Install Dependencies

In [102]:

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in ./anaconda3/lib/python
3.10/site-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/p
ython3.10/site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/pyth
on3.10/site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./anaconda3/lib/p
ython3.10/site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: numpy>=1.20 in ./anaconda3/lib/python
3.10/site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/
python3.10/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/pytho
n3.10/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/
python3.10/site-packages (from matplotlib) (4.39.4)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/l
ib/python3.10/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/py
thon3.10/site-packages (from matplotlib) (23.0)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.1
0/site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [103]:

```
!pip install numpy
```

```
Requirement already satisfied: numpy in ./anaconda3/lib/python3.10/s
ite-packages (1.24.3)
```

In [104]:

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in ./anaconda3/lib/python3.1
0/site-packages (0.12.2)
Requirement already satisfied: pandas>=0.25 in ./anaconda3/lib/pytho
n3.10/site-packages (from seaborn) (2.0.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in ./anaconda3/l
ib/python3.10/site-packages (from seaborn) (1.24.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in ./anaconda
3/lib/python3.10/site-packages (from seaborn) (3.7.1)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/py
thon3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.
0)
Requirement already satisfied: pyparsing>=2.3.1 in ./anaconda3/lib/p
ython3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.
0.9)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/
python3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.
39.4)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/pyth
on3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/l
ib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/
python3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.
4.4)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/pytho
n3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/p
ython3.10/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.
0.7)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/pytho
n3.10/site-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: tzdata>=2022.1 in ./anaconda3/lib/pyt
hon3.10/site-packages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.1
0/site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1-
>seaborn) (1.16.0)
```

In [105]:

```
!pip install scipy
```

```
Requirement already satisfied: scipy in ./anaconda3/lib/python3.10/s
ite-packages (1.10.1)
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in ./anaconda3/
lib/python3.10/site-packages (from scipy) (1.24.3)
```

In [167]:

```
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set(color_codes = True)

import datetime
```

Bernoulli Distributions

The probability mass function for `bernoulli` is:

$$f(k) = \begin{cases} 1-p & \text{if } k = 0 \\ p & \text{if } k = 1 \end{cases}$$

for k in $\{0, 1\}$, $0 \leq p \leq 1$

`bernoulli` takes p as shape parameter, where p is the probability of a single success and $1 - p$ is the probability of a single failure.

In [168]:

```
from scipy.stats import bernoulli
```

In [169]:

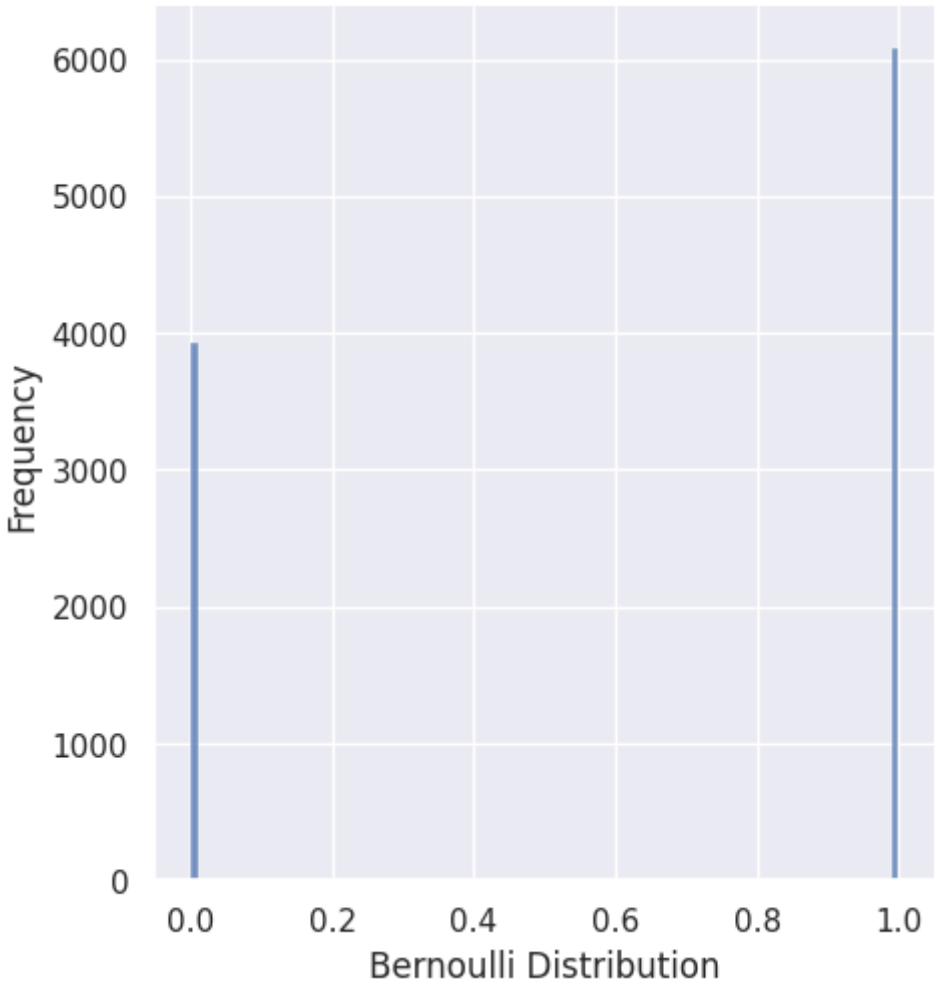
```
data_bern = bernoulli.rvs(size = 10000, p = 0.6)

ax = sns.displot(data_bern,
                 bins = 100,
                 kde = False,
                 )

ax.set(xlabel = "Bernoulli Distribution", ylabel = "Frequency")
```

Out[169]:

```
<seaborn.axisgrid.FacetGrid at 0x7f74bd3cf2b0>
```



In [170]:

```
for i in data_bern:
    print(i)
```

```
0
0
1
1
1
0
0
0
1
1
1
1
0
0
0
1
1
0
1
```

In [171]:

```
# Counting frequency of every event How many time is happened

from collections import Counter
cnt = Counter()

for i in data_bern:
    cnt[i] += 1


cnt.most_common()
```

Out[171]:

```
[(1, 6078), (0, 3922)]
```

Binomial Distribution

The probability mass function for `binom` is:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for $k \in \{0, 1, \ldots, n\}, 0 \leq p \leq 1$

`binom` takes n and p as shape parameters, where p is the probability of a single success and $1 - p$ is the probability of a single failure.

In [172]:

```
from scipy.stats import binom
```

In [174]:

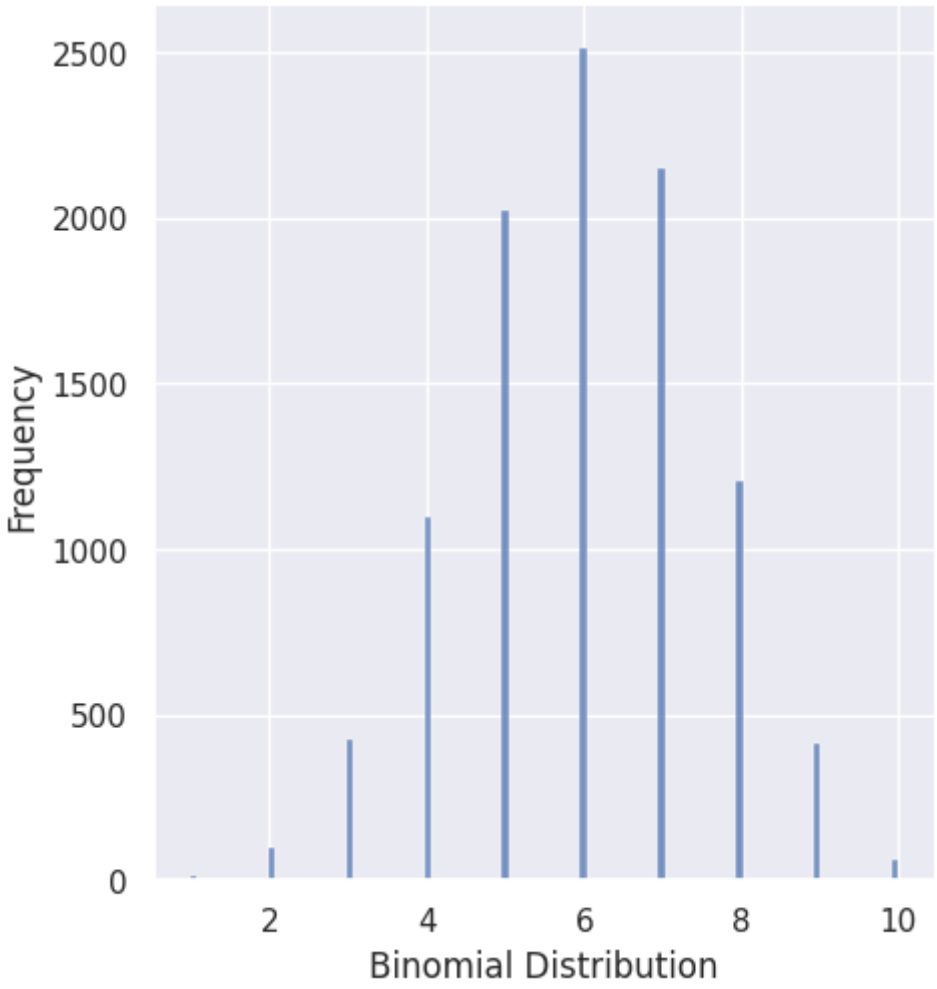
```
data_binom = binom.rvs(n = 10 , p = 0.6, size = 10000)

ax = sns.displot(data_binom,
                 bins = 100,
                 kde = False,
                 )

ax.set(xlabel = "Binomial Distribution", ylabel = "Frequency")
```

Out[174]:

```
<seaborn.axisgrid.FacetGrid at 0x7f74bc4aae30>
```



In [ ]:

In [175]:

```
for i in data_binom:
    print(i)
```

```
6
5
6
5
8
5
7
9
5
9
9
7
9
6
8
6
5
7
9
```

In [176]:

```
# Counting frequency of every event How many time is happened

from collections import Counter
cnt = Counter()

for i in data_binom:
    cnt[i] += 1


cnt.most_common()
```

Out[176]:

```
[(6, 2512),
 (7, 2151),
 (5, 2021),
 (8, 1204),
 (4, 1097),
 (3, 424),
 (9, 414),
 (2, 100),
 (10, 63),
 (1, 14)]
```

Poisson Distribution

The probability mass function for `poisson` is:

$$f(k) = \exp(-\mu)\frac{\mu^k}{k!}$$

for $k \geq 0$.

`poisson` takes $\mu \geq 0$ as shape parameter. When $\mu = 0$, the `pmf` method returns `1.0` at quantile $k = 0$.

In [177]:

```
from scipy.stats import poisson
```

In [178]:

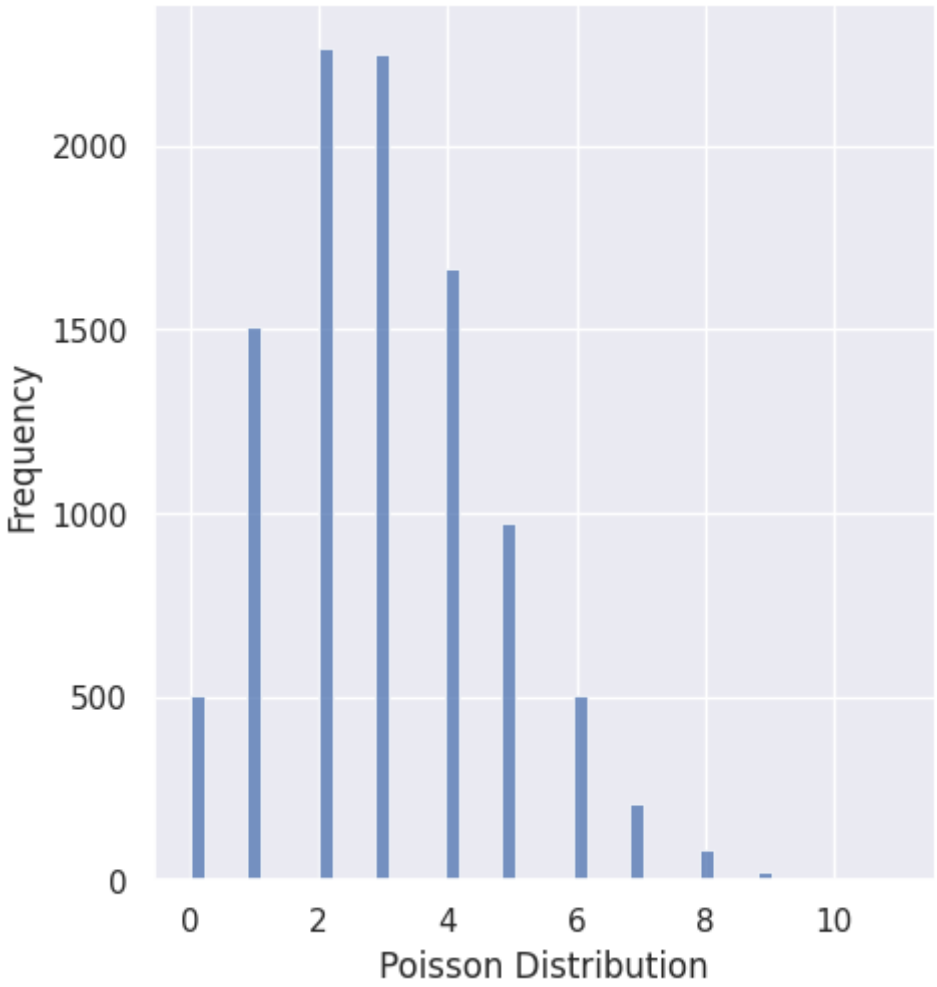
```
data_poisson = poisson.rvs(mu=3, size = 10000)

ax = sns.displot(data_poisson,
                 bins = 50,
                 kde = False,
                 )

ax.set(xlabel = "Poisson Distribution", ylabel = "Frequency")
```

Out[178]:

```
<seaborn.axisgrid.FacetGrid at 0x7f74bc06ea10>
```



In [179]:

```
for i in data_poisson:
    print(i)
```

```
5
6
5
2
5
3
3
2
6
0
2
5
0
4
6
5
5
7
3
5
```

In [180]:

```
# Counting frequency of every event How many time is happened

from collections import Counter
cnt = Counter()

for i in data_poisson:
    cnt[i] += 1


cnt.most_common()
```

Out[180]:

```
[(2, 2265),
 (3, 2251),
 (4, 1666),
 (1, 1509),
 (5, 972),
 (6, 506),
 (0, 505),
 (7, 207),
 (8, 84),
 (9, 24),
 (10, 8),
 (11, 3)]
```

In [181]:

```
# Counting frequency of every event How many time is happened

from collections import Counter
cnt = Counter()

for i in data_poisson:
    cnt[i] = cnt[i] + 1


cnt.most_common()
```

Out[181]:

```
[(2, 2265),
 (3, 2251),
 (4, 1666),
 (1, 1509),
 (5, 972),
 (6, 506),
 (0, 505),
 (7, 207),
 (8, 84),
 (9, 24),
 (10, 8),
 (11, 3)]
```

Geometric Distribution

The probability mass function for `geom` is:

$$f(k) = (1-p)^{k-1}p$$

for $k \geq 1, 0 < p \leq 1$

`geom` takes p as shape parameter, where p is the probability of a single success and $1 - p$ is the probability of a single failure.

In [182]:

```
from scipy.stats import geom
```

In [183]:

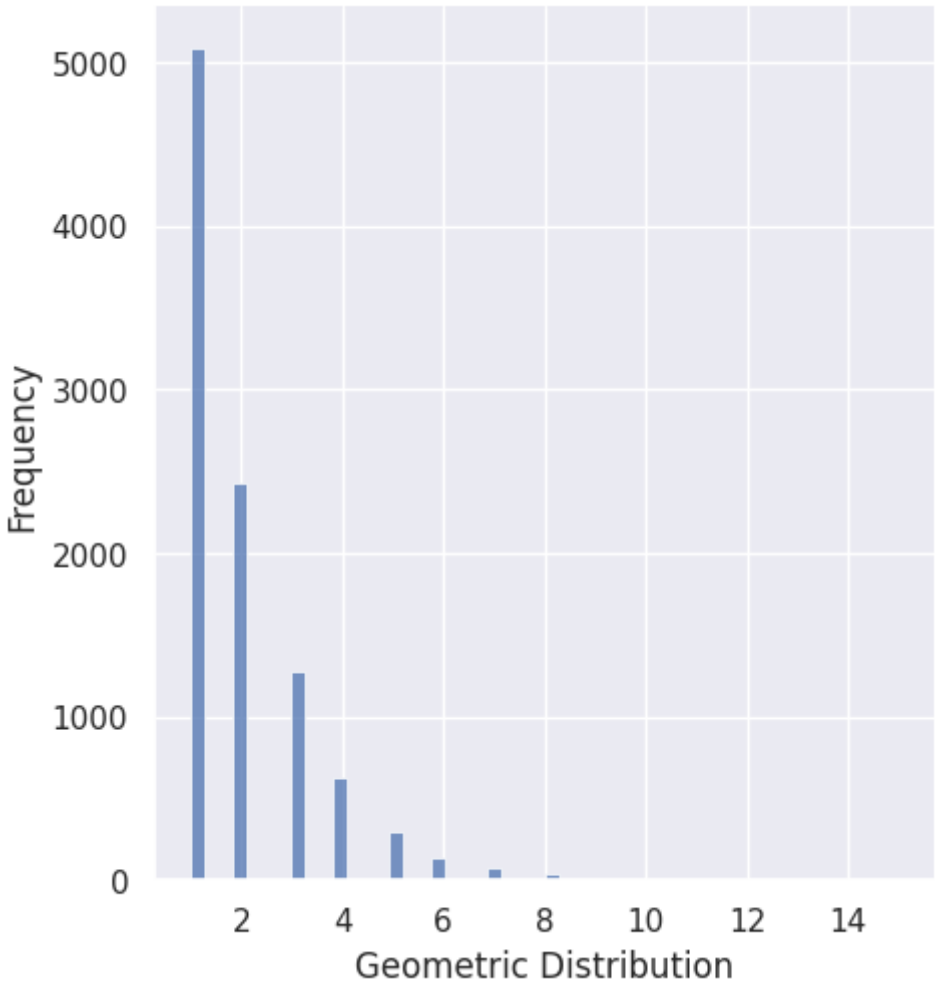
```
data_geometric = geom.rvs(p=0.5, size = 10000)

ax = sns.displot(data_geometric,
                 bins = 50,
                 kde = False,
                 )

ax.set(xlabel = "Geometric Distribution", ylabel = "Frequency")
```

Out[183]:

```
<seaborn.axisgrid.FacetGrid at 0x7f74bbdfc1c0>
```



In [184]:

```
# Counting frequency of every event How many time is happened

from collections import Counter
cnt = Counter()

for i in data_geometric:
    cnt[i] +=  1


cnt.most_common()
```

Out[184]:

```
[(1, 5084),
 (2, 2433),
 (3, 1274),
 (4, 624),
 (5, 301),
 (6, 135),
 (7, 73),
 (8, 41),
 (9, 16),
 (10, 6),
 (13, 4),
 (12, 4),
 (15, 2),
 (11, 2),
 (14, 1)]
```

In [ ]:

In [ ]: