Prepared by Asif Bhat

Data Visualization With Seaborn

In [674]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
```

Loading Datasets

In [675]:

```
pokemon = pd.read_csv("pokemon_updated.csv")
pokemon.head(10)
```

Out[675]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legen
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	I
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	I
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	I
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	I
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	I
5	5	Charmeleon	Fire	NaN	58	64	58	80	65	80	1	I
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	1	I
7	6	CharizardMega Charizard X	Fire	Dragon	78	130	111	130	85	100	1	I
8	6	CharizardMega Charizard Y	Fire	Flying	78	104	78	159	115	100	1	I
9	7	Squirtle	Water	NaN	44	48	65	50	64	43	1	I

In [676]:

```
stdperf = pd.read_csv("studentp.csv")
stdperf.head(10)
```

Out[676]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
5	female	group B	associate's degree	standard	none	71	83	78
6	female	group B	some college	standard	completed	88	95	92
7	male	group B	some college	free/reduced	none	40	43	39
8	male	group D	high school	free/reduced	completed	64	64	67
9	female	group B	high school	free/reduced	none	38	60	50

In [677]:

```
corona = pd.read_csv('C:/Users/DELL/Documents/GitHub/Public/COVID-19/covid/data/countries-a
                     index_col='Date' , parse_dates=True)
corona.head(10)
```

Out[677]:

Date	Country	Confirmed	Recovered	Deaths
2020-01-22	Afghanistan	0	0	0
2020-01-22	Albania	0	0	0
2020-01-22	Algeria	0	0	0
2020-01-22	Andorra	0	0	0
2020-01-22	Angola	0	0	0
2020-01-22	Antigua and Barbuda	0	0	0
2020-01-22	Argentina	0	0	0
2020-01-22	Armenia	0	0	0
2020-01-22	Australia	0	0	0
2020-01-22	Austria	0	0	0

In [678]:

```
spotify = pd.read_csv("spotify.csv" , index_col="Date")
spotify.head(10)
```

Out[678]:

Date	Shape of You	Despacito	Something Just Like This	HUMBLE.	Unforgettable
2017-01-06	12287078	NaN	NaN	NaN	NaN
2017-01-07	13190270	NaN	NaN	NaN	NaN
2017-01-08	13099919	NaN	NaN	NaN	NaN
2017-01-09	14506351	NaN	NaN	NaN	NaN
2017-01-10	14275628	NaN	NaN	NaN	NaN
2017-01-11	14372699	NaN	NaN	NaN	NaN
2017-01-12	14148108	NaN	NaN	NaN	NaN
2017-01-13	14536236	275178.0	NaN	NaN	NaN
2017-01-14	14173311	1144886.0	NaN	NaN	NaN
2017-01-15	12889849	1288198.0	NaN	NaN	NaN

In [679]:

```
housing = pd.read_csv('C:/Users/DELL/Documents/GitHub/Data-Visualization/housing.csv')
housing.tail()
```

Out[679]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	house
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

In [680]:

```
insurance = pd.read_csv('C:/Users/DELL/Documents/GitHub/Data-Visualization/insurance.csv')
insurance.head(10)
```

Out[680]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692

In [681]:

```
employment = pd.read_excel("unemployment.xlsx")
employment.head(10)
```

Out[681]:

	Age	Gender	Period	Unemployed
0	16 to 19 years	Men	2005-01-01	91000
1	20 to 24 years	Men	2005-01-01	175000
2	25 to 34 years	Men	2005-01-01	194000
3	35 to 44 years	Men	2005-01-01	201000
4	45 to 54 years	Men	2005-01-01	207000
5	55 to 64 years	Men	2005-01-01	101000
6	65 years and over	Men	2005-01-01	33000
7	16 to 19 years	Women	2005-01-01	38000
8	20 to 24 years	Women	2005-01-01	90000
9	25 to 34 years	Women	2005-01-01	142000

In [683]:

```
helpdesk = pd.read_csv("helpdesk.csv")
helpdesk.head(10)
```

Out[683]:

	ticket	requestor	RequestorSeniority	ITOwner	FiledAgainst	TicketType	Severity	Priority
0	1	1929	1 - Junior	50	Systems	Issue	2 - Normal	0 Unassigned
1	2	1587	2 - Regular	15	Software	Request	1 - Minor	1 - Low
2	3	925	2 - Regular	15	Access/Login	Request	2 - Normal	0 Unassigned
3	4	413	4 - Management	22	Systems	Request	2 - Normal	0 Unassigned
4	5	318	1 - Junior	22	Access/Login	Request	2 - Normal	1 - Low
5	6	858	4 - Management	38	Access/Login	Request	2 - Normal	3 - High
6	7	1978	3 - Senior	10	Systems	Request	2 - Normal	3 - High
7	8	1209	4 - Management	1	Software	Request	2 - Normal	0 Unassigned
8	9	887	2 - Regular	14	Software	Request	2 - Normal	2 - Medium
9	10	1780	3 - Senior	46	Access/Login	Request	2 - Normal	1 - Low

In [684]:

```
fish= pd.read_csv("Fish.csv")
fish.head(10)
```

Out[684]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340
5	Bream	450.0	26.8	29.7	34.7	13.6024	4.9274
6	Bream	500.0	26.8	29.7	34.5	14.1795	5.2785
7	Bream	390.0	27.6	30.0	35.0	12.6700	4.6900
8	Bream	450.0	27.6	30.0	35.1	14.0049	4.8438
9	Bream	500.0	28.5	30.7	36.2	14.2266	4.9594

In [685]:

```
exercise = pd.read_csv("exercise.csv")
exercise.head(10)
```

Out[685]:

	id	diet	pulse	time	kind
0	1	low fat	85	1 min	rest
1	1	low fat	85	15 min	rest
2	1	low fat	88	30 min	rest
3	2	low fat	90	1 min	rest
4	2	low fat	92	15 min	rest
5	2	low fat	93	30 min	rest
6	3	low fat	97	1 min	rest
7	3	low fat	97	15 min	rest
8	3	low fat	94	30 min	rest
9	4	low fat	80	1 min	rest

In [686]:

```
suicide = pd.read_csv("suicide.csv")
suicide.head(10)
```

Out[686]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,15
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,15
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,15
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,15
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,15
5	Albania	1987	female	75+ years	1	35600	2.81	Albania1987	NaN	2,15
6	Albania	1987	female	35-54 years	6	278800	2.15	Albania1987	NaN	2,15
7	Albania	1987	female	25-34 years	4	257200	1.56	Albania1987	NaN	2,15
8	Albania	1987	male	55-74 years	1	137500	0.73	Albania1987	NaN	2,15
9	Albania	1987	female	5-14 years	0	311000	0.00	Albania1987	NaN	2,15

In [687]:

```
canada = pd.read_csv("canada.csv")
canada.head()
```

Out[687]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	198
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	1
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	8
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	

5 rows × 43 columns

In [688]:

```
canada.columns
```

Out[688]:

```
Index(['Type', 'Coverage', 'OdName', 'AREA', 'AreaName', 'REG', 'RegName',
       'DEV', 'DevName', '1980', '1981', '1982', '1983', '1984', '1985',
       '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993', '199
4',
       '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '200
3',
       '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '201
2',
       '2013'],
      dtype='object')
```

In [689]:

```
canada.drop(columns=['AREA' , 'DEV', 'DevName' , 'REG', 'Type', 'Coverage' , 'AreaName', 'R
canada.head()
```

Out[689]:

	OdName	1980	1981	1982	1983	1984	1985	1986	1987	1988	...	2004	2005	2006	2
0	Afghanistan	16	39	39	47	71	340	496	741	828	...	2978	3436	3009	2
1	Albania	1	0	0	0	0	0	1	2	2	...	1450	1223	856	
2	Algeria	80	67	71	69	63	44	69	132	242	...	3616	3626	4807	3
3	American Samoa	0	1	0	0	0	0	0	1	0	...	0	0	1	
4	Andorra	0	0	0	0	0	0	2	0	0	...	0	0	1	

5 rows × 35 columns

In [690]:

```
canada.rename(columns={'OdName':'Country'} , inplace=True)
canada.set_index(canada.Country,inplace=True)
canada.head()
```

Out[690]:

	Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	...	2004	200
Country													
Afghanistan	Afghanistan	16	39	39	47	71	340	496	741	828	...	2978	343
Albania	Albania	1	0	0	0	0	0	1	2	2	...	1450	122
Algeria	Algeria	80	67	71	69	63	44	69	132	242	...	3616	362
American Samoa	American Samoa	0	1	0	0	0	0	0	1	0	...	0	
Andorra	Andorra	0	0	0	0	0	0	2	0	0	...	0	

5 rows × 35 columns

In [691]:

```
canada.index.name=None
canada.head()
```

Out[691]:

	Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	...	2004	200
Afghanistan	Afghanistan	16	39	39	47	71	340	496	741	828	...	2978	343
Albania	Albania	1	0	0	0	0	0	1	2	2	...	1450	122
Algeria	Algeria	80	67	71	69	63	44	69	132	242	...	3616	362
American Samoa	American Samoa	0	1	0	0	0	0	0	1	0	...	0	
Andorra	Andorra	0	0	0	0	0	0	2	0	0	...	0	

5 rows × 35 columns

In [692]:

```
del canada['Country']
canada.head()
```

Out[692]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	200
Afghanistan	16	39	39	47	71	340	496	741	828	1076	...	2978	3436	300
Albania	1	0	0	0	0	0	1	2	2	3	...	1450	1223	85
Algeria	80	67	71	69	63	44	69	132	242	434	...	3616	3626	480
American Samoa	0	1	0	0	0	0	0	1	0	1	...	0	0	
Andorra	0	0	0	0	0	0	2	0	0	0	...	0	0	

5 rows × 34 columns

In [693]:

```
canada = canada.transpose()
```

In [694]:

```
canada.head()
```

Out[694]:

	Afghanistan	Albania	Algeria	American Samoa	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia
1980	16	1	80	0	0	1	0	368	0
1981	39	0	67	1	0	3	0	426	0
1982	39	0	71	0	0	6	0	626	0
1983	47	0	69	0	0	6	0	241	0
1984	71	0	63	0	0	4	42	237	0

5 rows × 197 columns

In [695]:

```
led = pd.read_csv("Life Expectancy Data.csv")
led.head()
```

Out[695]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0

5 rows × 22 columns

In [696]:

```
adult = pd.read_csv("adult.csv")
adult.head(10)
```

Out[696]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family
2	66	?	186061	Some-college	10	Widowed	?	Unmarried
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried
7	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative
8	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried

In [697]:

```
adult = adult[adult["workclass"].isin (['Private', 'State-gov', 'Federal-gov'])]
adult.head()
```

Out[697]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried

In [698]:

```
iris = sns.load_dataset("iris")
iris.head()
```

Out[698]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [700]:

```
cars = pd.read_csv("cars.csv")
cars.head(10)
```

Out[700]:

	mpg	cylinders	cubicinches	hp	weightlbs	time-to-60	year	brand
0	14.0	8	350	165	4209	12	1972	US.
1	31.9	4	89	71	1925	14	1980	Europe.
2	17.0	8	302	140	3449	11	1971	US.
3	15.0	8	400	150	3761	10	1971	US.
4	30.5	4	98	63	2051	17	1978	US.
5	23.0	8	350	125	3900	17	1980	US.
6	13.0	8	351	158	4363	13	1974	US.
7	14.0	8	440	215	4312	9	1971	US.
8	25.4	5	183	77	3530	20	1980	Europe.
9	37.7	4	89	62	2050	17	1982	Japan.

Line Charts

In [27]:

```
col1 = np.linspace(0, 10, 1000)
col2 = np.sin(col1)
df = pd.DataFrame({"C1" : col1 , "C2" :col2})
df.head(10)
```

Out[27]:

	C1	C2
0	0.00000	0.000000
1	0.01001	0.010010
2	0.02002	0.020019
3	0.03003	0.030026
4	0.04004	0.040029
5	0.05005	0.050029
6	0.06006	0.060024
7	0.07007	0.070013
8	0.08008	0.079995
9	0.09009	0.089968

In [25]:

```
# Plotting lineplot using sns.lineplot()
plt.style.use('seaborn-darkgrid')
%matplotlib inline
sns.lineplot(x=df.C1,y=df.C2,data=df)
plt.show()
```

In [26]:

```
"""  - Adjusting background color using axes.facecolor
     - Changing label size using xtick.labelsize , ytick.labelsize """

plt.figure(figsize=(14,6))
sns.set(rc={"axes.facecolor":"#FFFDE7", "axes.grid":True,'xtick.labelsize':14,'ytick.labels
sns.lineplot(x=df.C1,y=df.C2,data=df , linewidth = 2)
sns.lineplot(x=df.C1,y=df.C2+1,data=df , linewidth = 2)
plt.show()
```

In [27]:

```
"""  - Adjusting background color using axes.facecolor
     - Changing label size using xtick.labelsize , ytick.labelsize """

plt.figure(figsize=(14,6))
sns.set(rc={"axes.facecolor":"#283747", "axes.grid":False,'xtick.labelsize':14,'ytick.label
sns.lineplot(x=df.C1,y=df.C2,data=df , color = "#FF5722" , linewidth = 2 )
sns.lineplot(x=df.C1,y=np.cos(df.C1),data=df , color = "#FFEB3B" , linewidth = 2)
plt.show()
```

In [28]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [32]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [30]:

```
plt.figure(figsize=(20,8))
sns.lineplot(data=spotify)
plt.show()
```

Look at the X-Asis. It is clealy not able to interpret the Index as Date. So the next step will be to convert the Index to Datetime.

In [31]:

```
spotify.index = pd.to_datetime(spotify.index) # Converting datatype of index to Datetime
```

In [32]:

```
plt.figure(figsize=(20,8))
sns.lineplot(data=spotify,linewidth = 2)
plt.title("Daily Global Streams")
plt.show()
```

In [33]:

```
# Using Matplotlib for same visualization
import matplotlib as mpl
plt.figure(figsize=(20,8))
plt.plot(spotify['Despacito'] , label="Despacito")
plt.plot(spotify['Shape of You'] , label="Shape of You")
plt.plot(spotify['Something Just Like This'] , label="Something Just Like This")
plt.plot(spotify['HUMBLE.'] , label="Humble")
plt.plot(spotify['Unforgettable'] , label="Unforgettable")
plt.legend()
plt.show()
```

So we can see using Seabon we can save many lines of code.

In [34]:

```
plt.figure(figsize=(20,6))
sns.lineplot(data=spotify['Despacito'],linewidth = 1.5 , label = 'Despacito')
sns.lineplot(data=spotify['HUMBLE.'],linewidth = 1.5 , label = 'Humble')
plt.show()
```

In [35]:

```
canada.head()
```

Out[35]:

	Afghanistan	Albania	Algeria	American Samoa	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia
1980	16	1	80	0	0	1	0	368	0
1981	39	0	67	1	0	3	0	426	0
1982	39	0	71	0	0	6	0	626	0
1983	47	0	69	0	0	6	0	241	0
1984	71	0	63	0	0	4	42	237	0

5 rows × 197 columns

In [701]:

```
plt.figure(figsize=(16,6))
plt.title("Immigrants from India",fontsize = 14)
sns.set(rc={"axes.facecolor":"#283747", "axes.grid":False,'xtick.labelsize':10,'ytick.label
plt.xticks(rotation=45) # Rotating X tickts by 45 degrees
sns.lineplot(x = canada.index.values, y = canada['India'])
plt.show()
```

In [702]:

```
# Plotting multiple sets of data (E.g Immigration data of multiple countries in one plot)
plt.figure(figsize=(16,6))
sns.set(rc={"axes.facecolor":"#283747", "axes.grid":False,'xtick.labelsize':10,'ytick.label
plt.title("Immigrants from India , Pakistan & Bangladesh",fontsize = 14)
plt.xticks(rotation=45) # Rotating X tickts by 45 degrees
sns.lineplot(x = canada.index.values, y = canada['India'] , color = '#ff9900' , label= 'Inc
sns.lineplot(x = canada.index.values, y = canada['Pakistan'] , color = '#4586ff' , label =
sns.lineplot(x = canada.index.values, y = canada['Bangladesh'] , color = '#a2ef44' , label
plt.legend(facecolor= 'grey' , fontsize='large' , edgecolor = 'black' ,shadow=True) # Leger
plt.show()
```

In [38]:

```
# Plotting multiple sets of data using for loop (E.g Immigration data of multiple countries
plt.figure(figsize=(16,6))
plt.title("Immigrants",fontsize = 16)
for i in canada.columns:
    if canada[i].name != 'Total' and canada[i].name != 'Unknown':
        x=canada.index.values
        y=canada[i]
        sns.lineplot(x,y)
plt.xlabel ('Year')
plt.ylabel ('Country')
plt.show()
```

In [39]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [40]:

```
employment.head()
```

Out[40]:

	Age	Gender	Period	Unemployed
0	16 to 19 years	Men	2005-01-01	91000
1	20 to 24 years	Men	2005-01-01	175000
2	25 to 34 years	Men	2005-01-01	194000
3	35 to 44 years	Men	2005-01-01	201000
4	45 to 54 years	Men	2005-01-01	207000

In [41]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(14,7))
plt.style.use('seaborn-darkgrid')
# Group variable using "hue" that will produce lines with different colors
sns.lineplot(x="Period" , y="Unemployed" , hue="Gender" , data=employment)
plt.show()
```

In [42]:

```
# Using markers to identify groups
plt.figure(figsize=(14,7))
plt.style.use('seaborn-darkgrid')
sns.lineplot(x="Period" , y="Unemployed" , hue = "Gender" ,style="Gender" , markers=True ,
plt.show()
```

In [43]:

```
plt.figure(figsize=(14,7))
plt.style.use('seaborn-darkgrid')
sns.lineplot(x="Period" , y="Unemployed" , hue = "Gender" ,style="Gender" ,  err_style="bar
plt.show()
```

In [44]:

```
plt.figure(figsize=(14,7))
sns.set(rc={'xtick.labelsize':17,'ytick.labelsize':10,'axes.labelsize':15 , "axes.grid":Fal

# Use "Pallete" to specify the colors to be used for different levels of the hue
sns.lineplot(x="Period" , y="Unemployed", data = employment, hue = "Gender",
            style = "Gender", dashes = False, palette = 'CMRmap' , markers = ["o", ">"])

plt.show()
```

In [708]:

```
# Color and line dashing to represent 2 different grouping variables using "hue" & "style"
plt.figure(figsize=(16,9))
plt.style.use('seaborn-darkgrid')
plt.gcf().text(.2, .84, "Line Plot", fontsize = 40, color='Black' ,ha='center', va='center'
sns.lineplot(x="Period" , y="Unemployed" , hue="Age" , style="Gender" , data=employment)
plt.show()
```

In [46]:

```
emp = employment[employment.Period.between('2005-01-01', '2006-01-01' , inclusive = True)]
emp.tail()
```

Out[46]:

	Age	Gender	Period	Unemployed
177	25 to 34 years	Women	2006-01-01	92000
178	35 to 44 years	Women	2006-01-01	134000
179	45 to 54 years	Women	2006-01-01	110000
180	55 to 64 years	Women	2006-01-01	85000
181	65 years and over	Women	2006-01-01	61000

In [47]:

```
#Showing all experiments instead of Aggregate using "units" and "estimator"
plt.figure(figsize=(14,7))
plt.style.use('seaborn-darkgrid')
sns.lineplot(x="Period" , y="Unemployed" , hue = "Age" ,style="Gender" ,
 units="Age" ,markers=True , dashes=False ,estimator=None, lw=1,data=emp)
plt.legend(bbox_to_anchor=(1.0, 1.0) , shadow=True, fontsize='large')
plt.show()
```

In [48]:

```
# Combining lineplots using relplot
plt.figure(figsize=(10,10))
sns.set(rc={'xtick.labelsize':17,'ytick.labelsize':17,'axes.labelsize':20  , "axes.grid":Fa
sns.relplot(x="Period" , y="Unemployed" , hue="Age" , col="Gender",kind='line', height=8.5,
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```

In [49]:

```
corona.head()
```

Out[49]:

Date	Country	Confirmed	Recovered	Deaths
2020-01-22	Afghanistan	0	0	0
2020-01-22	Albania	0	0	0
2020-01-22	Algeria	0	0	0
2020-01-22	Andorra	0	0	0
2020-01-22	Angola	0	0	0

In [50]:

```
plt.figure(figsize=(20,6))
sns.lineplot(data=corona[corona['Country'] == 'Italy']['Confirmed'] , label = "Confirmed")
sns.lineplot(data=corona[corona['Country'] == 'Italy']['Deaths'] , label = "Deaths")
sns.lineplot(data=corona[corona['Country'] == 'Italy']['Recovered'], label = "Recovered")
plt.title("Total Confirmed Cases / Deaths / Recovered in Italy")
plt.show()
```

In [51]:

```
plt.figure(figsize=(20,6))
sns.lineplot(data=corona[corona['Country'] == 'US']['Confirmed'] , label = "Confirmed")
sns.lineplot(data=corona[corona['Country'] == 'US']['Deaths'] , label = "Deaths")
sns.lineplot(data=corona[corona['Country'] == 'US']['Recovered'], label = "Recovered")
plt.title("Total Confirmed Cases / Deaths / Recovered in USA")
plt.show()
```

Plot Styling

In [52]:

```
# Enabling whitegrid style
sns.set_style("whitegrid")
plt.figure(figsize=(20,6))
sns.lineplot(data=corona[corona['Country'] == 'US']['Confirmed'] , label = "Confirmed")
sns.lineplot(data=corona[corona['Country'] == 'US']['Deaths'] , label = "Deaths")
sns.lineplot(data=corona[corona['Country'] == 'US']['Recovered'], label = "Recovered")
plt.title("Total Confirmed Cases / Deaths / Recovered in USA")
plt.show()
```

There will be horizontal grid lines in the background for "whitegrid" style.

In [53]:

```
# Enabling darkgrid style
sns.set_style("darkgrid")
plt.figure(figsize=(20,6))
sns.lineplot(data=corona[corona['Country'] == 'US']['Confirmed'] , label = "Confirmed")
sns.lineplot(data=corona[corona['Country'] == 'US']['Deaths'] , label = "Deaths")
sns.lineplot(data=corona[corona['Country'] == 'US']['Recovered'], label = "Recovered")
plt.title("Total Confirmed Cases / Deaths / Recovered in USA")
plt.show()
```

There will be horizontal grid lines in the background for "darkgrid" style. Also we will see a dark Background.

In [54]:

```
# Ticks Style
sns.set_style("ticks")
plt.figure(figsize=(20,6))
sns.lineplot(data=corona[corona['Country'] == 'US']['Confirmed'] , label = "Confirmed")
sns.lineplot(data=corona[corona['Country'] == 'US']['Deaths'] , label = "Deaths")
sns.lineplot(data=corona[corona['Country'] == 'US']['Recovered'], label = "Recovered")
plt.title("Total Confirmed Cases / Deaths / Recovered in USA")
plt.show()
```

There will be no horizontal grid lines in the background for "ticks" style. We will only see ticks in X & Y axis

Revert to default styling

In [55]:

```
sns.set_style("white")
plt.figure(figsize=(20,6))
sns.lineplot(data=corona[corona['Country'] == 'US']['Confirmed'] , label = "Confirmed")
sns.lineplot(data=corona[corona['Country'] == 'US']['Deaths'] , label = "Deaths")
sns.lineplot(data=corona[corona['Country'] == 'US']['Recovered'], label = "Recovered")
plt.title("Total Confirmed Cases / Deaths / Recovered in USA")
plt.show()
```

SCatter Plots

In [709]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [956]:

```
insurance.head(5)
```

Out[956]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [957]:

```
plt.figure(figsize=(7,7))
sns.scatterplot(x=insurance.bmi , y=insurance.charges )
plt.show()
```

In [958]:

```
#Changing the color of data points using "color" parameter
plt.figure(figsize=(7,7))
sns.scatterplot(x=insurance.bmi , y=insurance.charges , color='#91bd3a')
plt.show()
```

In [959]:

```
#Changing the shape of data points using "marker" parameter
plt.figure(figsize=(8,8))
sns.scatterplot(x=insurance.bmi , y=insurance.charges , color='#339900' , marker = ">"  )
plt.show()
```

In [960]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(7,7))
sns.scatterplot(x=insurance.bmi , y=insurance.charges , hue=insurance.sex )
plt.show()
```

In [961]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(7,7))
sns.scatterplot(x=insurance.bmi , y=insurance.charges , hue=insurance.smoker)
plt.show()
```

In [962]:

```
#Setting X limit using "plt.xlim"
plt.figure(figsize=(7,7))
plt.xlim([0,55])
sns.scatterplot(x=insurance.bmi , y=insurance.charges , hue=insurance.smoker)
plt.show()
```

In [970]:

```
# Showing two different grouping variables using "hue" amd "style" parameter
plt.figure(figsize=(10,10))
plt.gcf().text(.5, .9, "Scatter Plot", fontsize = 40, color='Black' ,ha='center', va='cente
sns.scatterplot(x=insurance.bmi , y=insurance.charges , hue=insurance.smoker,style=insuranc
plt.show()
```

In [220]:

```
stdperf.head()
```

Out[220]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [221]:

```
plt.figure(figsize=(9,9))
sns.scatterplot(x= stdperf['reading score'] , y= stdperf['writing score'] ,
                hue=stdperf['test preparation course'],size = stdperf['math score'] , sizes
plt.show()
```

In [222]:

```
pokemon.head()
```

Out[222]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legend
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	F
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	F
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	F
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	F
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	F

In [223]:

```
# Varying the size of the points for a quantitative variable using "size" parameter
plt.figure(figsize=(9,9))
sns.scatterplot(x= pokemon['Speed'] , y= pokemon['Attack'] ,hue= pokemon['Legendary'] ,
                size = pokemon['Sp. Atk'] , sizes = (20,200))
plt.show()
```

Relplot

Relplot provides access to several different axes-level functions like line & scatter

In [224]:

```
sns.set_style("white")
```

In [225]:

```
employment.head()
```

Out[225]:

	Age	Gender	Period	Unemployed
0	16 to 19 years	Men	2005-01-01	91000
1	20 to 24 years	Men	2005-01-01	175000
2	25 to 34 years	Men	2005-01-01	194000
3	35 to 44 years	Men	2005-01-01	201000
4	45 to 54 years	Men	2005-01-01	207000

In [226]:

```
# Drawing scatter plot using relplot
plt.figure(figsize=(14,16))
sns.relplot(x="Period" , y="Unemployed" , hue="Gender" ,data=employment , height=7 , aspect
plt.show()
```

```
<Figure size 1008x1152 with 0 Axes>
```

In [227]:

```
# Drawing scatter plot using relplot
plt.figure(figsize=(14,16))
sns.relplot(x="bmi" , y="charges" , hue="smoker" ,data=insurance , height=8 , aspect=1)
plt.show()
```

```
<Figure size 1008x1152 with 0 Axes>
```

In [228]:

```
# Use "Pallete" to specify the colors to be used for different levels of the hue
sns.set(rc={'xtick.labelsize':12,'ytick.labelsize':12,'axes.labelsize':12  , "axes.grid":Fa
sns.relplot(x="bmi" , y="charges" , hue="smoker" ,data=insurance , height=8 , aspect=1 ,
            palette=["#FF3333" ,"#00CC00"])
plt.show()
```

In [229]:

```
fish.head()
```

Out[229]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [981]:

```
plt.figure(figsize=(14,16))
sns.relplot(x="Width" , y="Height" , hue="Species" ,data=fish , height=8 , aspect=1)
plt.gcf().text(.5, .99, "Rel Plot", fontsize = 40, color='Black' ,ha='center', va='center')
plt.show()
```

```
<Figure size 1008x1152 with 0 Axes>
```

In [231]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(10,10))
sns.set(rc={'xtick.labelsize':12,'ytick.labelsize':10,'axes.labelsize':10  , "axes.grid":Fa
sns.relplot(x="Period" , y="Unemployed" , hue="Gender" , col="Age",kind='line', height=5,
            aspect=.5 ,data=employment)
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```

In [232]:

```
# Facet along the columns to show a categorical variable using "col" parameter
sns.set(rc={'xtick.labelsize':16,'ytick.labelsize':16,'axes.labelsize':16  , "axes.grid":Fa
sns.relplot(x="Period" , y="Unemployed" , hue="Gender" , col="Age",kind='line', height=6,
            aspect=1 ,data=employment, col_wrap = 4 , linewidth =2)
plt.show()
```

In [233]:

```
insurance.head()
```

Out[233]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [234]:

```
# Facet along the columns and rows to show a categorical variable using "col" and "row" par
sns.set(rc={'xtick.labelsize':12,'ytick.labelsize':12,'axes.labelsize':12  , "axes.grid":Fa
sns.relplot(x="bmi", y="charges", hue="sex",
                col="sex", row="region", data=insurance)
```

Out[234]:

```
<seaborn.axisgrid.FacetGrid at 0x1edd254a6d8>
```

In [971]:

```
# Facet along the columns and rows to show a categorical variable using "col" and "row" par
sns.set(rc={'xtick.labelsize':14,'ytick.labelsize':14,'axes.labelsize':14  , "axes.grid":Fa
sns.relplot(x="bmi", y="charges", hue="sex",height=7, aspect=.6,
                col="region", row="sex", data=insurance)
```

Out[971]:

```
<seaborn.axisgrid.FacetGrid at 0x1edd9586320>

<Figure size 432x288 with 0 Axes>
```

In [236]:

```
emp.tail()
```

Out[236]:

	Age	Gender	Period	Unemployed
177	25 to 34 years	Women	2006-01-01	92000
178	35 to 44 years	Women	2006-01-01	134000
179	45 to 54 years	Women	2006-01-01	110000
180	55 to 64 years	Women	2006-01-01	85000
181	65 years and over	Women	2006-01-01	61000

In [237]:

```
# Limiting the number of columns using "col_wrap"
sns.relplot(x="Period", y="Unemployed", hue="Gender",col="Age" , col_wrap=3,kind="line",lir
            height=5, aspect=1, estimator=None, data=employment)
```

Out[237]:

```
<seaborn.axisgrid.FacetGrid at 0x1edd56add30>
```

Bar Plot

Bar Plot shows the relationship between a numerical variable and a categorical variable.

In [731]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [239]:

```
plt.figure(figsize=(20,7))
sns.barplot(x=pokemon['Type 1'], y= pokemon['Attack'])
plt.show()
```

As per the above Bar plot pokemon with Type-1 as $Dragon$ are best attackers

In [240]:

```
plt.figure(figsize=(20,7))
sns.barplot(x=pokemon['Type 1'], y= pokemon['Defense'])
plt.show()
```

In [241]:

```
# Sorted Bar plot
plt.figure(figsize=(20,7))
order = pokemon.groupby(['Type 1']).mean().sort_values('Defense' , ascending = False).index
sns.barplot(x=pokemon['Type 1'], y= pokemon['Defense'] , order=order)
plt.show()
```

Pokemon with Type-2 as Steel have best defence

In [242]:

```
plt.figure(figsize=(20,7))
sns.barplot(x=pokemon['Type 1'], y= pokemon['Speed'])
plt.show()
```

Pokemons with Type-1 as Flying are the fastest

In [243]:

```
# Horizontal Bar plot
plt.figure(figsize=(8,10))
sns.barplot(x=pokemon['Speed'], y= pokemon['Type 1'])
plt.show()
```

In [244]:

```
plt.figure(figsize=(11,7))
sns.barplot(x="Age",y="Unemployed", data=employment)
plt.show()
```

In [245]:

```
plt.figure(figsize=(20,7))
sns.countplot(x=pokemon['Type 1'])
plt.show()
```

In [246]:

```
plt.figure(figsize=(20,7))
sns.countplot(x=pokemon['Type 1'] , order = pokemon['Type 1'].value_counts().index)
plt.show()
```

Pokemon with Type-1 as water & Normal are most common.

In [247]:

```
#Changing the background of bar plot
plt.figure(figsize=(20,7))
sns.set(rc={"axes.facecolor":"#283747", "axes.grid":False,'xtick.labelsize':14,'ytick.label
sns.countplot(x=pokemon['Type 1'] , order = pokemon['Type 1'].value_counts().index)
plt.show()
```

In [991]:

```
plt.figure(figsize=(20,7))
sns.set(rc={"axes.facecolor":"#283747", "axes.grid":False,'xtick.labelsize':14,'ytick.label
plt.gcf().text(.5, .93, "Bar Plot", fontsize = 60, color='Black' ,ha='center', va='center')
sns.countplot(x=pokemon['Type 1'] , order = pokemon['Type 1'].value_counts().index , palett
plt.show()
```

In [249]:

```
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [250]:

```
plt.figure(figsize=(20,7))
sns.countplot(x=pokemon['Type 2'])
plt.show()
```

As per above Count plot Type-2 Flying pokemon are most common

In [251]:

```
exercise.head()
```

Out[251]:

	id	diet	pulse	time	kind
0	1	low fat	85	1 min	rest
1	1	low fat	85	15 min	rest
2	1	low fat	88	30 min	rest
3	2	low fat	90	1 min	rest
4	2	low fat	92	15 min	rest

In [252]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(8,6))
sns.barplot(x=exercise.kind , y=exercise.pulse ,hue=exercise.diet)
plt.show()
```

In [253]:

```
helpdesk.head()
```

Out[253]:

	ticket	requestor	RequestorSeniority	ITOwner	FiledAgainst	TicketType	Severity	Priority
0	1	1929	1 - Junior	50	Systems	Issue	2 - Normal	0 Unassigned
1	2	1587	2 - Regular	15	Software	Request	1 - Minor	1 - Lov
2	3	925	2 - Regular	15	Access/Login	Request	2 - Normal	0 Unassigned
3	4	413	4 - Management	22	Systems	Request	2 - Normal	0 Unassigned
4	5	318	1 - Junior	22	Access/Login	Request	2 - Normal	1 - Lov

In [254]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(10,6))
sns.barplot(x=helpdesk.TicketType , y=helpdesk.daysOpen , hue=helpdesk.FiledAgainst)
sns.despine()
plt.show()
```

Cat Plot

Cat Plot provides access to several axes-level functions ("point", "bar", "strip", "swarm","box", "violin", "count" or "boxen") that show the relationship between a numerical and one or more categorical variables

In [255]:

```
# Use Count Plot to visualize data
sns.catplot(y = "Type 1", kind = "count",data = pokemon)
plt.show()
```

In [256]:

```
#Sorted Count Plot
sns.catplot(y = "Type 1", kind = "count",data = pokemon , order=pokemon['Type 1'].value_cou
plt.show()
```

In [257]:

```
# Use Voilin Plot to visualize data
plt.figure(figsize=(7,7))
sns.catplot(x="species" , y = "petal_width" ,kind="violin" ,data=iris)
plt.show()
```

```
<Figure size 504x504 with 0 Axes>
```

In [258]:

```
plt.figure(figsize=(11,9))
sns.catplot(x = "petal_width" , y="species" ,kind="violin" ,data=iris ,
            height=5, aspect=2)
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

In [259]:

```
exercise.head()
```

Out[259]:

	id	diet	pulse	time	kind
0	1	low fat	85	1 min	rest
1	1	low fat	85	15 min	rest
2	1	low fat	88	30 min	rest
3	2	low fat	90	1 min	rest
4	2	low fat	92	15 min	rest

In [260]:

```
# Facet along the columns to show a categorical variable using "col" parameter
sns.catplot(x='kind' , y='pulse' , data=exercise , col="diet")
```

Out[260]:

```
<seaborn.axisgrid.FacetGrid at 0x1edd1e14be0>
```

In [261]:

```
helpdesk.head()
```

Out[261]:

	ticket	requestor	RequestorSeniority	ITOwner	FiledAgainst	TicketType	Severity	Priority
0	1	1929	1 - Junior	50	Systems	Issue	2 - Normal	0 Unassigned
1	2	1587	2 - Regular	15	Software	Request	1 - Minor	1 - Low
2	3	925	2 - Regular	15	Access/Login	Request	2 - Normal	0 Unassigned
3	4	413	4 - Management	22	Systems	Request	2 - Normal	0 Unassigned
4	5	318	1 - Junior	22	Access/Login	Request	2 - Normal	1 - Low

In [738]:

```
""" Facet along the columns to show a categorical variable using "col" parameter and
Limiting the number of columns using "col_wrap" """

plt.figure(figsize=(20,10))
sns.catplot("Priority", col="FiledAgainst",col_wrap=2,
                data=helpdesk,kind="count",height=5, aspect=1)
plt.show()
```

```
<Figure size 1440x720 with 0 Axes>
```

In [263]:

```
employment.head()
```

Out[263]:

	Age	Gender	Period	Unemployed
0	16 to 19 years	Men	2005-01-01	91000
1	20 to 24 years	Men	2005-01-01	175000
2	25 to 34 years	Men	2005-01-01	194000
3	35 to 44 years	Men	2005-01-01	201000
4	45 to 54 years	Men	2005-01-01	207000

In [729]:

```
# Facet along the columns to show a categorical variable using "col" parameter
sns.set(rc={'xtick.labelsize':15,'ytick.labelsize':15,'axes.labelsize':15 , "axes.facecolor
sns.catplot(x="Gender", y="Unemployed", col="Age",data=employment,kind="bar",height=6 , asp
plt.show()
```

Dist Plot

Dist plot is used to plot a univariate distribution of observations.

In [740]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [269]:

```
num = np.random.normal(1,10,1000)
sns.distplot(num)
plt.show()
```

In [270]:

```
sns.distplot(num,kde=True) # Histogram with a kernel density estimate (Density Curve)
plt.show()
```

In [271]:

```
sns.distplot(num,kde=False) # Histogram without a kernel density estimate (Density Curve)
plt.show()
```

In [272]:

```
sns.distplot(num,kde=False,rug=True) # Histogram with Rugplot
plt.show()
```

In [273]:

```
sns.distplot(num,kde=True,rug=True,hist=False) # Histogram with Rugplot & KDE
plt.show()
```

In [274]:

```
#Housing Dataset
housing.head()
```

Out[274]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
0	-122.23	37.88	41.0	880.0	129.0	322.0	126
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259

In [275]:

```
housing.hist(bins=40 , figsize=(20,20)) #Pandas Hist function
plt.show()
```

In [276]:

```
plt.figure(figsize=(11,7))
sns.distplot( housing["median_income"])
plt.show()
```

In [277]:

```
plt.figure(figsize=(11,7))
sns.distplot( housing["median_income"] , bins=20)
plt.show()
```

In [278]:

```
plt.figure(figsize=(11,7))
sns.distplot( housing["median_income"] ,bins=20 , hist_kws=dict(edgecolor = '#9dab86' ,line
plt.show()
```

In [755]:

```
#Changing the color of all the plot elements using "color" parameter
plt.figure(figsize=(11,7))
sns.distplot( housing["median_income"] ,bins=20 , hist_kws=dict(edgecolor = '#9dab86' ,line
plt.show()
```

In [741]:

```
sns.set(style="white", color_codes=True)
```

In [742]:

```
# Displaying multiple distplots using subplot function.
fig1 , axes = plt.subplots(nrows=3,ncols=3 , figsize = (20,20))
sns.distplot( housing["longitude"] , ax=axes[0, 0])
sns.distplot( housing["latitude"] ,  ax=axes[0, 1])
sns.distplot( housing["housing_median_age"] , ax=axes[0, 2])
sns.distplot( housing["total_rooms"], ax=axes[1, 0] )
sns.distplot( housing["population"] , ax=axes[1, 1] )
sns.distplot( housing["households"] , ax=axes[1, 2] )
sns.distplot( housing["median_income"] , ax=axes[2, 0])
sns.distplot( housing["median_house_value"], ax=axes[2, 1])
sns.distplot( housing["median_house_value"], ax=axes[2, 2])
plt.show()
```

In [754]:

```
# Displaying multiple distplots using subplot function.
fig1 , axes = plt.subplots(nrows=3,ncols=3 , figsize = (20,20))
sns.distplot( housing["longitude"] , color="#00bcd4", ax=axes[0, 0] , kde=False , bins=20)
sns.distplot( housing["latitude"] , color="#937d14", ax=axes[0, 1] , kde=False,bins=20)
sns.distplot( housing["housing_median_age"] , color="#006600", ax=axes[0, 2],kde=False,bins
sns.distplot( housing["total_rooms"] , color="#ff1e56", ax=axes[1, 0] , kde=False,bins=20)
sns.distplot( housing["population"] , color="#216353", ax=axes[1, 1] , kde=False,bins=20)
sns.distplot( housing["households"] , color="#FF8F00", ax=axes[1, 2] , kde=False,bins=20)
sns.distplot( housing["median_income"] , color="#33FF00", ax=axes[2, 0] , kde=False,bins=20
sns.distplot( housing["median_house_value"] , color="#FF3300", ax=axes[2, 1], kde=False,bir
sns.distplot( housing["median_house_value"] , color="#CCCC00", ax=axes[2, 2] , kde=False,bi
plt.show()
```

KDE PLOT

KDE Plot is used to estimate the probability density function of a continuous random variable.

In [281]:

```
sns.set_style("darkgrid")
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (14,14))

x = np.random.normal(1,10,1000)

#Simple KDE Plot
axes[0,0].set_title("Simple KDE Plot")
sns.kdeplot(x,ax=axes[0,0])


# Shade under the density curve using the "shade" parameter
axes[0,1].set_title("KDE Plot (Shaded Area Under the Curve)")
sns.kdeplot(x,shade=True,ax=axes[0,1])


# Shade under the density curve using the "shade" parameter and use a different color.
axes[1,0].set_title("KDE Plot (Different Color)")
sns.kdeplot(x,ax=axes[1,0],color = 'r',shade=True,cut=0)

#Plotting the density on the vertical axis
axes[1,1].set_title("KDE Plot (Density on Vertical Axis)")
sns.kdeplot(x,vertical=True)

plt.show()
```

Simple KDE Plot
KDE Plot (Shaded Area Under the Curve)
KDE Plot (Different Color)
KDE Plot (Density on Vertical Axis)

In [282]:

```
plt.figure(figsize=(6,8))
x = np.linspace(0, 10, 100)
y = np.sin(x)
sns.kdeplot(x,y,shade=True,cmap="Reds", shade_lowest=False)
```

Out[282]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd24ac7f0>
```

In [283]:

```
insurance.head()
```

Out[283]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [284]:

```
plt.figure(figsize=(6,8))
sns.kdeplot(insurance.bmi,insurance.charges,shade=True,cmap="Reds", shade_lowest=False)
plt.show()
```

In [285]:

```
iris = sns.load_dataset("iris")
plt.figure(figsize=(8,6))
sns.kdeplot(iris.sepal_width, iris.sepal_length,cmap="Reds", shade=True, shade_lowest=False
plt.show()
```

Swarm Plot

Swarm plot is a categorical scatterplot with non-overlapping points

In [139]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("white")
```

In [287]:

```
insurance.head()
```

Out[287]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [288]:

```
#Simple Swarm plot
plt.figure(figsize=(7,7))
sns.swarmplot(x=insurance.smoker, y=insurance.charges)
plt.show()
```

In [289]:

```
# Draw horizontal swarm plot
plt.figure(figsize=(7,7))
sns.swarmplot(x=insurance.charges , y=insurance.smoker)
plt.show()
```

In [290]:

```
plt.figure(figsize=(7,7))
sns.swarmplot(x="species" , y = "petal_width" ,data=iris , size=6)
plt.show()
```

In [291]:

```
# Displaying multiple swarmplots using subplot function.
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (20,20))
sns.swarmplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris , size=8)
sns.swarmplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris , size=8)
sns.swarmplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris, size=8)
sns.swarmplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris , size=8)
plt.show()
```

In [292]:

```
exercise.head()
```

Out[292]:

	id	diet	pulse	time	kind
0	1	low fat	85	1 min	rest
1	1	low fat	85	15 min	rest
2	1	low fat	88	30 min	rest
3	2	low fat	90	1 min	rest
4	2	low fat	92	15 min	rest

In [293]:

```
plt.figure(figsize=(7,7))
sns.swarmplot(x= "kind",  y = "pulse", data = exercise)
```

Out[293]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd243b6d8>
```

In [294]:

```
# Changing size of data points using "size"parameter
plt.figure(figsize=(7,7))
sns.swarmplot(x= "kind",  y = "pulse", size = 9, data = exercise)
```

Out[294]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd6ec0898>
```

In [295]:

```
#Changing edge color , size and linewidth of data points
plt.figure(figsize=(7,7))
sns.swarmplot(x= "kind",  y = "pulse", size = 9 , linewidth= 2 , edgecolor="black" , data =
```

Out[295]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd243b7b8>
```

In [296]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(7,7))
sns.swarmplot(x= "kind",  y = "pulse", hue="diet", size = 8 , data = exercise)
```

Out[296]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd168e668>
```

In [297]:

```
employment.head()
```

Out[297]:

	Age	Gender	Period	Unemployed
0	16 to 19 years	Men	2005-01-01	91000
1	20 to 24 years	Men	2005-01-01	175000
2	25 to 34 years	Men	2005-01-01	194000
3	35 to 44 years	Men	2005-01-01	201000
4	45 to 54 years	Men	2005-01-01	207000

In [298]:

```
plt.figure(figsize=(11,8))
sns.swarmplot(x=employment.Age ,palette="Set1", y = employment.Unemployed)
plt.show()
```

In [299]:

```
# Facet along the columns to show a categorical variable using "col" parameter
sns.set(rc={'xtick.labelsize':10,'ytick.labelsize':10,'axes.labelsize':14})
sns.set_style("white")
sns.catplot(x="Age" , y = "Unemployed" , col= "Gender" , data=employment, kind="swarm" , he
plt.show()
```

In [130]:

```
pokemon.head()
```

Out[130]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legend
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	F
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	F
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	F
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	F
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	F

In [131]:

```
pokemon.columns
```

Out[131]:

```
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. At
k',
       'Sp. Def', 'Speed', 'Generation', 'Legendary', 'Total'],
      dtype='object')
```

In [136]:

```
pokemon1 = pokemon[[ 'HP', 'Attack', 'Defense', 'Sp. Atk','Sp. Def', 'Speed']]
pokemon1.head()
```

Out[136]:

	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
0	45	49	49	65	65	45
1	60	62	63	80	80	60
2	80	82	83	100	100	80
3	80	100	123	122	120	80
4	39	52	43	60	50	65

In [145]:

```
plt.figure(figsize=(14,12))
sns.swarmplot(data=pokemon1,size=6)
plt.show()
```

Violin Plot

Violinplots summarize numeric data over a set of categories.Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator.

In [759]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("white")
```

In [760]:

```
# Simple Violin Plot
plt.figure(figsize=(7,7))
sns.violinplot(x="species" , y = "petal_width" ,data=iris)
plt.show()
```

In [763]:

```
# Drawing the quartiles as horizontal lines instead of a mini-box using (inner="quartile")
plt.figure(figsize=(7,7))
sns.violinplot(x="species" , y = "petal_width" ,data=iris , inner="quartile")
plt.show()
```

In [767]:

```
# Remove interior section of the Violin plot
plt.figure(figsize=(7,7))
sns.violinplot(x="species" , y = "petal_width" ,data=iris , inner=None)
plt.show()
```

In [765]:

```
# horizontal violin plot
plt.figure(figsize=(9,7))
sns.violinplot(y="species" , x = "petal_width" ,data=iris , inner=None)
plt.show()
```

In [320]:

```
# Muted palette
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , palette="muted")
plt.show()
```

In [321]:

```
#Scale the density relative to the counts across all bins
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , palette="Set1" , scale="count
plt.show()
```

In [768]:

```
# Adjust the bandwidth of the KDE filtering parameter using "bw"
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , palette="Set1" , scale="count
plt.show()
```

In [774]:

```
# Adjust the bandwidth of the KDE filtering parameter using "bw"
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , palette="Set1" , scale="count
plt.show()
```

In [323]:

```
# Limit the violin range within the range of the observed data using "cut =0"
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , palette="Set1" , scale="count
plt.show()
```

In [324]:

```
#Show groups with different colors using "hue" and "palette"
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , hue= employment.Gender , pale
plt.show()
```

In [325]:

```
# Overlap the violinplots for different hue levels along the categorical axis using "dodge=
plt.figure(figsize=(18,9))
sns.violinplot(x=employment.Age , y = employment.Unemployed , hue= employment.Gender , pale
plt.show()
```

In [326]:

```
# Drawing split violins to compare across the hue variable using -> (split=True)
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , hue= employment.Gender , pale
plt.show()
```

In [327]:

```
plt.figure(figsize=(16,7))
sns.violinplot(x=employment.Age , y = employment.Unemployed , hue= employment.Gender ,
               palette="Set2" , split=True , inner="quartile")
plt.show()
```

In [328]:

```
#Showing each observation with a stick inside the violin using -> (inner="stick")
plt.figure(figsize=(16,11))
sns.violinplot(x=employment.Age , y = employment.Unemployed , hue= employment.Gender ,
               palette="Set2" , split=True , inner="stick")
plt.show()
```

In [329]:

```
# Displaying multiple violin plots using subplot function.
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (20,20))
sns.violinplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris , inner=None)
sns.violinplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris , inner=None)
sns.violinplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris, inner=None)
sns.violinplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris, inner=None )
plt.show()
```

In [330]:

```
# Displaying swarmplot on top of violin plot
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (20,20))
sns.swarmplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris , size=6 , color=
sns.violinplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris , inner=None)
sns.swarmplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris  , size=6 , colo
sns.violinplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris , inner=None)
sns.swarmplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris , size=6 , color
sns.violinplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris, inner=None)
sns.swarmplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris , size=6 , colo
sns.violinplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris, inner=None )
plt.show()
```

In [781]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(11,9))
sns.catplot(x="Age" , y = "Unemployed", col="Gender", kind="violin",palette="Set2" , height
            scale = "count",data=employment)
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

In [332]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(11,9))
sns.catplot(x="parental level of education" , y = "math score", hue= "test preparation cour
            col="lunch", kind="violin",palette="Set3" , height=8, aspect=1 ,data=stdperf ,
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

Strip Plot

Strip plot is a scatter plot where one of the variables is categorical.

In [787]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("white")
```

In [788]:

```
plt.figure(figsize=(11,8))
sns.stripplot(x=employment.Unemployed)
plt.show()
```

In [789]:

```
plt.figure(figsize=(16,8))
sns.stripplot(x=employment.Age ,palette="Set1", y = employment.Unemployed)
plt.show()
```

In [805]:

```
""" "Jitter" parameter signifies the amount of jitter to apply.
   This can be extremely useful when we have large clusters of data points"""

plt.figure(figsize=(16,5))
sns.stripplot(x=employment.Age ,palette="Set1", y = employment.Unemployed , jitter=.3)
plt.show()
```

In [303]:

```
# Flip x and y inputs to make a horizontal strip plot
plt.figure(figsize=(11,8))
sns.stripplot(y=employment.Age ,palette="Set1", x = employment.Unemployed , jitter=True)
plt.show()
```

In [304]:

```
plt.figure(figsize=(16,8))
sns.stripplot(x=employment.Age ,palette="Set1", y = employment.Unemployed , linewidth=1)
plt.show()
```

In [305]:

```
# Adjust the linewidth of the edges of the circles using "linewidth" parameter
# Adjust the size of the circles using the "size" parameter
plt.figure(figsize=(16,10))
sns.stripplot(x=employment.Age ,palette="Set1", y = employment.Unemployed , linewidth=2, si
plt.show()
```

In [306]:

```
# Using set2 pallete
plt.figure(figsize=(16,10))
sns.stripplot(x=employment.Age ,palette="Set2", y = employment.Unemployed , hue=employment.
plt.show()
```

In [307]:

```
# Separate the strips for different hue levels along the categorical axis using "dodge=True
plt.figure(figsize=(16,10))
sns.stripplot(x=employment.Age ,palette="Set2", y = employment.Unemployed , hue=employment.
plt.show()
```

In [308]:

```
plt.figure(figsize=(16,10))
sns.stripplot(x=employment.Age ,palette="Set2", y = employment.Unemployed ,
              hue=employment.Gender, marker = "o" , size = 30 , alpha = .15)
plt.show()
```

In [309]:

```
# Drawing stripplot on top of a box plot
plt.figure(figsize=(11,8))
sns.stripplot(x=employment.Age, y = employment.Unemployed , jitter=True , color="black" , s
sns.boxplot(x=employment.Age ,palette="Set1", y = employment.Unemployed , color='black')
plt.show()
```

In [483]:

```
plt.figure(figsize=(10,9))
sns.set(rc={"axes.facecolor":"#b0deff","axes.grid":False,
            'xtick.labelsize':15,'ytick.labelsize':15,
            'axes.labelsize':20,'figure.figsize':(20.0, 9.0)})
params = dict(data=exercise ,x = exercise.kind ,y = exercise.pulse ,hue=exercise.diet,dodge
sns.stripplot(**params , size=9,jitter=0.35,palette=['#FF8F00','#558B2F'],edgecolor='black'
sns.boxplot(**params ,palette=['#909497','#D0D3D4'],linewidth=4)
plt.show()
```

In [384]:

```
plt.figure(figsize=(16,11))
sns.set(rc={"axes.facecolor":"#b0deff","axes.grid":False,
            'xtick.labelsize':15,'ytick.labelsize':15,
            'axes.labelsize':20,'figure.figsize':(20.0, 9.0)})
params = dict(data=employment ,x = employment.Age ,y = employment.Unemployed ,hue=employmer
sns.stripplot(**params , size=8,jitter=0.35,palette=['#33FF66','#FF6600'],edgecolor='black'
sns.boxplot(**params ,palette=['#BDBDBD','#E0E0E0'],linewidth=6)
plt.show()
```

In [313]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("white")
```

In [314]:

```
# Drawing stripplot on top of a violin plot
plt.figure(figsize=(14,10))
sns.stripplot(x=employment.Age, y = employment.Unemployed , jitter=True , color="black" , s
sns.violinplot(x=employment.Age , y = employment.Unemployed , palette="Set1" , scale="count
plt.show()
```

In [315]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(11,9))
sns.catplot(x="Age" , y = "Unemployed", col="Gender", kind="strip",palette="Set1" , height=
            data=employment)
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

In [316]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(11,9))
sns.catplot(x="parental level of education" , y = "math score", hue= "test preparation cour
            col="lunch", kind="strip",palette="Set2" , height=8, aspect=1 ,data=stdperf , s
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

Box Plot

The box plot is a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum.

In [840]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("white")
```

In [841]:

```
insurance.head()
```

Out[841]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [842]:

```
# Simple boxplot
plt.figure(figsize=(8,6))
sns.boxplot(insurance.bmi)
```

Out[842]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd712cda0>
```

In [848]:

```
# Vertical boxplot
sns.set_style("darkgrid")
plt.figure(figsize=(8,6))
sns.boxplot(x= insurance.smoker , y= insurance.charges )
```

Out[848]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd8b64630>
```

In [849]:

```
#Explicit ordering using "order" parameter
plt.figure(figsize=(8,6))
sns.boxplot(x= insurance.smoker , y= insurance.charges , order = ['no' , 'yes'])
```

Out[849]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd8dfe080>
```

In [850]:

```
helpdesk.head(5)
```

Out[850]:

	ticket	requestor	RequestorSeniority	ITOwner	FiledAgainst	TicketType	Severity	Priority
0	1	1929	1 - Junior	50	Systems	Issue	2 - Normal	0 Unassigned
1	2	1587	2 - Regular	15	Software	Request	1 - Minor	1 - Low
2	3	925	2 - Regular	15	Access/Login	Request	2 - Normal	0 Unassigned
3	4	413	4 - Management	22	Systems	Request	2 - Normal	0 Unassigned
4	5	318	1 - Junior	22	Access/Login	Request	2 - Normal	1 - Low

In [851]:

```
plt.figure(figsize=(7,6))
sns.boxplot(helpdesk.daysOpen )
plt.show()
```

In [852]:

```
plt.figure(figsize=(8,6))
sns.boxplot(x= helpdesk.FiledAgainst , y= helpdesk.daysOpen )
```

Out[852]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd71db668>
```

In [853]:

```
plt.figure(figsize=(8,6))
sns.boxplot(x= fish.Species , y= fish.Weight)
```

Out[853]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd57ea208>
```

In [854]:

```
stdperf = pd.read_csv("studentp.csv")
stdperf.head(10)
```

Out[854]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
5	female	group B	associate's degree	standard	none	71	83	78
6	female	group B	some college	standard	completed	88	95	92
7	male	group B	some college	free/reduced	none	40	43	39
8	male	group D	high school	free/reduced	completed	64	64	67
9	female	group B	high school	free/reduced	none	38	60	50

In [855]:

```
# Adjust width of boxes
plt.subplots(figsize = (14,24))
plt.subplot(3,1,1)

sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,  width=.

plt.subplot(3,1,2)
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['reading score'] ,  widt

plt.subplot(3,1,3)
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['writing score'] ,  widt

plt.show()
```

100
80
60
40
20
0
math score
bachelor's degree
some college
master's degree
associate's degree
high school
some high school
parental level of education

100
80
60
40
20
reading score
bachelor's degree
some college
master's degree
associate's degree
high school
some high school
parental level of education

100
80
60
40
20
writing score
bachelor's degree
some college
master's degree
associate's degree
high school
some high school
parental level of education

In [856]:

```
plt.figure(figsize = (6,7))
sns.boxplot(x= exercise.kind , y= exercise.pulse)
plt.show()
```

In [857]:

```
# Show groups with different colors using "hue"
plt.figure(figsize = (6,7))
sns.boxplot(x= exercise.kind , y= exercise.pulse , hue=exercise.diet)
plt.show()
```

In [862]:

```
plt.figure(figsize = (14,8))
plt.text(1.5,105, "Box Plot", fontsize = 50, color='Black')
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'])
plt.show()
```

In [863]:

```
sns.set_style("white")
```

In [864]:

```
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] , palette= 'rainbow')
plt.show()
```

In [865]:

```
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} )
plt.show()
```

In [866]:

```
# Remove the top and right axis spines
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} )
sns.despine()
# More about sns.despine() here - https://seaborn.pydata.org/tutorial/aesthetics.html
plt.show()
```

In [867]:

```
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} )
sns.despine(left=True)
plt.show()
```

In [807]:

```
# Changing Linewidth
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} ,linewidth = 3)
sns.despine(left=True)
plt.show()
```

In [869]:

```
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} ,linewidth = 2)
sns.despine(left=True)
plt.xticks(rotation=45)
plt.show()
```

In [870]:

```
# Add a notch to the box
plt.figure(figsize = (14,6))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} ,linewidth = 2 , notch=True
sns.despine(left=True)
plt.xticks(rotation=45)
plt.show()
```

In [871]:

```
# Change the size of outlier markers
plt.figure(figsize = (16,8))
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} ,linewidth = 2  , fliersize
sns.despine(left=True)
plt.xticks(rotation=45)
plt.show()
```

In [872]:

```
# Change the size of outlier markers
sns.set(rc={"axes.facecolor":"#b0deff","axes.grid":False,
            'xtick.labelsize':15,'ytick.labelsize':15,
            'axes.labelsize':20,'figure.figsize':(20.0, 9.0)})
sns.boxplot(x= stdperf['parental level of education'] , y= stdperf['math score'] ,
            width=.7 , hue=stdperf['test preparation course'] ,
            palette= {"none":'#FFB74D' , "completed":'#9CCC65'} ,linewidth = 2  , fliersize
sns.despine(left=True)
plt.xticks(rotation=45)
plt.show()
```

In [873]:

```
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [874]:

```
plt.figure(figsize=(14,11))
sns.set(rc={"axes.grid":False,
            'xtick.labelsize':15,'ytick.labelsize':15,
            'axes.labelsize':20,'figure.figsize':(20.0, 9.0)})
params = dict(data=employment ,x = employment.Age ,y = employment.Unemployed)
sns.stripplot(**params , size=8,jitter=0.35,palette=['#33FF66','#FF6600'],edgecolor='black'
sns.boxplot(**params ,palette=['#BDBDBD','#E0E0E0'],linewidth=6)
plt.show()
```

In [875]:

```
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("darkgrid")
```

In [876]:

```
exercise.head()
```

Out[876]:

	id	diet	pulse	time	kind
0	1	low fat	85	1 min	rest
1	1	low fat	85	15 min	rest
2	1	low fat	88	30 min	rest
3	2	low fat	90	1 min	rest
4	2	low fat	92	15 min	rest

In [877]:

```
plt.figure(figsize=(18,16))
sns.catplot(x="kind", y="pulse",col="diet",data=exercise, kind="box");
plt.show()
```

```
<Figure size 1296x1152 with 0 Axes>
```

In [878]:

```
# Displaying multiple violin plots using subplot function.
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (20,24))
sns.swarmplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris , color=".10" , s
sns.boxplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris )
sns.swarmplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris  ,  color=".10",
sns.boxplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris)
sns.swarmplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris ,  color=".10",
sns.boxplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris)
sns.swarmplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris ,  color=".10",
sns.boxplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris)
plt.show()
```

In [879]:

```
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (20,24))
sns.swarmplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris, size=8,)
sns.boxplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris ,boxprops={'facecol
sns.swarmplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris  , size=8)
sns.boxplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris ,boxprops={'facecc
sns.swarmplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris ,  size=8)
sns.boxplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris ,boxprops={'facecc
sns.swarmplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris, size=8)
sns.boxplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris ,boxprops={'facec
plt.show()
```

In [880]:

```
fig1 , axes = plt.subplots(nrows=2,ncols=2 , figsize = (20,24))
sns.swarmplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris, size=8)
sns.boxplot(x="species" , y = "petal_width" , ax = axes[0,0] ,data=iris , showfliers=False,
sns.swarmplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris, size=8)
sns.boxplot(x="species" , y = "petal_length" ,ax = axes[0,1] , data=iris , showfliers=False
sns.swarmplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris, size=8)
sns.boxplot(x="species" , y = "sepal_width" , ax = axes[1,0] , data=iris , showfliers=False
sns.swarmplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris , size=8)
sns.boxplot(x="species" , y = "sepal_length" , ax = axes[1,1] , data=iris , showfliers=Fals
plt.show()
```

Boxen Plot

The Boxen Plot shows a large number of quantiles.By plotting more quantiles, it provides more information about the shape of the distribution, particularly in the tails.

In [881]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set_style("darkgrid")
```

In [882]:

```
# Simple Boxen Plot
plt.figure(figsize=(5,7))
sns.boxenplot(y=employment.Unemployed)
plt.show()
```

In [883]:

```
plt.figure(figsize=(7,9))
sns.boxenplot(x=employment.Gender , y = employment.Unemployed ,palette="Set1")
plt.show()
```

In [884]:

```
# Drawing a vertical boxenplot grouped by a categorical variable
plt.figure(figsize=(10,9))
sns.boxenplot(x=employment.Age , y = employment.Unemployed ,palette="Set1")
plt.show()
```

In [886]:

```
# Using spring palette
plt.figure(figsize=(11,9))
sns.boxenplot(x=employment.Age , y = employment.Unemployed,palette="spring")
plt.show()
```

In [888]:

```
sns.set(rc={"axes.facecolor":"#616161" , "axes.grid" : False})
plt.figure(figsize=(11,9))
sns.boxenplot(x=employment.Age , y = employment.Unemployed,palette="Set1")
plt.show()
```

In [939]:

```
sns.set(rc={"axes.facecolor":"#a1c45a" , "axes.grid" : False})
plt.figure(figsize=(11,9))
plt.gcf().text(.51, .84, "Box Plot", fontsize = 40, color='Black' ,ha='center', va='center'
sns.boxenplot(x=employment.Age , y = employment.Unemployed,palette="Set1")
plt.show()
```

In [940]:

```
# Horizontal Boxen plot
sns.set_style("darkgrid")
plt.figure(figsize=(10,9))
sns.boxenplot(x = employment.Unemployed ,y=employment.Age ,palette="Set1")
plt.show()
```

In [801]:

```
# Show groups with different colors using "hue" (Nested grouping by two categorical variabl
plt.figure(figsize=(11,9))
sns.boxenplot(x=employment.Age , y = employment.Unemployed, hue=employment.Gender, palette=
plt.show()
```

In [802]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(11,9))
sns.catplot(x="Age" , y = "Unemployed", col="Gender", kind="boxen",palette="Set2" , height=
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

In [803]:

```
stdperf.head()
```

Out[803]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [804]:

```
# Facet along the columns to show a categorical variable using "col" parameter
plt.figure(figsize=(11,9))
sns.catplot(x="parental level of education" , y = "math score", hue= "test preparation cour
            col="lunch", kind="boxen",palette="Set2" , height=8, aspect=1 ,data=stdperf)
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

Pair Plot

Pair Plot is used for plotting pairwise relationships in a dataset.

In [33]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [34]:

```
fish.head()
```

Out[34]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [35]:

```
fish1 = fish[fish['Species'].isin(['Bream', 'Perch', 'Pike'])]
fish1.head()
```

Out[35]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [39]:

```
# Draw scatterplots for joint relationships and histograms for univariate distributions
plt.figure(figsize=(11,9))
sns.pairplot(fish1,hue = 'Species')
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

In [40]:

```
# Show groups with different colors using "hue"
plt.figure(figsize=(11,9))
sns.pairplot(fish1,hue = 'Species')
plt.show()
```

```
<Figure size 792x648 with 0 Axes>
```

In [37]:

```
# Changing palettes
sns.pairplot(fish1,hue = 'Species',palette="husl",size=2)
plt.show()
```

In [97]:

```
# Plot a subset of variables
sns.pairplot(fish1,hue = 'Species',vars=["Length1", "Weight" , "Height" , "Width"] , height
plt.show()
```

In [98]:

```
# Use different markers for each level of the hue variable
sns.pairplot(fish1,hue = 'Species',vars=["Length1", "Weight" , "Height" , "Width"] ,
             markers= ['o' , 'D' , '<'] , height=3, aspect=1)
plt.show()
```

In [99]:

```
sns.pairplot(fish1,hue = 'Species',vars=["Weight" , "Height" , "Width"] ,
             markers= ['o' , 'D' , '<'] , height=5, aspect=1)
plt.show()
```

In [100]:

```
sns.pairplot(fish1,hue = 'Species',x_vars=["Weight" , "Height" , "Width"] , y_vars=["Length
             markers= ['o' , 'D' , '<'] , height=5, aspect=1)
plt.show()
```

In [103]:

```
# Fit linear regression models to the scatter plots
sns.pairplot(fish1,hue = 'Species',vars=["Weight" , "Height" , "Width"] , kind="reg",
             height=5, aspect=1)
plt.show()
```

In [104]:

```
# Changing size of circles in scatter plots using -> plot_kws=dict(s=140)
sns.pairplot(fish1,hue = 'Species',vars=["Weight" , "Height" , "Width"] , plot_kws=dict(s=1
             height=5, aspect=1)
plt.show()
```

In [105]:

```
sns.pairplot(fish1,hue = 'Species',vars=["Weight" , "Height" , "Width"] , plot_kws=dict(s=1
             height=5, aspect=1)
plt.show()
```

In [77]:

```
sns.pairplot(fish1,hue = 'Species',x_vars=["Weight" , "Height"] ,
             y_vars=["Weight" , "Height"] ,plot_kws=dict(s=140, linewidth=1,alpha= .6),
             height=5, aspect=1)
plt.show()
```

Pair Grid

Pair Grid is a Subplot grid for plotting pairwise relationships in a dataset. Different axes-level plotting functions can be used to draw bivariate plots in the upper and lower triangles, and the the marginal distribution of each variable can be shown on the diagonal.

In [412]:

```
#scatterplot for each pairwise relationship
g = sns.PairGrid(fish1)
g = g.map(plt.scatter)
plt.show()
```

In [413]:

```
# Histogram on the diagonal
g = sns.PairGrid(fish1)
g = g.map_offdiag(plt.scatter)
g = g.map_diag(plt.hist)
plt.show()
```

In [414]:

```
# Show groups with different colors using "hue"
g = sns.PairGrid(fish1 , hue='Species')
g = g.map_offdiag(plt.scatter)
g = g.map_diag(plt.hist)
g = g.add_legend()
plt.show()
```

In [415]:

```
g = sns.PairGrid(fish1 , hue='Species', palette="husl")
g = g.map_offdiag(plt.scatter)
g = g.map_diag(plt.hist)
g = g.add_legend()
plt.show()
```

In [416]:

```
# Changing histogram styling
g = sns.PairGrid(fish1 , hue='Species')
g = g.map_offdiag(plt.scatter)
g = g.map_diag(plt.hist , histtype="step", linewidth=2)
g = g.add_legend()
plt.show()
```

In [417]:

```
# Plot a subset of variables
g = sns.PairGrid(fish1 , hue='Species' ,vars=["Weight" , "Height" , "Width"],height=5, aspe
g = g.map_offdiag(plt.scatter , edgecolor="w", s=130)
g = g.map_diag(plt.hist , edgecolor ='w', linewidth=2)
g = g.add_legend()
plt.show()
```

In [418]:

```
# Plot a subset of variables
g = sns.PairGrid(fish1 , hue='Species' ,x_vars=["Weight" , "Height"],y_vars=["Weight" , "He
                 height=6, aspect=1)
g = g.map_offdiag(plt.scatter , edgecolor="w", s=130)
g = g.map_diag(plt.hist , edgecolor ='w', linewidth=2)
g = g.add_legend()
plt.show()
```

In [419]:

```
g = sns.PairGrid(fish1 , hue='Species' ,x_vars=["Weight" , "Height"],y_vars=["Weight" , "He
                 height=6, aspect=1 , hue_kws={"marker": ["o", "^", "v"]})
g = g.map(plt.scatter , edgecolor="w", s=130)
g = g.add_legend()
plt.show()
```

In [420]:

```
# Using different plots on the upper and lower triangles
g = sns.PairGrid(fish1 ,vars=["Length1" , "Height" , "Width"],height=5, aspect=1)
g = g.map_upper(sns.scatterplot , edgecolor="w", s=130)
g = g.map_lower(sns.kdeplot)
g = g.map_diag(sns.kdeplot , lw= 2)
g = g.add_legend()
plt.show()
```

Regression Plot

Regression plot is used to plot data and a linear regression model fit.

In [421]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [422]:

```
fish1.head()
```

Out[422]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [423]:

```
# Plot the relationship between two variables
plt.figure(figsize=(7,7))
sns.regplot(x=fish1.Weight , y=fish1.Height)
plt.show()
```

In [424]:

```
# Plot the relationship between two variables and use a different color
plt.figure(figsize=(9,9))
sns.regplot(x=fish1.Weight , y=fish1.Height , color='#FF6600')
plt.show()
```

In [425]:

```
plt.figure(figsize=(9,9))
sns.regplot(x=fish1.Weight , y=fish1.Height , color='#FF6600' , marker='>')
plt.show()
```

In [426]:

```
# Use 50% Confidnce interval
plt.figure(figsize=(9,9))
sns.regplot(x=fish1.Weight , y=fish1.Height , color='#FF6600' , ci=50 , truncate=False)
plt.show()
```

In [427]:

```
#Fit the regression model using log(x)
plt.figure(figsize=(9,9))
sns.regplot(x=fish1.Weight , y=fish1.Height , color='#FF6600' , logx=True)
plt.show()
```

In [428]:

```
# Change the color and width of regression line -> line_kws={"color":"#FF5722","lw":3}
plt.figure(figsize=(9,9))
sns.regplot(x=fish1.Weight , y=fish1.Height , logx=True , line_kws={"color":"#FF5722","alph
plt.show()
```

In [429]:

```
#Fit a regression model for a categorical variable
sns.lmplot(x="bmi" , y="charges" , hue="smoker" , data=insurance , height=8,aspect=1.2)
plt.show()
```

In [430]:

```
sns.lmplot(x="bmi" , y="charges" , hue="smoker" , data=insurance , height=8,aspect=1.2 ,
           scatter_kws ={'s':90 , 'alpha' : .4} ,  palette=["#FF3333" ,"#00CC00"])
plt.show()
```

In [431]:

```
insurance.head()
```

Out[431]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [432]:

```
# Plot the levels of the third variable across different columns
sns.lmplot(x="bmi", y="charges", hue="smoker", col="sex", data=insurance ,height=6,aspect=1
```

Out[432]:

```
<seaborn.axisgrid.FacetGrid at 0x22eade47048>
```

In [433]:

```
""" Facet along the columns to show a categorical variable using "col" parameter and
Limiting the number of columns using "col_wrap" """

sns.lmplot(x="bmi", y="charges", hue="smoker", col="region", data=insurance ,height=7,aspec
```

Out[433]:

```
<seaborn.axisgrid.FacetGrid at 0x22eadd524e0>
```

In [434]:

```
# Facet along the columns and rows to show a categorical variables using "col" and "row" pa
sns.lmplot(x="bmi", y="charges", row="smoker", col="region", data=insurance ,height=4,aspec
```

Out[434]:

```
<seaborn.axisgrid.FacetGrid at 0x22ebb75b320>
```

Heat Map

A heat map is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.

In [435]:

```
canada1 = canada.loc['2004': , ['Germany' , 'France' , 'Mexico' , 'Bangladesh' ,'Brazil','I
canada1.head()
```

Out[435]:

	Germany	France	Mexico	Bangladesh	Brazil	Israel	Egypt	Colombia	Iraq	Jamaica
2004	2020	4391	2259	2660	917	2788	2393	4566	1796	2237
2005	2226	4429	2837	4171	969	2446	2496	6424	2226	1945
2006	2767	4002	2844	4014	1181	2625	2190	6535	1788	1722
2007	2449	4290	3239	2897	1746	2401	2356	5357	2406	2141
2008	3833	4532	2856	2939	2138	2562	3347	5452	3543	2334

In [436]:

```
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1)
plt.yticks(rotation=0)
plt.show()
```

In [437]:

```
# Changing the limits of the colormap
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,vmin=0, vmax=8000)
plt.yticks(rotation=0)
plt.show()
```

In [438]:

```
# Plot a heatmap for data centered on 4000 with a diverging colormap
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,center=4000)
plt.yticks(rotation=0)
plt.show()
```

In [439]:

```
# Changing cmap
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="YlGnBu")
plt.yticks(rotation=0)
plt.show()
```

In [440]:

```
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="coolwarm")
plt.yticks(rotation=0)
plt.show()
```

In [441]:

```
# Add lines between each cell
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,linewidths=.1)
plt.yticks(rotation=0)
plt.show()
```

In [442]:

```
# Annotate each cell with the numeric value using integer formatting
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,annot=True, fmt="d")
plt.yticks(rotation=0)
plt.show()
```

In [443]:

```
# Annotate each cell with the numeric value using decimal formatting
plt.figure(figsize=(16,9))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="YlGnBu", annot=True ,fmt=".2f")
plt.yticks(rotation=0)
plt.show()
```

In [444]:

```
plt.figure(figsize=(20,11))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="YlGnBu", linewidths=.5, annot=True ,annot_
plt.yticks(rotation=0)
plt.show()
```

In [445]:

```
# Turn off the colorbar -> cbar=False
plt.figure(figsize=(20,11))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="YlGnBu", linewidths=.5,
                 annot=True ,annot_kws={'size':14} ,fmt=".1f" , cbar=False)
plt.yticks(rotation=0)
plt.show()
```

In [446]:

```
# Force the aspect ratio of the blocks to be equal using "square" parameter
plt.figure(figsize=(20,11))
ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="YlGnBu", linewidths=.5,
                 annot=True ,annot_kws={'size':14} ,fmt=".1f" , cbar=False ,square = True)
plt.yticks(rotation=0)
plt.show()
```

In [447]:

```
sns.set(rc={'xtick.labelsize':17,'ytick.labelsize':17,'axes.labelsize':20})
```

In [448]:

```
plt.figure(figsize=(20,12))
plt.rcParams['figure.facecolor'] = "#a1c45a"
plt.rcParams['axes.facecolor'] = "#a1c45a"
plt.rcParams[ 'axes.labelsize'] = 20
plt.rcParams['xtick.labelsize'] = 15
plt.rcParams['ytick.labelsize'] = 15

ax = sns.heatmap(canada1,vmin=0, vmax=8000,cmap="YlGnBu", linewidths=.5,annot=True ,annot_k
plt.yticks(rotation=0)
plt.text(4,-.2, "Heat Map", fontsize = 50, color='Black')
plt.show()
```

Facet Grid

Multi-plot grid for plotting. A FacetGrid can be drawn with up to three dimensions − row, col, and hue.

In [477]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
plt.rcParams[ 'axes.labelsize'] = 10
plt.rcParams['xtick.labelsize'] = 10
plt.rcParams['ytick.labelsize'] = 10
```

In [478]:

```
insurance.head()
```

Out[478]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [479]:

```
# Facet along the columns to show a categorical variables using "col" parameter (univariate
g = sns.FacetGrid(insurance, col="sex")
g = g.map(plt.hist, "bmi", edgecolor ='w', linewidth=2)
```

In [480]:

```
# Facet along the columns and rows to show a categorical variables using "col" and "row" pa
g = sns.FacetGrid(insurance, col="sex" , row = "region" , height=4, aspect=1)
g = g.map(plt.hist, "bmi",edgecolor ='w', linewidth=1)
```

In [484]:

```
g = sns.FacetGrid(insurance, col="region" , row = "sex" , height=3, aspect=1)
g = g.map(plt.hist, "bmi",edgecolor ='w', linewidth=1)
```

In [482]:

```
# Changing color of bars
g = sns.FacetGrid(insurance, col="region" , row = "sex" , height=3, aspect=1)
g = g.map(plt.hist, "bmi" , color = '#F39C12',edgecolor ='w', linewidth=1)
```

In [485]:

```
# Bivariate function on each facet
g = sns.FacetGrid(insurance, col="region" , row = "sex" , height=3, aspect=1)
g = g.map(plt.scatter, "bmi" , "charges")
```

In [490]:

```
# Changing size of dots -> s = 80
g = sns.FacetGrid(insurance, col="region" , row = "sex" , height=3, aspect=1)
g = g.map(plt.scatter, "bmi" , "charges" , color = '#FF6600',edgecolor="w", s=80)
```

In [492]:

```
# Show groups with different colors using "hue"
g = sns.FacetGrid(insurance, col="region" , row = "sex" , hue="smoker" ,height=3, aspect=1)
g = g.map(plt.scatter, "bmi" , "charges" , edgecolor="w", s=70)
```

In [493]:

```
# Showing legend -> g.add_legend()
g = sns.FacetGrid(insurance, col="region" , row = "sex" , hue="smoker" ,height=3, aspect=.9
g = g.map(plt.scatter, "bmi" , "charges",edgecolor="w", s=70)
g.add_legend()
```

Out[493]:

```
<seaborn.axisgrid.FacetGrid at 0x22ebb142358>
```

In [495]:

```
# Use a custom palette
pal1 = dict(yes="#ff7315", no="#8cba51")
g = sns.FacetGrid(insurance, col="region" , row = "sex" , hue="smoker" ,height=3, aspect=1,
g = g.map(plt.scatter, "bmi" , "charges",edgecolor="w", s=70)
g.add_legend()
plt.show()
```

In [498]:

```
pal1 = dict(yes="#ff7315", no="#8cba51")
kws = dict(s=80, linewidth=1, edgecolor="w")
kws1 =  dict(height=3, aspect=1,palette=pal1)
g = sns.FacetGrid(insurance, col="region" , row = "sex" , hue="smoker" , **kws1)
g = g.map(plt.scatter, "bmi" , "charges",**kws)
g.add_legend()
plt.show()
```

In [500]:

```
# Use different markers for each level of the hue variable
pal1 = dict(yes="#ff7315", no="#8cba51")
kws = dict(s=70, linewidth=1, edgecolor="w")
kws1 =  dict(height=3, aspect=1,palette=pal1 , hue_kws=dict(marker=["^", "v"]))
g = sns.FacetGrid(insurance, col="region" , row = "sex" , hue="smoker" , **kws1)
g = g.map(plt.scatter, "bmi" , "charges",**kws)
g.add_legend()
plt.show()
```

In [502]:

```
g = sns.FacetGrid(insurance, col="region" , row = "sex" ,height=3, aspect=1)
g = g.map(plt.bar, "smoker" , "charges",edgecolor="w" , color = '#8cba51')
g.add_legend()
```

Out[502]:

```
<seaborn.axisgrid.FacetGrid at 0x22ebf1605f8>
```

In [503]:

```
pokemon.head()
```

Out[503]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legenc
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	F
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	F
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	F
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	F
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	F

In [510]:

```
# Limiting the number of columns using "col_wrap"
g = sns.FacetGrid(pokemon, col="Type 2" , col_wrap=3 , height=2.8, aspect=1)
g = g.map(plt.scatter, "Attack" , "Sp. Atk",color = '#FF6600',edgecolor="w", s=70)
```

Joint Plot

Joint plot is used to draw a plot of two variables with bivariate graph and univariate graphs in the margin.

In [627]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
```

In [628]:

```
insurance.head()
```

Out[628]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [629]:

```
# scatterplot with marginal histograms
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10)
```

Out[629]:

```
<seaborn.axisgrid.JointGrid at 0x22ec4a64898>
```

In [630]:

```
# Regplot with marginal distplot
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="reg")
```

Out[630]:

```
<seaborn.axisgrid.JointGrid at 0x22ec4b755f8>
```

In [631]:

```
sns.set(style="white", color_codes=True)
```

In [632]:

```
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10)
```

Out[632]:

```
<seaborn.axisgrid.JointGrid at 0x22ec4dbc9e8>
```

In [633]:

```
# Resid Plot
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="resid")
```

Out[633]:

```
<seaborn.axisgrid.JointGrid at 0x22ec4edc780>
```

In [561]:

```
# "ratio" adjusts the relative size of the marginal plots
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10, ratio=2)
```

Out[561]:

```
<seaborn.axisgrid.JointGrid at 0x22ebbe178d0>
```

In [517]:

```
# Change the color of the joint plot
g = sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , color = '#FF6600',ec
```

In [518]:

```
plt.rcParams['figure.facecolor'] = "#E5E7E9"
plt.rcParams['axes.facecolor'] = "#E5E7E9"
plt.rcParams[ 'axes.labelsize'] = 20
plt.rcParams['xtick.labelsize'] = 15
plt.rcParams['ytick.labelsize'] = 15
g = sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , color = '#FF6600',ec
```

In [521]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
sns.set(style="white", color_codes=True)
```

In [541]:

```
# Replace the scatterplot with a joint histogram using hexagonal bins
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="hex")
```

Out[541]:

```
<seaborn.axisgrid.JointGrid at 0x22ec2387d68>
```

In [542]:

```
# Replace the scatterplot with a joint histogram using hexagonal bins
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="hex" , color="#fb9
```

Out[542]:

```
<seaborn.axisgrid.JointGrid at 0x22ec2728828>
```

In [540]:

```
""" Replace the scatterplot & joint histogram with kde plot
     in the margins and the interior into a shaded countour plot  """

sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" , color="#FF6
```

Out[540]:

```
<seaborn.axisgrid.JointGrid at 0x22ec2454a90>
```

In [580]:

```
# Create Space between the joint and marginal axes
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" , color="#FF6
```

Out[580]:

```
<seaborn.axisgrid.JointGrid at 0x22ec0202048>
```

In [609]:

```
# Set Axis Limits
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" ,
              color="#FF6600", space=2, xlim=(0,50),ylim=(-3000,70000))
```

Out[609]:

```
<seaborn.axisgrid.JointGrid at 0x22ebfdb1358>
```

In [589]:

```
# Change formatting of marginal graphs
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" ,
              color="#FF6600" , marginal_kws={'lw':5})
```

Out[589]:

```
<seaborn.axisgrid.JointGrid at 0x22ec0e29358>
```

In [613]:

```
# Change formatting of marginal graphs
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" ,
              color="#33CC00" , marginal_kws={'lw':5 , 'color' : "#FF3300"})
```

Out[613]:

```
<seaborn.axisgrid.JointGrid at 0x22ec2b800b8>
```

In [615]:

```
sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" ,
              color="#33CC00" , marginal_kws={'lw':5 , 'color' : "#FF3300" , 'bw' :.3})
```

Out[615]:

```
<seaborn.axisgrid.JointGrid at 0x22ec2cbcb38>
```

In [624]:

```
# Changing background of marginal graphs
g = sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="kde" ,
              color="#33CC00" , marginal_kws={'lw':5 , 'color' : "#FF3300"})
g.ax_marg_x.set_facecolor('#f8fab8')
g.ax_marg_y.set_facecolor('#f8fab8')
```

In [539]:

```
# Defining Axis Labels
(sns.jointplot(x="bmi", y="charges", data=insurance , height = 10 , kind="hex" , color="#a0
     .set_axis_labels("$BMI$", "$CHARGES$"))
```

Out[539]:

```
<seaborn.axisgrid.JointGrid at 0x22ebb1e42b0>
```

In [538]:

```
sns.jointplot("bmi", "charges", data=insurance,  s=70, edgecolor="w", linewidth=1, height =
              marginal_kws=dict(bins=15, rug=True))
plt.show()
```

Joint Grid

Grid for drawing a bivariate plot with marginal univariate plots.

In [639]:

```
#Add plots using default parameters
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 10)
g = g.plot(sns.regplot, sns.distplot)
```

In [650]:

```
# Changing color of plots
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 10)
g = g.plot_joint(sns.scatterplot, color="#FF6600",s=50)
g = g.plot_marginals(sns.distplot, color="#FF6600")
```

In [646]:

```
# Remove kernel density estimate from the marginal plots
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 10)
g = g.plot_joint(sns.scatterplot, color="#FF6600")
g = g.plot_marginals(sns.distplot,kde= False, color="#FF6600")
```

In [808]:

```
# kde plot in the margins and the interior into a shaded countour plot
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 8)
g = g.plot_joint(sns.kdeplot)
g = g.plot_marginals(sns.kdeplot, shade=True)
```

In [809]:

```
# Remove the space between the joint and marginal axes
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 8 , space =0)
g = g.plot_joint(sns.kdeplot)
g = g.plot_marginals(sns.kdeplot, shade=True)
```

In [658]:

```
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 8)
g = g.plot_joint(sns.kdeplot , cmap="OrRd_r")
g = g.plot_marginals(sns.kdeplot, shade=True , color = 'orange')
```

In [665]:

```
# Set Axis Limits
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 10 , xlim=(10,50) , ylim=(
g = g.plot_joint(sns.kdeplot , cmap="OrRd_r")
g = g.plot_marginals(sns.kdeplot, shade=True , color = 'orange')
```

In [673]:

```
# Changing background of marginal graphs
g = sns.JointGrid(x="bmi", y="charges", data=insurance, height = 10 , xlim=(10,50) , ylim=(
g = g.plot_joint(sns.scatterplot, color="#FF6600",s=50)
g= g.plot_marginals(sns.distplot,kde=False , color = 'Blue')
g.ax_marg_x.set_facecolor('#f8fab8')
g.ax_marg_y.set_facecolor('#f8fab8')
```

END