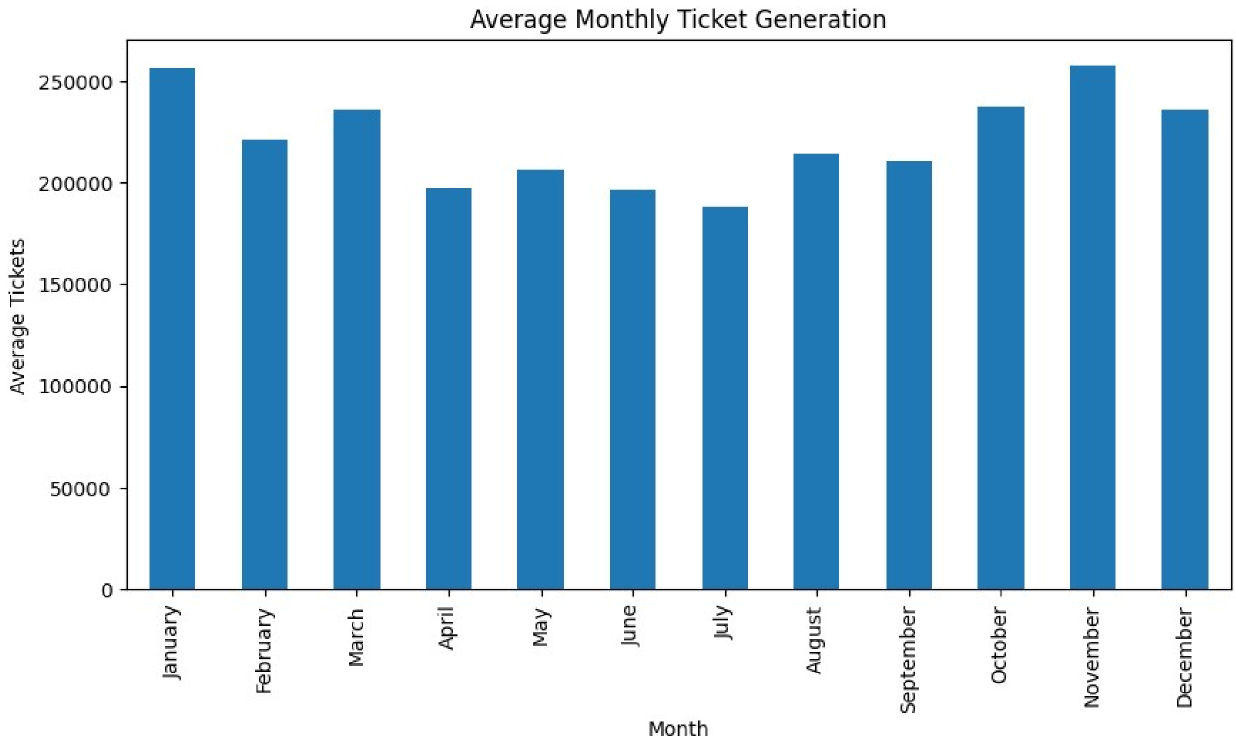
Ticket analysis from year 2022-2024

```
import pandas as pd
import matplotlib.pyplot as plt

# Data Preparation
data = {
    "Year": [2022]*12 + [2023]*12 + [2024]*4,
    "Month": ["January", "February", "March", "April", "May", "June",
"July", "August",
              "September", "October", "November", "December"]*2 +
             ["January", "February", "March", "April"],
    "Tickets": [133455, 122042, 151941, 151272, 160986, 165934,
166465, 175561,
                192221, 220097, 233420, 226298, 288409, 243657,
271454, 237398,
                251319, 227660, 209395, 252152, 229505, 254973,
281354, 245069,
                347618, 297993, 283800, 203942]
}
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' +
df['Month'])

# Descriptive Statistics
desc_stats = df['Tickets'].describe()
total_months = desc_stats['count']
average_tickets = desc_stats['mean']
std_dev_tickets = desc_stats['std']
min_tickets = desc_stats['min']
max_tickets = desc_stats['max']

print("Descriptive Statistics Summary:")
print(f"Total Months Analyzed: {total_months}")
print(f"Average Monthly Tickets Generated: {average_tickets:.0f}")
print(f"Variability in Ticket Generation: The standard deviation is
about {std_dev_tickets:.0f} tickets.")
print(f"Range of Ticket Generation: from {min_tickets:.0f} to
{max_tickets:.0f} tickets.")

# Yearly Overview
yearly_summary = df.groupby('Year')['Tickets'].agg(['sum', 'mean',
'std'])

print("\nYearly Ticket Generation Overview:")
for year, stats in yearly_summary.iterrows():
    print(f"{year}: Average monthly tickets generated were around
{stats['mean']:.0f}, with a total of {stats['sum']} tickets generated
throughout the year. Standard deviation was {stats['std']:.0f}.")
```

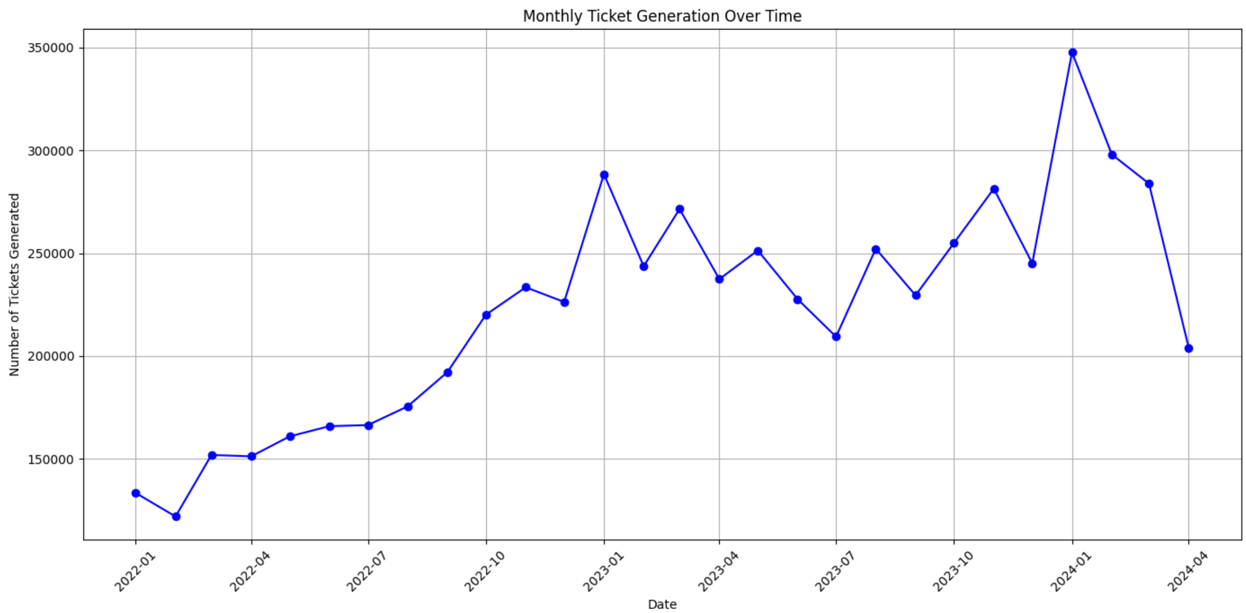
```python
# Seasonal Analysis
monthly_summary = df.groupby(df['Date'].dt.month_name())
['Tickets'].mean().reindex([
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
])

# Plotting monthly trends
plt.figure(figsize=(10, 5))
monthly_summary.plot(kind='bar')
plt.title('Average Monthly Ticket Generation')
plt.xlabel('Month')
plt.ylabel('Average Tickets')
plt.show()

df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' +
df['Month'])
plt.figure(figsize=(14, 7))
plt.plot(df['Date'], df['Tickets'], marker='o', linestyle='-',
color='b')
plt.title('Monthly Ticket Generation Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Tickets Generated')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
C:\Users\venu\AppData\Local\Temp\ipykernel_11556\2715160722.py:16:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is
consistent and as-expected, please specify a format.
  df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' +
df['Month'])

Descriptive Statistics Summary:
Total Months Analyzed: 28.0
Average Monthly Tickets Generated: 222335
Variability in Ticket Generation: The standard deviation is about
54876 tickets.
Range of Ticket Generation: from 122042 to 347618 tickets.

Yearly Ticket Generation Overview:
2022: Average monthly tickets generated were around 174974, with a
total of 2099692.0 tickets generated throughout the year. Standard
deviation was 36121.
2023: Average monthly tickets generated were around 249362, with a
total of 2992345.0 tickets generated throughout the year. Standard
deviation was 22836.
```

```
2024: Average monthly tickets generated were around 283338, with a
total of 1133353.0 tickets generated throughout the year. Standard
deviation was 59584.
```




```
C:\Users\venu\AppData\Local\Temp\ipykernel_11556\2715160722.py:53:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is
consistent and as-expected, please specify a format.
  df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' +
df['Month'])
```



Forcast analysis Model Comparison and Best Fit: To determine which model is the best fit, one would typically look at various accuracy metrics (like RMSE, MAE, etc.), plot the residuals to check for patterns, and compare how well the forecasts align with known trends and seasonal patterns. However, based on typical use cases:

SARIMA is generally robust for datasets with strong seasonal patterns and can adapt well to the complexity of different time cycles. Exponential Smoothing is often best when the seasonal pattern changes proportionally over time. ARIMA is straightforward and effective but may underperform if strong seasonality is present, as it doesn't inherently model this.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA

# Prepare the data
data = {
    "Year": [2022]*12 + [2023]*12 + [2024]*4,
    "Month": ["January", "February", "March", "April", "May", "June",
"July", "August",
              "September", "October", "November", "December"]*2 +
             ["January", "February", "March", "April"],
    "Tickets": [133455, 122042, 151941, 151272, 160986, 165934,
166465, 175561,
                192221, 220097, 233420, 226298, 288409, 243657,
271454, 237398,
                251319, 227660, 209395, 252152, 229505, 254973,
281354, 245069,
                347618, 297993, 283800, 203942]
```

```python
}
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Year'].astype(str) + df['Month'],
format='%Y%B')
df.set_index('Date', inplace=True)

# Create a time series
time_series = df['Tickets']

# Model setups
sarima_model = SARIMAX(time_series, order=(1, 1, 1),
seasonal_order=(1, 1, 1, 12))
sarima_results = sarima_model.fit()
exp_smoothing_model = ExponentialSmoothing(time_series, trend='add',
seasonal='mul', seasonal_periods=12)
exp_smoothing_results = exp_smoothing_model.fit()
arima_model = ARIMA(time_series, order=(1, 1, 1))
arima_results = arima_model.fit()

# Predictions
future_periods = pd.date_range('2024-05-01', '2024-12-01', freq='MS')
sarima_forecast = sarima_results.predict(start=future_periods[0],
end=future_periods[-1])
exp_smoothing_forecast =
exp_smoothing_results.predict(start=future_periods[0],
end=future_periods[-1])
arima_forecast = arima_results.predict(start=future_periods[0],
end=future_periods[-1])

# Create a DataFrame for the forecasted values
forecast_df = pd.DataFrame({
    'Month': future_periods.month_name(),
    'SARIMA Forecast': sarima_forecast,
    'Exponential Smoothing Forecast': exp_smoothing_forecast,
    'ARIMA Forecast': arima_forecast
}, index=future_periods)

# Display the DataFrame
print(forecast_df)
```

```
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
```

```
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:966: UserWarning: Non-
stationary starting autoregressive parameters found. Using zeros as
starting parameters.
  warn('Non-stationary starting autoregressive parameters'
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-
invertible starting MA parameters found. Using zeros as starting
parameters.
  warn('Non-invertible starting MA parameters found.'
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:866: UserWarning: Too
few observations to estimate starting parameters for seasonal ARMA.
All parameters except for variances will be set to zeros.
  warn('Too few observations to estimate starting parameters%s.'
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)

                Month  SARIMA Forecast  Exponential Smoothing Forecast  
\
2024-05-01        May    199666.424726                   230822.855929   

2024-06-01       June    162792.701373                   209765.725912   

2024-07-01       July    130561.164823                   213157.471900   

2024-08-01     August    154392.880523                   214373.706820   
```

```
2024-09-01  September    124363.347602                          226204.279818

2024-10-01    October    139715.021215                          251937.597793

2024-11-01   November    152916.664751                          261718.595813

2024-12-01   December    107290.947714                          249892.819854


            ARIMA Forecast
2024-05-01   228663.464687
2024-06-01   213473.919697
2024-07-01   222806.792159
2024-08-01   217072.419704
2024-09-01   220595.775197
2024-10-01   218430.929082
2024-11-01   219761.069827
2024-12-01   218943.794923
```

```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Prepare the data
data = {
    "Year": [2022]*12 + [2023]*12 + [2024]*4,
    "Month": ["January", "February", "March", "April", "May", "June",
"July", "August",
              "September", "October", "November", "December"]*2 +
             ["January", "February", "March", "April"],
    "Tickets": [133455, 122042, 151941, 151272, 160986, 165934,
166465, 175561,
                192221, 220097, 233420, 226298, 288409, 243657,
271454, 237398,
                251319, 227660, 209395, 252152, 229505, 254973,
281354, 245069,
                347618, 297993, 283800, 203942]
}
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Year'].astype(str) + df['Month'],
format='%Y%B')
df.set_index('Date', inplace=True)

# Splitting data into train and test
train = df.loc[:'2023-12']
test = df.loc['2024-01':]
```

```python
# Fit models
sarima_model = SARIMAX(train['Tickets'], order=(1, 1, 1),
seasonal_order=(1, 1, 1, 12))
sarima_results = sarima_model.fit()
exp_smoothing_model = ExponentialSmoothing(train['Tickets'],
trend='add', seasonal='mul', seasonal_periods=12)
exp_smoothing_results = exp_smoothing_model.fit()
arima_model = ARIMA(train['Tickets'], order=(1, 1, 1))
arima_results = arima_model.fit()

# Forecasting
sarima_forecast = sarima_results.forecast(steps=len(test))
exp_smoothing_forecast =
exp_smoothing_results.forecast(steps=len(test))
arima_forecast = arima_results.forecast(steps=len(test))

# Calculate accuracy
sarima_rmse = np.sqrt(mean_squared_error(test['Tickets'],
sarima_forecast))
exp_smoothing_rmse = np.sqrt(mean_squared_error(test['Tickets'],
exp_smoothing_forecast))
arima_rmse = np.sqrt(mean_squared_error(test['Tickets'],
arima_forecast))

# Print results
print(f"SARIMA RMSE: {sarima_rmse}")
print(f"Exponential Smoothing RMSE: {exp_smoothing_rmse}")
print(f"ARIMA RMSE: {arima_rmse}")
```

```
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:866: UserWarning: Too
few observations to estimate starting parameters for seasonal ARMA.
All parameters except for variances will be set to zeros.
  warn('Too few observations to estimate starting parameters%s.'
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
```

```
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\holtwinters\model.py:917: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)
c:\Users\venu\AppData\Local\Programs\Python\Python38\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No
frequency information was provided, so inferred frequency MS will be
used.
  self._init_dates(dates, freq)

SARIMA RMSE: 120472.54955205708
Exponential Smoothing RMSE: 40130.63043613415
ARIMA RMSE: 59111.710741575705
```