eda_lab2_3_jan

January 7, 2025

1 3 January, 2024

1.0.1 pd.qcut is a function in pandas that splits the data into equal-sized groups (quantiles).

[7]:
```
import numpy as np
import pandas as pd
randnum = np.random.rand(2000)
cat3 = pd.qcut(randnum,4)
```

[8]:
```
cat3
```

[8]:
```
[(-0.000789, 0.248], (0.248, 0.51], (0.51, 0.768], (-0.000789, 0.248], (0.51,
0.768], …, (0.768, 1.0], (-0.000789, 0.248], (-0.000789, 0.248], (0.768, 1.0],
(-0.000789, 0.248]]
Length: 2000
Categories (4, interval[float64, right]): [(-0.000789, 0.248] < (0.248, 0.51] <
(0.51, 0.768] < (0.768, 1.0]]
```

[9]:
```
pd.Series(cat3).value_counts()
```

[9]:
```
(-0.000789, 0.248]    500
(0.248, 0.51]         500
(0.51, 0.768]         500
(0.768, 1.0]          500
Name: count, dtype: int64
```

[10]:
```
randumnum=np.random.rand(2000)
cat3=pd.qcut(randumnum,10)
cat3
```

[10]:
```
[(0.801, 0.889], (0.801, 0.889], (0.602, 0.707], (0.214, 0.308], (0.112, 0.214],
…, (0.409, 0.507], (0.308, 0.409], (0.308, 0.409], (0.00021999999999999993,
0.112], (0.602, 0.707]]
Length: 2000
Categories (10, interval[float64, right]): [(0.00021999999999999993, 0.112] <
(0.112, 0.214] < (0.214, 0.308] < (0.308, 0.409] … (0.602, 0.707] < (0.707,
0.801] < (0.801, 0.889] < (0.889, 0.999]]
```

[11]:
```
pd.Series(cat3).value_counts()
```

[11]:
```
(0.00021999999999999993, 0.112]    200
(0.112, 0.214]                     200
(0.214, 0.308]                     200
(0.308, 0.409]                     200
(0.409, 0.507]                     200
(0.507, 0.602]                     200
(0.602, 0.707]                     200
(0.707, 0.801]                     200
(0.801, 0.889]                     200
(0.889, 0.999]                     200
Name: count, dtype: int64
```

1.0.2 Here instead of diving the data into equal intervals we are defining our own custom intervals

[12]:
```
cat4=pd.qcut(randumnum,[0,0.3,0.5,0.7,1.0])
pd.Series(cat4).value_counts()
```

[12]:
```
(0.00021999999999999993, 0.308]    600
(0.707, 0.999]                     600
(0.308, 0.507]                     400
(0.507, 0.707]                     400
Name: count, dtype: int64
```

1.0.3 Reading a dataset

[13]:
```
df = pd.read_csv(r"D:\study material\VIT_Data_Science\Winter_Sem\Exploratory␣
 ↪Data Analysis Lab\Dataset\sales.csv")
df.head(10)
```

[13]:

	Account	Company	Order	SKU	Country	Year	\
0	123456779	Kulas Inc	99985	s9-supercomputer	Aruba	1981	
1	123456784	GitHub	99986	s4-supercomputer	Brazil	2001	
2	123456782	Kulas Inc	99990	s10-supercomputer	Montserrat	1973	
3	123456783	My SQ Man	99999	s1-supercomputer	El Salvador	2015	
4	123456787	ABC Dogma	99996	s6-supercomputer	Poland	1970	
5	123456778	Super Sexy Dingo	99996	s9-supercomputer	Costa Rica	2004	
6	123456783	ABC Dogma	99981	s11-supercomputer	Spain	2006	
7	123456785	ABC Dogma	99998	s9-supercomputer	Belarus	2015	
8	123456778	Loolo INC	99997	s8-supercomputer	Mauritius	1999	
9	123456775	Kulas Inc	99997	s7-supercomputer	French Guiana	2004	

	Quantity	UnitPrice	transactionComplete
0	5148	545	False
1	3262	383	False

```
2      9119        407                True
3      3097        615               False
4      3356         91                True
5      2474        136                True
6      4081        195               False
7      6576        603               False
8      2460         36               False
9      1831        664                True
```

1.0.4 Introducing a new column total price

[15]:
```
df['Total_price'] = df['UnitPrice']*df['Quantity']
df
```

[15]:
```
        Account           Company  Order                SKU  \
0     123456779         Kulas Inc  99985   s9-supercomputer
1     123456784            GitHub  99986   s4-supercomputer
2     123456782         Kulas Inc  99990  s10-supercomputer
3     123456783         My SQ Man  99999   s1-supercomputer
4     123456787         ABC Dogma  99996   s6-supercomputer
…         …               …    …                 …
9995  123456784       Pryianka Ji  99987   s1-supercomputer
9996  123456775          Will LLC  99985   s3-supercomputer
9997  123456774         Kulas Inc  99982   s2-supercomputer
9998  123456781         Loolo INC  99986   s5-supercomputer
9999  123456775  Super Sexy Dingo  99987   s1-supercomputer

                       Country  Year  Quantity  UnitPrice  \
0                        Aruba  1981      5148        545
1                       Brazil  2001      3262        383
2                   Montserrat  1973      9119        407
3                  El Salvador  2015      3097        615
4                       Poland  1970      3356         91
…                          …   …       …        …
9995                   Jamaica  1983       886        475
9996                   Vietnam  2002      9995        302
9997  Northern Mariana Islands  1979      1421        249
9998                      Mali  1991      2342        506
9999                Luxembourg  1982      5282        222

      transactionComplete  Total_price
0                   False      2805660
1                   False      1249346
2                    True      3711433
3                   False      1904655
4                    True       305396
…                    …          …
```

```
9995                False       420850
9996                 True      3018490
9997                 True       353829
9998                False      1185052
9999                False      1172604

[10000 rows x 10 columns]
```

- Remove the 'TotalPrice' column from its current position and insert it at the first (0th) position in the DataFrame.

[17]:
```
df.insert(0,'Total_price',df.pop('Total_price'))
df
```

[17]:
```
      Total_price    Account           Company  Order                SKU  \
0         2805660  123456779         Kulas Inc  99985   s9-supercomputer
1         1249346  123456784            GitHub  99986   s4-supercomputer
2         3711433  123456782         Kulas Inc  99990  s10-supercomputer
3         1904655  123456783         My SQ Man  99999   s1-supercomputer
4          305396  123456787         ABC Dogma  99996   s6-supercomputer
…             …          …                 …      …                  …
9995       420850  123456784       Pryianka Ji  99987   s1-supercomputer
9996      3018490  123456775          Will LLC  99985   s3-supercomputer
9997       353829  123456774         Kulas Inc  99982   s2-supercomputer
9998      1185052  123456781         Loolo INC  99986   s5-supercomputer
9999      1172604  123456775  Super Sexy Dingo  99987   s1-supercomputer

                        Country  Year  Quantity  UnitPrice  transactionComplete
0                         Aruba  1981      5148        545                False
1                        Brazil  2001      3262        383                False
2                    Montserrat  1973      9119        407                 True
3                   El Salvador  2015      3097        615                False
4                        Poland  1970      3356         91                 True
…                           …     …         …          …                  …
9995                    Jamaica  1983       886        475                False
9996                    Vietnam  2002      9995        302                 True
9997  Northern Mariana Islands  1979      1421        249                 True
9998                       Mali  1991      2342        506                False
9999                 Luxembourg  1982      5282        222                False

[10000 rows x 10 columns]
```

- Arguments: 0 - index to insert the column, 'TotalPrice' - column name, df.pop('TotalPrice') - data to insert

1.0.5 Filtering out data where totaltransaction > 3000000

[18]:
```
totaltransaction = df['Total_price']
totaltransaction[np.abs(totaltransaction)>3000000]
```

[18]:
```
2       3711433
7       3965328
13      4758900
15      5189372
17      3989325
        ...
9977    3475824
9984    5251134
9987    5670420
9991    5735513
9996    3018490
Name: Total_price, Length: 2094, dtype: int64
```

1.0.6 filtering rows from the dataframe where totaltransaction (i.e total price) is greater than 674112

[19]:
```
df[np.abs(totaltransaction)>6741112]
```

[19]:

	Total_price	Account	Company	Order	SKU	\
818	6746328	123456781	Gen Power	99991	s1-supercomputer	
1402	6841580	123456778	Will LLC	99985	s11-supercomputer	
2242	6784368	123456770	Name IT	99997	s9-supercomputer	
2876	6745865	123456772	Gen Power	99992	s10-supercomputer	
3210	6841112	123456782	Loolo INC	99991	s8-supercomputer	
3629	6785800	123456779	My SQ Man	99980	s3-supercomputer	
7674	6828462	123456781	Loolo INC	99989	s6-supercomputer	
8645	6809658	123456789	Gen Power	99996	s11-supercomputer	
8684	6804670	123456785	Gen Power	99989	s2-supercomputer	

	Country	Year	Quantity	UnitPrice	transactionComplete
818	Burkina Faso	1985	9693	696	False
1402	Austria	1990	9844	695	True
2242	Myanmar	1979	9804	692	False
2876	Mali	2007	9935	679	False
3210	Kuwait	2006	9886	692	False
3629	Hong Kong	1994	9694	700	False
7674	Sri Lanka	1994	9882	691	False
8645	Suriname	2005	9742	699	False
8684	Kenya	2013	9805	694	False

- create an array from 0-79 and then reshape it to 10x8

[22]:
```
dat = np.arange(80).reshape(10,8)
df = pd.DataFrame(dat)
df
```

[22]:

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47
6	48	49	50	51	52	53	54	55
7	56	57	58	59	60	61	62	63
8	64	65	66	67	68	69	70	71
9	72	73	74	75	76	77	78	79

- randomply permute np.arange(x)

[23]:
```
sampler = np.random.permutation(10)
sampler
```

[23]:
```
array([1, 2, 5, 3, 4, 7, 0, 6, 9, 8])
```

- randomly selects and returns all rows from df in a new order(in the order of sample array), effectively shuffling the DataFrame.

[24]:
```
df.take(sampler)
```

[24]:

	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
5	40	41	42	43	44	45	46	47
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39
7	56	57	58	59	60	61	62	63
0	0	1	2	3	4	5	6	7
6	48	49	50	51	52	53	54	55
9	72	73	74	75	76	77	78	79
8	64	65	66	67	68	69	70	71

- To randomly choose n rows

[26]:
```
df.take(np.random.permutation(len(df))[:3])
```

[26]:

	0	1	2	3	4	5	6	7
9	72	73	74	75	76	77	78	79
7	56	57	58	59	60	61	62	63
0	0	1	2	3	4	5	6	7

1.1 Random sampling with replacement

[29]:
```
sack = np.array([4,8,-2,7,5])
sampler = np.random.randint(0,len(sack),size=10)
sampler
```

[29]:
```
array([4, 2, 2, 2, 2, 3, 4, 0, 2, 1])
```

- Select elements from 'sack' at indices specified by 'sampler'

[31]:
```
draw = sack.take(sampler)
draw
```

[31]:
```
array([ 5, -2, -2, -2, -2,  7,  5,  4, -2,  8])
```

1.2 Computing Indicators/ Dummy variable

[33]:
```
df=pd.DataFrame({'gender':
  ↪['female','female','male','unknown','male','female'],'votes':range(6,12,1)})
df
```

[33]:

	gender	votes
0	female	6
1	female	7
2	male	8
3	unknown	9
4	male	10
5	female	11

[34]:
```
pd.get_dummies(df['gender'])
```

[34]:

	female	male	unknown
0	True	False	False
1	True	False	False
2	False	True	False
3	False	False	True
4	False	True	False
5	True	False	False

[35]:
```
pd.get_dummies(df['gender']).astype(int)
```

[35]:

	female	male	unknown
0	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0

[36]:
```
dummies=pd.get_dummies(df['gender'],prefix='gender')
dummies
```

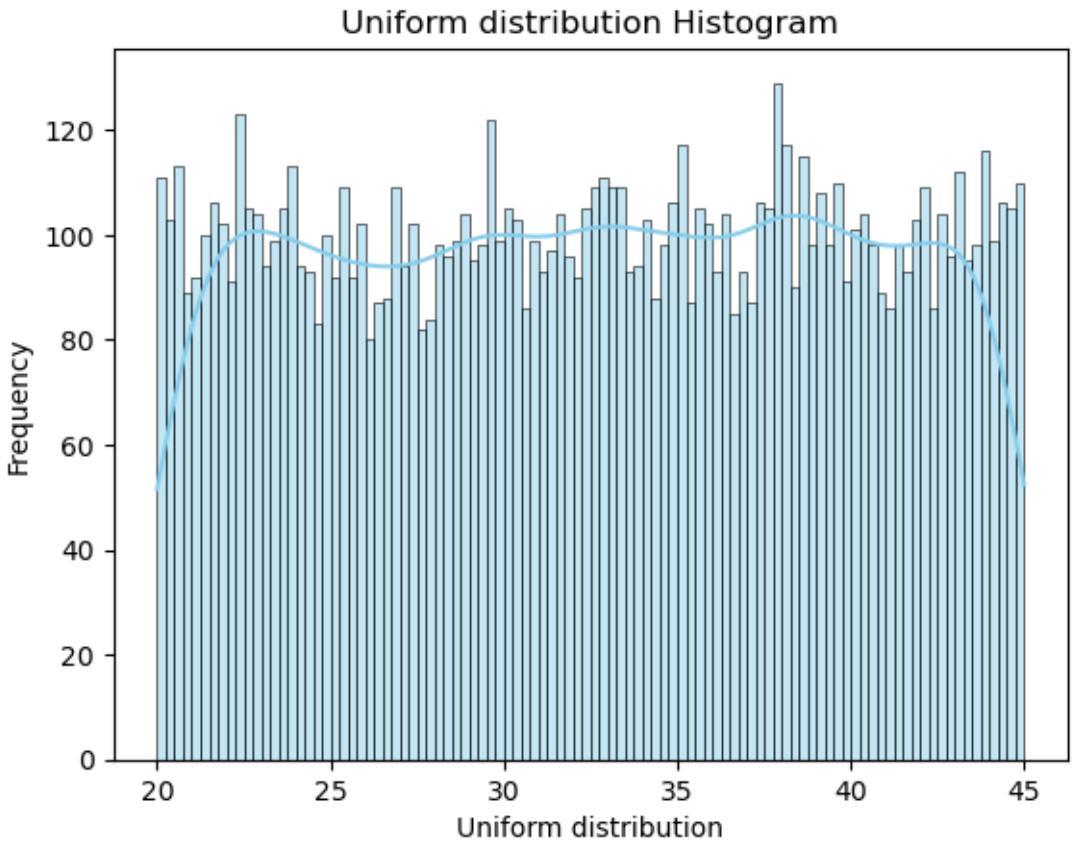
[36]:

	gender_female	gender_male	gender_unknown
0	True	False	False
1	True	False	False
2	False	True	False
3	False	False	True
4	False	True	False
5	True	False	False

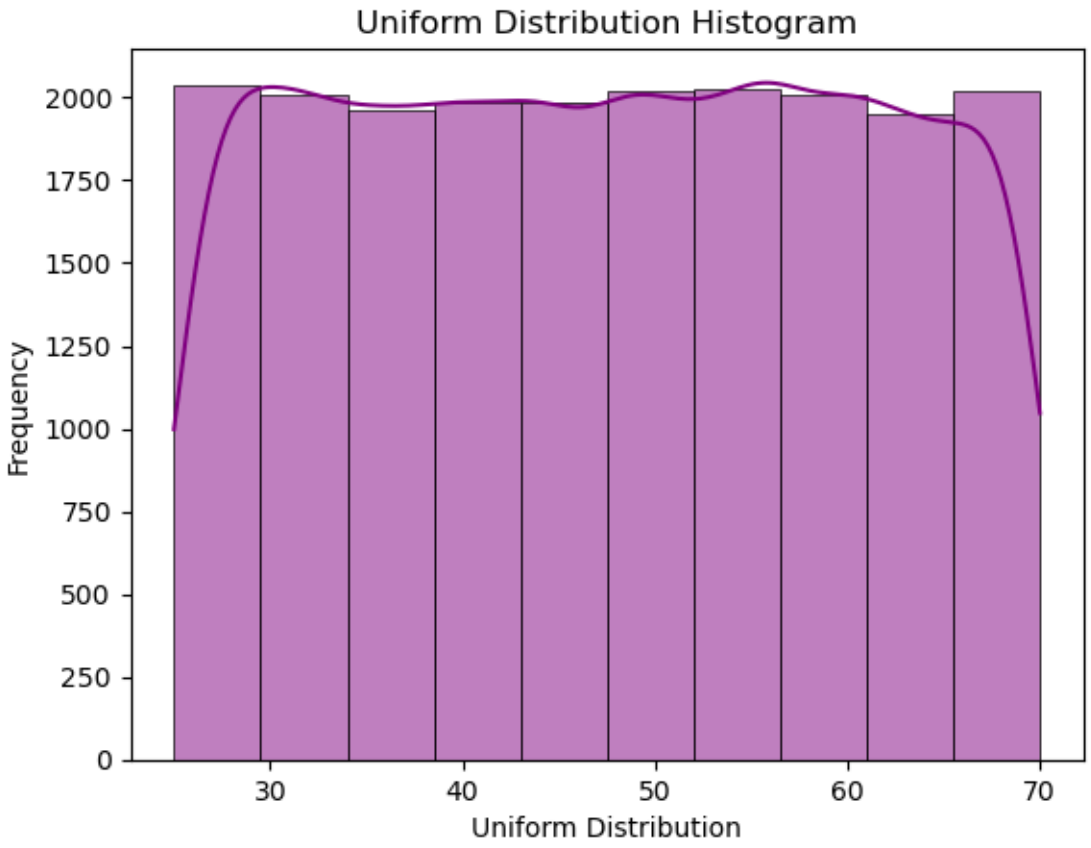
1.3 Descriptive Statistics

1.3.1 Unifrom distribution

[39]:
```
import numpy as np
import seaborn as sns
from scipy.stats import uniform
import matplotlib.pyplot as plt

number = 10000
start = 20
width = 25

## Creating sample dataset
u_d = uniform.rvs(size = number,loc = start, scale = width)
## upperbound of the function is start+width

##plotting using histplot
sns.histplot(u_d,bins = 100,kde= True,color = 'skyblue',linewidth = 0.4)
plt.xlabel('Uniform distribution')
plt.ylabel('Frequency')
plt.title('Uniform distribution Histogram')
plt.show()
```

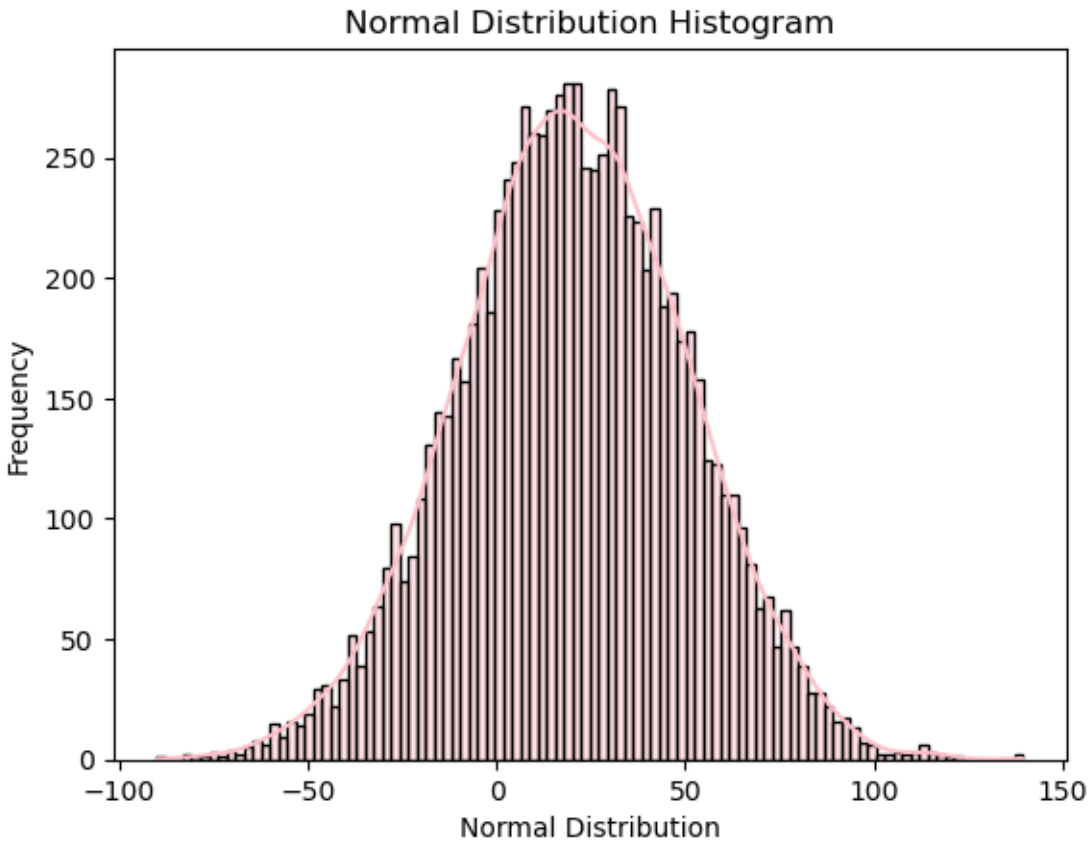


[40]:
```
number=20000
start=25
width=45
#Create sample dataset
u_d=uniform.rvs(size=number,loc=start,scale=width)
#Plot using histplot
sns.histplot(u_d,bins=10,kde=True,color='purple',linewidth= 0.5)
plt.xlabel('Uniform Distribution')
plt.ylabel('Frequency')
plt.title('Uniform Distribution Histogram')
plt.show()
```



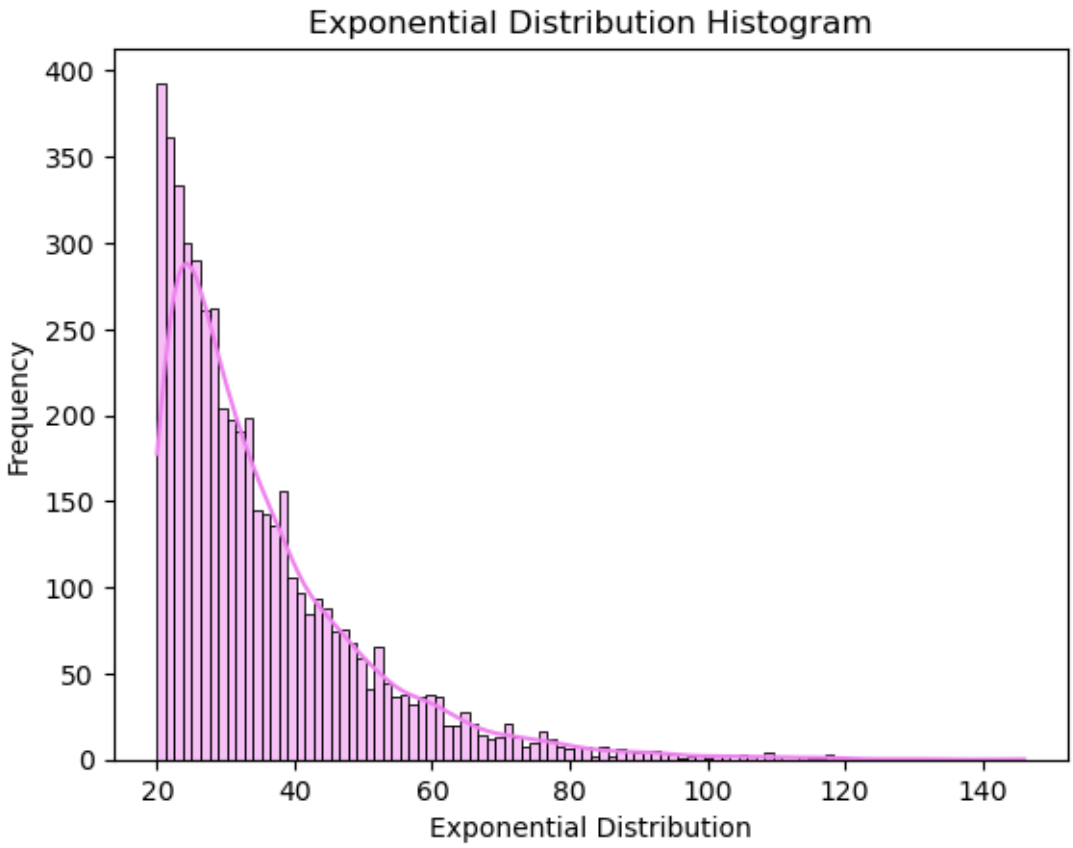
1.3.2 Normal distribution

[41]:
```
from scipy.stats import norm

# create sample dataset
norm_d = norm.rvs(size = 9000, loc = 20, scale = 30)

# plotting using histplot
sns.histplot(norm_d,bins = 100, kde = True,color = 'pink', linewidth = 1)
plt.xlabel('Normal Distribution')
plt.ylabel('Frequency')
plt.title('Normal Distribution Histogram')
plt.show()
```



1.3.3 Exponential distribution

[43]:
```
from scipy.stats import expon
#Create sample dataset
e_d=expon.rvs(size=5000,loc=20,scale=15)
#Plotting using histplot
sns.histplot(e_d,bins=100,kde=True,color='violet',linewidth= 0.6)
plt.xlabel('Exponential Distribution')
plt.ylabel('Frequency')
plt.title('Exponential Distribution Histogram')
plt.show()
```


Exponential Distribution Histogram
Frequency
Exponential Distribution
0
50
100
150
200
250
300
350
400
20
40
60
80
100
120
140