform(data['COMPANY_STATUS'])
data['COMPANY_CLASS'] = label_encoder.fit_transform(data['COMPANY_CLASS'])
data['COMPANY_CATEGORY'] = label_encoder.fit_transform(data['COMPANY_CATEGORY'])
data['COMPANY_SUB_CATEGORY'] = label_encoder.fit_transform(data['COMPANY_SUB_CATEGORY'])
data['REGISTERED_STATE'] = label_encoder.fit_transform(data['REGISTERED_STATE'])
data['INDUSTRIAL_CLASS'] = label_encoder.fit_transform(data['INDUSTRIAL_CLASS'])
data['PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN'] = label_encoder.fit_transform(data['PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN'])
data['LATEST_YEAR_ANNUAL_RETURN'] = label_encoder.fit_transform(data['LATEST_YEAR_ANNUAL_RETURN'])
data['LATEST_YEAR_FINANCIAL_STATEMENT'] = label_encoder.fit_transform(data['LATEST_YEAR_FINANCIAL_STATEMENT'])
data.fillna(data.mean(), inplace=True)
print(data.mean())
```

```
    COMPANY_STATUS                            6.889205e-01
    COMPANY_CLASS                             1.272727e+00
    COMPANY_CATEGORY                          3.721591e-01
    COMPANY_SUB_CATEGORY                      7.471591e-01
    REGISTERED_STATE                          0.000000e+00
    AUTHORIZED_CAP                            1.354964e+08
    PAIDUP_CAPITAL                            1.114212e+08
    INDUSTRIAL_CLASS                          2.365625e+01
    PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN    2.926136e-01
    LATEST_YEAR_ANNUAL_RETURN                 1.527841e+01
    LATEST_YEAR_FINANCIAL_STATEMENT           1.427557e+01
    dtype: float64
    <ipython-input-64-7409e9b4c871>:15: FutureWarning: DataFrame.mean and DataFrame.median with numeric_only=None will include datetime64 a
      data.fillna(data.mean(), inplace=True)
    <ipython-input-64-7409e9b4c871>:15: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future versi
      data.fillna(data.mean(), inplace=True)
    <ipython-input-64-7409e9b4c871>:16: FutureWarning: DataFrame.mean and DataFrame.median with numeric_only=None will include datetime64 a
      print(data.mean())
    <ipython-input-64-7409e9b4c871>:16: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future versi
      print(data.mean())
```

```
import pandas as pd
```

```
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
data['DATE_OF_REGISTRATION'] = pd.to_datetime(data['DATE_OF_REGISTRATION'])
data['YearMonth'] = data['DATE_OF_REGISTRATION'].dt.to_period('M')
data['YearMonthDatetime'] = data['YearMonth'].dt.to_timestamp()
data['MonthsSinceStart'] = (data['YearMonthDatetime'] - data['YearMonthDatetime'].min())
activity_counts = data.groupby('MonthsSinceStart')['PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN'].sum()
model = ARIMA(activity_counts, order=(5, 1, 0))
results = model.fit()
forecast = results.get_forecast(steps=12)
forecast_values = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()
plt.figure(figsize=(12, 6))
plt.plot(activity_counts.index, activity_counts.values, label='Actual')
plt.plot(forecast_values.index, forecast_values, linestyle='--', color='green', label='Predicted')
plt.fill_between(forecast_conf_int.index, forecast_conf_int.iloc[:, 0], forecast_conf_int.iloc[:, 1], color='red', alpha=0.5, label='Confider
plt.xlabel('Months Since Start')
plt.ylabel('Principal Business Activity Trends')
plt.title('Principal Business Activity Trends and Predictions (ARIMA)')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWar
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWar
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWar
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: ValueWar
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: FutureWa
  return get_prediction_index(
```

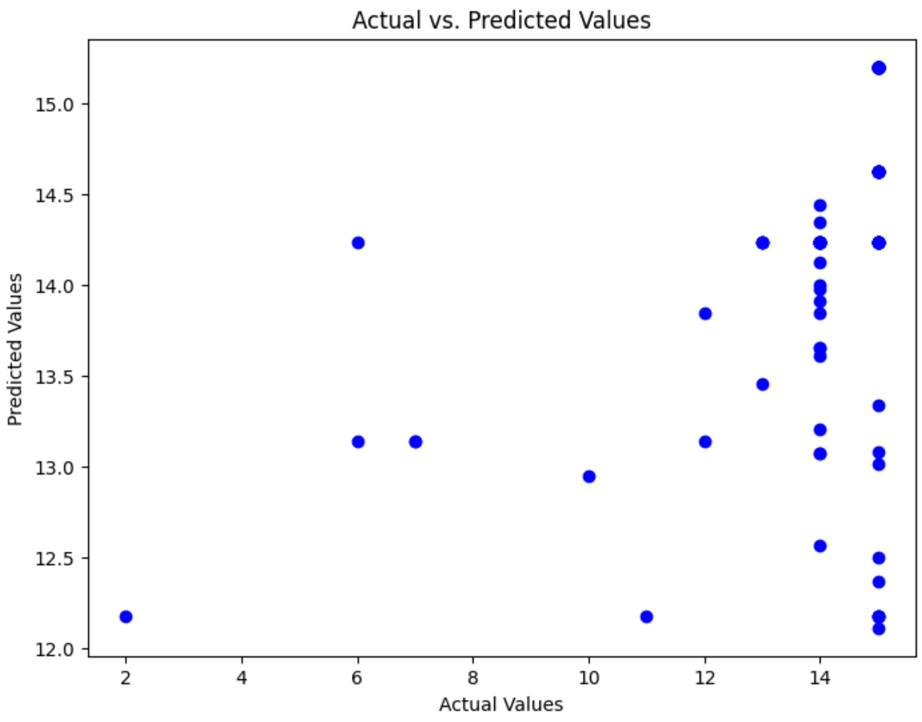


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

independent_vars = ['COMPANY_STATUS', 'COMPANY_CLASS', 'COMPANY_CATEGORY', 'COMPANY_SUB_CATEGORY',
                    'INDUSTRIAL_CLASS', 'PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN']
dependent_var = 'LATEST_YEAR_FINANCIAL_STATEMENT'
X = data[independent_vars]
y = data[dependent_var]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

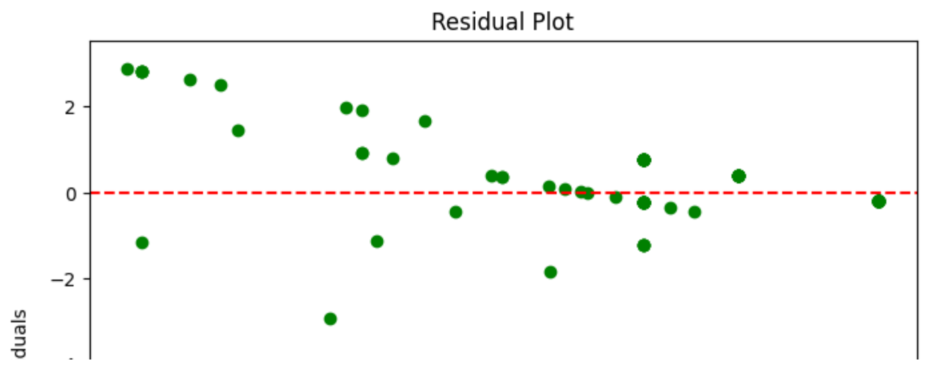
```
y_pred = model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
     Mean Squared Error: 2.8628794989463033
     R-squared: 0.17920934300658375
```

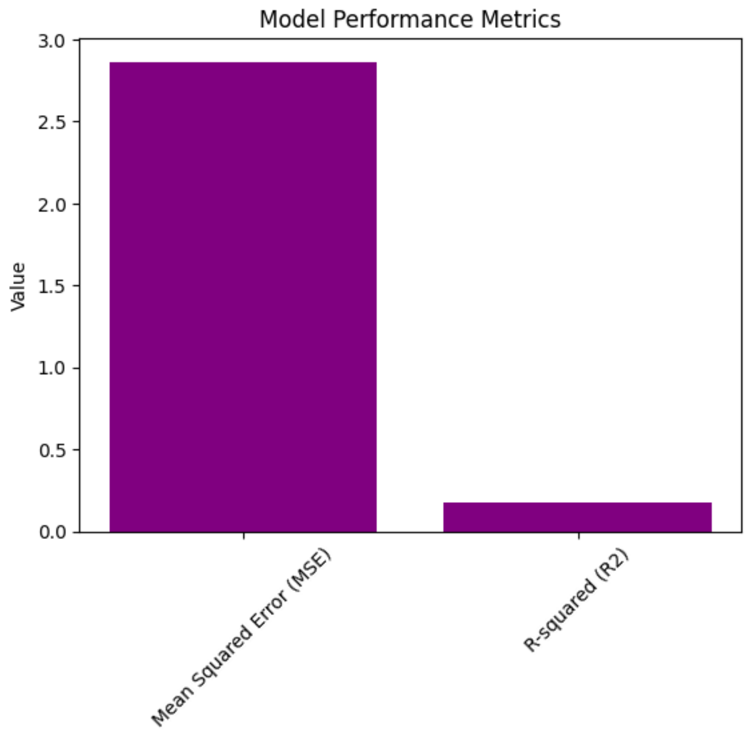
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```



```
plt.figure(figsize=(8, 6))
residuals = y_test - y_pred
plt.scatter(y_pred, residuals, color='green')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.axhline(0, color='red', linestyle='--')
plt.show()
```



```
performance_metrics = {'Mean Squared Error (MSE)': mse, 'R-squared (R2)': r2}
plt.bar(performance_metrics.keys(), performance_metrics.values(), color='purple')
plt.ylabel('Value')
plt.title('Model Performance Metrics')
plt.xticks(rotation=45)
plt.show()
```


Model Performance Metrics
Value
Mean Squared Error (MSE)
R-squared (R2)