**PYTHON RAPORU**

**Kod Açıklamaları**:

print(data.head())

print(data.describe())

print(data.head()): Bu komut, veri setinin ilk 5 satırını yazdırır. Veri setinin başlangıcındaki örnek verileri görmek için kullanılır.

print(data.describe()): Bu komut, sayısal sütunlar için temel istatistiksel verileri (ortalama, standart sapma, min, max, çeyrekler vb.) hesaplar ve yazdırır. Veri setinin genel dağılımı ve özellikleri hakkında bilgi verir.

# Tarih ve yıl sütunlarını düzenleme

data['Date'] = pd.to\_datetime(data['Date'])

data['Year'] = data['Date'].dt.year

Date sütunundaki veriyi pd.to\_datetime() fonksiyonu ile tarih formatına dönüştürür. Ardından, data['Date'].dt.year ifadesi ile her tarih için yılı çıkarır ve bunu Year sütununa ekler. Böylece, veri setinde her olayın yılına ulaşılabilir.

**1.**

# Felaket türlerinin dağılımı

disaster\_type\_counts = data["Disaster\_Type"].value\_counts()

disaster\_type\_counts.plot(kind="pie", autopct="%1.1f%%", title="Felaket Türlerinin Dağılımı")

plt.show()

Bir veri setindeki felaket türlerinin dağılımını analiz eder ve pasta grafiği olarak görselleştirir. İlk adımda, **Disaster\_Type** sütunundaki türlerin sıklığı hesaplanır. Daha sonra, bu veriler pasta grafiğiyle görselleştirilir, her dilim üzerine yüzdelik oranları yazılır ve grafik ekrana gösterilir.

**2.**

# Aylara göre toplam ekonomik kayıplar

data["Month"] = pd.to\_datetime(data["Date"]).dt.month

annual\_loss = data.groupby("Month")["Economic\_Loss($)"].sum()

plt.figure(figsize=(10, 6))

annual\_loss.plot(kind="line", marker="o", color="darkorange")

plt.title("Aylara Göre Ekonomik Kayıplar", fontsize=14)

plt.xlabel("Ay", fontsize=12)

plt.ylabel("Ekonomik Kayıp ($)", fontsize=12)

plt.grid(alpha=0.5)

plt.show()

Verilerdeki tarihlerden ay bilgilerini çıkarır ve her ay için toplam ekonomik kaybı hesaplar. Ardından, bu veriler bir çizgi grafiğiyle görselleştirilir. Grafikte aylar x ekseninde, ekonomik kayıplar ($) ise y ekseninde yer alır ve veri noktaları daha belirgin olması için işaretlenmiştir. Sonuç olarak, aylara göre ekonomik kayıpların değişimi net bir şekilde gösterilir.

**3.**

# Aylara göre felaket sayısı

monthly\_disasters = data.groupby("Month").size()

plt.figure(figsize=(10, 6))

monthly\_disasters.plot(kind="bar", color="teal", alpha=0.7)

plt.title("Aylara Göre Felaket Sayısı", fontsize=14)

plt.xlabel("Ay", fontsize=12)

plt.ylabel("Felaket Sayısı", fontsize=12)

plt.xticks(ticks=range(12), labels=["Ocak", "Şubat", "Mart", "Nisan", "Mayıs", "Haziran", "Temmuz", "Ağustos", "Eylül", "Ekim", "Kasım", "Aralık"], rotation=45)

plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.show()

Bu kod, her ay meydana gelen toplam felaket sayısını hesaplar ve bunu çubuk grafiğiyle görselleştirir. Grafikte x ekseni ayları (Ocak’tan Aralık’a), y ekseni ise felaket sayılarını gösterir. Çubuklar daha estetik görünmesi için renk ve saydamlık özellikleriyle düzenlenmiştir. Bu grafik, felaketlerin hangi aylarda yoğunlaştığını kolayca anlamayı sağlar.

**4.**

# 2024 Yılındaki Aylara Göre Ortalama Can Kaybı

# Tarih sütunundan ay bilgisini çıkarın

data['Month'] = pd.to\_datetime(data['Date']).dt.month\_name()

# Aylara göre ortalama can kaybını hesaplayın

fatalities\_by\_month = data.groupby('Month')['Fatalities'].mean().reset\_index()

# Ayları sıralamak için bir sözlük kullanın

month\_order = {

'January': 1, 'February': 2, 'March': 3, 'April': 4,

'May': 5, 'June': 6, 'July': 7, 'August': 8,

'September': 9, 'October': 10, 'November': 11, 'December': 12

}

fatalities\_by\_month['Month\_Num'] = fatalities\_by\_month['Month'].map(month\_order)

fatalities\_by\_month = fatalities\_by\_month.sort\_values('Month\_Num')

# Çubuk grafiğini oluşturun

plt.figure(figsize=(12, 6)) # Genişliği artırın

sns.barplot(x='Month', y='Fatalities', data=fatalities\_by\_month, palette="viridis")

plt.title("2024 Yılında Aylara Göre Ortalama Can Kaybı")

plt.xlabel("Ay")

plt.ylabel("Ortalama Can Kaybı")

plt.xticks(rotation=45, ha='right') # X eksenindeki etiketleri döndürün

plt.tight\_layout() # Etiketlerin kesilmesini önleyin

plt.show()

2024 yılındaki verilerdeki aylık ortalama can kayıplarını hesaplar ve çubuk grafiğiyle gösterir. İlk olarak, **Date** sütunundaki tarih bilgisi ay ismine dönüştürülür. Ardından, her ay için **Fatalities** sütununun ortalaması alınır. Aylar doğru sırada yer alması için bir sıralama yapılır ve ay numaralarına göre düzenlenir. Son olarak, **seaborn** ile ortalama can kayıplarının görselleştirildiği bir çubuk grafiği oluşturulur.

**5.**

# Bölgelere göre ortalama ekonomik kayıp

avg\_loss\_by\_region = data.groupby("Location")["Economic\_Loss($)"].mean().nlargest(10)

plt.figure(figsize=(12, 6))

avg\_loss\_by\_region.plot(kind="barh", color="seagreen", alpha=0.8, edgecolor="black")

plt.title("Bölgelere Göre Ortalama Ekonomik Kayıp", fontsize=16)

plt.xlabel("Ortalama Ekonomik Kayıp ($)", fontsize=14)

plt.ylabel("Bölge", fontsize=14)

plt.grid(axis="x", linestyle="--", alpha=0.7)

plt.tight\_layout()

plt.show()

Verilerdeki her bölge için ortalama ekonomik kaybı hesaplar ve en yüksek 10 bölgeyi yatay çubuk grafiğiyle gösterir. **groupby("Location")** ile bölgelere göre gruplama yapılır, ardından **mean()** ile her bölge için ortalama ekonomik kayıp hesaplanır. **nlargest(10)** ile en yüksek 10 bölge seçilir. Çubuk grafiği, **seagreen** renk paletiyle oluşturulur ve **tight\_layout()** ile etiketler düzenlenir.

**6.**

# Bölgeye göre ekonomik kayıplar

region\_loss = data.groupby("Location")["Economic\_Loss($)"].sum().nlargest(10)

plt.figure(figsize=(12, 6))

region\_loss.plot(kind="barh", color="dodgerblue", alpha=0.8)

plt.title("En Çok Ekonomik Kayıp Yaşanan 10 Bölge", fontsize=14)

plt.xlabel("Toplam Ekonomik Kayıp ($)", fontsize=12)

plt.ylabel("Bölge", fontsize=12)

plt.grid(axis="x", linestyle="--", alpha=0.7)

plt.show()

Her bölge için toplam ekonomik kaybı hesaplar ve en yüksek 10 bölgeyi yatay çubuk grafiğiyle gösterir. **groupby("Location")** ile veriler bölgelere göre gruplanır, ardından **sum()** ile her bölge için toplam ekonomik kayıp hesaplanır. **nlargest(10)** ile en yüksek 10 bölge seçilir. Grafikte **dodgerblue** rengi kullanılır ve yatay eksende toplam ekonomik kayıplar, dikey eksende ise bölgeler gösterilir.

**7.**

earthquake\_df = data[data['Disaster\_Type'] == 'Earthquake']

# Büyüklük aralıklarını oluşturun ve ekonomik kayıpları gruplayın

bins = [0, 2, 4, 6, 8, 10] # Büyüklük aralıkları

labels = ['0-2', '2-4', '4-6', '6-8', '8-10'] # Aralık etiketleri

earthquake\_df['Magnitude\_Range'] = pd.cut(earthquake\_df['Magnitude'], bins=bins, labels=labels)

grouped\_df = earthquake\_df.groupby('Magnitude\_Range')['Economic\_Loss($)'].sum().reset\_index()

# Çubuk grafiğini oluşturun

plt.figure(figsize=(10, 6))

sns.barplot(

x='Magnitude\_Range',

y='Economic\_Loss($)',

data=grouped\_df,

palette="viridis"

)

# Başlık ve eksen etiketlerini ekleyin

title = "Deprem Büyüklüğü Aralıklarına Göre Toplam Ekonomik Kayıp"

plt.title(title, fontsize=14)

plt.xlabel("Büyüklük Aralığı", fontsize=12)

plt.ylabel("Toplam Ekonomik Kayıp ($)", fontsize=12)

# Düzeni ayarlayın ve grafiği gösterin

plt.tight\_layout()

plt.show()

Deprem verilerini alır ve farklı büyüklük aralıklarına göre toplam ekonomik kaybı hesaplar. İlk olarak, Magnitude sütunundaki değerler belirli aralıklara göre pd.cut() fonksiyonu ile kategorilere ayrılır. Bu büyüklük aralıkları bins ile belirlenir ve her bir aralık için etiketler atanır. Ardından, her büyüklük aralığı için Economic\_Loss($) sütununun toplamı groupby() ve sum() ile hesaplanır. Çubuk grafiği seaborn kütüphanesi ile oluşturulur ve her bir büyüklük aralığının toplam ekonomik kaybı gösterilir. Grafikte renk paleti viridis kullanılır.

**8.**

# Yıllara göre ölümlerin dağılımı

yearly\_fatalities = data.groupby('Year')['Fatalities'].sum()

plt.figure(figsize=(10, 6))

sns.lineplot(x=yearly\_fatalities.index, y=yearly\_fatalities.values, marker="o", color="g")

plt.title("Yıllara Göre Ölümlerin Dağılımı")

plt.xlabel("Yıl")

plt.ylabel("Toplam Ölümler")

plt.grid(True)

plt.show()

Yıllara göre toplam ölümleri hesaplar ve bir çizgi grafiğiyle gösterir. groupby('Year') ile veriler yıllara göre gruplanır ve her yıl için Fatalities (ölümler) sütununun toplamı hesaplanır. sns.lineplot() fonksiyonu ile yıllara göre ölümlerin dağılımı çizilir. Grafikte yıllar x ekseninde, toplam ölümler ise y ekseninde yer alır. Çizgi üzerindeki noktalar marker="o" parametresiyle işaretlenir ve grid(True) ile grid çizgileri eklenir.

**9.**

En çok ölüm yaşanan ilk 5 lokasyon

top\_5\_locations = query13.head(5)

plt.figure(figsize=(10, 6))

sns.barplot(x=top\_5\_locations.index, y=top\_5\_locations.values, palette="coolwarm")

plt.title("En Çok Ölüm Yaşanan İlk 5 Lokasyon")

plt.xlabel("Lokasyon")

plt.ylabel("Toplam Ölümler")

plt.xticks(rotation=45)

plt.show()

En çok ölüm yaşanan ilk 5 lokasyonu ve bunlardaki toplam ölümleri bir çubuk grafiğiyle gösterir. İlk olarak, query13.head(5) ile en yüksek ölüm sayısına sahip ilk 5 lokasyon seçilir. Ardından, sns.barplot() fonksiyonu ile bu lokasyonlar ve ölüm sayıları görselleştirilir. coolwarm renk paleti kullanılır, x eksenindeki lokasyon isimleri döndürülerek (rotation=45) okunabilirlik artırılır. Grafikte lokasyonlar x ekseninde, toplam ölümler ise y ekseninde yer alır.

**10.**

Ortalama büyüklük ve afet türü ilişkisi

plt.figure(figsize=(10, 6))

sns.barplot(x=query3.index, y=query3.values, palette="cool")

plt.title("Afet Türlerine Göre Ortalama Büyüklük")

plt.xlabel("Afet Türü")

plt.ylabel("Ortalama Büyüklük")

plt.xticks(rotation=45)

plt.show()

Afet türlerine göre ortalama büyüklükleri gösteren bir çubuk grafiği oluşturur. query3.index ve query3.values ile afet türleri ve bu türlere ait ortalama büyüklükler alınır. sns.barplot() fonksiyonu ile bu veriler görselleştirilir ve cool renk paleti kullanılır. X eksenindeki afet türlerinin isimleri döndürülerek (rotation=45) okunabilirlik artırılır. Grafikte afet türleri x ekseninde, ortalama büyüklükler ise y ekseninde yer alır.

**11.**

# 1. Ortalama Ekonomik Kayıp (Afet Türüne Göre)

plt.subplot(3, 2, 1)

avg\_economic\_loss\_by\_disaster.plot(kind='bar', color='skyblue')

plt.title("Ortalama Ekonomik Kayıp (Afet Türüne Göre)")

plt.xlabel("Afet Türü")

plt.ylabel("Ortalama Ekonomik Kayıp ($)")

Afet türlerine göre ortalama ekonomik kayıpları gösteren bir çubuk grafiği oluşturur. avg\_economic\_loss\_by\_disaster adlı veri seti, her afet türü için hesaplanan ortalama ekonomik kayıpları içerir. plt.subplot(3, 2, 1) komutu, grafiğin bir alt grafik olarak (3 satır, 2 sütunluk bir düzen içinde) yerleştirilmesini sağlar. Çubuklar skyblue rengiyle gösterilir ve başlık, x ve y eksen etiketleri eklenir. Bu şekilde, her afet türü için ortalama ekonomik kayıplar kolayca karşılaştırılabilir.

**12.**

# 2. Her Ülke için Toplam Afet Sayısı

plt.subplot(3, 2, 2)

disasters\_per\_country.head(10).plot(kind='bar', color='orange')

plt.title("Her Ülke için Toplam Afet Sayısı")

plt.xlabel("Ülke")

plt.ylabel("Afet Sayısı")

Her ülke için toplam afet sayısını gösteren bir çubuk grafiği oluşturur. disasters\_per\_country adlı veri çerçevesi, her ülke için hesaplanan toplam afet sayısını içerir. plt.subplot(3, 2, 2) ile bu grafiğin bir alt grafik olarak (3 satır, 2 sütunluk bir düzen içinde) yerleştirilmesi sağlanır. Çubuklar orange rengiyle gösterilir ve başlık, x ve y eksen etiketleri eklenir. Böylece, en fazla afet yaşanan ilk 10 ülke kolayca görülebilir.

**13.**

# 4. En Çok Afet Görülen 5 Ülke için Ortalama Ölüm Sayısı

plt.subplot(3, 2, 4)

avg\_fatalities\_top\_countries.plot(kind='bar', color='red')

plt.title("En Çok Afet Görülen 5 Ülke için Ortalama Ölüm Sayısı")

plt.xlabel("Ülke")

plt.ylabel("Ortalama Ölüm Sayısı")

En çok afet görülen 5 ülke için ortalama ölüm sayısını gösteren bir çubuk grafiği oluşturur. avg\_fatalities\_top\_countries adlı veri seti, her ülke için hesaplanan ortalama ölüm sayısını içerir. plt.subplot(3, 2, 4) komutu, grafiğin 3 satır, 2 sütunluk bir düzen içinde dördüncü alt grafik olarak yerleştirilmesini sağlar. Çubuklar red rengiyle gösterilir ve başlık, x ve y eksen etiketleri eklenir. Bu şekilde, en çok afet yaşayan ülkelerdeki ortalama ölüm sayıları karşılaştırılabilir.

**14.**

# 5. Afet Türlerine Göre En Yüksek Şiddetli 3 Afet

plt.subplot(3, 2, 5)

sns.barplot(data=highest\_magnitude\_by\_disaster, x='Magnitude', y='Disaster\_Type', hue='Location')

plt.title("Afet Türlerine Göre En Yüksek Şiddetli 3 Afet")

plt.xlabel("Şiddet")

plt.ylabel("Afet Türü")

plt.tight\_layout()

plt.show()

Afet türlerine göre en yüksek şiddetli üç afeti görselleştirir. highest\_magnitude\_by\_disaster veri seti, her afet türü için en yüksek şiddeti ve bu şiddetin yaşandığı yerleri içerir. sns.barplot() fonksiyonu, her afet türü için en yüksek şiddetli afetleri Magnitude (şiddet) ve Disaster\_Type (afet türü) eksenlerine göre çubuk grafiğiyle gösterir. hue='Location' parametresi, her afetin gerçekleştiği farklı lokasyonları farklı renklerle gösterir. plt.tight\_layout() komutu ise düzenin sıkışık olmasını engeller. Bu grafik, en yüksek şiddetli afetlerin türlerini ve bulundukları bölgeleri karşılaştırır.

**15.**

# 5. En fazla afete sahip 5 ülkenin afet türü dağılımı

plt.subplot(3, 2, 5)

top\_5\_countries = data['Location'].value\_counts().nlargest(5).index

top\_countries\_disasters = data[data['Location'].isin(top\_5\_countries)]

sns.countplot(data=top\_countries\_disasters, x='Location', hue='Disaster\_Type', palette='muted')

plt.title("En Fazla Afete Sahip 5 Ülkenin Afet Türü Dağılımı")

plt.xlabel("Ülke")

plt.ylabel("Afet Sayısı")

En fazla afete sahip 5 ülkenin afet türü dağılımını görselleştirir. İlk olarak, **data['Location'].value\_counts().nlargest(5).index** ile en fazla afete sahip 5 ülke belirlenir. Ardından, bu ülkelerdeki afet türlerinin dağılımı **sns.countplot()** fonksiyonu ile çubuk grafiği olarak gösterilir. **hue='Disaster\_Type'** parametresi, her ülke için farklı afet türlerini renklerle ayırır. **palette='muted'** rengi kullanılır. Grafikte x ekseninde ülkeler, y ekseninde ise afet sayıları yer alır. Bu grafik, en fazla afete sahip 5 ülkenin afet türlerini karşılaştırır.

**16.**

# Belirli bir ülkede en yüksek ekonomik kayıp

location = "Japan" # Örnek ülke

max\_loss\_event = data[data["Location"] == location].nlargest(1, "Economic\_Loss($)")

Belirli bir ülkedeki (örneğin "Japan") en yüksek ekonomik kaybı yaşayan afet olayını bulur. İlk olarak, **data[data["Location"] == location]** ile sadece "Japan" ülkesine ait veriler filtrelenir. Ardından, **nlargest(1, "Economic\_Loss($)")** fonksiyonu, "Economic\_Loss($)" sütunundaki en yüksek kaybı içeren olayı seçer. Sonuç **print(max\_loss\_event)** ile yazdırılır ve en yüksek ekonomik kayba sahip afetin detayları gösterilir.

**17.**

#Bu kod en çok can kaybına sebep olan 10 depremi bulur

earthquake\_df = earthquake\_df.rename(columns={

'Disaster\_Type': 'Afet\_Türü',

'Magnitude': 'Büyüklük',

'Date': 'Tarih',

'Location': 'Yer',

'Fatalities': 'Ölümler',

'Economic\_Loss($)': 'Ekonomik\_Kayıp($)'

})

# En çok can kaybına neden olan 10 deprem olayını bulun

most\_fatal\_earthquakes = earthquake\_df.nlargest(10, 'Ölümler')[['Büyüklük', 'Afet\_Türü', 'Tarih', 'Yer', 'Ölümler']]

# Sonuçları yazdır

print("En çok can kaybına neden olan 10 deprem olayı:")

print(most\_fatal\_earthquakes)

#Bu kod en çok can kaybına sebep olan 10 depremi bulur

earthquake\_df = earthquake\_df.rename(columns={

'Disaster\_Type': 'Afet\_Türü',

'Magnitude': 'Büyüklük',

'Date': 'Tarih',

'Location': 'Yer',

'Fatalities': 'Ölümler',

'Economic\_Loss($)': 'Ekonomik\_Kayıp($)'

})

# En çok can kaybına neden olan 10 deprem olayını bulun

most\_fatal\_earthquakes = earthquake\_df.nlargest(10, 'Ölümler')[['Büyüklük', 'Afet\_Türü', 'Tarih', 'Yer', 'Ölümler']]

# Sonuçları yazdır

print("En çok can kaybına neden olan 10 deprem olayı:")

print(most\_fatal\_earthquakes)

En çok can kaybına sebep olan 10 depremi bulur ve bunları yazdırır. İlk olarak, earthquake\_df.rename() fonksiyonu ile veri çerçevesindeki sütun isimleri Türkçe'ye çevrilir. Ardından, earthquake\_df.nlargest(10, 'Ölümler') komutu ile "Ölümler" (Fatalities) sütununa göre en fazla can kaybı yaşanan ilk 10 deprem olayı seçilir. [['Büyüklük', 'Afet\_Türü', 'Tarih', 'Yer', 'Ölümler']] ile sadece önemli sütunlar (Büyüklük, Afet Türü, Tarih, Yer, Ölümler) alınır. Son olarak, print() ile en çok can kaybına neden olan 10 deprem olayı ekrana yazdırılır.

**18.**

# En büyük 5 afetin gerçekleştiği yeri ve bilgilerini bulun

most\_fatal\_disasters = data.nlargest(5, 'Fatalities')[['Disaster\_Type', 'Location', 'Date', 'Fatalities', 'Magnitude']]

# Başlıkları Türkçe olarak değiştirilir

most\_fatal\_disasters = most\_fatal\_disasters.rename(columns={

'Disaster\_Type': 'Afet Türü',

'Location': 'Yer',

'Date': 'Tarih',

'Fatalities': 'Ölü Sayısı',

'Magnitude': 'Büyüklük'

})

# Sonuçları yazdırın

print("En büyük 5 afetin gerçekleştiği yer ve bilgileri:")

print(most\_fatal\_disasters)

En büyük 5 afetin gerçekleştiği yer ve ilgili bilgileri bulur ve yazdırır. İlk olarak, data.nlargest(5, 'Fatalities') komutu, "Fatalities" (Ölümler) sütununa göre en fazla can kaybı yaşanan 5 afet olayını seçer. Seçilen olaylar arasında sadece [['Disaster\_Type', 'Location', 'Date', 'Fatalities', 'Magnitude']] sütunları alınır. Daha sonra, rename() fonksiyonu ile sütun isimleri Türkçe'ye çevrilir. Son olarak, print() komutu ile en büyük 5 afetin bilgileri ekrana yazdırılır.

**19.**

# Büyüklüğü en fazla olan 10 olayı bulur

largest\_magnitude\_events = data.nlargest(10, 'Magnitude')[['Disaster\_Type', 'Location', 'Date', 'Fatalities', 'Magnitude']]

# Başlıkları Türkçe olarak değiştirilir

largest\_magnitude\_events = largest\_magnitude\_events.rename(columns={

'Disaster\_Type': 'Afet Türü',

'Location': 'Yer',

'Date': 'Tarih',

'Fatalities': 'Ölü Sayısı',

'Magnitude': 'Büyüklük'

})

# Sonuçları yazdırın

print("Büyüklüğü en fazla olan 10 olay:")

print(largest\_magnitude\_events)

Büyüklüğü en fazla olan 10 afet olayını bulur ve ilgili bilgileri yazdırır. İlk olarak, data.nlargest(10, 'Magnitude') komutu ile "Magnitude" (Büyüklük) sütununa göre en büyük büyüklüğe sahip ilk 10 afet olayı seçilir. Seçilen olaylar arasında [['Disaster\_Type', 'Location', 'Date', 'Fatalities', 'Magnitude']] sütunları alınır. Daha sonra, rename() fonksiyonu ile sütun isimleri Türkçe'ye çevrilir. Son olarak, print() komutu ile büyüklüğü en fazla olan 10 olayın bilgileri ekrana yazdırılır.

**20.**

# Depremleri filtreleyi

earthquake\_data = data[data['Disaster\_Type'] == 'Earthquake']

# Büyüklüğü en fazla olan 10 depremi bulun

largest\_magnitude\_earthquakes = earthquake\_data.nlargest(10, 'Magnitude')[['Disaster\_Type', 'Location', 'Date', 'Fatalities', 'Magnitude']]

# Başlıkları Türkçe olarak değiştirin

largest\_magnitude\_earthquakes = largest\_magnitude\_earthquakes.rename(columns={

'Disaster\_Type': 'Afet Türü',

'Location': 'Yer',

'Date': 'Tarih',

'Fatalities': 'Ölü Sayısı',

'Magnitude': 'Büyüklük'

})

# Sonuçları yazdırın

print("Büyüklüğü en fazla olan 10 deprem:")

print(largest\_magnitude\_earthquakes)

Depremleri filtreler ve büyüklüğü en fazla olan 10 depremi bulur. İlk olarak, data[data['Disaster\_Type'] == 'Earthquake'] ile yalnızca "Earthquake" (Deprem) afet türüne ait veriler seçilir ve earthquake\_data adında yeni bir veri çerçevesi oluşturulur. Ardından, nlargest(10, 'Magnitude') komutu ile "Magnitude" (Büyüklük) sütununa göre en büyük büyüklüğe sahip ilk 10 deprem olayı seçilir. Seçilen olaylar arasında sadece [['Disaster\_Type', 'Location', 'Date', 'Fatalities', 'Magnitude']] sütunları alınır. Daha sonra, rename() fonksiyonu ile sütun isimleri Türkçe'ye çevrilir. Son olarak, print() komutu ile büyüklüğü en fazla olan 10 deprem olayının bilgileri ekrana yazdırılır.

**21.**

# Afet türlerinin sayısını hesaplayın

disaster\_counts = data['Disaster\_Type'].value\_counts()

# En sık karşılaşılan 5 afet türünü seçin

top\_5\_disasters = disaster\_counts.head(5)

# Başlıkları Türkçe olarak değiştirin

top\_5\_disasters = top\_5\_disasters.rename\_axis('Afet Türü').reset\_index(name='Sayı')

# Sonuçları yazdırın

print("En sık karşılaşılan 5 afet türü:")

print(top\_5\_disasters)

Afet türlerinin sayısını hesaplar ve en sık karşılaşılan 5 afet türünü bulur. İlk olarak, data['Disaster\_Type'].value\_counts() komutu ile "Disaster\_Type" (Afet Türü) sütunundaki her bir afet türünün kaç kez tekrar ettiğini hesaplar. Bu sayede her afet türünün sıklığı elde edilir ve disaster\_counts adında bir seri oluşturulur. Daha sonra, head(5) komutu ile en sık karşılaşılan ilk 5 afet türü seçilir. Seçilen afet türleri, rename\_axis('Afet Türü').reset\_index(name='Sayı') fonksiyonu ile sütun isimleri Türkçe'ye çevrilir ve daha düzgün bir görünüm elde edilir. Son olarak, print() komutu ile en sık karşılaşılan 5 afet türü ve bunların sayıları ekrana yazdırılır.

**22.**

# Şehirlerdeki afet sayısını hesaplayın

city\_counts = data['Location'].value\_counts()

# En çok afet görülen 10 şehri seçin

top\_10\_cities = city\_counts.head(10)

# Başlıkları Türkçe olarak değiştirin

top\_10\_cities = top\_10\_cities.rename\_axis('Şehir').reset\_index(name='Afet Sayısı')

# Sonuçları yazdırın

print("En fazla doğal afet görülen 10 şehir:")

print(top\_10\_cities)

Şehirlerdeki afet sayısını hesaplar ve en fazla doğal afet görülen 10 şehri bulur. value\_counts() ile her şehrin kaç kez afet yaşadığı hesaplanır. head(10) ile en çok afet görülen 10 şehir seçilir. Sütun isimleri Türkçeye çevrilir ve sonuçlar ekrana yazdırılır.

**23.**

# Her afet türü için en sık görülen yeri bulun

most\_frequent\_locations = data.groupby('Disaster\_Type')['Location'].agg(lambda x: x.value\_counts().index[0])

# Başlıkları Türkçe olarak değiştirin

most\_frequent\_locations = most\_frequent\_locations.rename\_axis('Afet Türü').reset\_index(name='En Sık Görülen Yer')

# Sonuçları yazdırın

print("Afet türlerine göre en sık görülen yerler:")

print(most\_frequent\_locations)

Her afet türü için en sık görülen yeri bulur. groupby('Disaster\_Type') ile afet türlerine göre gruplama yapılır, ardından agg(lambda x: x.value\_counts().index[0]) kullanılarak her grup için en sık görülen yer hesaplanır. reset\_index() ile veri çerçevesi yeniden düzenlenir ve sütun isimleri Türkçeye çevrilir. Son olarak, print() komutu ile her afet türüne göre en sık görülen yerler ekrana yazdırılır.

**24.**

# En çok mali zarara uğratan 20 olayı bulun

top\_20\_costliest\_events = data.nlargest(20, 'Economic\_Loss($)')

# İlgili sütunları seçin ve başlıkları Türkçe olarak değiştirin

top\_20\_costliest\_events = top\_20\_costliest\_events[['Disaster\_Type', 'Location', 'Magnitude', 'Economic\_Loss($)']].rename(columns={

    'Disaster\_Type': 'Afet Türü',

    'Location': 'Yer',

    'Magnitude': 'Büyüklük',

    'Economic\_Loss($)': 'Ekonomik Zarar ($)'

})

# Sonuçları yazdırın

print("En çok mali zarara uğratan 20 olay:")

print(top\_20\_costliest\_events)

İlk olarak, data.nlargest(20, 'Economic\_Loss($)') ifadesi, "Economic\_Loss($)" (Ekonomik Kayıp) sütununa göre en büyük 20 değeri olan satırları seçer. Bu, ekonomik kaybı en yüksek 20 afet olayını getirir.

Sonra, .loc[] veya sütun seçimi kullanılarak sadece "Disaster\_Type" (Afet Türü), "Location" (Yer), "Magnitude" (Büyüklük) ve "Economic\_Loss($)" (Ekonomik Zarar) sütunları alınır. Bu sütunlar, olaylarla ilgili önemli bilgileri içerir.

Daha sonra, .rename(columns={...}) fonksiyonu kullanılarak sütun başlıkları Türkçeye çevrilir:"Disaster\_Type" → "Afet Türü" …

**25.**

disaster\_summary = data.groupby('Disaster\_Type')[['Fatalities', 'Economic\_Loss($)']].sum().reset\_index()

disaster\_summary = disaster\_summary.rename(columns={

    'Disaster\_Type': 'Afet Türü',

    'Fatalities': 'Toplam Ölümler',

    'Economic\_Loss($)': 'Toplam Ekonomik Kayıp ($)'

})

print(disaster\_summary)

her afet türü için toplam ölümler ve toplam ekonomik kayıpları hesaplar ve bunları bir özet halinde sunar.

İlk olarak, data.groupby('Disaster\_Type')[['Fatalities', 'Economic\_Loss($)']].sum() ifadesi ile veriler "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır. Bu gruplar için "Fatalities" (Ölüm Sayısı) ve "Economic\_Loss($)" (Ekonomik Kayıp) sütunlarının toplamları (sum()) hesaplanır. Bu işlem, her afet türü için toplam ölüm sayısı ve toplam ekonomik kayıpları verir.

Sonra, .reset\_index() fonksiyonu ile indeks sıfırlanır, böylece daha düzenli bir veri çerçevesi elde edilir.

Ardından, disaster\_summary.rename(columns={...}) ile sütun adları değiştirilir. Bu, daha anlamlı ve okunabilir hale getirilmesi için "Disaster\_Type" → "Afet Türü", "Fatalities" → "Toplam Ölümler" ve "Economic\_Loss($)" → "Toplam Ekonomik Kayıp ($)" şeklinde isimlendirilir. Son olarak, print() komutu ile her afet türü için toplam ölümler ve toplam ekonomik kayıpların yer aldığı özet tablo ekrana yazdırılır.

**26.**

top\_10\_affected\_countries = data['Location'].value\_counts().head(10).reset\_index()

top\_10\_affected\_countries.columns = ['Ülke', 'Afet Sayısı']

print(top\_10\_affected\_countries)

en fazla afet yaşanan 10 ülkeyi belirler ve bunları listeler.

İlk olarak, data['Location'].value\_counts() ifadesi, her bir lokasyon (ülke veya bölge) için kaç afet meydana geldiğini sayar. Sonra, .head(10) fonksiyonu ile en fazla afet yaşanan ilk 10 ülke seçilir.

reset\_index() fonksiyonu, sıralanan veri çerçevesinin indeksini sıfırlar ve yeni bir indeks ekler. Bu sayede, ülke ve afet sayısı bilgilerini daha düzenli bir formatta alırız.

Son olarak, top\_10\_affected\_countries.columns = ['Ülke', 'Afet Sayısı'] ile sütun adları daha anlaşılır hale getirilir: "Ülke" ve "Afet Sayısı".

Sonuçta, print() komutu ile en fazla afet yaşanan 10 ülke ve bunların afet sayıları ekrana yazdırılır.

**27.**

# En yıkıcı 10 olay

most\_damaging\_events = data.nlargest(10, "Economic\_Loss($)")[["Disaster\_Type", "Location", "Magnitude", "Economic\_Loss($)"]]

print(most\_damaging\_events)

en büyük ekonomik kayba yol açan 10 afet olayını belirler ve bunları listeler.

İlk olarak, data.nlargest(10, "Economic\_Loss($)") ifadesi, "Economic\_Loss($)" (Ekonomik Kayıp) sütununa göre en yüksek 10 değeri olan satırları seçer. Bu, ekonomik kayıpları en büyük olan ilk 10 afet olayını getirir.

Sonrasında, [["Disaster\_Type", "Location", "Magnitude", "Economic\_Loss($)"]] ile sadece "Disaster\_Type" (Afet Türü), "Location" (Lokasyon), "Magnitude" (Şiddet) ve "Economic\_Loss($)" (Ekonomik Kayıp) sütunları seçilir. Bu sütunlar, en büyük ekonomik kayıplara sahip afetlerle ilgili gerekli bilgileri içerir.

Son olarak, print() komutu ile en fazla ekonomik kayba yol açan 10 afetin türü, lokasyonu, şiddeti ve ekonomik kaybı ekrana yazdırılır.

**28.**

 #2024 yılında gerçekleşen toplam ölümler

query5 = data[data['Year'] == 2024]['Fatalities'].sum()

print("2024 Yılında Gerçekleşen Toplam Ölümler:", query5)

Bu Python kodu, 2024 yılında gerçekleşen toplam ölümleri hesaplar. İlk olarak, data[data['Year'] == 2024] ifadesi ile sadece 2024 yılına ait veriler filtrelenir. Yani, yalnızca 2024 yılında meydana gelen afetler seçilir.

Daha sonra, ['Fatalities'] ile bu verilerin "Fatalities" (Ölüm Sayısı) sütununa erişilir ve .sum() fonksiyonu ile bu sütundaki değerlerin toplamı hesaplanır. Bu, 2024 yılı boyunca yaşanan tüm afetlerdeki toplam ölüm sayısını verir.

Son olarak, print() komutu ile 2024 yılındaki toplam ölümler ekrana yazdırılır.

**29.**

# En fazla ekonomik kayıp oluşturan afet türü

query4 = data.groupby('Disaster\_Type')['Economic\_Loss($)'].sum().idxmax()

print("En Fazla Ekonomik Kayıp Oluşturan Afet Türü:", query4)

en fazla ekonomik kayba yol açan afet türünü belirler. İlk olarak, veri "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır ve her grup için "Economic\_Loss($)" (Ekonomik Kayıp) sütunundaki toplam değer (sum()) hesaplanır.

Daha sonra, idxmax() fonksiyonu kullanılarak toplam ekonomik kaybı en yüksek olan afet türü seçilir. idxmax() fonksiyonu, en büyük toplam kayba sahip olan grubun indeksini döndürür, yani en fazla ekonomik kayba yol açan afet türünü.

Son olarak, print() komutu ile en fazla ekonomik kayba yol açan afet türü ekrana yazdırılır.

**30.**

# En ölümcül afet türü

query6 = data.groupby('Disaster\_Type')['Fatalities'].sum().idxmax()

print("En Ölümcül Afet Türü:", query6)

Bu Python kodu, en ölümcül afet türünü belirler. İlk olarak, veri "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır ve her grup için "Fatalities" (Ölüm Sayısı) sütunundaki toplam değer (sum()) hesaplanır.

Daha sonra, idxmax() fonksiyonu kullanılarak toplam ölüm sayısı en yüksek olan afet türü seçilir. idxmax() fonksiyonu, en büyük toplam ölüme sahip olan grubun indeksini döndürür, yani en ölümcül afet türünü.

Son olarak, print() komutu ile en ölümcül afet türü ekrana yazdırılır.

**31.**

# En düşük büyüklükteki afetin türü  ve yılı nerde gereçekleştigi yapcam daha bunu

query7 = data.loc[data['Magnitude'].idxmin(), 'Disaster\_Type']

print("En Düşük Büyüklükteki Afetin Türü:", query7)

Bu kod en düşük büyüklüğe sahip afet türünü belirler. İlk olarak, data['Magnitude'].idxmin() ifadesi, "Magnitude" (Şiddet) sütunundaki en küçük değeri taşıyan satırın indeksini bulur. idxmin() fonksiyonu, en küçük değere sahip olan satırın indeksini döndürür.

Daha sonra, data.loc[...] ile bu indeks kullanılarak ilgili satır seçilir. Seçilen satırdaki "Disaster\_Type" (Afet Türü) sütunundaki değer alınır. Bu, en düşük büyüklüğe sahip afetin türünü belirler.

Son olarak, print() komutu ile en düşük büyüklükteki afetin türü ekrana yazdırılır.

**32.**

# Belirli bir lokasyonda (örneğin 'Brazil') meydana gelen afet sayısı

query8 = data[data['Location'] == 'Brazil'].shape[0]

print("Brezilya'da Meydana Gelen Afet Sayısı:", query8)

Bu kod, Brezilya'da meydana gelen afet sayısını belirler. İlk olarak, data[data['Location'] == 'Brazil'] ifadesi ile veri çerçevesinde "Location" (Lokasyon) sütununda "Brazil" (Brezilya) olan satırlar filtrelenir. Yani, sadece Brezilya'ya ait veriler seçilir.

Sonrasında, .shape[0] ifadesi, bu filtrelenmiş verinin satır sayısını döndürür, yani Brezilya'da meydana gelen afetlerin sayısını verir.

Son olarak, print() komutu ile Brezilya'daki toplam afet sayısı ekrana yazdırılır.

**33.**

# 2024 yılında en çok ekonomik kayıp oluşturan afet türü

query12 = data[data['Year'] == 2024].groupby('Disaster\_Type')['Economic\_Loss($)'].sum().idxmax()

print("2024 Yılında En Çok Ekonomik Kayıp Oluşturan Afet Türü:", query12)

2024 yılında en çok ekonomik kayba yol açan afet türünü belirler. İlk olarak, data[data['Year'] == 2024] ifadesi ile yalnızca 2024 yılına ait veriler filtrelenir. Ardından, bu veriler "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır ve her grup için "Economic\_Loss($)" (Ekonomik Kayıp) sütunundaki toplam değer (sum()) hesaplanır.

Son olarak, idxmax() fonksiyonu kullanılarak en yüksek toplam ekonomik kayba sahip afet türü seçilir. idxmax() fonksiyonu, en büyük değeri taşıyan grup (yani en yüksek ekonomik kayba sahip afet türü) için indeks değerini döndürür.

print() komutu ile 2024 yılında en fazla ekonomik kayba yol açan afet türü ekrana yazdırılır.

**34.**

# 13. Sorgu: Lokasyonlara göre toplam ölümler

query13 = data.groupby('Location')['Fatalities'].sum().sort\_values(ascending=False)

print("Lokasyonlara Göre Toplam Ölümler:\n", query13.head(10))

her lokasyon (ülke veya bölge) için toplam ölüm sayısını hesaplar ve en fazla ölüme sahip ilk 10 lokasyonu sıralar. İlk olarak, veri "Location" (Lokasyon) sütununa göre gruplandırılır ve her grup için "Fatalities" (Ölüm Sayısı) sütunundaki toplam değer (sum()) hesaplanır.

Daha sonra, .sort\_values(ascending=False) fonksiyonu ile toplam ölüm sayıları azalan sırayla sıralanır. Bu, en fazla ölüme sahip lokasyonların üstte yer almasını sağlar.

Son olarak, query13.head(10) ile en fazla ölüm sayısına sahip ilk 10 lokasyon seçilir ve print() komutu ile bu lokasyonların toplam ölüm sayıları ekrana yazdırılır.

**35.**

# 15. Sorgu: Ortalama büyüklüğü en yüksek olan afet türü

query15 = data.groupby('Disaster\_Type')['Magnitude'].mean().idxmax()

print("Ortalama Büyüklüğü En Yüksek Olan Afet Türü:", query15)

afet türleri arasından ortalama büyüklüğü en yüksek olanı belirler. İlk olarak, veri "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır ve her grup için "Magnitude" (Şiddet) sütununun ortalaması (mean()) hesaplanır.

Sonrasında, idxmax() fonksiyonu kullanılarak ortalama büyüklüğü en yüksek olan afet türü seçilir. idxmax() fonksiyonu, en büyük ortalama değere sahip olan grubun indeksini döndürür, yani en büyük ortalama büyüklüğe sahip afet türünü.

Son olarak, print() komutu ile ortalama büyüklüğü en yüksek olan afet türü ekrana yazdırılır.

**36.**

#En çok ölüm yaşanan 5 ülkenin toplam ekonomik kaybı

query17 = data.groupby('Location')[['Fatalities', 'Economic\_Loss($)']].sum().nlargest(5, 'Fatalities')

print("En Çok Ölüm Yaşanan 5 Ülkenin Toplam Ekonomik Kaybı:\n", query17)

Bu kod, en çok ölüm yaşanan 5 ülkenin toplam ekonomik kayıplarını hesaplar. İlk olarak, data.groupby('Location')[['Fatalities', 'Economic\_Loss($)']].sum() ile ülkelere göre gruplama yapılır ve her grup için "Fatalities" (Ölüm Sayısı) ve "Economic\_Loss($)" (Ekonomik Kayıp) sütunlarındaki toplam değerler hesaplanır.

Daha sonra, .nlargest(5, 'Fatalities') fonksiyonu ile toplam ölüm sayısı en yüksek olan ilk 5 ülke seçilir. Bu, en fazla ölüm yaşanan 5 ülkenin toplam verilerini alır.

Son olarak, print() komutu ile bu 5 ülkenin toplam ölüm sayıları ve ekonomik kayıpları ekrana yazdırılır.

**37.**

# 18. Sorgu: Her yıl en fazla büyüklüğe sahip afetin türü

query18 = data.loc[data.groupby('Year')['Magnitude'].idxmax()][['Year', 'Disaster\_Type', 'Magnitude']]

print("Her Yıl En Fazla Büyüklüğe Sahip Afetin Türü:\n", query18)

Bu kod, her yılın en büyük şiddete sahip afet türünü belirler. İlk olarak, data.groupby('Year')['Magnitude'].idxmax() ile her yıl için "Magnitude" (Şiddet) sütunundaki en yüksek değeri taşıyan satırların indeksleri bulunur. idxmax() fonksiyonu, her yılın en büyük şiddete sahip afetinin indeksini döndürür.

Daha sonra, data.loc[...] ile bu indeksler kullanılarak ilgili satırlar seçilir. Bu satırlar, her yılın en büyük şiddetli afetine ait "Year", "Disaster\_Type" (Afet Türü) ve "Magnitude" (Şiddet) sütunlarını içerir.

Son olarak, print() komutu ile her yılın en büyük şiddetli afet türü ve şiddet değerleri ekrana yazdırılır.

**38.**

#Ekonomik kaybı 500 milyon dolardan fazla olan afetlerin sayısı

query19 = data[data['Economic\_Loss($)'] > 1e4].shape[0]

print("Ekonomik Kaybı 500 Milyon Dolardan Fazla Olan Afetlerin Sayısı:", query19)

Bu kod, ekonomik kaybı 10.000 dolardan (1e4) fazla olan afetlerin sayısını belirler. İlk olarak, data[data['Economic\_Loss($)'] > 1e4] ifadesi ile "Economic\_Loss($)" (Ekonomik Kayıp) sütunundaki değeri 10.000'den büyük olan afetler filtrelenir. Burada 1e4, 10.000 sayısının bilimsel gösterimidir.

Sonrasında, .shape[0] ifadesi, bu filtrelenmiş verinin satır sayısını (yani, 10.000 dolardan büyük ekonomik kayba sahip afetlerin sayısını) döndürür.

Son olarak, print() komutu ile bu afetlerin sayısı ekrana yazdırılır.

**40.**

# Ortalama ekonomik kaybı en düşük olan afet türü

query20 = data.groupby('Disaster\_Type')['Economic\_Loss($)'].mean().idxmin()

print("Ortalama Ekonomik Kaybı En Düşük Olan Afet Türü:", query20)

afet türleri arasından ortalama ekonomik kaybı en düşük olanı belirler. İlk olarak, veri "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır. Ardından, her grup için "Economic\_Loss($)" (Ekonomik Kayıp) sütununun ortalaması (mean()) hesaplanır.

Sonrasında, idxmin() fonksiyonu kullanılarak ortalama ekonomik kaybı en düşük olan afet türü seçilir. idxmin() fonksiyonu, en küçük değere sahip olan grubun indeksini döndürür, yani en düşük ortalama kayba sahip afet türünü.

Son olarak, print() komutu ile ortalama ekonomik kaybı en düşük olan afet türü ekrana yazdırılır.

**41.**

#2025 yılında en ölümcül afet türü

query24 = data[data['Year'] == 2025].groupby('Disaster\_Type')['Fatalities'].sum().idxmax()

print("2025 Yılında En Ölümcül Afet Türü:", query24)

Bu kod, 2025 yılında en ölümcül afet türünü belirler. İlk olarak, data[data['Year'] == 2025] ile 2025 yılına ait veriler filtrelenir. Ardından, groupby('Disaster\_Type') ile bu veriler afet türüne göre gruplanır ve her grup için "Fatalities" (Ölüm Sayısı) sütunundaki toplam değer (sum()) hesaplanır.

Sonrasında, idxmax() fonksiyonu kullanılarak toplam ölüm sayısı en yüksek olan afet türü seçilir. idxmax(), en büyük değeri taşıyan satırın indeksini döndürür, yani en ölümcül afet türünü.

Son olarak, print() komutu ile 2025 yılında en ölümcül afet türü ekrana yazdırılır.

**42.**

# 1. Her afet türü için ortalama ekonomik kayıp

avg\_economic\_loss\_by\_disaster = data.groupby('Disaster\_Type')['Economic\_Loss($)'].mean().sort\_values(ascending=False)

print("Ortalama Ekonomik Kayıp (Afet Türüne Göre):")

print(avg\_economic\_loss\_by\_disaster)

Bu kod, her afet türü için ortalama ekonomik kaybı hesaplar. İlk olarak, veri "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır. Ardından, her grup için "Economic\_Loss($)" (Ekonomik Kayıp) sütununun ortalaması (mean()) hesaplanır. Son olarak, .sort\_values(ascending=False) fonksiyonu ile ortalama ekonomik kayıplar azalan sırayla sıralanır.

Sonuç olarak, print() komutları ile her afet türüne göre ortalama ekonomik kayıplar ekrana yazdırılır.

**43.**

# 2. Her ülke için toplam afet sayısı

disasters\_per\_country = data['Location'].value\_counts()

print("Her Ülke için Toplam Afet Sayısı:")

print(disasters\_per\_country)

her ülke için toplam afet sayısını hesaplar. data['Location'].value\_counts() fonksiyonu, "Location" (Ülke) sütunundaki her bir ülkenin kaç kez geçtiğini sayar, yani her ülkedeki afet sayısını belirler. Sonuç olarak, her ülke için toplam afet sayıları bir seri (Series) olarak elde edilir.

Son olarak, print() komutu ile her ülke için toplam afet sayısı ekrana yazdırılır.

**44.**

# 4. En çok afet görülen 5 ülke için ölüm sayısı ortalaması

top\_5\_countries = data['Location'].value\_counts().head(5).index

avg\_fatalities\_top\_countries = data[data['Location'].isin(top\_5\_countries)].groupby('Location')['Fatalities'].mean()

print("En Çok Afet Görülen 5 Ülke için Ortalama Ölüm Sayısı:")

print(avg\_fatalities\_top\_countries)

en çok afet görülen 5 ülkenin ortalama ölüm sayısını hesaplar. İlk olarak, data['Location'].value\_counts().head(5).index ile veri çerçevesindeki "Location" (Ülke) sütunundaki en fazla afet görülen 5 ülke belirlenir. value\_counts() fonksiyonu, her ülke için afet sayısını hesaplar ve head(5) ile en yüksek 5 değeri seçer.

Sonra, data[data['Location'].isin(top\_5\_countries)] ile bu 5 ülkeye ait veriler filtrelenir. groupby('Location') ile ülkelere göre gruplama yapılır ve ['Fatalities'].mean() ile her grup için "Fatalities" (Ölüm Sayısı) sütununun ortalaması hesaplanır.

Son olarak, print() komutları ile en çok afet görülen 5 ülke için ortalama ölüm sayıları ekrana yazdırılır.

**45.**

# 5. Afet türlerine göre en yüksek şiddetli 3 afet

highest\_magnitude\_by\_disaster = data.groupby('Disaster\_Type').apply(lambda x: x.nlargest(3, 'Magnitude')).reset\_index(drop=True)

print("Afet Türlerine Göre En Yüksek Şiddetli 3 Afet:")

print(highest\_magnitude\_by\_disaster[['Disaster\_Type', 'Location', 'Magnitude']])

Bu kod, her afet türü için en yüksek şiddete sahip üç afetin verilerini çıkarır. İlk olarak, veri "Disaster\_Type" (Afet Türü) sütununa göre gruplandırılır. Ardından, her grup için nlargest(3, 'Magnitude') fonksiyonu ile "Magnitude" (Şiddet) sütununa göre en büyük üç değeri seçeriz. apply() fonksiyonu, bu işlemi her grup için uygular. Sonrasında reset\_index(drop=True) ile sıralanan veri çerçevesinin indeksleri yeniden düzenlenir.

Son olarak, print() komutları ile afet türü, ülke ve şiddet sütunlarından oluşan ilk üç en büyük şiddete sahip afetler ekrana yazdırılır.

**46.**

# 5. En çok afet türüne sahip ilk 5 ülke

most\_common\_disaster\_by\_country = data.groupby('Location')['Disaster\_Type'].agg(pd.Series.mode)

print("En Çok Afet Türüne Sahip İlk 5 Ülke:")

print(most\_common\_disaster\_by\_country.head(5))

Bu kod, her ülke için en yaygın afet türünü bulur. İlk olarak, veri "Location" (Ülke) sütununa göre gruplandırılır. Ardından, her grup için "Disaster\_Type" (Afet Türü) sütununda en sık görülen değer (pd.Series.mode) belirlenir. agg() fonksiyonu, bu işlemi her grup için uygular. Son olarak, head(5) ile en fazla görülen afet türlerine sahip ilk 5 ülke ekrana yazdırılır.

**Kütüphaneler ve Kullanım Amacı**

**1. pandas (pd):**  
pandas Python programlama dilinde veri analizi ve veri işleme için kullanılan çok güçlü bir kütüphanedir. pandas, veriyi kolayca işleyebilmenizi ve analiz etmenizi sağlar. Veri çerçeveleri (DataFrame) ve seriler (Series) gibi veri yapıları sunar. Verileri filtrelemek, gruplamak, sıralamak, birleştirmek gibi birçok işlemi hızlı ve verimli bir şekilde yapmanıza olanak tanır.

Kullanım amacına örnekler:

* groupby() fonksiyonu ile verilerin gruplanması ve özetlenmesi.
* sum(), mean(), max() gibi fonksiyonlar ile verilerin analizi.
* Veri manipülasyonu için indeksleme ve filtreleme işlemleri.
* Veriyi içeri aktarmak ve dışarı aktarmak (CSV, Excel gibi).

**Kullanım Amacı:**

* Veriyi organize etme, analiz etme ve işleme.
* Veri kümesindeki ilişkileri keşfetme ve özetleme.
* Veriyi görselleştirmek için uygun hale getirme.

**2. matplotlib.pyplot (plt):**  
matplotlib, Python'da veri görselleştirme için kullanılan bir kütüphanedir. matplotlib.pyplot, özellikle grafikler ve görseller oluşturmak için kullanılan bir modüldür. Çizgi grafikleri, çubuk grafikleri, histogramlar, daire grafikleri gibi çok çeşitli grafikler oluşturulabilir.

**Kullanım amacına örnekler:**

* plt.plot(), plt.bar(), plt.hist() gibi fonksiyonlarla grafikler oluşturmak.
* Grafiklerin başlıklarını, eksenlerini, etiketlerini ve renklerini özelleştirmek.
* Verilerin daha anlaşılır hale getirilmesini sağlamak.

**Kullanım Amacı:**

* Veri analizi sonuçlarını görsel olarak sunmak.
* Verilerin eğilimlerini ve dağılımlarını daha iyi anlamak.
* Grafikler aracılığıyla sonuçların görsel olarak yorumlanmasını sağlamak.

**3. seaborn (sns):**  
seaborn, matplotlib üzerine inşa edilmiş ve istatistiksel veri görselleştirmeleri için geliştirilmiş bir kütüphanedir. seaborn, veri kümelerinin görselleştirilmesini kolaylaştırır ve estetik olarak hoş grafikler üretir. Özellikle karmaşık veri kümelerinin analizini kolaylaştıran fonksiyonlar sunar.

Kullanım amacına örnekler:

* sns.heatmap(), sns.boxplot(), sns.pairplot() gibi fonksiyonlarla gelişmiş görselleştirmeler oluşturmak.
* Verilerin daha anlamlı ve anlaşılır şekilde görselleştirilmesini sağlamak.
* İstatistiksel ilişki ve dağılım analizi yapmak.

**Kullanım Amacı:**

* İstatistiksel verilerin görselleştirilmesi.
* Veriler arasındaki ilişkilerin görsel olarak analiz edilmesi.
* Veri görselleştirmelerini daha estetik ve anlaşılır hale getirme.

**VeriTabanı İncelenmesi:**

**Veri Seti İçeriği:**

* Bu veri seti doğal afetler ile ilgili bilgileri içermektedir. Her satır bir doğal afet olayını temsil eder.

**Sütunlar ve Türleri:**

* **Disaster\_ID (int)**: Her bir doğal afetin benzersiz bir kimliği.
* **Disaster\_Type (object)**: Doğal afet türü (örneğin, Wildfire, Hurricane, Tornado, vb.).
* **Location (object)**: Doğal afetin meydana geldiği yer.
* **Magnitude (float)**: Doğal afetin şiddet veya büyüklük derecesi.
* **Date (object)**: Doğal afetin gerçekleştiği tarih.
* **Fatalities (int)**: Doğal afette ölen kişi sayısı.
* **Economic\_Loss($) (float)**: Doğal afetin neden olduğu ekonomik kayıp (ABD Doları cinsinden).

**Genel Bilgiler:**

* **Toplam veri sayısı**: 10,000 satır.