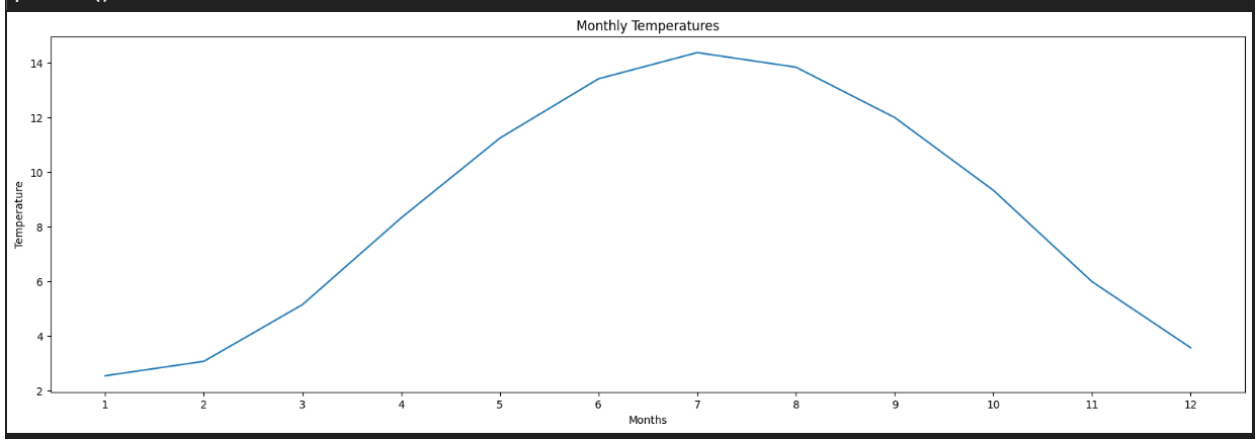
Data Collection and Preprocessing Phase

Date	9 July 2024
Team ID	SWTID1720111029
Project Title	Unveiling Climate Change Dynamics through Earth Surface Temperature Analysis
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

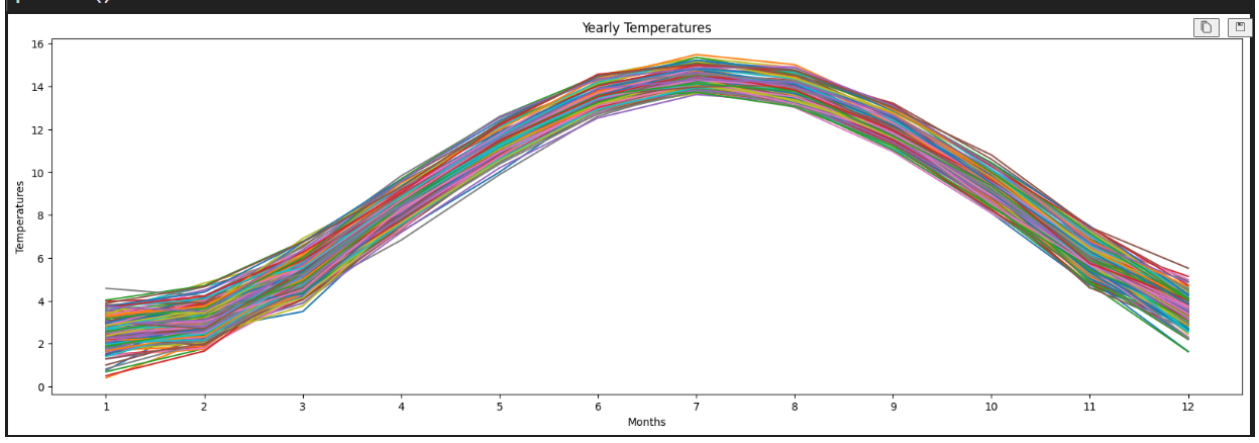
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	RangeIndex: 3192 entries, 0 to 3191 Data columns (total 9 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 dt 3192 non-null object 1 LandAverageTemperature 3180 non-null float64 2 LandAverageTemperatureUncertainty 3180 non-null float64 3 LandMaxTemperature 1992 non-null float64 4 LandMaxTemperatureUncertainty 1992 non-null float64 5 LandMinTemperature 1992 non-null float64 6 LandMinTemperatureUncertainty 1992 non-null float64 7 LandAndOceanAverageTemperature 1992 non-null float64 8 LandAndOceanAverageTemperatureUncertainty 1992 non-null float64 dtypes: float64(8), object(1) memory usage: 224.6+ KB

Univariate Analysis	`pivot = data.pivot_table(values='LandAverageTemperature', index=data.index.year, columns=data.index.month)`
	`# Plot the monthly seasonality`
	`monthly_seasonality = pivot.mean(axis=0)`
	`monthly_seasonality.plot(figsize=(20, 6))`
	`plt.title('Monthly Temperatures')`
	`plt.xlabel('Months')`
	`plt.ylabel('Temperature')`
	`plt.xticks(range(1, 13))`
	`plt.show()`
	 
	
	Seasonal fluctuations in temperature on a monthly basis. Monthly temperature variations reflecting seasonal patterns.

Bivariate Analysis

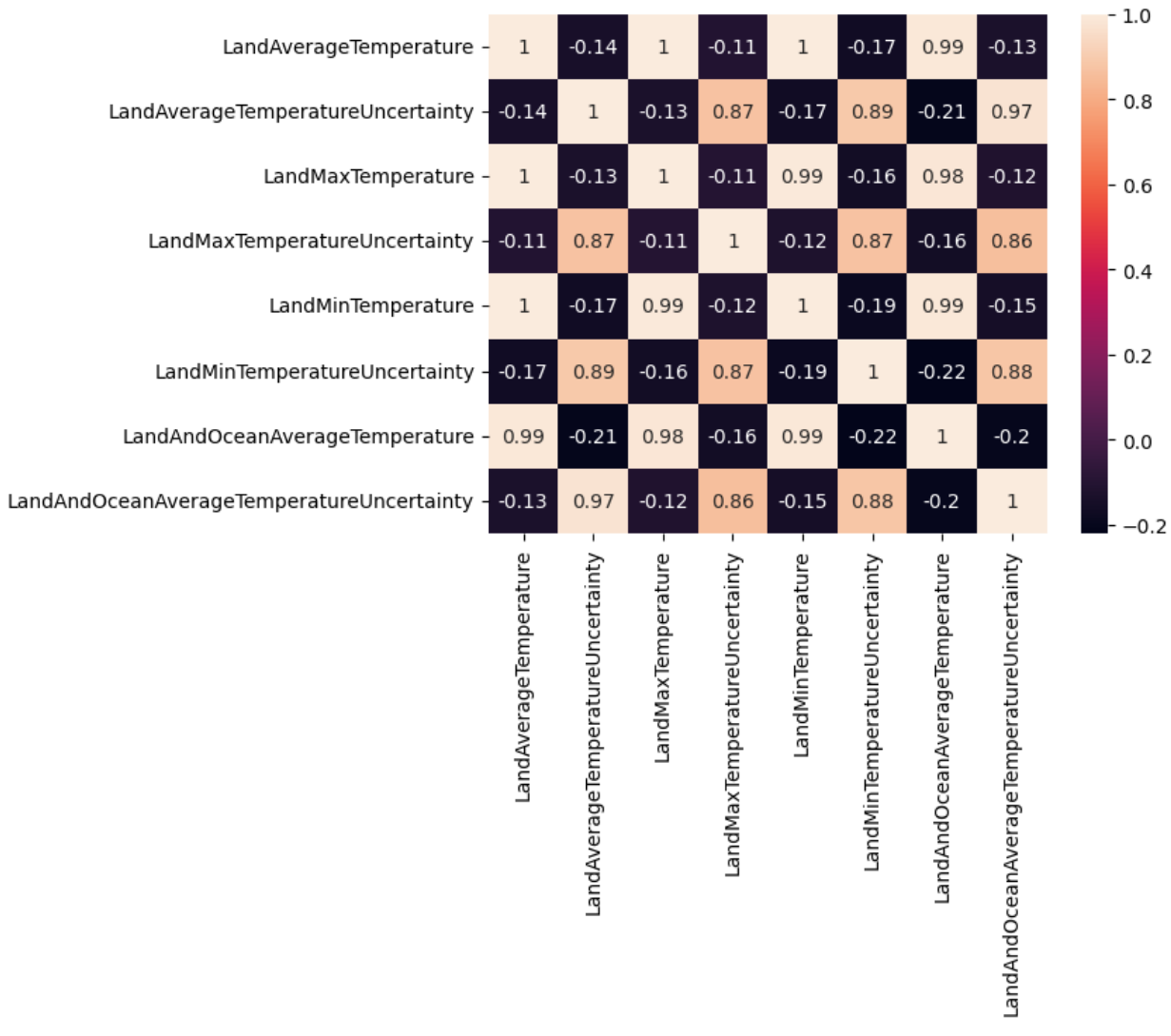
```python
# Extract the month from the index and create a new 'month' column
data['month'] = data.index.month

# Extract the year from the index and create a new 'year' column
data['year'] = data.index.year

# Create a pivot table with 'LandAverageTemperature' as values,
# months as rows (index), and years as columns, aggregating by mean
pivot = pd.pivot_table(data, values='LandAverageTemperature', index='month', columns='year', aggfunc='mean')

# Plot the pivot table
pivot.plot(figsize=(20, 6))

# Set the title of the plot
plt.title('Yearly Temperatures')

# Set the x-axis label
plt.xlabel('Months')

# Set the y-axis label
plt.ylabel('Temperatures')

# Set the x-axis ticks to represent the months (1 to 12)
plt.xticks(range(1, 13))

# Remove the legend
plt.legend().remove()

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



The mean temperatures across the land for each of the 12 months over multiple years.

Multivariate Analysis

```python
# Correlation Heatmap

# Calculate the correlation matrix of the DataFrame, considering only numeric columns
hm = df.corr(numeric_only=True)

# Create a heatmap of the correlation matrix with annotations
sns.heatmap(hm, annot=True)

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



Data Preprocessing Code Screenshots

Loading Data

```
df = pd.read_csv('GlobalTemperatures.csv')
```

Handling Missing Data

Missing values in LandAverageTemperature and LandAverageTemperatureUncertainity columns were replaced with the mean and missing values in other columns were dropped.

```
df.isnull().sum()
```

```
dt                                             0
LandAverageTemperature                        12
LandAverageTemperatureUncertainty             12
LandMaxTemperature                          1200
LandMaxTemperatureUncertainty               1200
LandMinTemperature                          1200
LandMinTemperatureUncertainty               1200
LandAndOceanAverageTemperature              1200
LandAndOceanAverageTemperatureUncertainty   1200
dtype: int64
```

```
# Impute LandAverageTemperature and LandAverageTemperatureUncertainty with mean

# Fill missing values in the 'LandAverageTemperature' column with the mean of that column
df['LandAverageTemperature'].fillna(df['LandAverageTemperature'].mean(), inplace=True)

# Fill missing values in the 'LandAverageTemperatureUncertainty' column with the mean of that column
df['LandAverageTemperatureUncertainty'].fillna(df['LandAverageTemperatureUncertainty'].mean(), inplace=True)
```

```
# For columns with 1200 missing values, drop those rows

# List of columns to check for missing values
cols_to_dropna = ['LandMaxTemperature', 'LandMaxTemperatureUncertainty', 'LandMinTemperature', 'LandMinTemperatureUncertainty', 'LandAndOceanAverageTemperature', 'LandAndOceanAverageTemperatureUncertainty']

# Loop through each column in the list
for col in cols_to_dropna:
    # Drop rows where the specified column has missing values
    df.dropna(subset=[col], inplace=True)
```

```
# Verify if there are any remaining missing values
print(df.isnull().sum())
```

```
dt                                             0
LandAverageTemperature                         0
LandAverageTemperatureUncertainty              0
LandMaxTemperature                             0
LandMaxTemperatureUncertainty                  0
LandMinTemperature                             0
LandMinTemperatureUncertainty                  0
LandAndOceanAverageTemperature                 0
LandAndOceanAverageTemperatureUncertainty      0
dtype: int64
```

Data Transformation

Scaling the Xtrain, Xtest , Ytrain & Ytest values to be suitable for LSTM model.

```
# Scale X and y using MinMaxScaler
scaler_x = MinMaxScaler()
scaler_y = MinMaxScaler()

# Fit and transform the training data
X_train_scaled = scaler_x.fit_transform(X_train)
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1))  # Reshape y_train to a 2D array

# Only transform the testing data
X_test_scaled = scaler_x.transform(X_test)
y_test_scaled = scaler_y.transform(y_test.values.reshape(-1, 1))  # Reshape y_test to a 2D array
```

Feature Engineering

Additionally adding year and month columns on the basis of the date column.

```
# Add Year and Month columns based on 'dt' column
df['Year'] = pd.to_datetime(df['dt']).dt.year
df['Month'] = pd.to_datetime(df['dt']).dt.month
```

Save Processed Data	```python
	# Save the scaled data
	joblib.dump({
	    'X_train_scaled': X_train_scaled,
	    'y_train_scaled': y_train_scaled,
	    'X_test_scaled': X_test_scaled,
	    'y_test_scaled': y_test_scaled
	}, 'scaled_data.pkl')
	``
	Save the scaled data