

Gezinomia Teknik Dökümanı - Veri Ön Hazırlığı ve İş Probleminin Belirlenmesi

İş Problemi = State Group isimli şirketler grubu turizm ve otelcilik alanında farklı yatırımları bulunan bir firma. Firma yeni bir tatil köyü için yatırım yapmayı düşünüyor.

Otelin; çalışma konsepti, açılacağı şehir ve fiyatlandırma konusunda destege ihtiyacı var. Yatırım danışmanlığı için firmamız Gezinomia'ya gelen şirket yatırım depertmanına gerekli bilgileri sağlayacağız.

Bunun için firmamız elde ettiği veri setini düzenleyerek anlamlı bir çıktı alacak hale getirecek. Çıktıları firmaya anlamlı bir şekilde ifade edebilmek için görselleştirmeler yapılacak.

Daha sonra bir tahmin modeli geliştirerek farklı şehir seçimlerine göre hangi konseptte ne kadar başarılı olunabileceğini tahmin eden bir tahmin modeli geliştireceğiz. Bu model şehir isminin girdisiyle tavsiye edilen konseptin ve tavsiye edilen fiyatlandırmanın çıktısını verecek.

In [1]: # Gerekli kütüphanelerin import edilmesi ile başlıyoruz.

```
import numpy as np
import pandas as pd
import re
import locale
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

In [2]: df_ = pd.read_csv("Desktop/MIUUL Veri Analisti/Gezinomia Final/gezinomi_csv.csv", encoding="utf-8")
Dosyayı tarattıktan sonra Türkçe karakterlerin hata yarattığını gördük. Bu yüzden encoding düzenlemesi yapıyorum.
df_.head(10)
Veri setimize genel bir bakış atıyoruz.

Out[2]:

| | SaleId | SaleDate | CheckInDate | Price | ConceptName | SaleCityName | CInDay | SaleCheckInDayDiff | Seasons | SaleDate.1 | Ct |
|---|--------|--------------------------------|--------------------------------|-------------|--------------------|--------------|----------|--------------------|---------|--------------------------|--------------------------------|
| 0 | 415122 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 79,3040293 | Her_ey Dahil | Antalya | Saturday | | 0 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 1 | 415103 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 45,97069597 | Yarışm Pansiyon | Antalya | Saturday | | 0 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 2 | 404034 | 2022-09-12 00:00:00.0000000 | 2022-09-13 00:00:00.0000000 | 77,83882784 | Her_ey Dahil | Antalya | Tuesday | | 1 | High 00:00:00.0000000 | 2022-09-12 00:00:00.0000000 |
| 3 | 415094 | 2022-12-03 00:00:00.0000000 | 2022-12-10 00:00:00.0000000 | 222,7106227 | Yarışm Pansiyon | İzmir | Saturday | | 7 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 4 | 414951 | 2022-12-01 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 140,4761905 | Yarışm Pansiyon | İzmir | Saturday | | 2 | Low 00:00:00.0000000 | 2022-12-01 00:00:00.0000000 |
| 5 | 415091 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 70,26862027 | Yarışm Pansiyon | İzmir | Saturday | | 0 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 6 | 415085 | 2022-12-03 00:00:00.0000000 | 2022-12-09 00:00:00.0000000 | 45,78754579 | Yarışm Pansiyon | Antalya | Friday | | 6 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 7 | 415084 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 51,23931624 | Her_ey Dahil | Antalya | Saturday | | 0 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 8 | 415081 | 2022-12-03 00:00:00.0000000 | 2022-12-04 00:00:00.0000000 | 77,28937729 | Yarışm Pansiyon | İzmir | Sunday | | 1 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |
| 9 | 415079 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 68,68131868 | Yarışm Pansiyon | Diğer | Saturday | | 0 | Low 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 |

In [3]: # Orjinal veri setini korumak ve işlemlerimizi uygulamak için ilk aşamada bir kopya oluşturuyoruz.
df = df_.copy()

In [4]: # encoding denemesinden sonra gördük ki Türkçe karakter sorunu halen mevcut.
Türkçe karakterlere düzenleme yapmak için bir fonksiyon oluşturuyoruz. Bu fonksiyonu oluştururken if döngüsü içinde replace fonksiyonunu kullanıyoruz.
def fix_turkish_characters(text):
 if isinstance(text, str):
 # İlk harfi "_" olan yerleri "İ" ile değiştirelim
 if text.startswith("_"):
 text = "İ" + text[1:]
 text = text.replace("ş", "ı")
 text = re.sub(r"(?<=[a-zA-Z])_(?=([a-zA-Z])", "ş", text)
 # girdiğimiz "_"yi "ş" ile değiştirme fonksiyonu Birkaç şehir değişkenini bozduğu için onları o şekilde koruyarak replace fonksiyonunu kullanıyoruz.
 text = text.replace("Dişer", "Diğer")
 text = text.replace("Muşla", "Muğla")

 return text

Daha sonra bu fonksiyonu Türkçe karakterlerin sorun çıkarttığı sütunlara uyguluyoruz.
df["ConceptName"] = df["ConceptName"].apply(fix_turkish_characters)
df["SaleCityName"] = df["SaleCityName"].apply(fix_turkish_characters)
df["ConceptName.1"] = df["ConceptName.1"].apply(fix_turkish_characters)
df["SaleCityName.1"] = df["SaleCityName.1"].apply(fix_turkish_characters)

```
df.head(20)
```

| Out[4]: | SaleId | SaleDate | CheckInDate | Price | ConceptName | SaleCityName | CInDay | SaleCheckInDayDiff | Seasons | SaleDate.1 |
|---------|--------|--------------------------------|--------------------------------|-------------|----------------|--------------|-----------|--------------------|---------|--------------------------|
| 0 | 415122 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 79,3040293 | Herşey Dahil | Antalya | Saturday | | 0 | Low 00:00:00.0000000 |
| 1 | 415103 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 45,97069597 | Yarım Pansiyon | Antalya | Saturday | | 0 | Low 00:00:00.0000000 |
| 2 | 404034 | 2022-09-12 00:00:00.0000000 | 2022-09-13 00:00:00.0000000 | 77,83882784 | Herşey Dahil | Antalya | Tuesday | | 1 | High 00:00:00.0000000 |
| 3 | 415094 | 2022-12-03 00:00:00.0000000 | 2022-12-10 00:00:00.0000000 | 222,7106227 | Yarım Pansiyon | İzmir | Saturday | | 7 | Low 00:00:00.0000000 |
| 4 | 414951 | 2022-12-01 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 140,4761905 | Yarım Pansiyon | İzmir | Saturday | | 2 | Low 00:00:00.0000000 |
| 5 | 415091 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 70,26862027 | Yarım Pansiyon | İzmir | Saturday | | 0 | Low 00:00:00.0000000 |
| 6 | 415085 | 2022-12-03 00:00:00.0000000 | 2022-12-09 00:00:00.0000000 | 45,78754579 | Yarım Pansiyon | Antalya | Friday | | 6 | Low 00:00:00.0000000 |
| 7 | 415084 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 51,23931624 | Herşey Dahil | Antalya | Saturday | | 0 | Low 00:00:00.0000000 |
| 8 | 415081 | 2022-12-03 00:00:00.0000000 | 2022-12-04 00:00:00.0000000 | 77,28937729 | Yarım Pansiyon | İzmir | Sunday | | 1 | Low 00:00:00.0000000 |
| 9 | 415079 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 68,68131868 | Yarım Pansiyon | Diğer | Saturday | | 0 | Low 00:00:00.0000000 |
| 10 | 415076 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 103,021978 | Yarım Pansiyon | Diğer | Saturday | | 0 | Low 00:00:00.0000000 |
| 11 | 415078 | 2022-12-03 00:00:00.0000000 | 2022-12-07 00:00:00.0000000 | 39,68253968 | Oda + Kahvaltı | Diğer | Wednesday | | 4 | Low 00:00:00.0000000 |
| 12 | 415075 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 53,35775336 | Oda + Kahvaltı | İzmir | Saturday | | 0 | Low 00:00:00.0000000 |
| 13 | 415072 | 2022-12-03 00:00:00.0000000 | 2023-01-23 00:00:00.0000000 | 37,39052726 | Yarım Pansiyon | İzmir | Monday | | 51 | Low 00:00:00.0000000 |
| 14 | 415071 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 70,26862027 | Yarım Pansiyon | İzmir | Saturday | | 0 | Low 00:00:00.0000000 |
| 15 | 415066 | 2022-12-02 00:00:00.0000000 | 2022-12-31 00:00:00.0000000 | 108,7454212 | Yarım Pansiyon | Diğer | Saturday | | 29 | Low 00:00:00.0000000 |
| 16 | 415066 | 2022-12-02 00:00:00.0000000 | 2022-12-31 00:00:00.0000000 | 144,993895 | Yarım Pansiyon | Diğer | Saturday | | 29 | Low 00:00:00.0000000 |
| 17 | 415064 | 2022-12-02 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 70,26862027 | Yarım Pansiyon | İzmir | Saturday | | 1 | Low 00:00:00.0000000 |
| 18 | 415055 | 2022-12-02 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 42,57224257 | Yarım Pansiyon | İzmir | Saturday | | 1 | Low 00:00:00.0000000 |
| 19 | 415043 | 2022-12-02 00:00:00.0000000 | 2023-01-27 00:00:00.0000000 | 77,21179625 | Yarım Pansiyon | Diğer | Friday | | 56 | Low 00:00:00.0000000 |

* Veri setini anlaşılır ve daha kolay okunabilir hale getirdik. Şimdi veri setini daha yakından tanıyoruz.

```
In [6]: # Veri setinin genişliği hakkında genel bilgi ediniyoruz.  
df.shape
```

```
Out[6]: (59164, 17)
```

```
In [7]: # Veri setini tanıtmak ve kolon isimlerini gözlemek için "info" komutunu kullanıyoruz. Null Count özelliği ve data type göstergesi de bize yardımcı oluyor.  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59164 entries, 0 to 59163
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   SaleId             59164 non-null   int64  
 1   SaleDate           59164 non-null   object  
 2   CheckInDate        59164 non-null   object  
 3   Price              59151 non-null   object  
 4   ConceptName        59164 non-null   object  
 5   SaleCityName       59164 non-null   object  
 6   CInDay             59164 non-null   object  
 7   SaleCheckInDayDiff 59164 non-null   int64  
 8   Seasons            59164 non-null   object  
 9   SaleDate.1         59164 non-null   object  
 10  CheckInDate.1     59164 non-null   object  
 11  Price.1            59151 non-null   object  
 12  ConceptName.1     59164 non-null   object  
 13  SaleCityName.1    59164 non-null   object  
 14  CInDay.1          59164 non-null   object  
 15  SaleCheckInDayDiff.1 59164 non-null   int64  
 16  Seasons.1         59164 non-null   object  
dtypes: int64(3), object(14)
memory usage: 7.7+ MB
```

```
In [8]: # Değişkenlerin data tiplerini teker teker öğreniyoruz.
df.dtypes
```

```
Out[8]: SaleId          int64
SaleDate         object
CheckInDate      object
Price            object
ConceptName      object
SaleCityName     object
CInDay           object
SaleCheckInDayDiff int64
Seasons          object
SaleDate.1        object
CheckInDate.1    object
Price.1          object
ConceptName.1    object
SaleCityName.1   object
CInDay.1         object
SaleCheckInDayDiff.1 int64
Seasons.1        object
dtype: object
```

```
In [9]: # Datada null değer var mı ? Varsa hangi Kolonda ve kaç tane var sorusunu gerçekleştiriyoruz.
df.isnull().sum()
```

```
Out[9]: SaleId          0
SaleDate         0
CheckInDate      0
Price            13
ConceptName      0
SaleCityName     0
CInDay           0
SaleCheckInDayDiff 0
Seasons          0
SaleDate.1        0
CheckInDate.1    0
Price.1          13
ConceptName.1    0
SaleCityName.1   0
CInDay.1         0
SaleCheckInDayDiff.1 0
Seasons.1        0
dtype: int64
```

* Price isimli değişkende 13 null değer olduğunu görüyoruz.

Bu değerler şehirlerin ve konseptlerin karşılığında elde edeceğimiz ve müşterimize tavsiye edeceğimiz stratejik hareketleri kötü etkileyebilir. Şehir ve Konsept değişkenine bağlı olan bir değişken olduğu için bu iki değeri gruplayarak bir ortalama alacağız ve bu şekilde dolduracağız.

```
In [11]: # Price'Ları float a dönüştürmeden agg function failed hatası alıyordu. Bu yüzden başına bir düzeltme ekledim.
df["Price"] = df["Price"].str.replace(",",".").astype(float)
df["Price"] = df["Price"].round(2)

mean_prices = df.groupby(["SaleCityName", "ConceptName"])["Price"].transform("mean")

# Null değerleri bu ortalamalarla dolduralım
df["Price"].fillna(mean_prices, inplace=True)
```

```
In [12]: df.isnull().sum()
```

```
Out[12]: SaleId          0
SaleDate          0
CheckInDate      0
Price             0
ConceptName       0
SaleCityName      0
CInDay            0
SaleCheckInDayDiff 0
Seasons           0
SaleDate.1        0
CheckInDate.1    0
Price.1           13
ConceptName.1     0
SaleCityName.1    0
CInDay.1          0
SaleCheckInDayDiff.1 0
Seasons.1         0
dtype: int64
```

* Bunlarla birlikte birde sonunda .1 ifadesi olan değişkenler mevcut. Ortalama değer alma, null doldurma vb. gibi ön hazırlık adımlarında bu değerlerin duplicate oluşu bir soruna yol açabilir. Birebir aynı olduğunu teyit ettiğim bu verileri düşüreceğim.

```
In [14]: # Önce drop olacak değişken isimlerini yazıyorum.
columns_to_drop = [
    "SaleDate.1", "CheckInDate.1", "Price.1", "ConceptName.1",
    "SaleCityName.1", "CInDay.1", "SaleCheckInDayDiff.1", "Seasons.1" ]
# Daha sonra inplace özelliğini True yaparak devam ediyoruz.
df.drop(columns=columns_to_drop, inplace=True)

df.head()
```

| | SaleId | SaleDate | CheckInDate | Price | ConceptName | SaleCityName | CInDay | SaleCheckInDayDiff | Seasons |
|---|--------|-----------------------------|-----------------------------|--------|----------------|--------------|----------|--------------------|---------|
| 0 | 415122 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 79.30 | Herşey Dahil | Antalya | Saturday | 0 | Low |
| 1 | 415103 | 2022-12-03 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 45.97 | Yarım Pansiyon | Antalya | Saturday | 0 | Low |
| 2 | 404034 | 2022-09-12 00:00:00.0000000 | 2022-09-13 00:00:00.0000000 | 77.84 | Herşey Dahil | Antalya | Tuesday | 1 | High |
| 3 | 415094 | 2022-12-03 00:00:00.0000000 | 2022-12-10 00:00:00.0000000 | 222.71 | Yarım Pansiyon | İzmir | Saturday | 7 | Low |
| 4 | 414951 | 2022-12-01 00:00:00.0000000 | 2022-12-03 00:00:00.0000000 | 140.48 | Yarım Pansiyon | İzmir | Saturday | 2 | Low |

* Şimdi devam edebiliriz. İkinci bir kontrol olarak da SaleId değişkeninde unique olmayan değerleri gözden geçiriyoruz

```
In [19]: tekrarlayan_count = df.duplicated(subset=["SaleId"]).sum()

# Sonuçlar:
print(tekrarlayan_count)
```

7303

* Veri setindeki tekrarlayan SaleID'leri incelediğimizde, tarihler, şehir, konsept birebir aynıken price değerlerinde farklı "gruplanmalar" gördük. Bu gruplanmalar birebir aynı gözükmüyorlar. Bu durumu veriyi sağlayan firma ile görüşüğümüzde tek bir kullanıcidan gelen ve farklı fiyatlandırımlar ile satış yapılan girişler olduğunu gördük.

örneğin:

18 yaş üstü : Yetişkin Fiyatı

12-18 yaş arası : Genç Fiyatı

12 yaş altı : Çocuk Fiyatı

gibi.

Bu gibi bir durum bizim veri setimiz için çok doğal olduğundan duplicate gözüken değerleri droplamanın sonuçlara ve toplam satışlara zarar vereceğini düşünüyoruz. Bu da aynı şekilde müşterimize tavsiye edeceğimiz bölgelere ve karlılık oranlarına kötü yansıyacaktır.

| | SaleId | SaleDate | CheckInDate | Price | ConceptName | SaleCityName | CInDay | SaleCheckInDayDiff | Seasons |
|---|--------|----------------------------|----------------------------|-------------|--------------|--------------|---------|--------------------|---------|
| 1 | 324498 | 2021-08-08 00:00:00.000000 | 2021-08-10 00:00:00.000000 | 59,5701459 | Her_ey Dahil | Antalya | Tuesday | 2 | High |
| 2 | 324498 | 2021-08-08 00:00:00.000000 | 2021-08-10 00:00:00.000000 | 41,37514029 | Her_ey Dahil | Antalya | Tuesday | 2 | High |
| 3 | 324498 | 2021-08-08 00:00:00.000000 | 2021-08-10 00:00:00.000000 | 41,37514029 | Her_ey Dahil | Antalya | Tuesday | 2 | High |
| 4 | 324498 | 2021-08-08 00:00:00.000000 | 2021-08-10 00:00:00.000000 | 30,64814815 | Her_ey Dahil | Antalya | Tuesday | 2 | High |
| 5 | 324498 | 2021-08-08 00:00:00.000000 | 2021-08-10 00:00:00.000000 | 30,64814815 | Her_ey Dahil | Antalya | Tuesday | 2 | High |
| 6 | 324498 | 2021-08-08 00:00:00.000000 | 2021-08-10 00:00:00.000000 | 59,5701459 | Her_ey Dahil | Antalya | Tuesday | 2 | High |

In [30]: df.index

Out[30]: RangeIndex(start=0, stop=59164, step=1)

```
In [32]: # Unique şehir sayısını ve frekanslarını hesaplayarak veri setimizdeki unique şehirleri öğrenip frekanslarını göreceğiz.
unique_cities = df["SaleCityName"].value_counts()

print("Unique Şehir Sayısı:", unique_cities.count())
print("\nŞehirlerin Frekansları:\n", unique_cities)
```

Unique Şehir Sayısı: 6

Şehirlerin Frekansları:

| | |
|--------------|-------|
| SaleCityName | |
| Antalya | 31649 |
| Muğla | 10662 |
| Aydın | 10646 |
| Diğer | 3245 |
| İzmir | 2507 |
| Girne | 455 |

Name: count, dtype: int64

Toplamda 6 adet unique şehir var. 5 farklı şehir diyebiliriz. 1 adet "Diğer" isimli değişkenimiz var.

```
In [35]: # Kaç unique Konsept olduğu ve Konsept başına kaçar adet satış olduğunu öğreniyoruz.
unique_concepts = df["ConceptName"].value_counts()

print("Unique konsept sayısı:", unique_concepts.count())
print("\nKonsept Başına Toplam Satış Sayıları:\n", unique_concepts)
```

Unique konsept sayısı: 3

Konsept Başına Toplam Satış Sayıları:

| | |
|----------------|-------|
| ConceptName | |
| Hersey Dahil | 53186 |
| Yarım Pansiyon | 3559 |
| Oda + Kahvaltı | 2419 |

Name: count, dtype: int64

In [37]: df["Price"] = df["Price"].astype(str)

```
In [39]: # Şehir başına ne kadar kar elde edildiğini öğreniyoruz.
# Price sütunu object olarak gözükmektedir. Bu da Count ile elde ettiğimiz satış adetlerini price ile çarpmamızı,
# yani toplam karı hesaplamamızı engelliyor. Bunun için onu int değere dönüştürüyoruz.
df["Price"] = df["Price"].str.replace(",",".").astype(float)
df["Price"] = df["Price"].round(2)

# Daha sonra işlemimize devam ediyoruz.
city_sales = df.groupby("SaleCityName").agg(
    Toplam_Satis=("SaleId", "count"),
    Toplam_Kar=("Price", "sum")
).reset_index()
# Toplam karları daha anlaşılır şekilde göstermek için virgülü göstermek istiyorum. Bu yüzden bi format değişikliği yapıyorum.
locale.setlocale(locale.LC_NUMERIC, 'en_US.UTF-8')
city_sales["Toplam_Kar"] = city_sales["Toplam_Kar"].apply(lambda x: locale.format_string("%.2f", x, grouping=True))

city_sales_df = pd.DataFrame(city_sales, columns = ["Toplam_Kar", "Toplam_Satis", "SaleCityName"])

# Bu değişikliği Toplam_Satis için yapmıyorum. Bunun nedeni fiyat bilgisinde anlamlı bir bilgi çıktısı veriyor. Ancak topla satış adet bazında
city_sales_df.sort_values(by=['Toplam_Satis'], ascending = False)
```

Out[39]: Toplam_Kar Toplam_Satis SaleCityName

| | | | |
|---|--------------|-------|---------|
| 0 | 2,042,044.49 | 31649 | Antalya |
| 4 | 665,966.03 | 10662 | Muğla |
| 1 | 573,351.89 | 10646 | Aydın |
| 2 | 154,808.06 | 3245 | Diğer |
| 5 | 166,129.41 | 2507 | İzmir |
| 3 | 27,065.21 | 455 | Girne |

```
In [41]: # Konseptlerin hangilerinden ne kadar satış elde edildiğini öğreniyoruz.
concept_sales = df.groupby("ConceptName").agg(
    Toplam_Satis_Adedi=("SaleId", "count"),
    Toplam_Kar=("Price", "sum")
).reset_index()
```

```
# Sütun isimlerini düzenliyoruz.
concept_sales = concept_sales.rename(columns={"ConceptName": "Konsept İsmi", "Toplam_Kar": "Toplam Kar"})

# Sonuçlar:
concept_sales.head()
```

Out[41]:

| | Konsept İsmi | Toplam_Satis_Adedi | Toplam Kar |
|---|----------------|--------------------|------------|
| 0 | Hersey Dahil | 53186 | 3333324.59 |
| 1 | Oda + Kahvaltı | 2419 | 121518.02 |
| 2 | Yarım Pansiyon | 3559 | 174522.48 |

In [43]: #Şehir başına olan ortalama kazançları sıralıyoruz.

city_avg_price = df.groupby("SaleCityName")["Price"].mean().reset_index()

Ortalama Price değerine göre büyükten küçüğe sıralama yapıyoruz. Böylece daha anlamlı bir çıktı alacağız.
city_avg_price = city_avg_price.sort_values(by="Price", ascending=False)

Sütun isimlerini düzenliyoruz.
city_avg_price = city_avg_price.rename(columns={"SaleCityName": "Şehir", "Price": "Ortalama Price"})

Sonuçlar:
city_avg_price.head()

Out[43]:

| | Şehir | Ortalama Price |
|---|---------|----------------|
| 5 | İzmir | 66.266219 |
| 0 | Antalya | 64.521612 |
| 4 | Muğla | 62.461642 |
| 3 | Girne | 59.483978 |
| 1 | Aydın | 53.856086 |

In [45]: #Konsept başına olan ortalama kazançları sıralıyoruz.

concept_avg_price = df.groupby("ConceptName")["Price"].mean().reset_index()

Price ortalamalarına göre büyükten küçüğe sıralayalım
concept_avg_price = concept_avg_price.sort_values(by="Price", ascending=False)

Sütun isimlerini değiştirelim
concept_avg_price = concept_avg_price.rename(columns={"ConceptName": "Konsept İsmi", "Price": "Ortalama Price"})

Sonuçları:
concept_avg_price.head()

Out[45]:

| | Konsept İsmi | Ortalama Price |
|---|----------------|----------------|
| 0 | Hersey Dahil | 62.672970 |
| 1 | Oda + Kahvaltı | 50.234816 |
| 2 | Yarım Pansiyon | 49.036943 |

In [47]: concept_sum_price = df.groupby("ConceptName")["Price"].sum().reset_index()
concept_sum_price = concept_sum_price.rename(columns={"ConceptName": "Konsept İsmi", "Price": "Toplam Kazanç"})

In [49]: # Şehir ve Konsept kırılımına göre Price ortalamalarını görmek istiyoruz. Bunun için bi groupby yöntemi uygulayacağız.

city_concept_avg_price = df.groupby(["SaleCityName", "ConceptName"])["Price"].mean().reset_index()

Price ortalamalarına göre büyükten küçüğe sıralayalım
city_concept_avg_price = city_concept_avg_price.sort_values(by="Price", ascending=False)

Sütun isimlerini değiştirelim
city_concept_avg_price = city_concept_avg_price.rename(columns={"SaleCityName": "Şehir", "ConceptName": "Konsept İsmi", "Price": "Ortalama Price"})

Sonuçlar:
city_concept_avg_price.head()

Out[49]:

| | Şehir | Konsept İsmi | Ortalama Price |
|----|---------|----------------|----------------|
| 9 | Girne | Hersey Dahil | 97.681132 |
| 6 | Diğer | Hersey Dahil | 84.771225 |
| 15 | İzmir | Hersey Dahil | 74.701751 |
| 2 | Antalya | Yarım Pansiyon | 67.191172 |
| 0 | Antalya | Hersey Dahil | 64.519163 |

* SaleCheckInDayDiff değişkenini kategorik bir değişkene çeviriyoruz.

In [52]: # Aralıkları ve bunlara karşılık gelen isimleri tanımlayalım
bins = [0, 7, 30, 90, df['SaleCheckInDayDiff'].max()]
labels = ["Last Minuters", "Potential Planners", "Planners", "Early Bookers"]

SaleCheckInDayDiff değişkenini bu aralıklara göre sınıflandırıyalım
df["Booking_Type"] = pd.cut(df["SaleCheckInDayDiff"], bins=bins, labels=labels, right=False)

```

# Her Booking_Type için ortalama Price değerlerini hesaplayalım
booking_type_avg_price = df.groupby("Booking_Type")["Price"].mean().reset_index()

# Sonuçlar:
print(booking_type_avg_price)

   Booking_Type      Price
0    Last Minuters  59.613394
1  Potential Planners  61.050886
2        Planners  64.605135
3   Early Bookers  63.898807

C:\Users\hp\AppData\Local\Temp\ipykernel_15864\3918050763.py:10: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  booking_type_avg_price = df.groupby("Booking_Type")["Price"].mean().reset_index()

In [54]: # Aldığımız çıktıya göre görüyoruz ki Planner ve Early Booker olarak sınıflandırdığımız müşteriler,
# daha çok getiri sağlıyorlar.

In [56]: # SaleCheckInDayDiff değişkenini bu aralıklara göre sınıflandıralım
df["EB_Score"] = pd.cut(df["SaleCheckInDayDiff"], bins=bins, labels=labels, right=False)

# Gerekli kırımlarda gruplama ve özetleme işlemleri
agg_df = df.groupby(["SaleCityName", "ConceptName", "EB_Score", "Seasons", "CInDay"]).agg(
    Avg_Price=("Price", "mean"),
    Transaction_Count=("SaleId", "count")
).reset_index()

# Sütun isimlerini daha anlamlı hale getirelim
agg_df = agg_df.rename(columns={
    "SaleCityName": "Şehir",
    "ConceptName": "Konsept",
    "EB_Score": "EB Skoru",
    "Seasons": "Sezon",
    "CInDay": "Giriş Günü",
    "Avg_Price": "Ortalama Fiyat",
    "Transaction_Count": "İşlem Sayısı"
})

# agg_df'i csv olarak kaydediyoruz. Türkçe karakter sıkıntısızı çekmemek için encoding ekliyoruz.
agg_df.to_csv("agg_df.csv", index=False, encoding='utf-8')

# Sonuçlar:
print(agg_df)

      Şehir      Konsept      EB Skoru Sezon Giriş Günü  Ortalama Fiyat \
0  Antalya  Hersey Dahil  Last Minuters  High     Friday      62.714887
1  Antalya  Hersey Dahil  Last Minuters  High    Monday      59.224316
2  Antalya  Hersey Dahil  Last Minuters  High  Saturday      62.879485
3  Antalya  Hersey Dahil  Last Minuters  High   Sunday      63.779343
4  Antalya  Hersey Dahil  Last Minuters  High Thursday      64.211553
...     ...     ...
1003  İzmir  Yarım Pansiyon Early Bookers  Low  Saturday      37.440000
1004  İzmir  Yarım Pansiyon Early Bookers  Low   Sunday       NaN
1005  İzmir  Yarım Pansiyon Early Bookers  Low Thursday       NaN
1006  İzmir  Yarım Pansiyon Early Bookers  Low Tuesday       NaN
1007  İzmir  Yarım Pansiyon Early Bookers  Low Wednesday       NaN

      İşlem Sayısı
0            1776
1            1770
2            1651
3            1264
4            1558
...           ...
1003           1
1004           0
1005           0
1006           0
1007           0

[1008 rows x 7 columns]

C:\Users\hp\AppData\Local\Temp\ipykernel_15864\595642491.py:5: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  agg_df = df.groupby(["SaleCityName", "ConceptName", "EB_Score", "Seasons", "CInDay"]).agg(

```

In [58]: agg_df.head(20)

Out[58]:

| | Şehir | Konsept | EB Skoru | Sezon | Giriş Günü | Ortalama Fiyat | İşlem Sayısı |
|----|---------|--------------|--------------------|-------|------------|----------------|--------------|
| 0 | Antalya | Hersey Dahil | Last Minuters | High | Friday | 62.714887 | 1776 |
| 1 | Antalya | Hersey Dahil | Last Minuters | High | Monday | 59.224316 | 1770 |
| 2 | Antalya | Hersey Dahil | Last Minuters | High | Saturday | 62.879485 | 1651 |
| 3 | Antalya | Hersey Dahil | Last Minuters | High | Sunday | 63.779343 | 1264 |
| 4 | Antalya | Hersey Dahil | Last Minuters | High | Thursday | 64.211553 | 1558 |
| 5 | Antalya | Hersey Dahil | Last Minuters | High | Tuesday | 64.934828 | 1392 |
| 6 | Antalya | Hersey Dahil | Last Minuters | High | Wednesday | 68.486118 | 1319 |
| 7 | Antalya | Hersey Dahil | Last Minuters | Low | Friday | 54.703259 | 494 |
| 8 | Antalya | Hersey Dahil | Last Minuters | Low | Monday | 64.953843 | 281 |
| 9 | Antalya | Hersey Dahil | Last Minuters | Low | Saturday | 54.195010 | 519 |
| 10 | Antalya | Hersey Dahil | Last Minuters | Low | Sunday | 53.665777 | 296 |
| 11 | Antalya | Hersey Dahil | Last Minuters | Low | Thursday | 57.789053 | 359 |
| 12 | Antalya | Hersey Dahil | Last Minuters | Low | Tuesday | 72.017973 | 296 |
| 13 | Antalya | Hersey Dahil | Last Minuters | Low | Wednesday | 67.463416 | 322 |
| 14 | Antalya | Hersey Dahil | Potential Planners | High | Friday | 64.827030 | 1091 |
| 15 | Antalya | Hersey Dahil | Potential Planners | High | Monday | 62.697974 | 2058 |
| 16 | Antalya | Hersey Dahil | Potential Planners | High | Saturday | 64.620982 | 1181 |
| 17 | Antalya | Hersey Dahil | Potential Planners | High | Sunday | 65.510760 | 1119 |
| 18 | Antalya | Hersey Dahil | Potential Planners | High | Thursday | 63.055759 | 1193 |
| 19 | Antalya | Hersey Dahil | Potential Planners | High | Tuesday | 65.805897 | 1048 |

In [88]: `df.groupby(["SaleCityName", "ConceptName"]).agg({"Price" : "mean"})`

Out[88]:

| Price | | |
|--------------|-----------------------|-----------|
| SaleCityName | ConceptName | |
| Antalya | Hersey Dahil | 64.519163 |
| | Oda + Kahvaltı | 63.504883 |
| | Yarım Pansiyon | 67.191172 |
| Aydın | Hersey Dahil | 53.995646 |
| | Oda + Kahvaltı | 34.458684 |
| | Yarım Pansiyon | 30.016452 |
| Düger | Hersey Dahil | 84.771225 |
| | Oda + Kahvaltı | 37.599146 |
| | Yarım Pansiyon | 42.113686 |
| Girne | Hersey Dahil | 97.681132 |
| | Oda + Kahvaltı | 39.776475 |
| | Yarım Pansiyon | 53.248000 |
| Muğla | Hersey Dahil | 63.020239 |
| | Oda + Kahvaltı | 59.037878 |
| | Yarım Pansiyon | 45.120160 |
| İzmir | Hersey Dahil | 74.701751 |
| | Oda + Kahvaltı | 41.320088 |
| | Yarım Pansiyon | 59.610491 |

In [90]: `#Yeni seviye tabanlı satışları tanımlayınız ve veri setine değişken olarak ekleyeceğim. bu seviye tabanlı katman "sales_Level_based" olacak # yani örnek olarak "Antalya_HerseyDahil_High" şeklinde yapacağız.`

```
agg_df["sales_level_based"] = agg_df.apply(
    lambda row: f'{row["Şehir"]}-{row["Konsept"]}-{row["Sezon"]}', axis=1
)

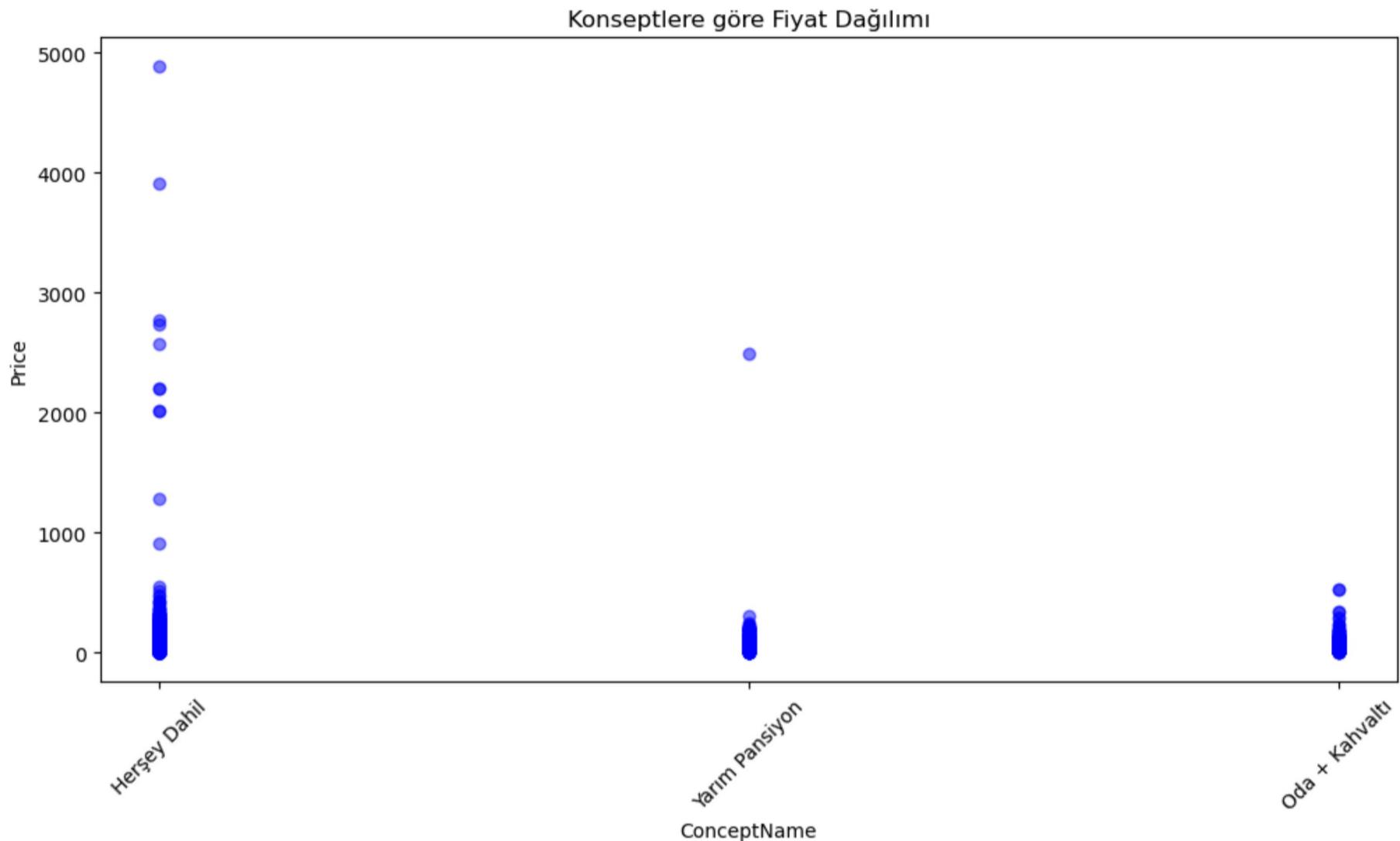
# agg_df'i kaydedelim
agg_df.to_csv("agg_df_with_sales_level_based.csv", index=False, encoding="utf-8")

# Sonuçlar:
agg_df.head()
```

| | Şehir | Konsept | EB Skoru | Sezon | Giriş Günü | Ortalama Fiyat | İşlem Sayısı | sales_level_based |
|---|---------|--------------|---------------|-------|------------|----------------|--------------|---------------------------|
| 0 | Antalya | Herşey Dahil | Last Minuters | High | Friday | 62.714887 | 1776 | Antalya-Herşey Dahil-High |
| 1 | Antalya | Herşey Dahil | Last Minuters | High | Monday | 59.224316 | 1770 | Antalya-Herşey Dahil-High |
| 2 | Antalya | Herşey Dahil | Last Minuters | High | Saturday | 62.879485 | 1651 | Antalya-Herşey Dahil-High |
| 3 | Antalya | Herşey Dahil | Last Minuters | High | Sunday | 63.779343 | 1264 | Antalya-Herşey Dahil-High |
| 4 | Antalya | Herşey Dahil | Last Minuters | High | Thursday | 64.211553 | 1558 | Antalya-Herşey Dahil-High |

In [64]: #Konseptler ve Price değişkeni arasındaki ilişkiyi görmek için scatterplot kullanıyoruz.

```
plt.figure(figsize=(12, 6))
plt.scatter(df['ConceptName'], df['Price'], color='blue', alpha=0.5)
plt.title("Konseptlere göre Fiyat Dağılımı")
plt.xlabel("ConceptName")
plt.ylabel("Price")
plt.xticks(rotation=45)
plt.show()
```

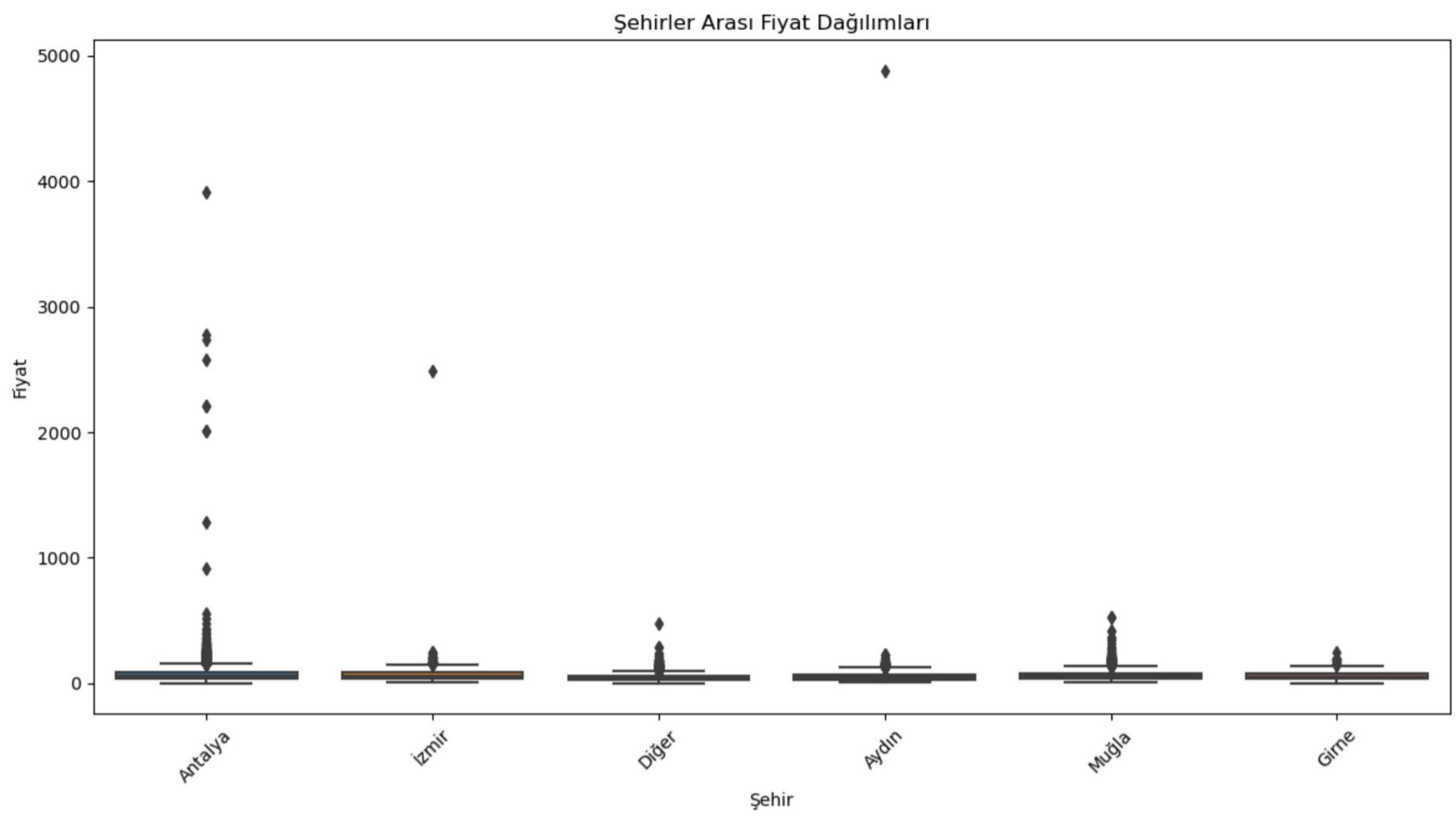


In []:

In [67]: # SaleCityName değişkenini Label encoder kullanarak sayısal değere dönüştürüyoruz.
label_encoder = LabelEncoder()
df["SaleCityName_encoded"] = label_encoder.fit_transform(df["SaleCityName"])

Şehir ve Price değişkeni arasındaki ilişkiyi görmek için scatterplot kullanıyoruz.

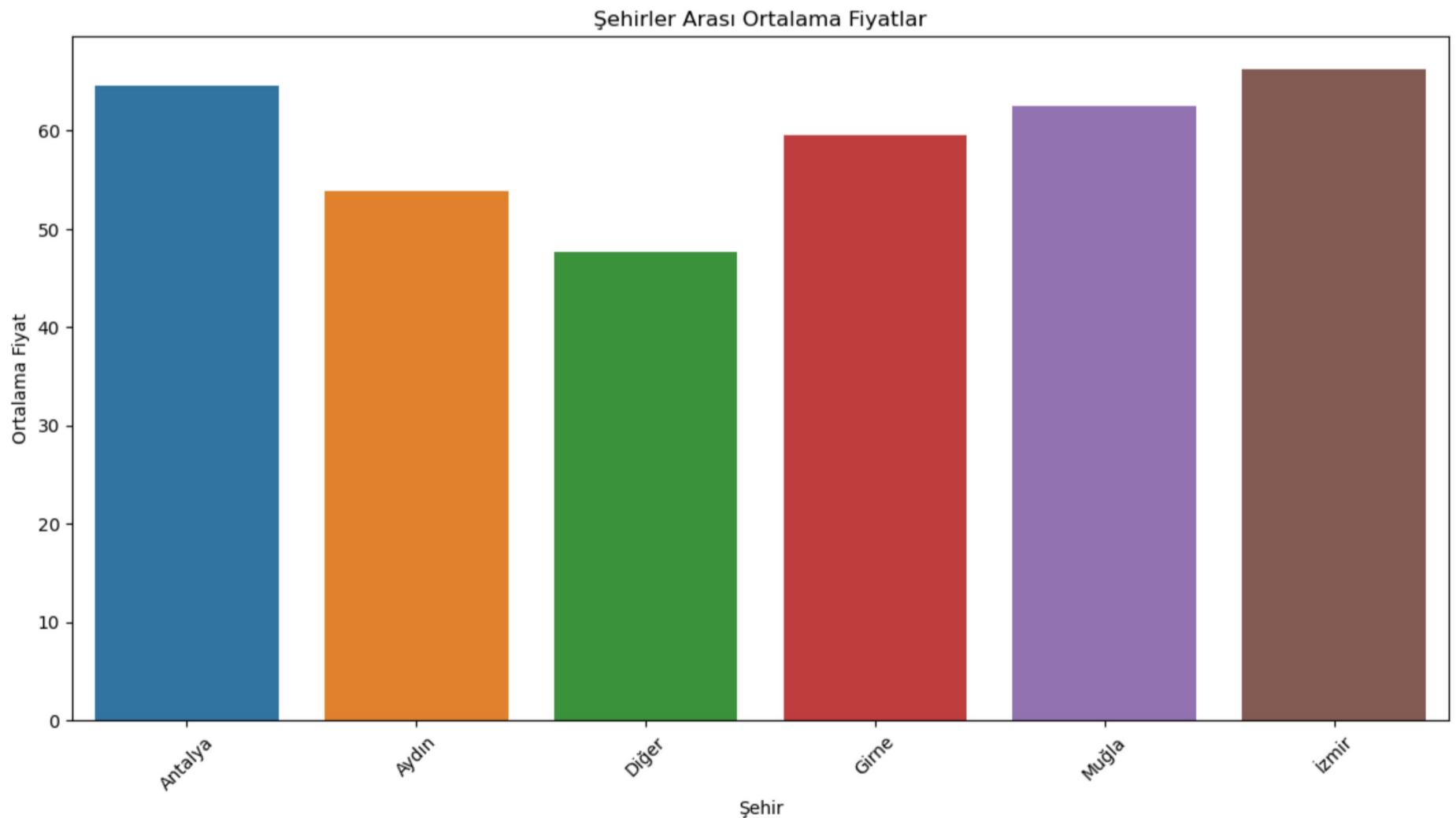
```
plt.figure(figsize=(14, 7))
sns.boxplot(x="SaleCityName", y="Price", data=df)
plt.title("Şehirler Arası Fiyat Dağılımları")
plt.xlabel("Şehir")
plt.ylabel("Fiyat")
plt.xticks(rotation=45)
plt.show()
```



* Konsept-Price ve Şehir-Price dağılımları birbirine benzer tablolar veriyorlar. Bunun temel nedeni Antalya'nın daha fazla Hersey Dahil otele sahip olması. Böylece daha fazla "yüksek" price'a sahip bir dağılım oluşuyor.

```
In [70]: # Şehirler arası fiyat ortalamalarını hesaplayalım
city_price_mean = df.groupby("SaleCityName")["Price"].mean().reset_index()

# Bar plot ile şehirler arası fiyat ortalamalarını görselleştirelim
plt.figure(figsize=(14, 7))
sns.barplot(x="SaleCityName", y="Price", data=city_price_mean)
plt.title("Şehirler Arası Ortalama Fiyatlar")
plt.xlabel("Şehir")
plt.ylabel("Ortalama Fiyat")
plt.xticks(rotation=45)
plt.show()
```

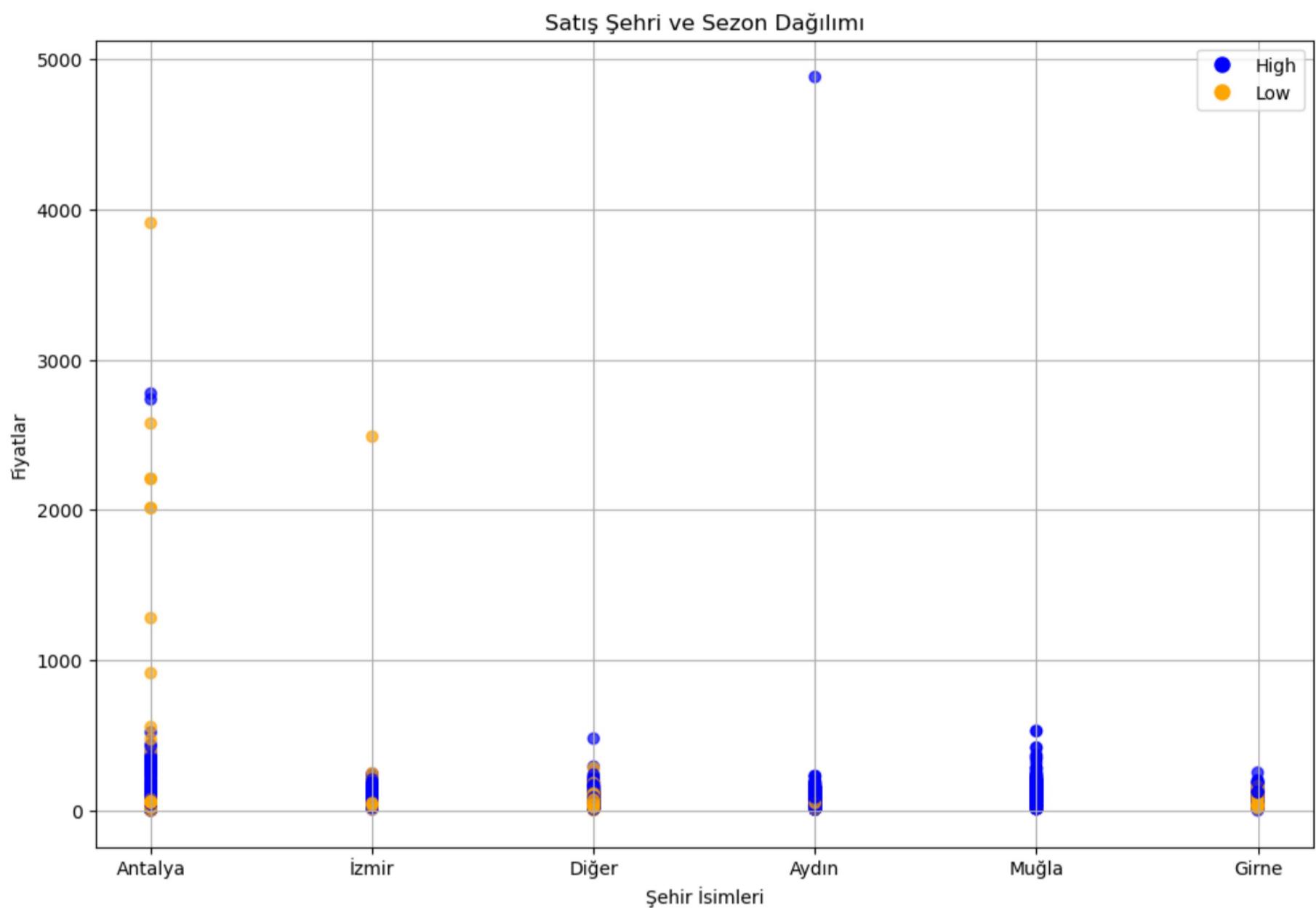


* Şehirler Arası Fiyat Ortalamalarına baktığımızda daha önceki ortalama değer tablomuzda gördüğümüz gibi; İzmir'in ortalamada en pahalı, daha sonra Antalya, daha sonra da Muğla şehirlerini görüyoruz.

* Ancak burada gördüğümüz fiyatlar şehirlerdeki satışlarla orantılı olmadığı için tam olarak doğru bir izlenim vermiyor.

Satışlarla doğru orantılı bir değerlendirmede hangi şehrin daha karlı olduğunu önceki çıktıımızda görmüştük.

```
In [74]: # tablomuz
plt.figure(figsize=(12, 8))
colors = {"High": "blue", "Low": "orange"}
plt.scatter(df["SaleCityName"], df["Price"], c=df["Seasons"].apply(lambda x: colors[x]), alpha=0.7)
plt.title("Satış Şehri ve Sezon Dağılımı")
plt.xlabel("Şehir İsimleri")
plt.ylabel("Fiyatlar")
plt.legend(handles=[plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=v, markersize=10, label=k) for k, v in colors.items()])
plt.grid(True)
plt.show()
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```