Name: ISHAN CHASKAR URK21CS1181

Aim: To obtain statistical inference from the csv file, diamonds.csv

Description: In data science, statistical inference empowers data professionals to make informed decisions, validate models, and extract valuable insights from data while quantifying uncertainty. It plays a foundational role in the entire data science pipeline, from data exploration and preprocessing to modeling and decision-making. Population: The entire group you want to draw conclusions about Sample: A specific group you want to draw conclusions about. The size of a sample will always be less than the size o the entire population. Mean: It is important to contrast the population mean (the estimand) with the sample mean (the estimator). The sample mean estimates the population mean. Not coincidentally, since the population mean is the center of mass of the population distribution, the sample mean is the center of mass of the data. Confidence Interval: A confidence interval is a statistical concept and a range of values used to estimate an unknown population parameter, such as the population mean or proportion, based on a sample from that population.

In [6]:
```
import pandas as pd
df=pd.read_csv('/dataset/idslab/diamonds.csv')
df.head(10)
```

Out[6]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39

Q1: Calculate the sample mean for 'price' column with n=500 and observe

In [7]:
```
print('URK21CS1181')
import numpy as np
s1=500
sample=np.random.choice(a=df['price'],size=s1)
m_sample_1=sample.mean();
print("Mean of 500 samples is ",m_sample_1)
```

```
URK21CS1181
Mean of 500 samples is  3820.636
```

Q2: Calculate the sample mean for 'price' column with n=1000 and observe

In [8]:
```
print('URK21CS1181')
sample2=np.random.choice(a=df['price'],size=1000)
mean2=sample2.mean();
print("Mean of 1000 samples is ",mean2)
```

```
URK21CS1181
Mean of 1000 samples is  3664.032
```

Q3: Calculate the population mean for 'price' column

In [9]:
```
print('URK21CS1181')
pm=df['price'].mean()
print("Mean of the entire population is ",pm)
```

```
URK21CS1181
Mean of the entire population is  3932.799721913237
```

Q4: Calculate the confidence interval (CI) with sample mean for 'price' column of n=500 and confidence level of 95%. Observe whether the population mean lies in CI.

In [10]:
```
print('URK21CS1181')
import scipy.stats as stats
import math
SD=sample.std()
print("Standard Deviation ", SD)
CL=0.95
alpha=(1-CL)/2
z=round(stats.norm.ppf(1-alpha),2)
print("Z score ", z)
er=z*(SD/math.sqrt(s1))
L=m_sample_1-er
H=m_sample_1+er
print("confidence interval",L,H)
print("[",L,pm,H,"]")
```

```
URK21CS1181
Standard Deviation  3861.6065666382947
Z score  1.96
confidence interval 3482.1512604137783 4159.120739586222
[ 3482.1512604137783 3932.799721913237 4159.120739586222 ]
```

Q5: Change the confidence level to 99% and observe the confidence interval for the same sample mean for 'price' column of n=500.

In [11]:
```
print('URK21CS1181')
SD=sample.std()
print("Standard Deviation ", SD)
CL=0.99
alpha=(1-CL)/2
z=round(stats.norm.ppf(1-alpha),2)
```

```
print("Z score ", z)
er=z*(SD/math.sqrt(s1))
L=m_sample_1-er
H=m_sample_1+er
print("confidence interval",L,H)
print("[",L,pm,H,"]")
```

```
URK21CS1181
Standard Deviation  3861.6065666382947
Z score  2.58
confidence interval 3375.07955707528 4266.19244292472
[ 3375.07955707528 3932.799721913237 4266.19244292472 ]
```

Q6: Calculate and plot the Confidence Intervals for 25 Trials with n=500 and CI=95% for 'price' column. Observe the results.

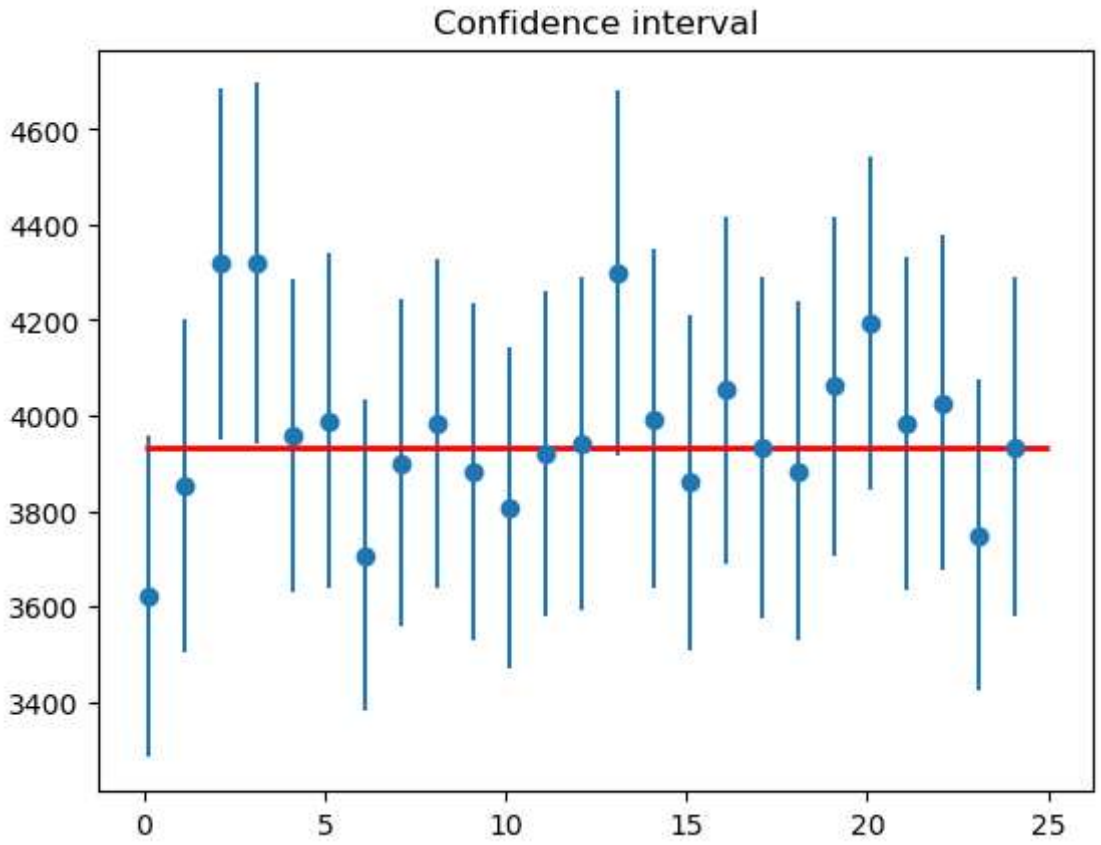
In [12]:
```
print('URK21CS1181')
sample_size=500
intervals=[]
sample_means=[]

CL=0.95
alpha=(1-CL)/2
z=round(stats.norm.ppf(1-alpha),2)

for sample in range(25):
    sample=np.random.choice(a=df['price'],size=sample_size)
    sample_mean=sample.mean()
    sample_means.append(sample_mean)
    SD=sample.std()
    er=z*(SD/math.sqrt(sample_size))
    L=sample_mean-er
    H=sample_mean+er
    confidence_interval=(L,H)
    intervals.append(confidence_interval)
print(sample_means)
print(pm)
print(intervals)
```

```
URK21CS1181
[3621.354, 3852.254, 4318.598, 4319.682, 3958.164, 3988.622, 3705.918, 3900.596, 3
984.622, 3882.276, 3806.49, 3920.798, 3940.75, 4299.476, 3993.854, 3859.308, 4052.
948, 3932.278, 3883.164, 4061.922, 4192.922, 3984.598, 4026.296, 3749.658, 3933.1
3]
3932.799721913237
[(3283.107166315436, 3959.6008336845634), (3502.890515601241, 4201.617484398759),
(3951.226052013937, 4685.969947986063), (3942.931424842278, 4696.432575157722), (3
628.5615035534415, 4287.766496446559), (3638.1352949094908, 4339.108705090509), (3
379.8482829506074, 4031.987717049393), (3558.5127553244884, 4242.679244675512), (3
639.2873333410707, 4329.956666658929), (3528.62254277248, 4235.92945722752), (347
0.773059485644, 4142.206940514356), (3581.204801098926, 4260.391198901074), (3592.
3142054402538, 4289.185794559746), (3917.170136170748, 4681.781863829252), (3636.5
49131447728, 4351.158868552271), (3509.4640361197467, 4209.151963880254), (3688.09
03179003944, 4417.805682099605), (3575.0632938939175, 4289.492706106082), (3528.14
3882457956, 4238.184117542045), (3707.2831658452988, 4416.560834154701), (3843.173
452922013, 4542.670547077986), (3634.746645140895, 4334.4493548591045), (3675.5959
59123173, 4376.996040876827), (3424.029641289033, 4075.286358710967), (3577.566139
1086564, 4288.693860891343)]
```

In [13]:
```
print('URK21CS1181')
import matplotlib.pyplot as plt
plt.errorbar(x=np.arange(0.1,25,1),
             y=sample_means,
             yerr = [(bot - top) / 2 for top, bot in intervals],
             fmt="o")
plt.hlines(xmin=0,xmax=25,
           y=pm,
           linewidth=2.0,
           color="red")
plt.title("Confidence interval")
```

```
URK21CS1181
```

Out[13]:
```
Text(0.5, 1.0, 'Confidence interval')
```



Q7: Calculate the Correlation Coefficient using Pearson for the given table

In [14]:
```
print('URK21CS1181')
x=pd.Series([150,169,175,180,200])
y=pd.Series([125,130,160,169,150])

p=x.corr(y,method='pearson')
print('Pearson correlation: %.3f'%p)
```

```
URK21CS1181
Pearson correlation: 0.610
```

Q8: Calculate the Correlation Coefficient using Spearman for the given table

In [15]:
```
print('URK21CS1181')
s=x.corr(y,method='spearman')
print('Spearman correlation: %.3f'%s)
```

```
URK21CS1181
Spearman correlation: 0.700
```

Q9: Calculate the Covariance Matrix and plot the Correlation in Seaborn Heatmap for the given data and analyse it

In [16]:
```
print('URK21CS1181')
df=pd.DataFrame({'Math':[90,90,60,60,30],'English':[60,90,60,60,30],'Art':[90,30,60
cov_matrix=df.cov()
print(cov_matrix)
```

```
URK21CS1181
          Math  English    Art
Math     630.0    450.0  225.0
English  450.0    450.0    0.0
Art      225.0      0.0  900.0
```

Q10: Perform a hypothesis testing with Z-test The mean breaking strength of the cables supplied by a manufacture is 1800 with a S.D of 100. By a new technique in the manufacturing process, it is claimed that the breaking strength of the cable has increased. In order to test this claim, a sample of 50 cables is tested and it is found that the mean breaking strength is 1850. Can we support the claim at 1 % level? (Z-critical=2.33)

In [17]:
```
xbar=1850
mu=1800
n=50
sd=100
z=abs(((xbar-mu)/(sd/math.sqrt(n))))
if(z>2.33):
    print("Reject H0")
else:
    print("Accept H0")
print(z)
```

```
Reject H0
3.5355339059327378
```