In [13]:

```
#Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [3]:

```
#Importing the dataset
df=pd.read_excel("C:/Users/soumy/OneDrive/Documents/Lottery_Powerball_Winning_Numbers__B
df.head()
```

Out[3]:

	Draw Date	Winning Numbers	Multiplier
0	09/26/2020	11 21 27 36 62 24	3.0
1	09/30/2020	14 18 36 49 67 18	2.0
2	2020-03-10 00:00:00	18 31 36 43 47 20	2.0
3	2020-07-10 00:00:00	06 24 30 53 56 19	2.0
4	2020-10-10 00:00:00	05 18 23 40 50 18	3.0

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1429 entries, 0 to 1428
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Draw Date        1429 non-null   object 
 1   Winning Numbers  1429 non-null   object 
 2   Multiplier       1219 non-null   float64
dtypes: float64(1), object(2)
memory usage: 33.6+ KB
```

In [5]:

```
#Converting the datetime to date
df['Draw Date']=pd.to_datetime(df['Draw Date']).dt.date
df.head()
```

Out[5]:

	Draw Date	Winning Numbers	Multiplier
0	2020-09-26	11 21 27 36 62 24	3.0
1	2020-09-30	14 18 36 49 67 18	2.0
2	2020-03-10	18 31 36 43 47 20	2.0
3	2020-07-10	06 24 30 53 56 19	2.0
4	2020-10-10	05 18 23 40 50 18	3.0

In [6]:

```
df=df.sort_values("Draw Date")
df
```

Out[6]:

	Draw Date	Winning Numbers	Multiplier
1144	2010-01-05	16 23 25 49 58 20	4.0
1109	2010-01-09	17 20 21 40 51 19	3.0
1083	2010-01-12	05 10 11 12 20 02	3.0
1135	2010-02-06	04 09 14 39 43 38	4.0
1100	2010-02-10	12 20 30 36 47 25	4.0
...	...	...	...
1427	2023-06-02	05 11 22 23 69 07	2.0
1414	2023-07-01	35 36 44 45 67 14	3.0
1428	2023-08-02	52 58 59 64 66 09	2.0
1415	2023-09-01	18 43 48 60 69 14	3.0
1416	2023-11-01	04 08 46 47 48 05	3.0

1429 rows × 3 columns

In [7]:

```
df['Winning Numbers']
```

Out[7]:

```
1144    16 23 25 49 58 20
1109    17 20 21 40 51 19
1083    05 10 11 12 20 02
1135    04 09 14 39 43 38
1100    12 20 30 36 47 25
              ...
1427    05 11 22 23 69 07
1414    35 36 44 45 67 14
1428    52 58 59 64 66 09
1415    18 43 48 60 69 14
1416    04 08 46 47 48 05
Name: Winning Numbers, Length: 1429, dtype: object
```

In [8]:

```
#Splitting the Winning Numbers columns in 6 separate columns
df['Winning Numbers']=df['Winning Numbers'].str.split()
df[['ball1','ball2','ball3','ball4','ball5','ball6']]=df['Winning Numbers'].apply(lambda
df[['ball1','ball2','ball3','ball4','ball5','ball6']]=df[['ball1','ball2','ball3','ball4
df.head()
```

Out[8]:

	Draw Date	Winning Numbers	Multiplier	ball1	ball2	ball3	ball4	ball5	ball6
1144	2010-01-05	[16, 23, 25, 49, 58, 20]	4.0	16	23	25	49	58	20
1109	2010-01-09	[17, 20, 21, 40, 51, 19]	3.0	17	20	21	40	51	19
1083	2010-01-12	[05, 10, 11, 12, 20, 02]	3.0	5	10	11	12	20	2
1135	2010-02-06	[04, 09, 14, 39, 43, 38]	4.0	4	9	14	39	43	38
1100	2010-02-10	[12, 20, 30, 36, 47, 25]	4.0	12	20	30	36	47	25

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1429 entries, 1144 to 1416
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Draw Date        1429 non-null   object 
 1   Winning Numbers  1429 non-null   object 
 2   Multiplier       1219 non-null   float64
 3   ball1            1429 non-null   int32  
 4   ball2            1429 non-null   int32  
 5   ball3            1429 non-null   int32  
 6   ball4            1429 non-null   int32  
 7   ball5            1429 non-null   int32  
 8   ball6            1429 non-null   int32  
dtypes: float64(1), int32(6), object(2)
memory usage: 78.1+ KB
```

TASK 1: Exploratory Data Analysis

QUESTION 1:How can you identify hidden trends and patterns in the Powerball winning numbers dataset? Are there any specific techniques or algorithms that can be used to uncover these trends?

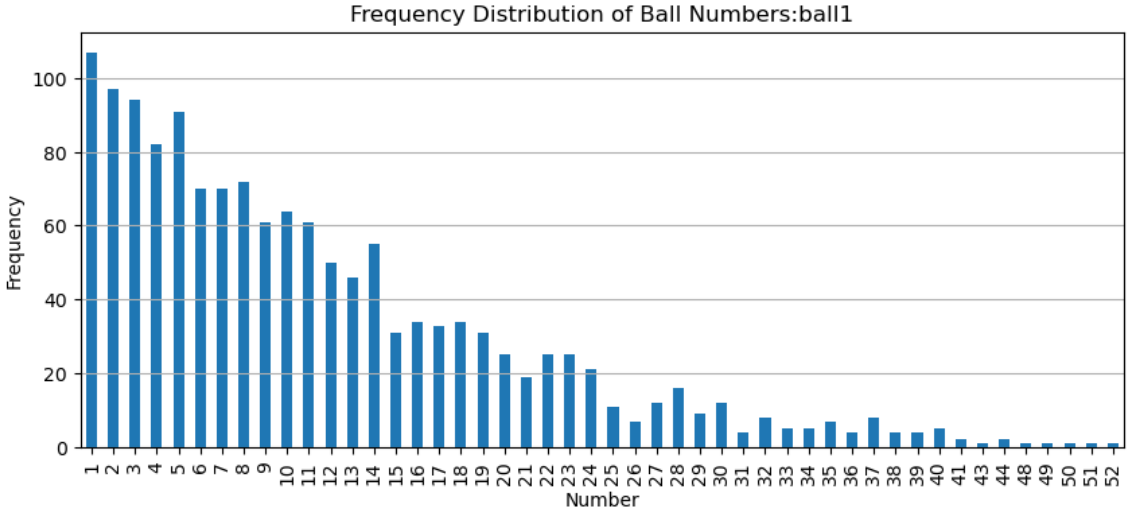
SOLUTION 1: The hidden trends and patterns can be identified using visualisation i.e plotting the curve of the winning trend over the years and identifying the highest occurring number

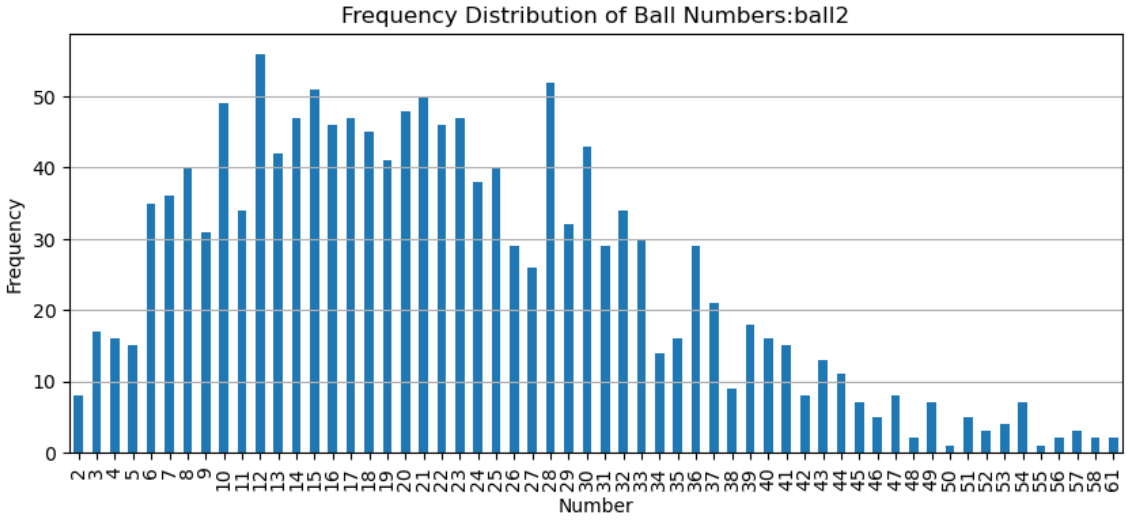
In [10]:

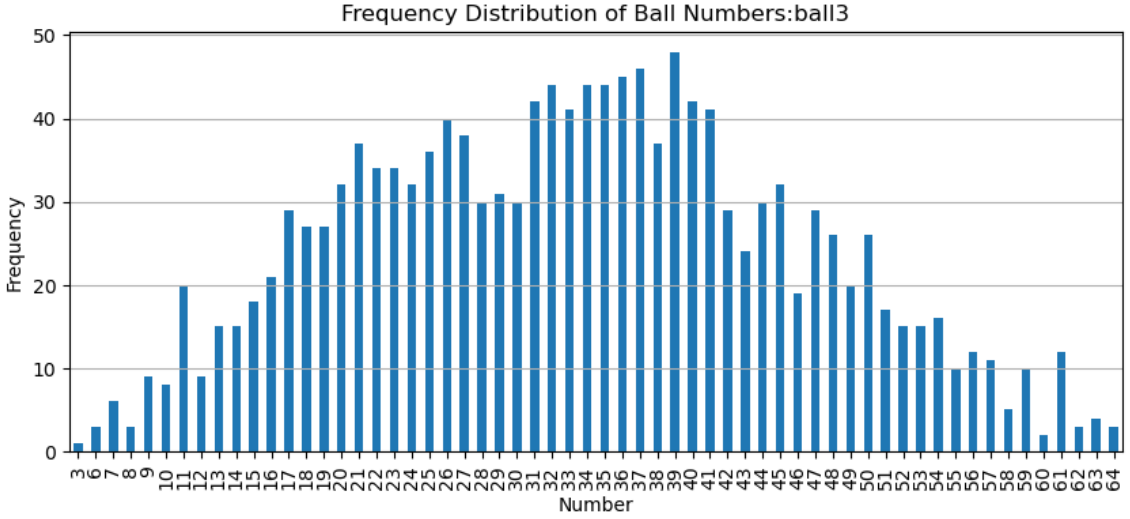
```
balls_data = df[['ball1', 'ball2', 'ball3', 'ball4', 'ball5', 'ball6']]
```

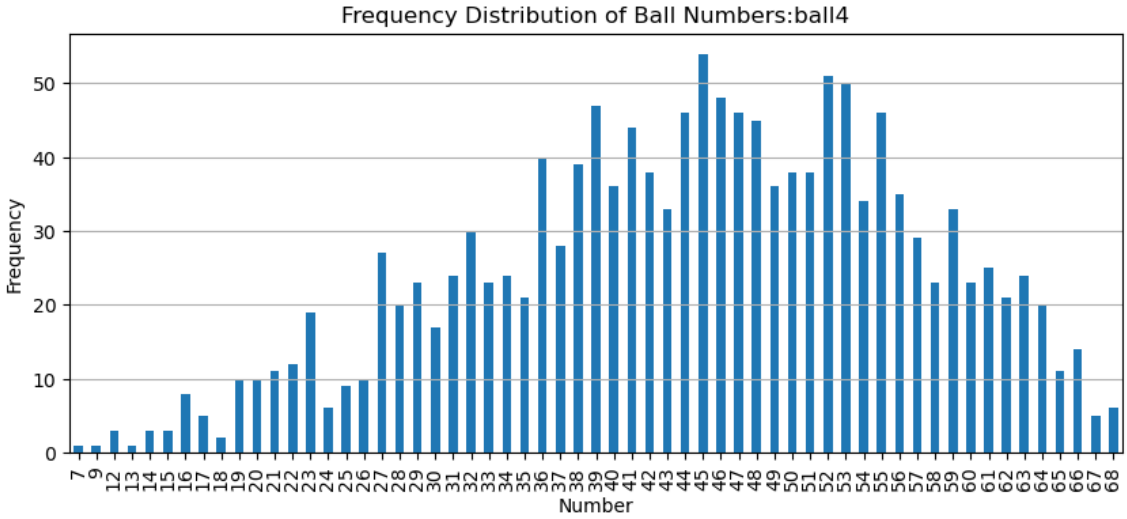
In [11]:

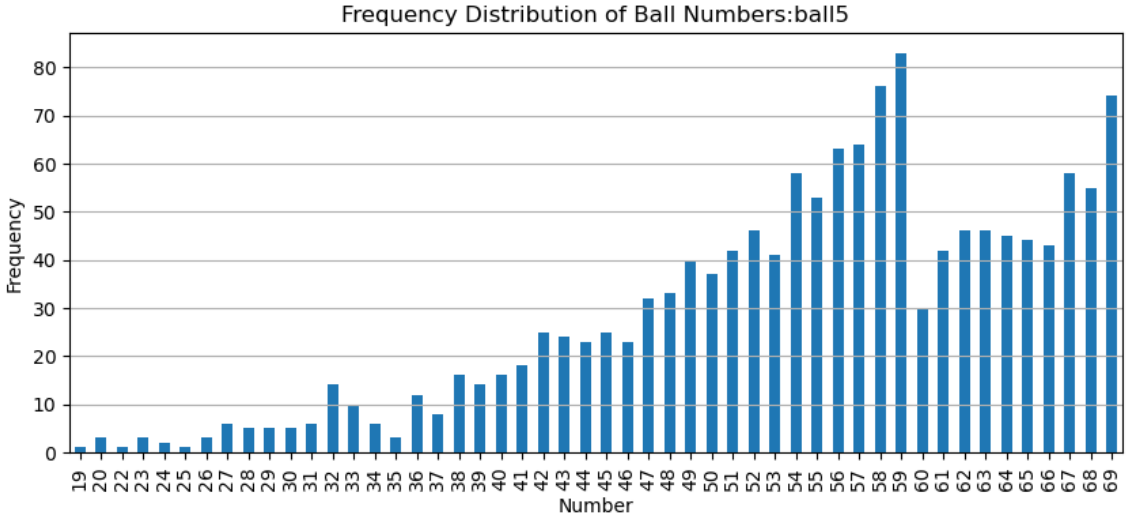
```
#Frequency distribution of different balls
for column in balls_data:
    plt.figure(figsize=(10,4))
    balls_data[column].value_counts().sort_index().plot(kind='bar')
    plt.xlabel('Number')
    plt.ylabel('Frequency')
    plt.title('Frequency Distribution of Ball Numbers:'+column)
    plt.grid(axis='y')
    plt.show()
```

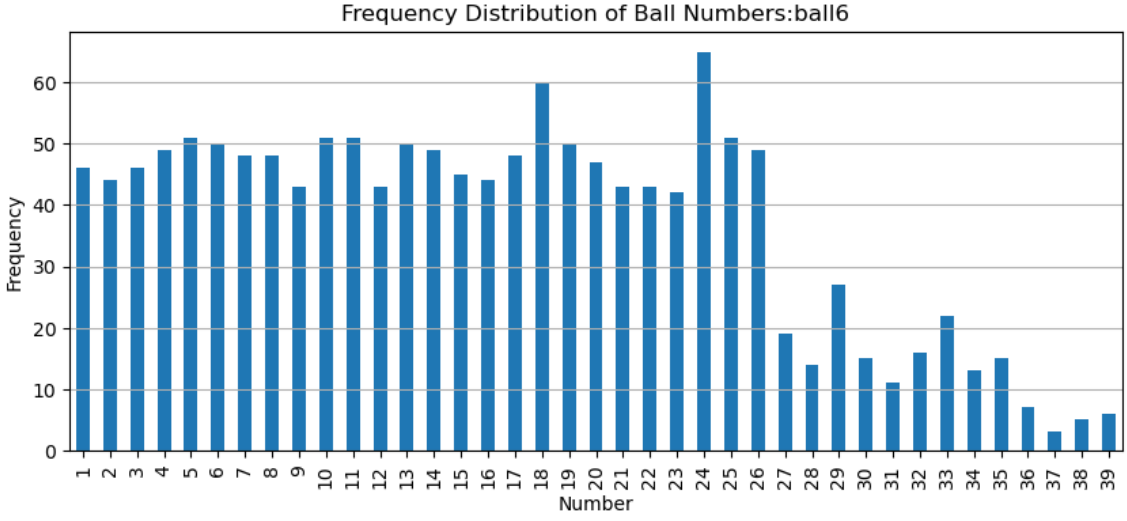












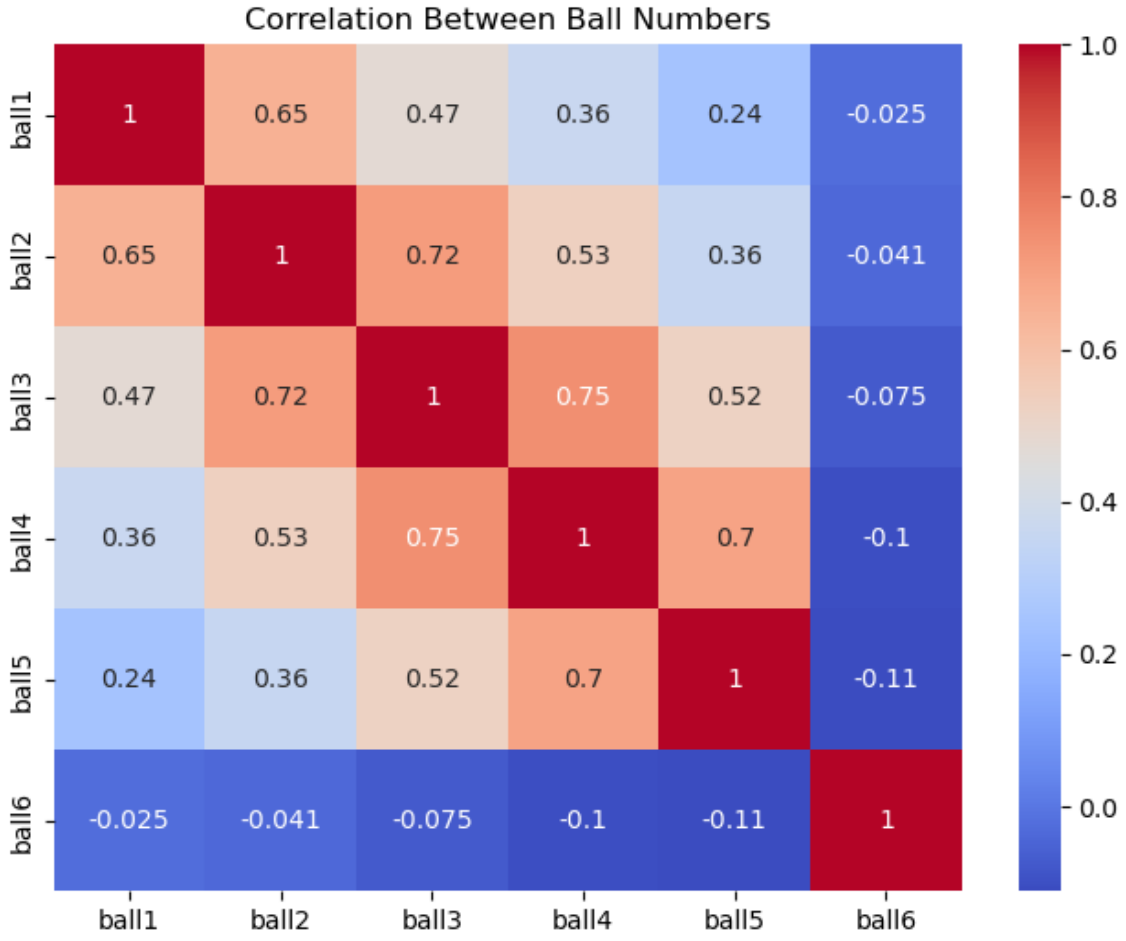
From the above plots, it is clearly visible that:

1) First ball (most of the times) falls within 1 to 15 2) Second ball falls within 10 to 30 3) Third ball has maximum frequency for numbers within 30 to 50 4) Fourth ball falls with 40 to 55 5) Fifth ball has maximum frequencies for numbers 58 and 59 6) Sixth ball has a very constant plot for numbers within 1 to 25

QUESTION 2: Can you provide a visualization of the winning numbers over time? For example, a line plot showing the frequency of each number being drawn or a heat map showing the correlation between different numbers.

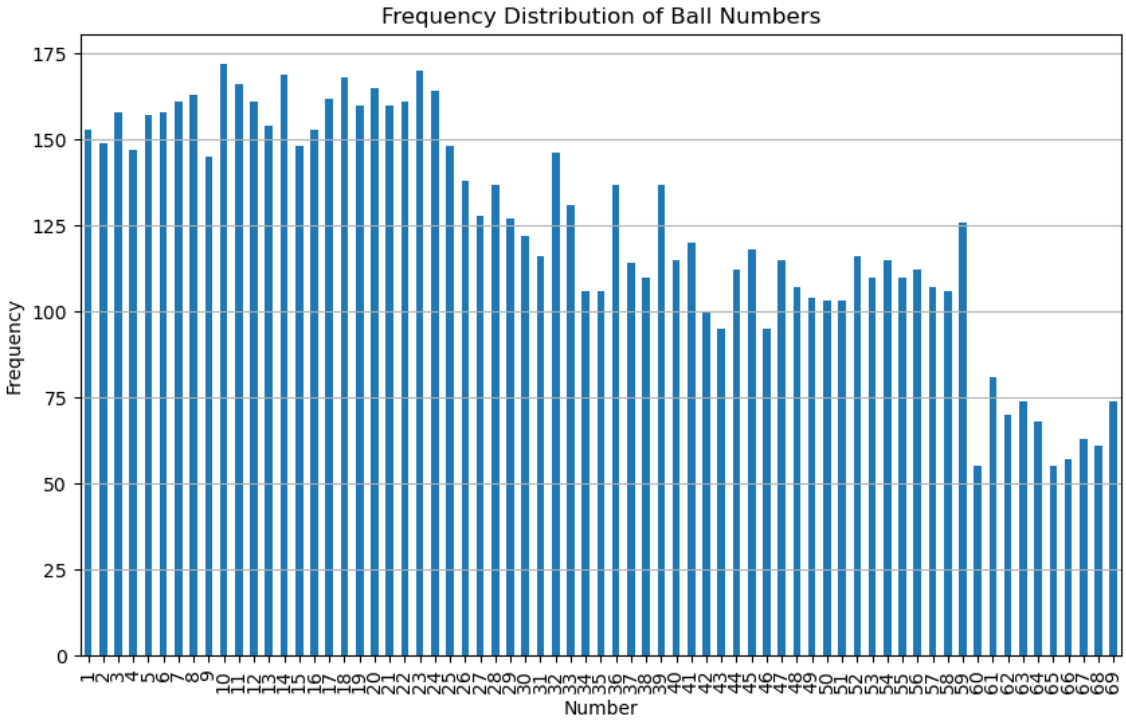
In [14]:

```
#Heatmap showing the correlation betweeen the winning numbers
balls_correlation = balls_data.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(balls_correlation, annot=True, cmap='coolwarm')
plt.title('Correlation Between Ball Numbers')
plt.show()
```



In [15]:

```
#Frequency of all the winning numbers over the years
plt.figure(figsize=(10, 6))
balls_data.stack().value_counts().sort_index().plot(kind='bar')
plt.xlabel('Number')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Ball Numbers')
plt.grid(axis='y')
plt.show()
```



QUESTION 3: What are the characteristics of the luckiest numbers in the Powerball lottery? Are there any specific numbers that appear more frequently than others? Can you provide insights into the frequency distribution of the numbers?

SOLUTION 3: From the above distribution, it is clearly visible that most of the numbers drawn fall under the range [1,25]

QUESTION 4: How can you predict the luckiest number for future Powerball drawings? Are there any machine learning algorithms or statistical techniques that can be employed for this prediction task?

SOLUTION 4: To predict the luckiest number, i've used Prophet Forecasting Model which is an ADDITIVE REGRESSIVE MODEL

QUESTION 5: How can you evaluate the performance of the luckiest number prediction model? What metrics or methods would be appropriate to assess the accuracy of the predicted lucky numbers compared to the actual winning numbers?

SOLUTION 5: To evaluate the performance of the luckiest number prediction model, i've used mean_absolute error and root mean_squared error from sklearn.metrics

TASK 2: Regression

In [16]:

```
# Select the columns 'Draw Date' and 'ball1'
data = df[['Draw Date', 'ball1']]
data.columns = ['ds', 'y']  # Rename columns as required by Prophet
```

In [17]:

```
# Initialize and fit the Prophet model
model = Prophet(daily_seasonality=True)
model.fit(data)
```

```
C:\Users\soumy\anaconda3\lib\site-packages\prophet\forecaster.py:896: Futu
reWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  components = components.append(new_comp)
```

Out[17]:

```
<prophet.forecaster.Prophet at 0x27510786eb0>
```

In [18]:

```
# Make future predictions
future = model.make_future_dataframe(periods=1)
forecast = model.predict(future)

# Print the forecasted values
print(forecast[['ds', 'yhat']].tail(1))
```

```
C:\Users\soumy\anaconda3\lib\site-packages\prophet\forecaster.py:896: Futu
reWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  components = components.append(new_comp)
C:\Users\soumy\anaconda3\lib\site-packages\prophet\forecaster.py:896: Futu
reWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  components = components.append(new_comp)
```

```
            ds       yhat
1429 2023-11-02  11.251244
```

In [20]:

```
# Calculate accuracy metrics
actual_values = data['y'].values  # Actual values from the dataset
predicted_values = forecast['yhat'].values[:len(actual_values)]  # Forecasted values fro

mae = mean_absolute_error(actual_values, predicted_values)
mse = mean_squared_error(actual_values, predicted_values)
rmse = np.sqrt(mse)

print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

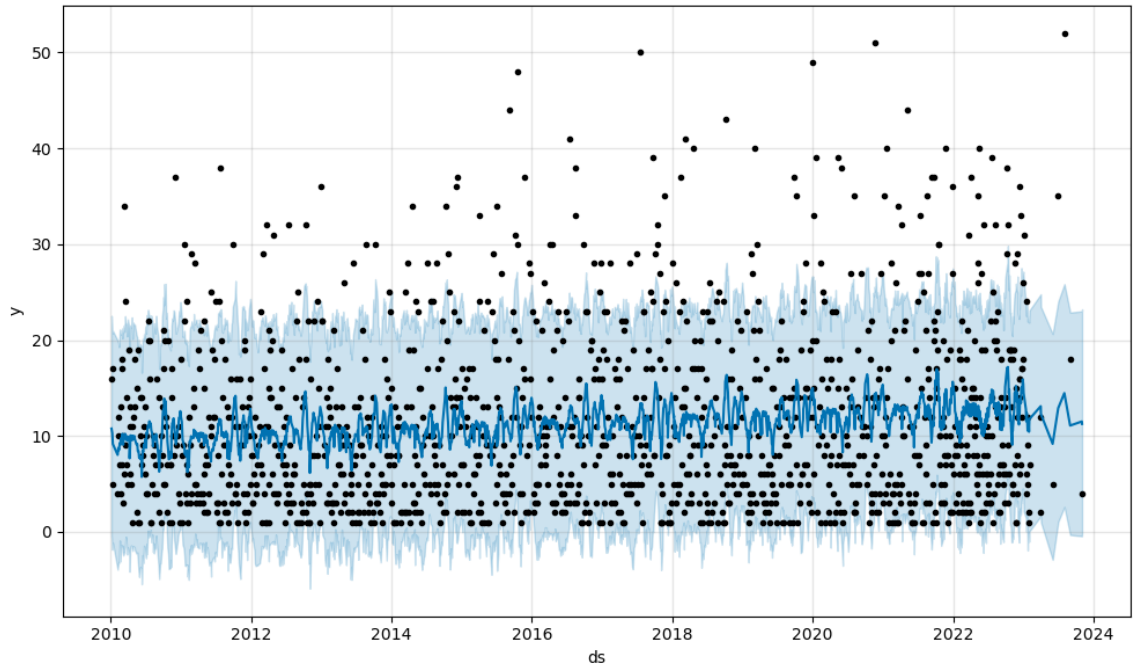
```
Mean Absolute Error (MAE): 7.0671137820011465
Mean Squared Error (MSE): 80.96983816567582
Root Mean Squared Error (RMSE): 8.998324186518056
```

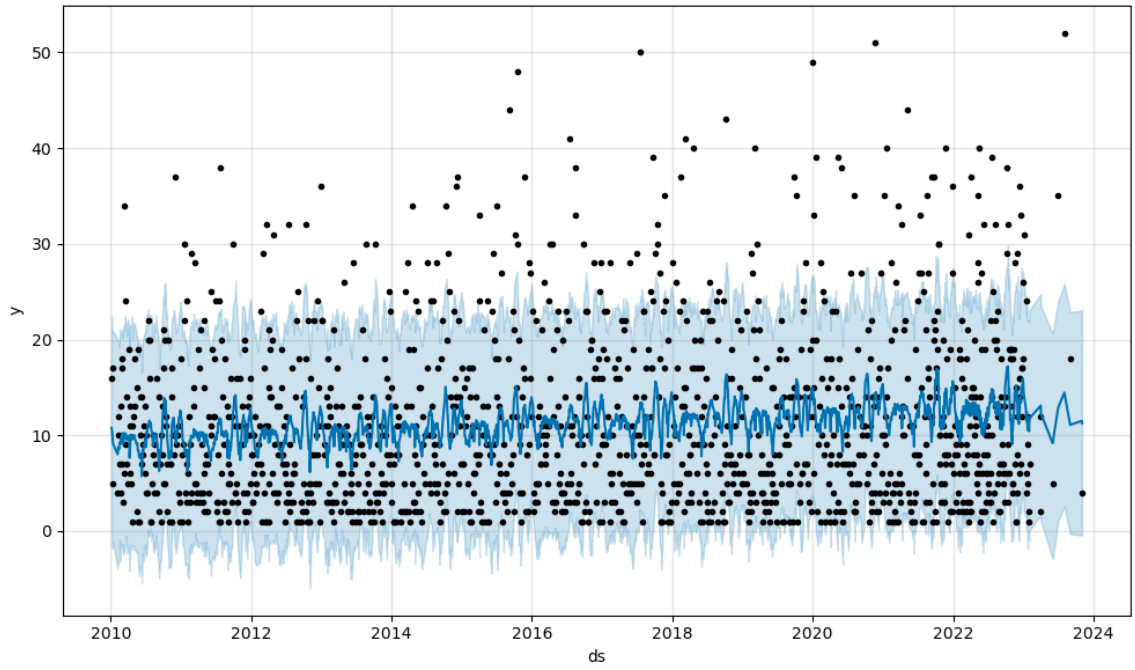
R2 may not be directly applicable for evaluating the accuracy of the Prophet model, as it is primarily designed for evaluating the variance explained by the model in relation to the total variance in the data.

In [21]:

```
model.plot(forecast)
```

Out[21]:



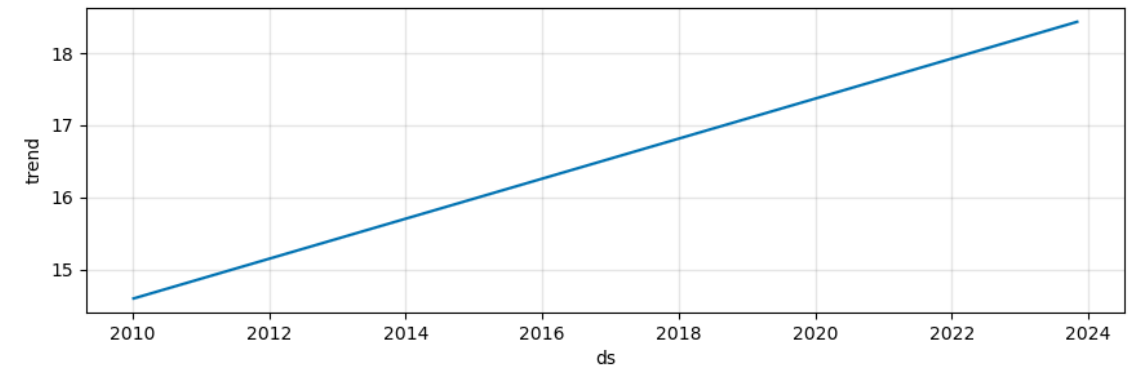


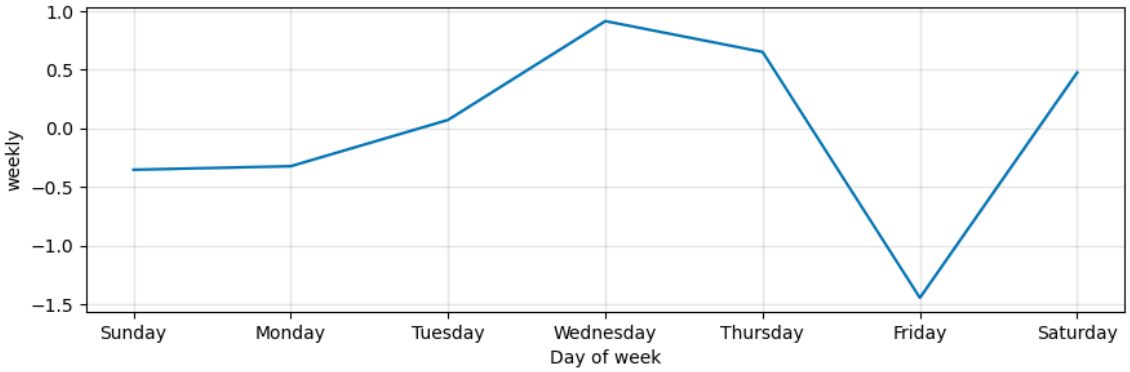
In [22]:

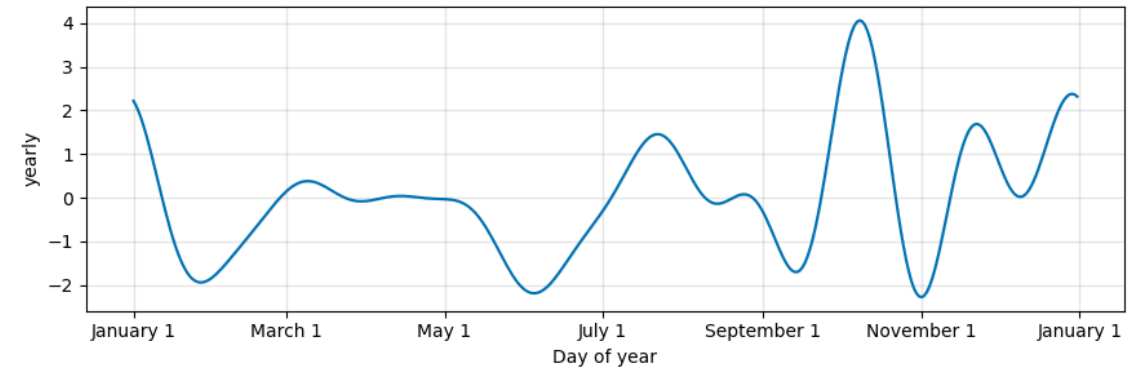
```
model.plot_components(forecast)
```

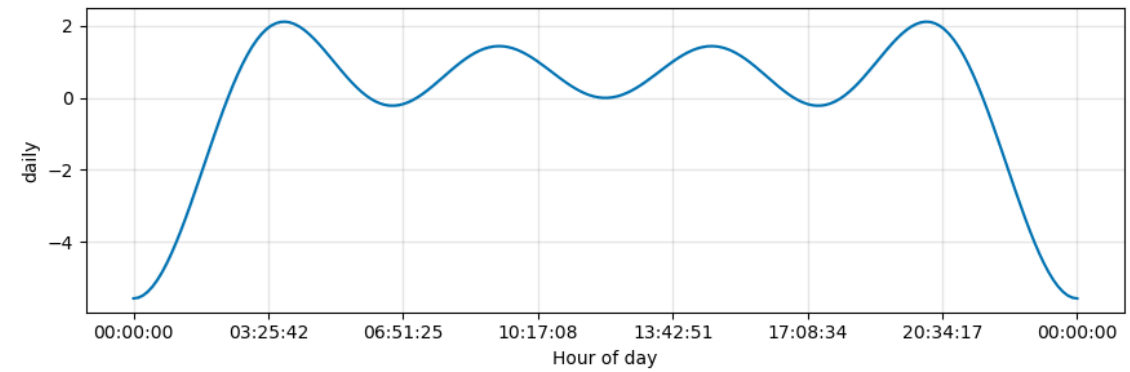
```
C:\Users\soumy\anaconda3\lib\site-packages\prophet\forecaster.py:896: Futu
reWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  components = components.append(new_comp)
C:\Users\soumy\anaconda3\lib\site-packages\prophet\forecaster.py:896: Futu
reWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  components = components.append(new_comp)
C:\Users\soumy\anaconda3\lib\site-packages\prophet\forecaster.py:896: Futu
reWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  components = components.append(new_comp)
```

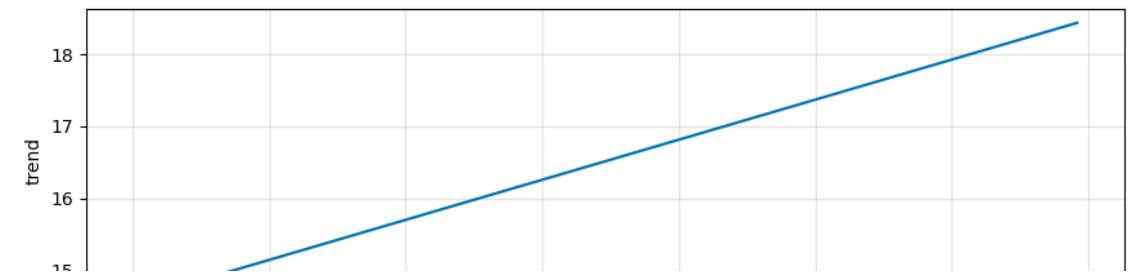
Out[22]:


trend
18
17
16
15
2010
2012
2014
2016
2018
2020
2022
2024
ds


weekly
1.0
0.5
0.0
−0.5
−1.0
−1.5
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
4
3
2
1
0
−1
−2
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
2
0
−2
−4
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day


trend
18
17
16