In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.offline as pyo
import plotly.graph_objs as go
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Load all 5 datasets
city1 = pd.read_csv(r"C:\Users\payal\Downloads\Beijing.csv")
city2 = pd.read_csv(r"C:\Users\payal\Downloads\Chengdu.csv")
city3 = pd.read_csv(r"C:\Users\payal\Downloads\Guangzhou.csv")
city4 = pd.read_csv(r"C:\Users\payal\Downloads\Shanghai.csv")
city5 = pd.read_csv(r"C:\Users\payal\Downloads\Shenyang.csv")

# Add 'city' column to each of the datasets
city1["city"] = "Beijing"
city2["city"] = "Chengdu"
city3["city"] = "Guangzhou"
city4["city"] = "Shanghai"
city5["city"] = "Shenyang"
```

Que 1) Perform data cleaning, impute missing values, do feature engineering for “Season” feature map it with seasons.

TASK 1: Merge all 5 datasets

In [3]:
```
# Merge all 5 datasets
chinese_cities = pd.concat([city1, city2, city3, city4, city5], ignore_index=True)
chinese_cities.sample(5)
```

Out[3]:

	year	month	day	hour	season	PM	DEWP	HUMI	PRES	TEMP	Iws	precipitation	I
150574	2015	3	7	22	1.0	66.0	14.2	90.00	1009.8	15.9	1.00	0.0	
204733	2015	5	12	13	1.0	26.0	8.0	31.97	1010.0	26.0	10.00	0.0	
178683	2012	5	22	3	1.0	47.0	15.0	82.65	1013.9	18.0	0.00	0.0	
168223	2011	3	13	7	1.0	NaN	8.0	100.00	1019.0	8.0	56.00	0.0	
12253	2011	5	26	13	1.0	NaN	9.0	30.00	1010.0	28.0	59.45	0.0	

In [4]:
```
# Add 'date' column to the merged dataset
chinese_cities["date"] = pd.to_datetime(chinese_cities[["month", "day", "year"]])
```

In [5]:
```
chinese_cities.sample(3)
```

Out[5]:

	year	month	day	hour	season	PM	DEWP	HUMI	PRES	TEMP	Iws	precipitation	Ip
174018	2011	11	9	18	3.0	NaN	10.0	72.04	1019.0	15.0	419.0	0.0	
116268	2011	4	8	12	1.0	NaN	17.9	57.00	1009.7	27.1	2.3	0.0	
200939	2014	12	5	11	4.0	33.0	-16.0	17.54	1029.0	7.0	213.0	0.0	

In [6]:
```
# Check for datatypes & shape of the dataset
chinese_cities.info()
chinese_cities.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262920 entries, 0 to 262919
Data columns (total 15 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   year           262920 non-null  int64
 1   month          262920 non-null  int64
 2   day            262920 non-null  int64
 3   hour           262920 non-null  int64
 4   season         262919 non-null  float64
 5   PM             167358 non-null  float64
 6   DEWP           261680 non-null  float64
 7   HUMI           261352 non-null  float64
 8   PRES           261339 non-null  float64
 9   TEMP           261682 non-null  float64
 10  Iws            261677 non-null  float64
 11  precipitation  242708 non-null  float64
 12  Iprec          242708 non-null  float64
 13  city           262920 non-null  object
 14  date           262920 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(9), int64(4), object(1)
memory usage: 30.1+ MB
```

Out[6]:
```
(262920, 15)
```

TASK 2: Perform Data Cleaning

In [7]:
```
# Check for missing values
chinese_cities.isnull().sum()
```

Out[7]:
```
year                0
month               0
day                 0
hour                0
season              1
PM              95562
DEWP             1240
HUMI             1568
PRES             1581
TEMP             1238
Iws              1243
precipitation   20212
Iprec           20212
city                0
date                0
dtype: int64
```

TASK 3: Impute missing values

In [8]:
```
# Copy the dataframe to a new dataframe
imputed_df = chinese_cities.copy()

# Filling missing values with mean, median, mode, ffill, bfill, or interpolate method
imputed_df = imputed_df.interpolate(method="bfill")
imputed_df.head()
```

Out[8]:

	year	month	day	hour	season	PM	DEWP	HUMI	PRES	TEMP	Iws	precipitation	Iprec
0	2010	1	1	0	4.0	129.0	-21.0	43.0	1021.0	-11.0	1.79	0.0	0.0
1	2010	1	1	1	4.0	129.0	-21.0	47.0	1020.0	-12.0	4.92	0.0	0.0
2	2010	1	1	2	4.0	129.0	-21.0	43.0	1019.0	-11.0	6.71	0.0	0.0
3	2010	1	1	3	4.0	129.0	-21.0	55.0	1019.0	-14.0	9.84	0.0	0.0
4	2010	1	1	4	4.0	129.0	-20.0	51.0	1018.0	-12.0	12.97	0.0	0.0

In [9]:
```
# Check again for missing values after imputation
imputed_df.isnull().sum()
```

Out[9]:
```
year             0
month            0
day              0
hour             0
season           0
PM               0
DEWP             0
HUMI             0
PRES             0
TEMP             0
Iws              0
precipitation    0
Iprec            0
city             0
date             0
dtype: int64
```

In [10]:
```
# Check for duplicate records
duplicate = imputed_df[imputed_df.duplicated()]
print("Duplicate Rows :")
duplicate
```

```
Duplicate Rows :
```

Out[10]:

year	month	day	hour	season	PM	DEWP	HUMI	PRES	TEMP	Iws	precipitation	Iprec	city

TASK 4: Do feature engineering for "Season" and feature map it with seasons

In [11]:
```
# Get the unique values in column "season"
unique_values = imputed_df["season"].unique()

# Print the unique values
print("Unique Seasons")
print(unique_values)
```

```
Unique Seasons
[4. 1. 2. 3.]
```

In [12]:
```
# Map the values of the "season" feature to season names
seasons = {1: "Spring", 2: "Summer", 3: "Fall", 4: "Winter"}
imputed_df["season"] = imputed_df["season"].map(seasons)
```

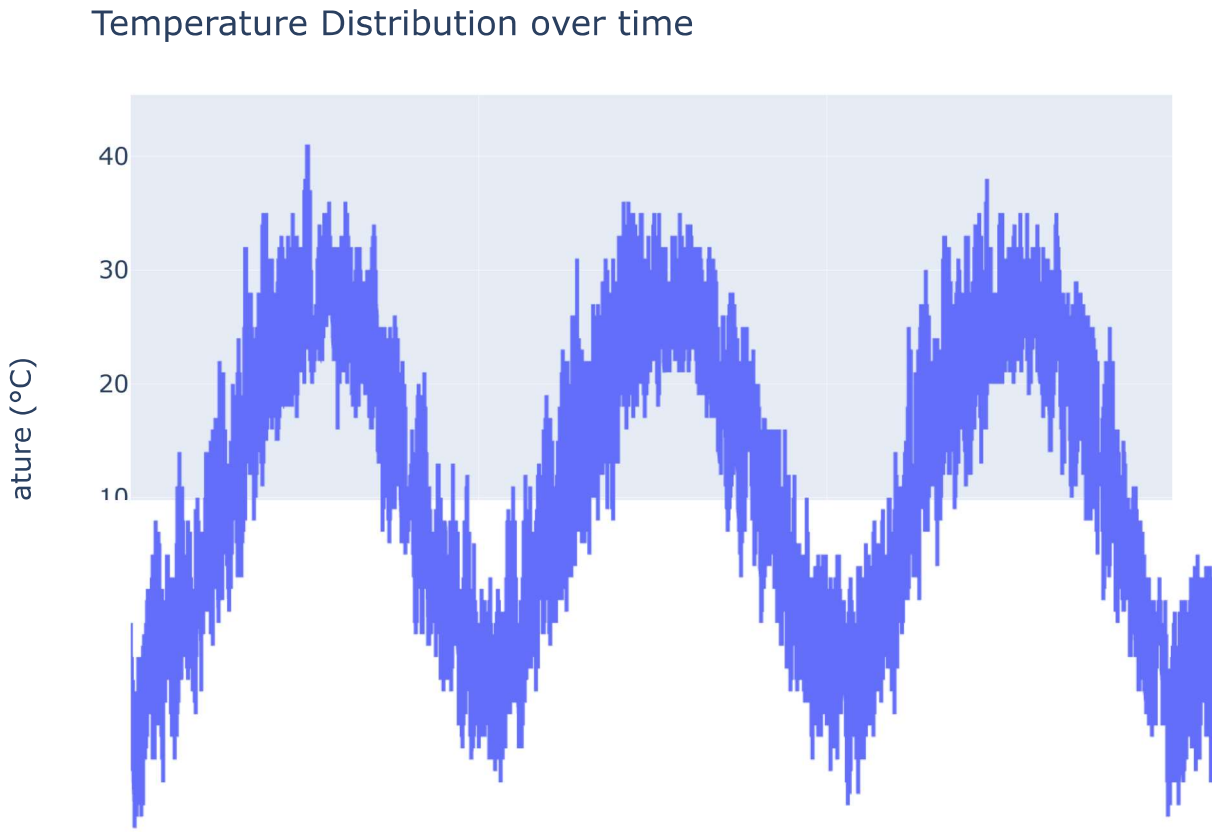
In [13]:
```
imputed_df.sample(5)
```

Out[13]:

	year	month	day	hour	season	PM	DEWP	HUMI	PRES	TEMP	Iws	precipitation
2320	2010	4	7	16	Spring	90.0	-7.0	15.00	1015.0	20.0	42.91	0.0
94818	2014	10	26	18	Fall	174.0	16.0	73.14	1018.0	21.0	3.00	0.0
111057	2010	9	3	9	Fall	48.0	24.9	100.00	998.3	24.9	11.50	5.0
97948	2015	3	6	4	Spring	88.0	5.0	87.09	1021.0	7.0	0.00	0.0
194535	2014	3	13	15	Spring	39.0	-7.0	27.58	1022.0	11.0	109.00	0.0

Que 2) Generate a line chart showing the temperature (y-axis) and dates (x-axis) for one of the five cities. Is there a noticeable seasonal pattern?

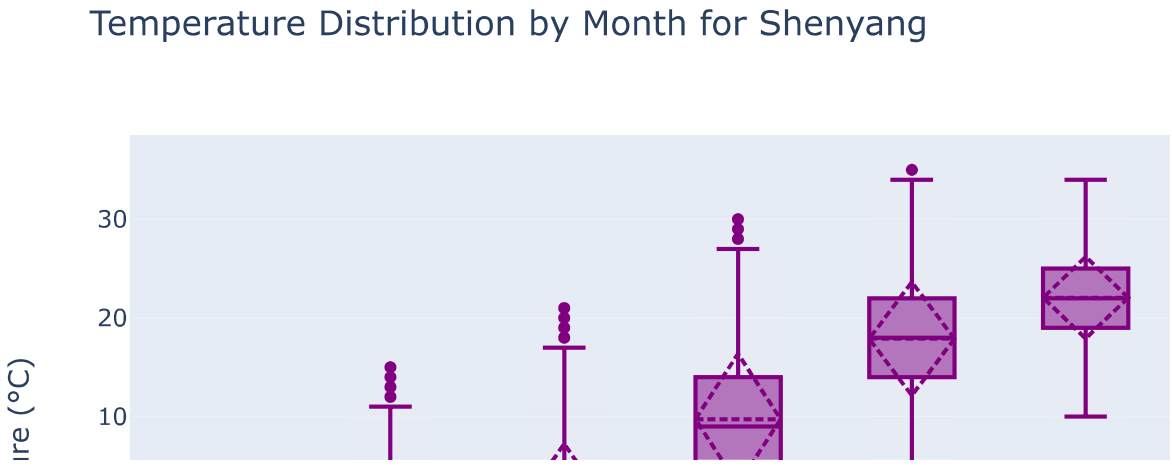
In [14]:
```
# Filter the DataFrame to include only records with the name 'Beijing'
filtered_df = imputed_df[imputed_df["city"] == "Beijing"]

# Plot the temperature values
fig = px.line(filtered_df, x="date", y="TEMP")

# Display the plot
fig.update_xaxes(title_text="Date")
fig.update_yaxes(title_text="Temperature (°C)")
fig.update_layout(title_text="Temperature Distribution over time")
fig.show()
```



Insights: The above line plot shows the Distribution of Temperature(°C) over the period of years for the city 'Beijing'. We can see that the temperature over the years is same with roughly min-temp = -19°C and max-temp = 42°C.

Que 3) Create a boxplot showing the temperature values aggregated by month for one of the five cities. (Use plotly)

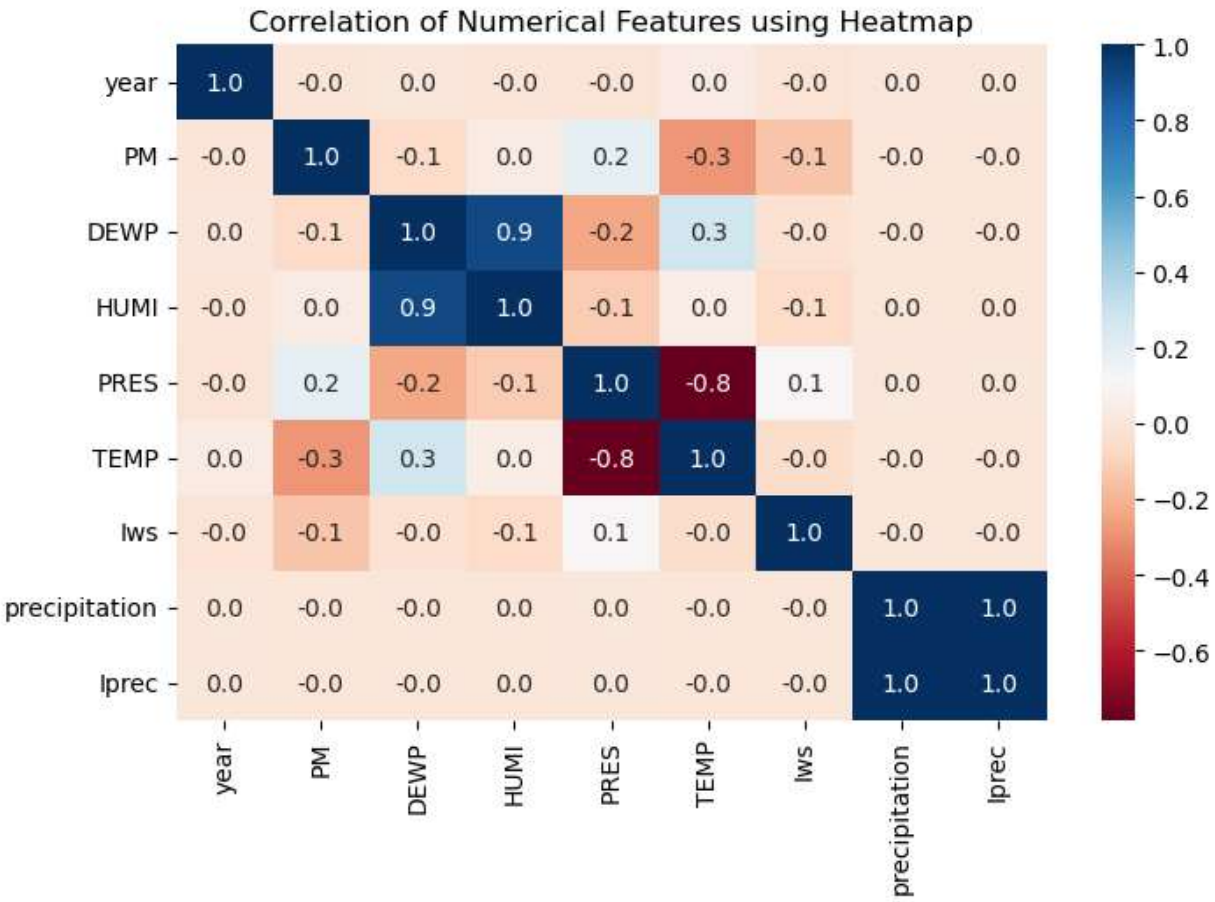
In [15]:
```
# Filter the DataFrame to include only records with the name 'Shenyang'
filtered_df = imputed_df[imputed_df["city"] == "Shenyang"]

px.box(filtered_df, x="month", y="TEMP", points="all")

# Adding standard deviation and mean
fig = go.Figure()
fig.add_trace(go.Box(x=filtered_df.month, y=filtered_df.TEMP, marker_color="purple", b

# Add labels and title
fig.update_xaxes(title_text="Month")
fig.update_yaxes(title_text="Temperature (°C)")
fig.update_layout(title_text="Temperature Distribution by Month for Shenyang")
```



Insights: The box plot shows the Temperature Distribution by months for the city 'Shenyang'. We can see that there is a variation of temperatures for each month. The temperatures in Shenyang are high for the months July and August & low for the months December and January.

Que 4) Create a heatmap to generate correlation between numeric features.

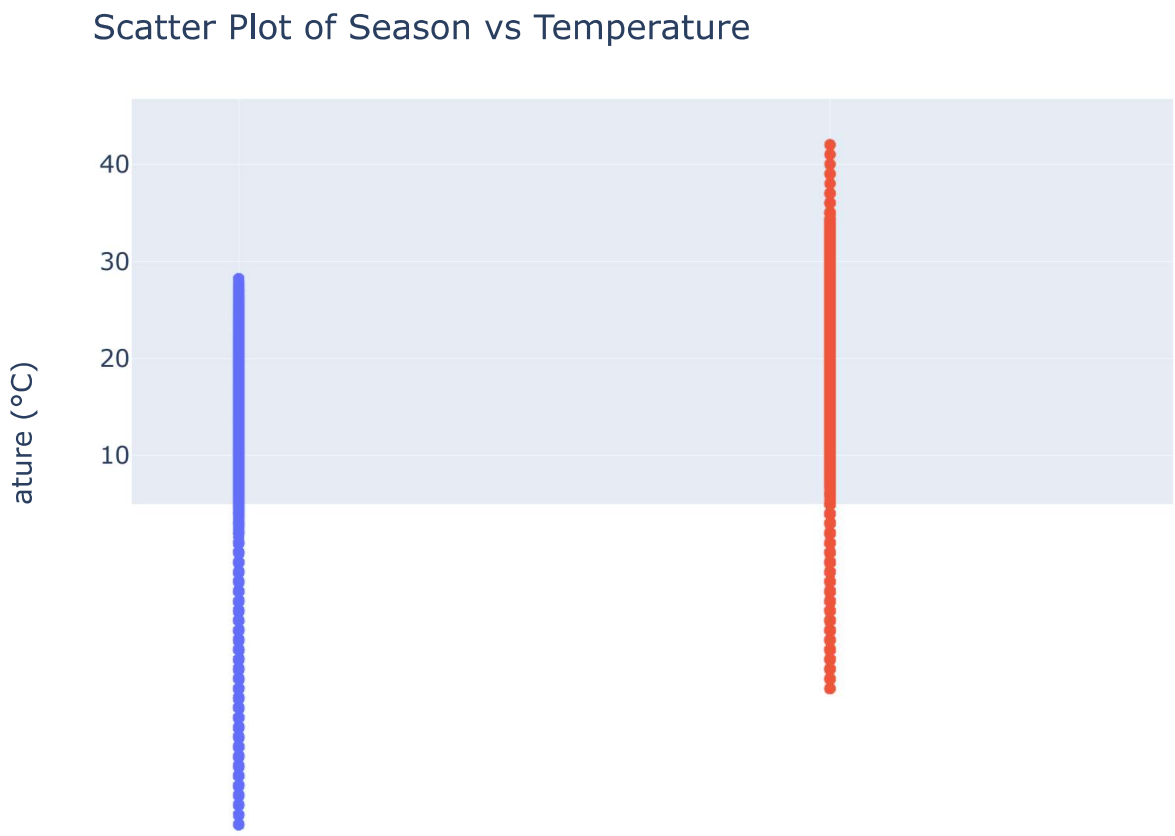
In [16]:
```
# Select numerical features from "chinese_cities" dataset
num_features = ["year", "PM", "DEWP", "HUMI", "PRES", "TEMP", "Iws", "precipitation",

# Create correlation matrix of numerical features
corr_matrix = imputed_df[num_features].corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(8, 5))
sns.heatmap(corr_matrix, annot=True, cmap="RdBu", fmt=".1f")
plt.title("Correlation of Numerical Features using Heatmap")
plt.show()
```



Insights: The heatmap shows the correlation between pairs of numerical features, with positive correlations shown in blue and negative correlations shown in red. The color scale ranges from -1 to 1, with -1 indicating a perfectly negative correlation, 0 indicates no correlation, and 1 indicates a perfectly positive correlation. We can see that there is a strong positive correlation between "Humidity" and "Dew Point". Also, there is a strong positive correlation between "Precipitation" and "Iprec". There is a strong negative correlation between "Temperature" and "Pressure".

Que 5) Create a scatter plot using two features of your choice. Choose a pair of features that you believe have some correlation between them. Based on your visualization, do they seem to be correlated? (Use plotly)

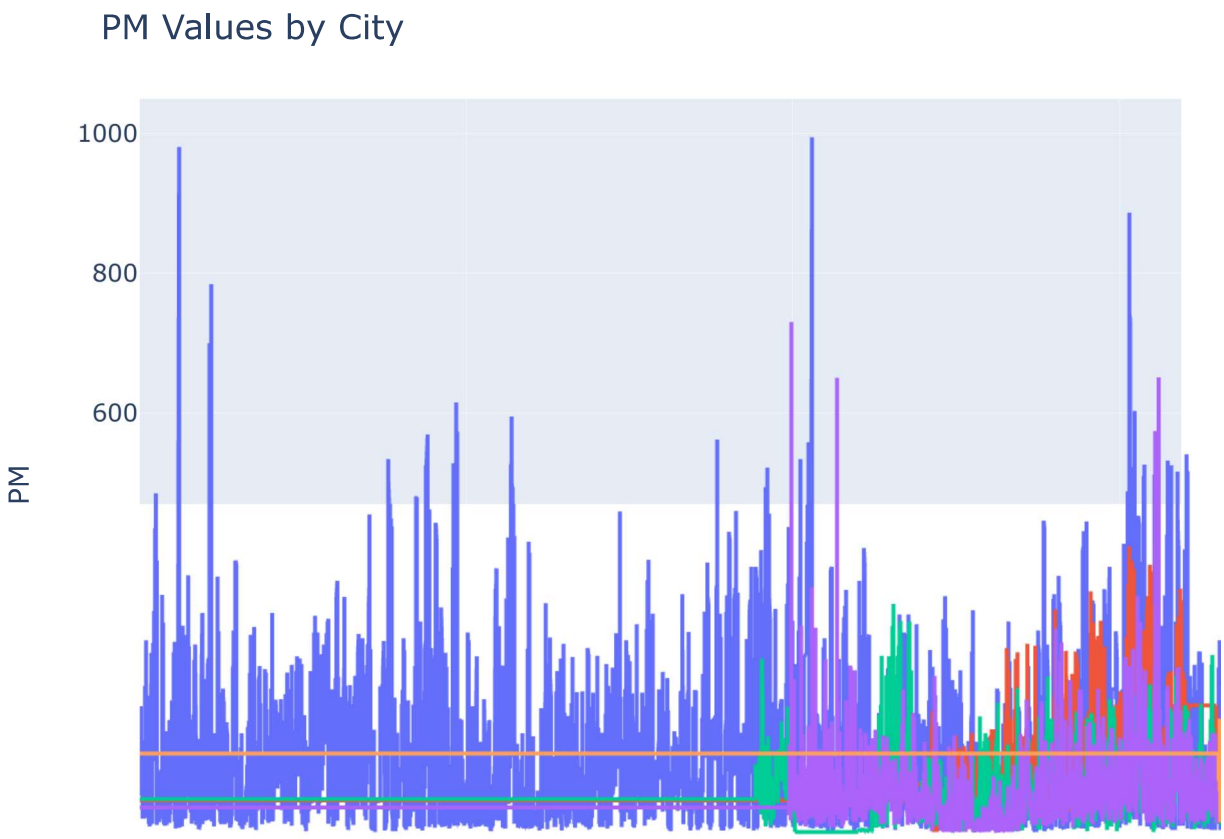
In [17]:
```
# Create a scatter plot for Season vs. Temperature
fig = px.scatter(imputed_df, x="season", y="TEMP", color="season")

# Display the plot
fig.update_xaxes(title_text="Season")
fig.update_yaxes(title_text="Temperature (°C)")
fig.update_layout(title_text="Scatter Plot of Season vs Temperature")
fig.show()
```



Insights: The scatter plot of Temperature(y-axis) vs Season(x-axis) shows that the temperatures are highest in the summer, and lowest in the winter. This is as expected in general. So we can see that there is a strong correlation between the two features "Temperature" and "Seasons".

Que 6) Create a single plot that illustrates the value of the PM column over time for each of the four cities. Color and label each city differently so that they can be distinguished easily.

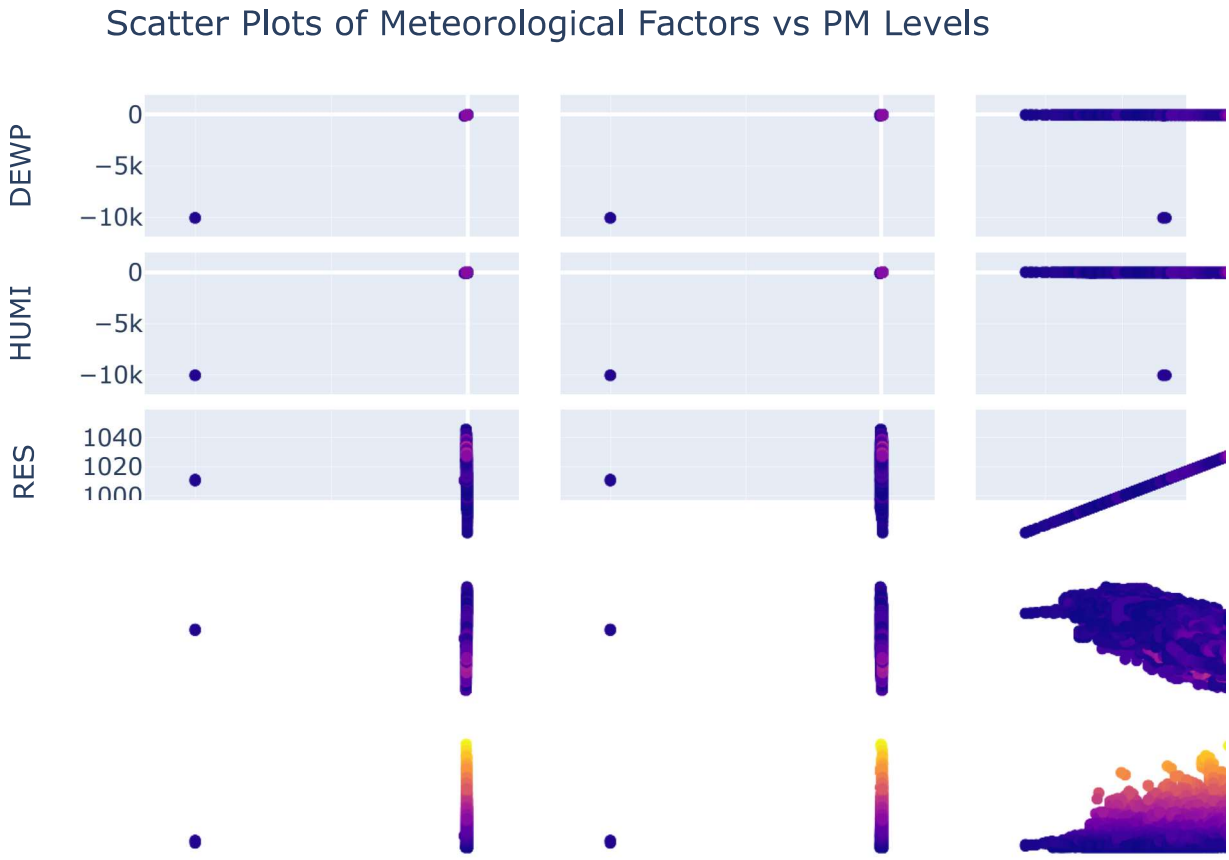
In [18]:
```
# Create a line chart for the PM column of each city
fig = px.line(imputed_df, x="date", y="PM", color="city")

# Display the plot
fig.update_xaxes(title_text="Date")
fig.update_yaxes(title_text="PM")
fig.update_layout(title_text="PM Values by City")
fig.show()
```



Insights: The line chart shows the distribution of PM values over time for each of the five cities. The date(x-axis) and PM(y-axis) is denoting that there is a variation of PM levels over a period for each of the cities. Some cities have consistently higher or lower PM levels than others.

Que 7) How do meteorological factors (DEWP, HUMI, PRES, TEMP) correlate with PM levels? Create scatter plots to explore relationships. (Use plotly)

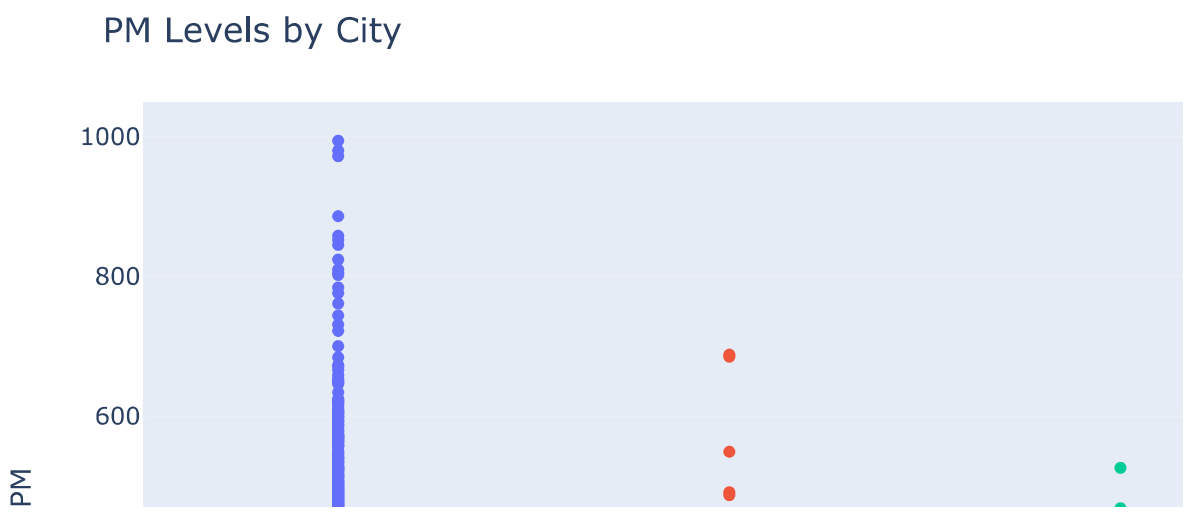
In [19]:
```
#Reference- https://plotly.com/python/splom/
# Create scatter plots for each pair of the features to check the correlation
fig = px.scatter_matrix(imputed_df, dimensions=["DEWP", "HUMI", "PRES", "TEMP", "PM"],

# Display the plot
fig.update_layout(title_text="Scatter Plots of Meteorological Factors vs PM Levels")
fig.show()
```



Insights: The scatter plot shows the correlation between the meteorological factors((DEWP, HUMI, PRES, TEMP) with the PM levels. The scatter plot matrix shows the pairwise relationship between the features. We can notice that - 1)DEWP vs PM have strong positive correlation. 2)HUMI vs PM have strong positive correlation. 3)PRES vs PM have no correlation. 4)TEMP vs PM have no correlation.

Que 8) How do PM levels compare between the five cities? Create bar charts or boxplots for a city-to-city comparison. (Use plotly)

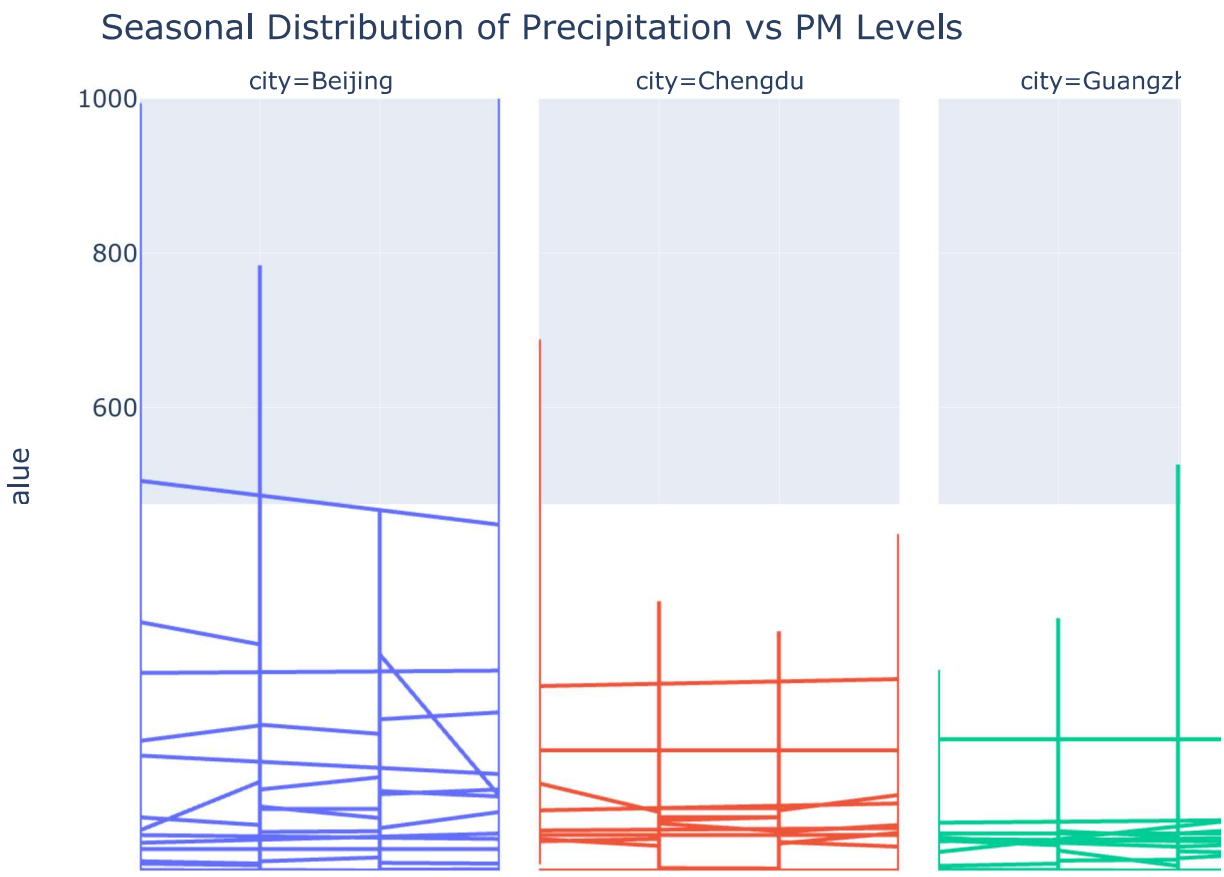
In [20]:
```
# Create box plots for each city
fig = px.box(imputed_df, x="city", y="PM", color="city")

# Display the plot
fig.update_xaxes(title_text="City")
fig.update_yaxes(title_text="PM")
fig.update_layout(title_text="PM Levels by City")
fig.show()
```



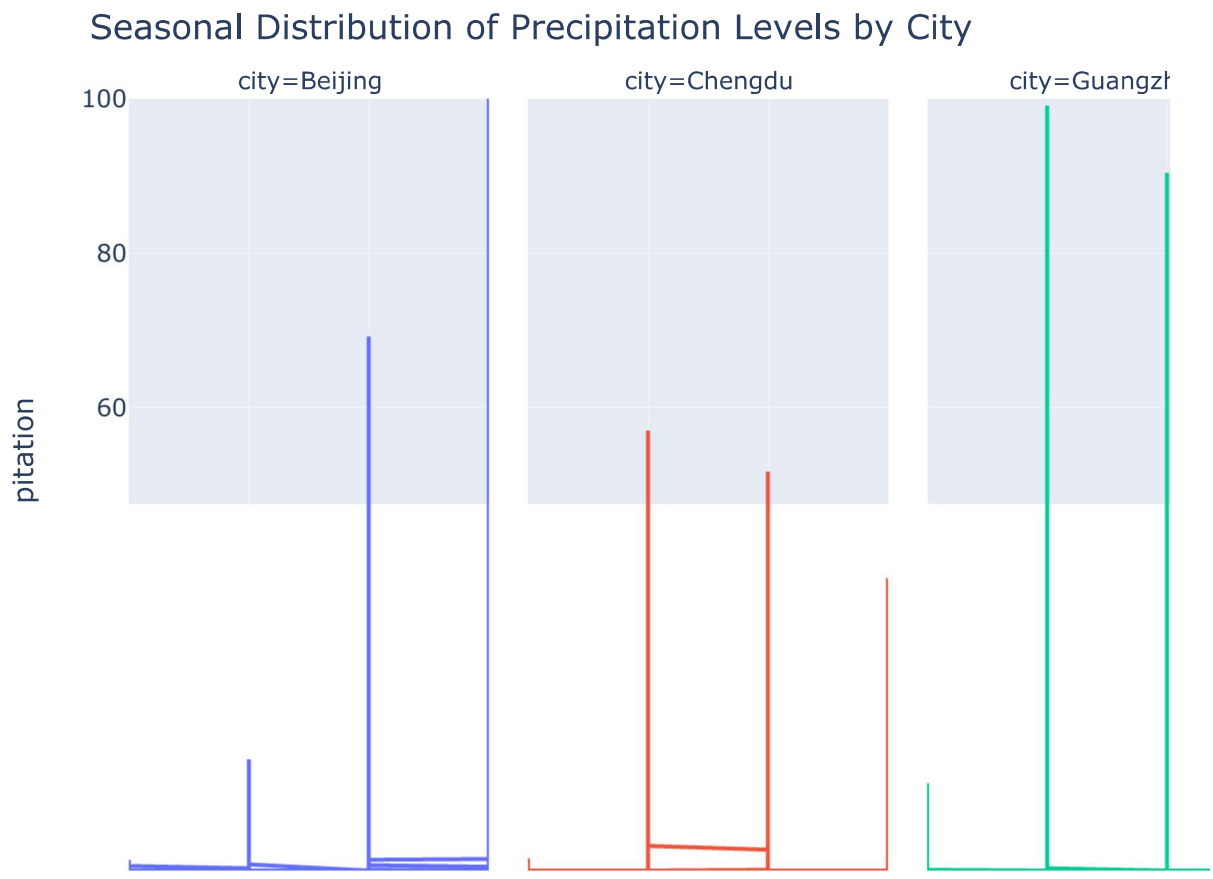
Insights: The box plot shows the comparison of PM levels by city. We can interpret that the PM levels are higher for cities Beijing and Shenyang compared to the other cities.

Que 9) Create a line plot to show the seasonal distribution of precipitation levels and examine how it relates to PM levels. (Use plotly)

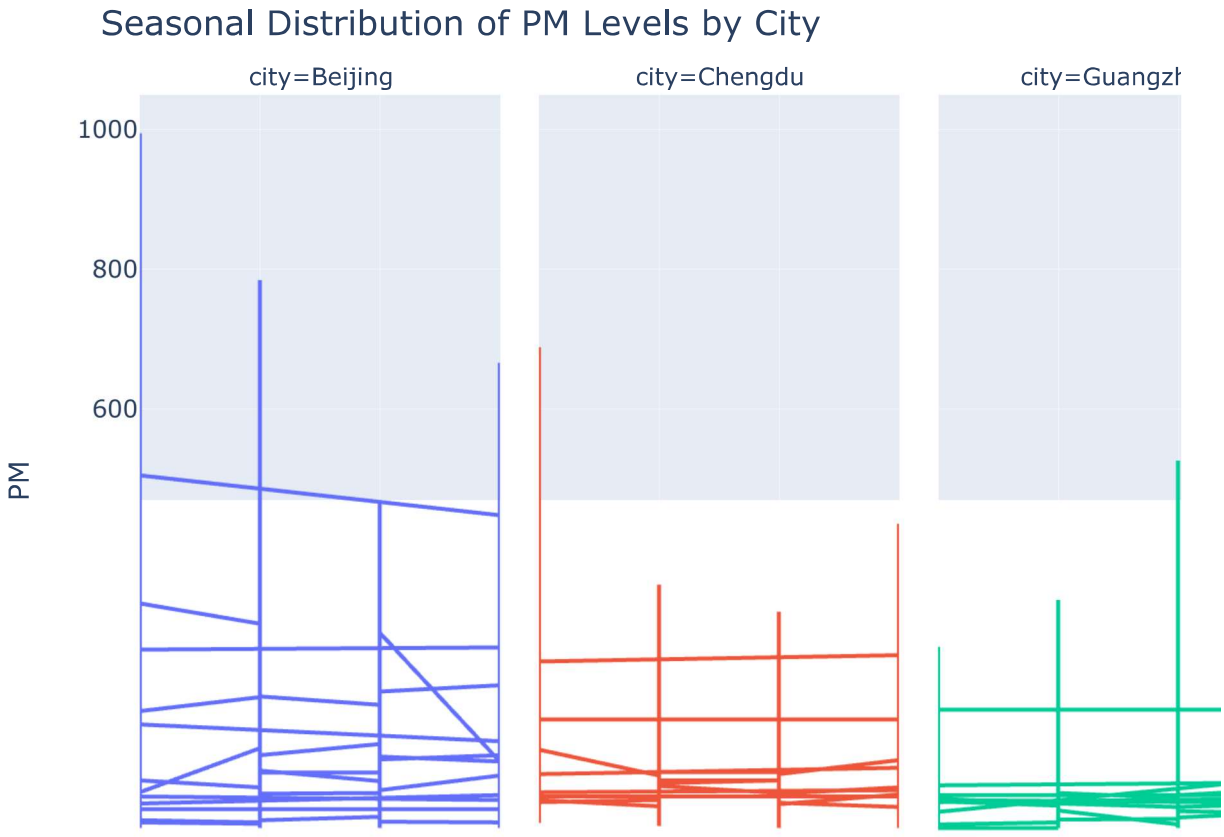
In [21]:
```
# Create a line plot for the seasonal distribution of precipitation and PM levels
fig = px.line(imputed_df, x="season", y=["precipitation", "PM"], color="city", facet_c

# Change the scale of the y-axis
fig.update_yaxes(range=[0, 1000])

# Display the plot
fig.update_xaxes(title_text="Season")
fig.update_layout(title_text="Seasonal Distribution of Precipitation vs PM Levels")
fig.show()
```



In [22]:
```
# Create a line plot of Season vs Precipitation Levels
fig = px.line(imputed_df, x="season", y="precipitation", color="city", facet_col="city

# Change the scale of the y-axis
fig.update_yaxes(range=[0, 100])

# Display the plot
fig.update_layout(title_text="Seasonal Distribution of Precipitation Levels by City")
fig.show()
```



In [23]:
```
# Create a line plot of Season vs PM Levels
fig = px.line(imputed_df, x="season", y="PM", color="city", facet_col="city")

# Display the plot
fig.update_layout(title_text="Seasonal Distribution of PM Levels by City")
fig.show()
```



Insights: The line plots shows the seasonal distribution of each city with respect to precipitation and PM levels. Each line represents different city, and the color indicates which city it belongs to. The season(x-axis) and Precipitation and PM(y-axis) shows that generally the precipitation is highest in summer and lowest in winter. On the contrary, PM levels are generally highest in winter and lowest in summer, having variations for some cities.

Que 10) Create a simple dashboard using pydash tutorial 1.

In [36]:
```
from dash import Dash, html, dcc, callback, Output, Input
import dash_core_components as dcc
import dash_html_components as html

app = Dash(__name__)

app.layout = html.Div([
    html.H1(children = "Dashboard of Precipitation over time by City", style = {"textA
    dcc.Dropdown(imputed_df.city.unique(), "Shanghai", id = "dropdown-selection"),
    dcc.Graph(id="graph-content")
])

@callback(
    Output("graph-content", "figure"),
```

```
    Input("dropdown-selection", "value")
)

def update_graph(value):
    dff = imputed_df[imputed_df["city"] == value]
    return px.line(dff, x="date", y="Iprec")


if __name__ == "__main__":
    app.run_server(debug=True, port=8051)
```

Loading...

Insights: This simple pydash dashboard shows the distribution of precipitation over a period for each of the 5 cities.The dashboard consists of a dropdown menu to select the city and a line plot showing the precipitation over time for the selected city. And the plot represents date(x-axis) and cumulative precipitation:Iprec(y-axis). There is a variation of precipitation over time for each cities.