MEASURE ENERGY CONSUMPTION

NAME: M.RAGHUL

REG NO: 110521106327

PROJECT TITTLE: **MEASURE ENERGY CONSUMPTION**

PHASE 3: **Development Part 1**

TOPIC: **TO measure energy consumption by loading and pre-processing the data set.**

INTRODUCTION:

- **Measuring energy consumption is essential for understanding and managing the use of energy resources in various settings, from individual homes to large industrial facilities. Accurate energy consumption measurements are crucial for optimizing energy efficiency, reducing costs, and minimizing environmental impact. This introduction provides an overview of the key concepts and methods involved in measuring energy consumption.**

- **Cost Management:** Energy is a significant expense for households and businesses. Measuring energy consumption allows you to track costs and identify areas for potential savings.
- **Environmental Impact:** Tracking energy use helps reduce your carbon footprint by identifying opportunities to reduce energy consumption and consequently, greenhouse gas emissions.

GIVEN DATASET:

S.NO	DATE TIME	PJM LOAD MW
1	31-12-1998 01:00	29309
2	31-12-1998 02:00	29309
3	31-12-1998 03:00	29309
4	31-12-1998 04:00	29309
5	31-12-1998 05:00	29309
----	----	----
32892	01-01-2001 19:00	35497
32893	02-01-2001 19:00	35209
32894	03-01-2001 19:00	34791
32895	04-01-2001 19:00	33699
32896	05-01-2001 19:00	31809
32897	06-01-2001 19:00	29506

NECESSARY STEP TO FOLLOW:

1.Import Libraries: Start by importing the necessary libraries.

PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn model_selection import train_test_split
from sklearn preprocessing import StandardScaler
```

LOADING AND PRE-PROCESSING DATASET:

Loading and preprocessing a dataset for measuring energy consumption typically involves several steps. Here's a general guide to help you get started:

COLLECTING THE DATASET:

Identify reliable sources for energy consumption data. This could be utility companies, government databases, or research institutions.

LOADING THE DATASET:

Use Python libraries like Pandas to load the dataset into a DataFrame.

```
import pandas as pd
data = pd.read_csv('PJM_LOAD_HOURLY.csv')
pd.read()
```

EXPLORATARY DATA ANALYSIS(EDA):

Perform EDA to understand your data better. This includes checking for missing values,exploring the datas statistics,and visualizing it to identify patterns.

PROGRAM:

```
# Check for missing values
print(df.isnull().sum())
# Explore statistic
print(df.describe())
```

IMPORTANCE OF LOADING AND PREPROCESSING DATASET:

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are

often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

HOW TO OVERCOME THE CHALLENGES OF LOADING AND PREPROCESSING THE MEASURE ENERGY CONSUMPTION:

Loading and preprocessing energy consumption data can be challenging due to the volume, variety, and quality of the data. Here are some steps and strategies to overcome these challenges:

DATA COLLECTION AND SOURCES:

Identify reliable data sources: Ensure you have access to accurate and comprehensive energy consumption data from various sources like sensors, smart meters, utility companies, or government agencies.

Data formats: Understand the format of the data, whether it's structured (e.g., CSV, Excel) or unstructured (e.g., text logs), and be prepared to handle different formats.

DATA CLEANING:

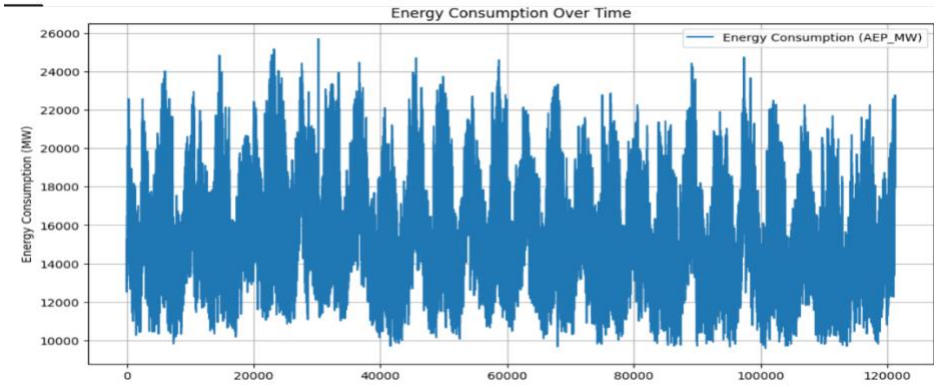
Handle missing data: Energy data can often have missing values. Decide on a strategy to deal with missing data, whether it's through imputation, interpolation, or removal.

Outlier detection: Identify and handle outliers in the data that might be caused by measurement errors or anomalies.

PROGRAM:

```
print(BLUE + "\nDATA CLEANING" + RESET)
# --- Check for missing values
missing_values = df.isnull().sum()
print(GREEN + "Missing Values : " + RESET)
```

```
print(missing_values)
# --- Handle missing values
df.dropna(inplace=True)
# --- Check for duplicate values
duplicate_values = df.duplicated().sum()
print(GREEN + "Duplicate Values : " + RESET)
print(duplicate_values)
# --- Drop duplicate values
df.drop_duplicates(inplace=True)
```

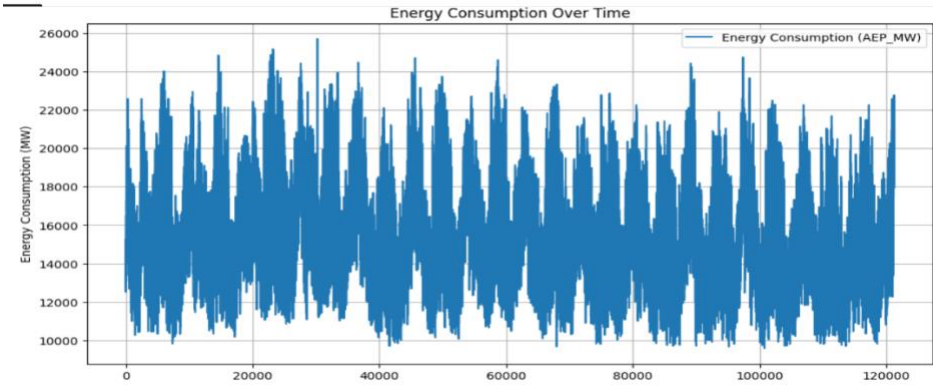


DATA ANALYSIS:

Calculate the cost of energy consumption by incorporating tariff rates, taxes, and any other associated costs. This helps in budgeting and cost optimization.

PROGRAM:

```
print(BLUE + "\nDATA ANALYSIS" + RESET)
# --- Summary Statistics
summary_stats = df.describe()
print(GREEN + "Summary Statistics : " + RESET)
print(summary_stats)
```



DATA PREPROSESSING:

First we import some basic python libraries like pandas numpy and matplotlib in our project and then we initialized our dataset in our project file by using the read csv command in the pandas as our project is in the csv file.

And then we started our processing of data in the project by dropping nan values and unnecessary columns in the data for the prediction . Thus this is the preprocessing that has been done in our project. let me attach the snipset of the data processing using pandas.

PROGRAM:

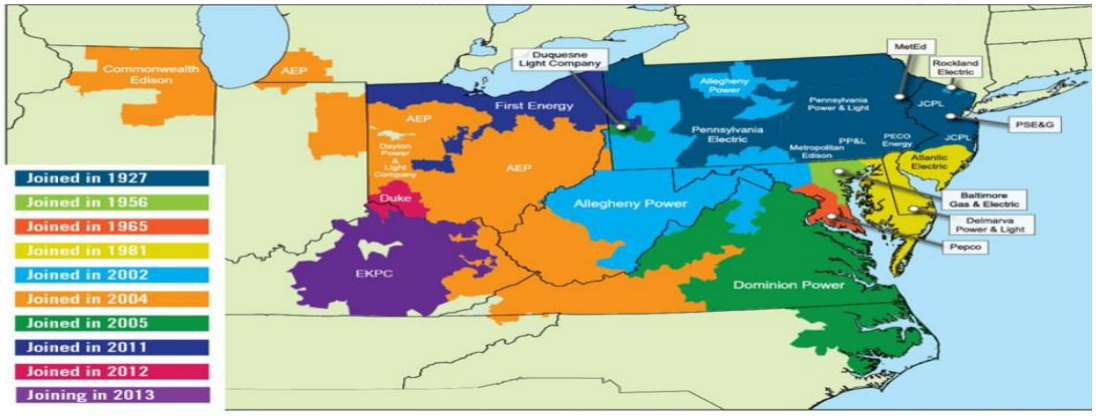
```
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
plt.style.use('ggplot') # Make it pretty
```

```
# Data is saved in parquet format so schema is preserved.
df = pd.read_parquet('../input/est_hourly.paruqet')
```

Data index is the date/hour, columns are for different regions within PJM.

Regions joined at different times, so not all have data for all dates. Regions also split (PJM_Load split to East and West)

```
#Show PJM Regions
from IPython.display import Image
Image(url= "http://slideplayer.com/4238181/14/images/4/PJM+Evolution.jpg")
```




```
df.head()
```

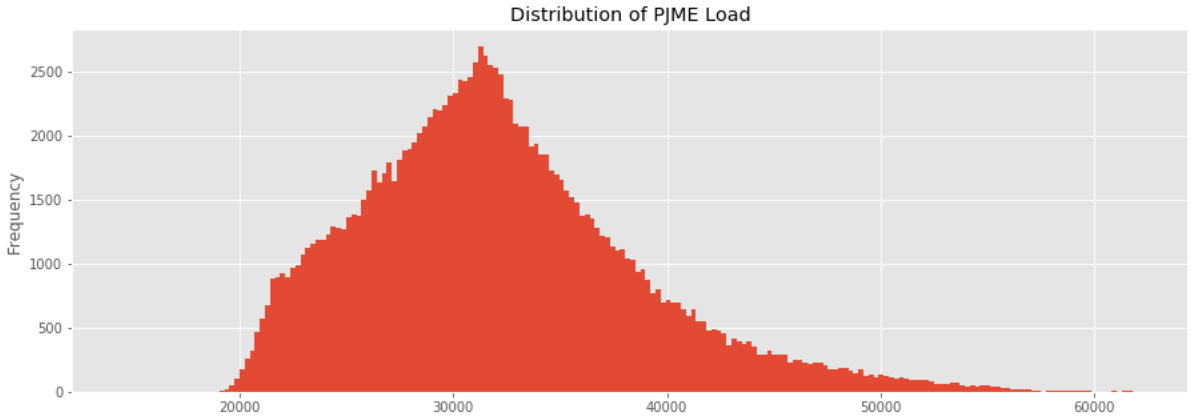
OUTPUT:

	AEP	COMED	DAYTON	DEOK	DOM	DUQ	EKPC	FE	NI	PJME	PJMW	PJM_Load
Datetime												
1998-12-31 01:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	29309.0
1998-12-31 02:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	28236.0
1998-12-31 03:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	27692.0
1998-12-31 04:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	27596.0
1998-12-31 05:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	27888.0

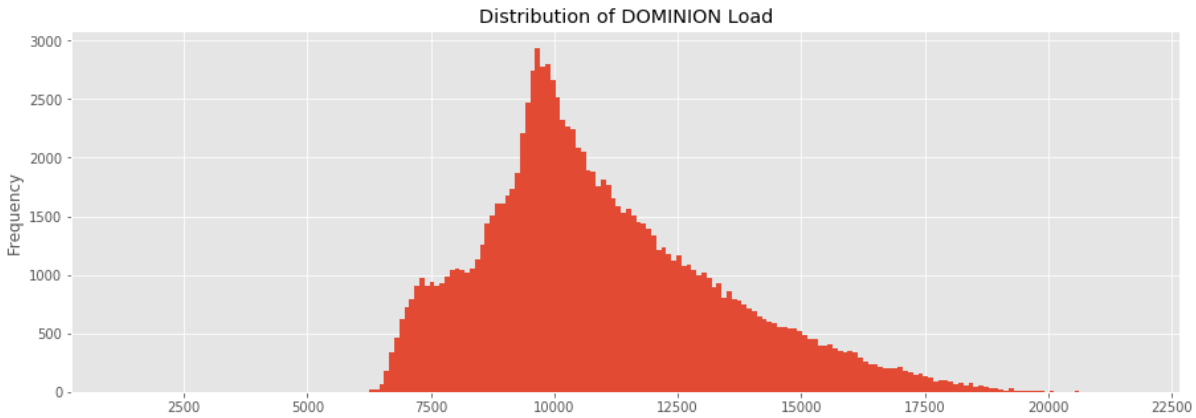
```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
AEP	121273.0	15499.513717	2591.399065	9581.0	13630.0	15310.0	17200.00	25695.0
COMED	66497.0	11420.152112	2304.139517	7237.0	9780.0	11152.0	12510.00	23753.0
DAYTON	121275.0	2037.851140	393.403153	982.0	1749.0	2009.0	2279.00	3746.0
DEOK	57739.0	3105.096486	599.859026	907.0	2687.0	3013.0	3449.00	5445.0
DOM	116189.0	10949.203625	2413.946569	1253.0	9322.0	10501.0	12378.00	21651.0
DUQ	119068.0	1658.820296	301.740640	1014.0	1444.0	1630.0	1819.00	3054.0
EKPC	45334.0	1464.218423	378.868404	514.0	1185.0	1386.0	1699.00	3490.0
FE	62874.0	7792.159064	1331.268006	0.0	6807.0	7700.0	8556.00	14032.0
NI	58450.0	11701.682943	2371.498701	7003.0	9954.0	11521.0	12896.75	23631.0
PJME	145366.0	32080.222831	6464.012166	14544.0	27573.0	31421.0	35650.00	62009.0
PJMW	143206.0	5602.375089	979.142872	487.0	4907.0	5530.0	6252.00	9594.0
PJM_Load	32896.0	29766.427408	5849.769954	17461.0	25473.0	29655.0	33073.25	54030.0

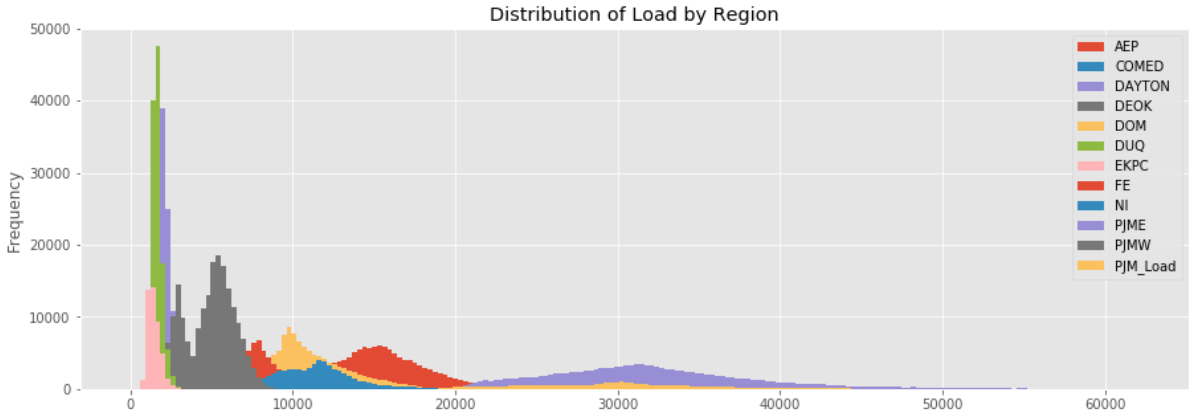
```
_ = df['PJME'].plot.hist(figsize=(15, 5), bins=200, title='Distribution of PJME Load')
```




```
_ = df['DOM'].plot.hist(figsize=(15, 5), bins=200, title='Distribution of DOMINION Load')
```

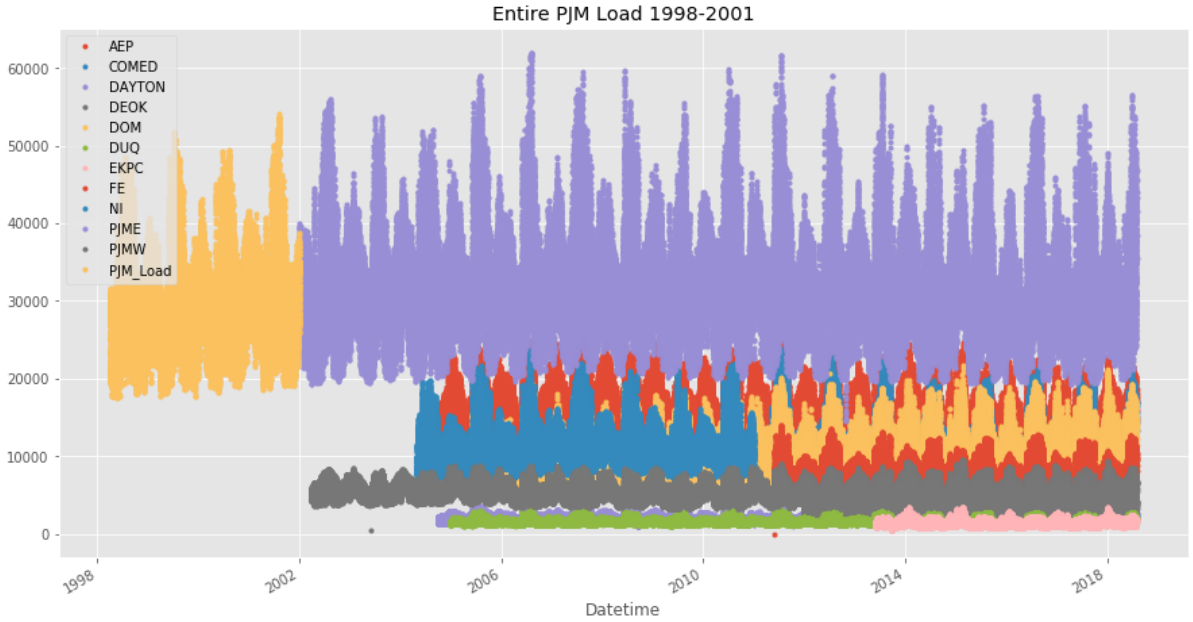


```
_ = df.plot.hist(figsize=(15, 5), bins=200, title='Distribution of Load by
Region')
```



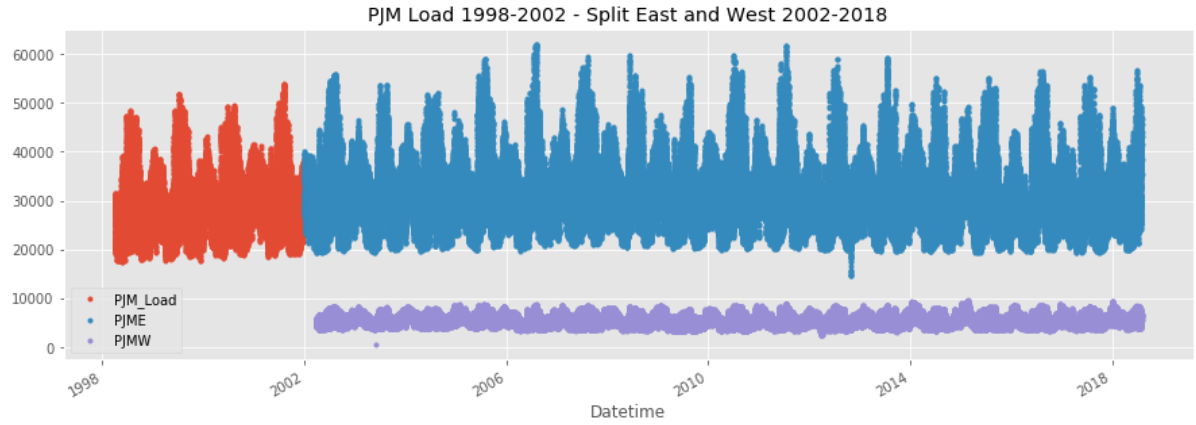
PLOT TIME SERIES:

```
plot = df.plot(style='.', figsize=(15, 8), title='Entire PJM Load 1998-2001
')
```



PLOTTING REGIONS:

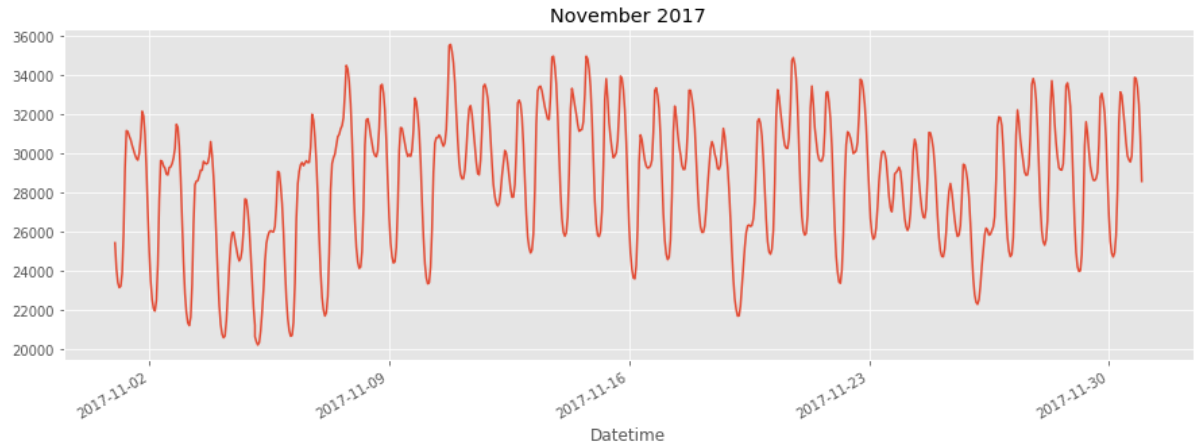
```
_ = df[['PJM_Load','PJME','PJMW']] \
    .plot(style='.', figsize=(15, 5), title='PJM Load 1998-2002 - Split Eas
t and West 2002-2018')
```



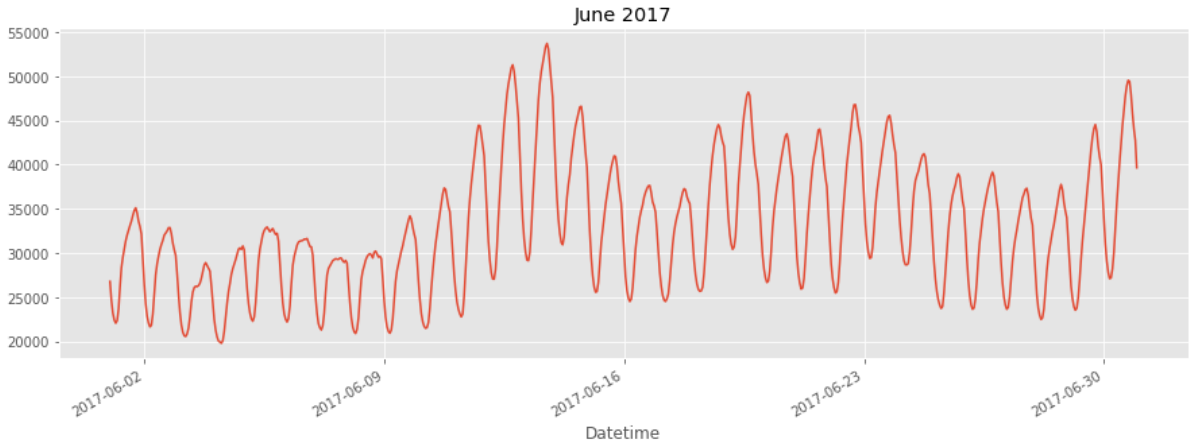
SUMMER DEMAND VS WINTER DEMAND:

Note the dips mid-day in the winter months. Conversely in summer months the daily load is more bell shaped. This is due to high mid-day energy consumtion by air conditioning. In winter months people tend to use less energy mid-day.

```
_= df['PJME'].loc[(df['PJME'].index >= '2017-11-01') &
                  (df['PJME'].index < '2017-12-01')] \
    .plot(figsize=(15, 5), title = 'November 2017')
```




```
_ = df['PJME'].loc[(df['PJME'].index >= '2017-06-01') &
                  (df['PJME'].index < '2017-07-01')] \
    .plot(figsize=(15, 5), title = 'June 2017')
```

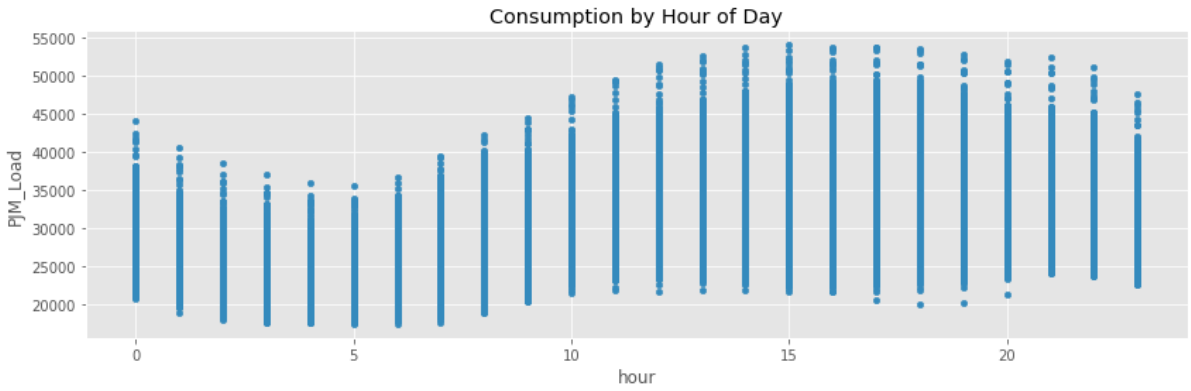


CREATE TIME SERIES FEATURE:

```
df['dow'] = df.index.dayofweek
df['doy'] = df.index.dayofyear
df['year'] = df.index.year
df['month'] = df.index.month
df['quarter'] = df.index.quarter
df['hour'] = df.index.hour
df['weekday'] = df.index.weekday_name
df['woy'] = df.index.weekofyear
df['dom'] = df.index.day # Day of Month
df['date'] = df.index.date
```

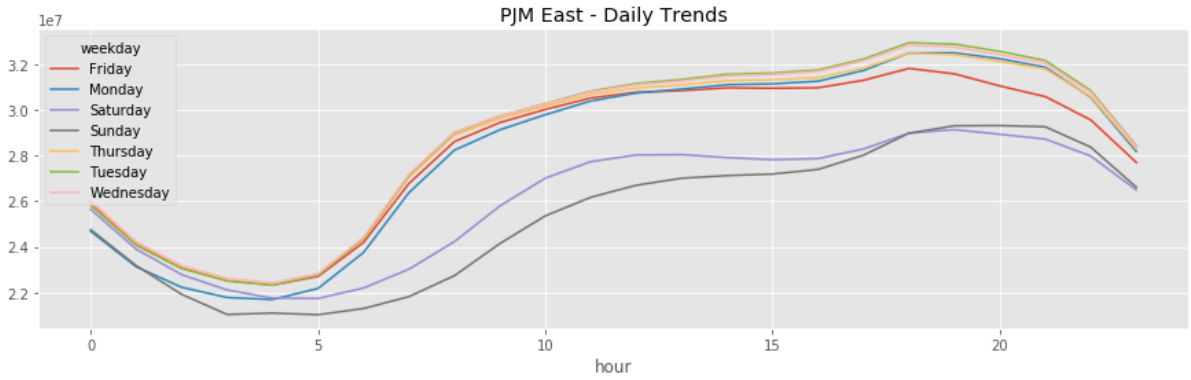
linkcode

```
_ = df[['PJM_Load','hour']].plot(x='hour',
                                 y='PJM_Load',
                                 kind='scatter',
                                 figsize=(14,4),
                                 title='Consumption by Hour of Day')
```



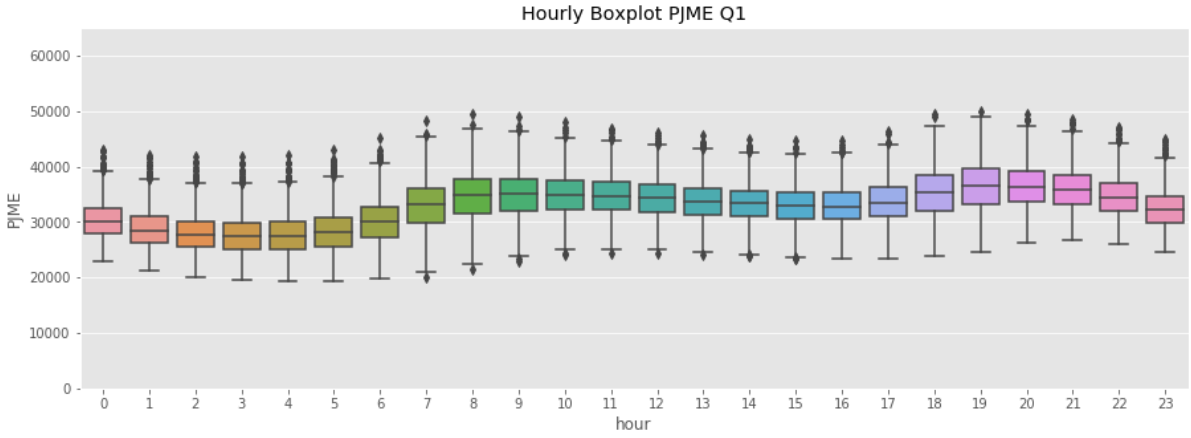

```
_ = df.pivot_table(index=df['hour'],
                   columns='weekday',
                   values='PJME',
```

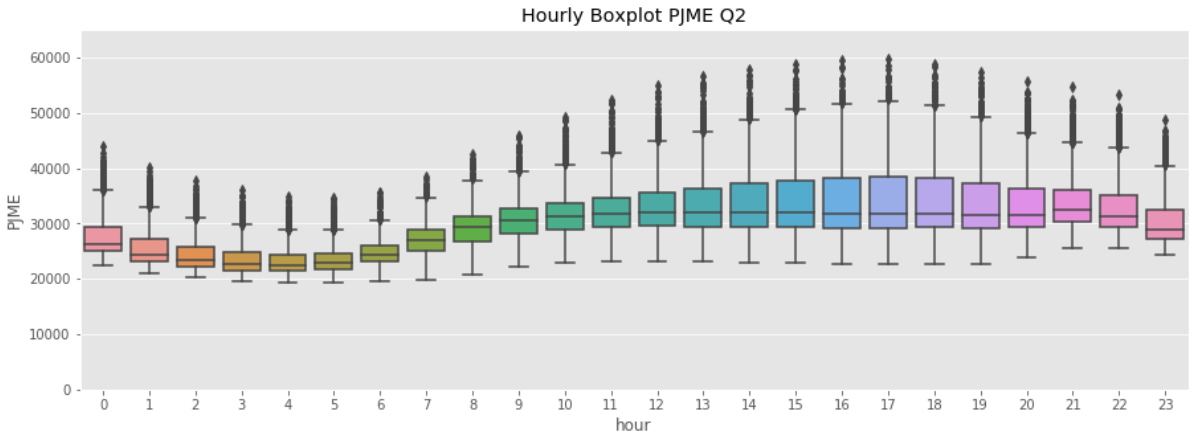
```
aggfunc='sum').plot(figsize=(15,4),
title='PJM East - Daily Trends')
```



TREANDS CHANGE DEPENDING ON TIME OF YEAR

```
 ax = plt.subplots(figsize=(15,5))
sns.boxplot(df.loc[df['quarter']==1].hour, df.loc[df['quarter']==1].PJME)
ax.set_title('Hourly Boxplot PJME Q1')
ax.set_ylim(0,65000)
fig, ax = plt.subplots(figsize=(15,5))
sns.boxplot(df.loc[df['quarter']==2].hour, df.loc[df['quarter']==2].PJME)
ax.set_title('Hourly Boxplot PJME Q2')
ax.set_ylim(0,65000)
fig, ax = plt.subplots(figsize=(15,5))
sns.boxplot(df.loc[df['quarter']==3].hour, df.loc[df['quarter']==3].PJME)
ax.set_title('Hourly Boxplot PJME Q3')
ax.set_ylim(0,65000)
fig, ax = plt.subplots(figsize=(15,5))
sns.boxplot(df.loc[df['quarter']==4].hour, df.loc[df['quarter']==4].PJME)
ax.set_title('Hourly Boxplot PJME Q4')
_ = ax.set_ylim(0,65000)
```





CONCLUSION:

In the quest to measure energy consumption model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.

Understanding the data structure any potential issues through exploratory data analysis is essential for informed decision making.

Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.

With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training the measure energy consumption.