Table of Contents

Overview of the dataset

The dataset contains 49945 transactions of 23708 different customers taken from 30th November 2016 to 6th December 2017. The first column uniquely identifies the customer, the second column has the date and time of the transaction and the last column has the amount a customer spent during that transaction.

Introduction about the library and the models

The lifetimes library is useful in predicting when a customer will make a purchase, when a person will come back and visit a website and most importantly predict the lifetime value of the customer. It consists of different types of fitters for this purpose like the BG model, modified BG model and the Pareto/NBD model. In this notebook, we are using the modified BG model to predict the customer behaviour.

The modified BG model has all the assumptions of the BG model except an additional assumption that a customer may drop out after the first purchase. The model assumes that after each transaction, the customer tosses a buy coin (not a fair coin) and depending on the result he chooses to continue his purchases or drop-out. The customer will stay alive after each purchase until the coin toss result is 'T'. Hence the customer's relationship with the firm is geometrically distributed (number of coin tosses). The customers probability of dropping out or dying can take a value between 0 and 1 and this is achieved by using a beta distribution.

The Gamma-Gamma model is used to calculate the economic value of each transaction. It gives us an understanding what a customer will be worth in a future time period.

In this dataset we see that the relationship between the customer and the ecommerce store is non-contractual. In a non-contractual setting, differentiating between those customers who have ended their relationship with the firm versus those who are in the middle of a break between transactions is very difficult.

Loading the data, viewing and checking the data types of columns

Loading the required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifetimes.utils import *
from lifetimes import ModifiedBetaGeoFitter
from lifetimes.plotting import plot_frequency_recency_matrix
from lifetimes.plotting import plot_probability_alive_matrix
from lifetimes.plotting import plot_period_transactions
from lifetimes import GammaGammaFitter
import random
from autograd.scipy.special import gamma
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
sns.set_palette("husl")
sns.set(rc={'image.cmap': 'coolwarm'})
%matplotlib inline
pd.options.display.float_format = '{:.4f}'.format
```

Loading the dataset

In [2]:
```
df = pd.read_csv('data-science-exercise-data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	CustomerID	Timestamp	PurchaseValue
0	5611860a94204a4078659f1667372cc4	2017-02-09T18:31:00Z	4.7200
1	07b260307114f9cffc5601eb7ad4b565	2017-06-28T19:26:00Z	5.3300
2	1ede55062d0d68c70fc4d355a5328d11	2017-07-28T17:47:00Z	5.3500
3	42b8e86e8da5e35628bcdc146c358665	2016-12-17T22:59:00Z	6.0000
4	1047f2787b7efc759d7ffcbc40ef7e19	2016-12-05T17:23:00Z	6.2800

In [4]:
```
df.shape
```

Out[4]:
```
(49945, 3)
```

In [5]:
```
df.dtypes
```

Out[5]:
```
CustomerID        object
Timestamp         object
PurchaseValue    float64
dtype: object
```

Converting 'date' column to datetime datatype

In [6]:
```
df['Timestamp'] = pd.to_datetime(df['Timestamp']).dt.date
```

In [7]:
```
df['Timestamp'].min()
```

Out[7]:
```
datetime.date(2016, 11, 30)
```

In [8]:
```
df['Timestamp'].max()
```

Out[8]:
```
datetime.date(2017, 12, 6)
```

Implement the modified BG model from the lifetimes package using the data we provide.

Creating a summary data from the given transaction data with a daily frequency for data aggregation

This will give us the RFM values of each customer.

Here,

Frequency : Number of repeat purchases a customer has made. If the frequency is 0, it means that the customer has made only 1 purchase.

Recency : It is the time between the customer's first purchase and last purchase. It tell the age of the customer when he did his most recent purchase.

T : It is the age of customer in the units of the dataset (here days). It is the time between the customer's first purchase and the end of observation time period.

In [9]:
```
data = summary_data_from_transaction_data(df, 'CustomerID', 'Timestamp', monetary_value_col='PurchaseValue', observation_period_end='2017-12-6', freq = 'D')
data.head(10)
```

Out[9]:

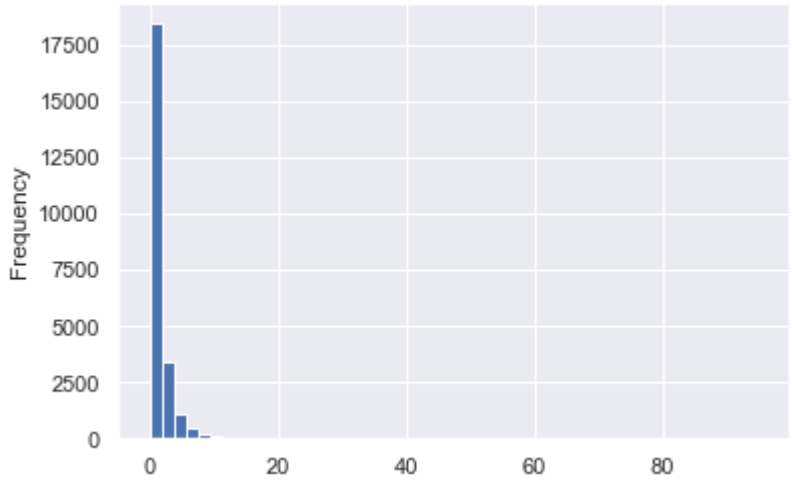
CustomerID	frequency	recency	T	monetary_value
0001117ff1305c1fe840697166e61564	1.0000	11.0000	344.0000	87.2800
00028502859fd7e111d88c20456b59d5	0.0000	0.0000	342.0000	0.0000
000306d200fd6e9b03de48d15348f5c2	0.0000	0.0000	33.0000	0.0000
0003f3458a6e7b495a975c2d9ddda559	2.0000	285.0000	343.0000	88.0650
000784b838b807ad589d4bc69c0c562f	0.0000	0.0000	210.0000	0.0000
0008e6b90a8f191089e8a0757fabf968	1.0000	69.0000	77.0000	145.5400
00090b9fa28029c4839ca0a0306b6b11	1.0000	61.0000	82.0000	349.4400
000ad0f90e9fcb6ff5a0bc480cccbdb3	5.0000	68.0000	200.0000	229.7820
000af852b3020e67ba4385e6477d4e58	0.0000	0.0000	353.0000	0.0000
000c6eab0c01eddd654c20fab4182dae	4.0000	165.0000	183.0000	119.2825

From the above result, we can see that the first customer with **Customer ID '0001117ff1305c1fe840697166e61564'** has made 2 purchases, i.e. he has just made **1 repeat purchase. So his frequency is 1. The time duration between the customer's first purchase and his latest purchase is 11 days.** T represents the customers age, i.e. the time between his first purchase to the end of observation period. Here the **customer we are considering has a age of 344 days.**

Plotting the histogram of frequency of purchases

In [10]:
```
data['frequency'].plot(kind='hist', bins = 50)
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2403a29e2b0>
```



In [11]:
```
print(sum(data['frequency'] == 0)/float(len(data)))
```

```
0.5930909397671672
```

The above result shows that in our dataset, nearly 59.30% of the customers have made a purchase only once, i.e. their frequency is 0

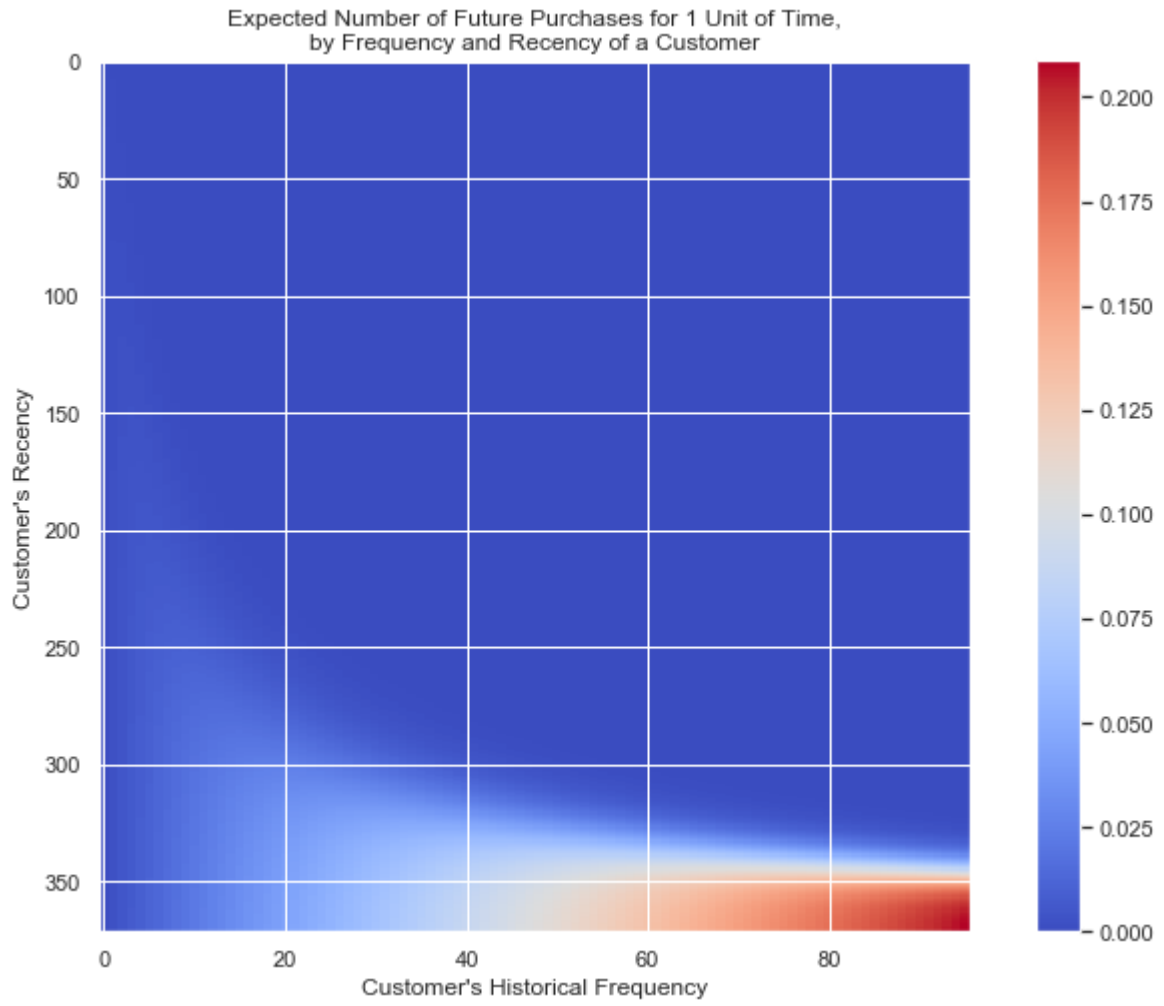
Fitting the data to Modified beta-geometric model

In [12]:
```
mbg = ModifiedBetaGeoFitter(penalizer_coef=0.0)
mbg.fit(data['frequency'], data['recency'], data['T'])
print(mbg)
```

```
<lifetimes.ModifiedBetaGeoFitter: fitted with 23708 subjects, a: 0.54, alpha:
85.27, b: 1.45, r: 0.79>
```

In [13]:
```
fig = plt.figure(figsize=(12,8))
plot_frequency_recency_matrix(mbg)
```

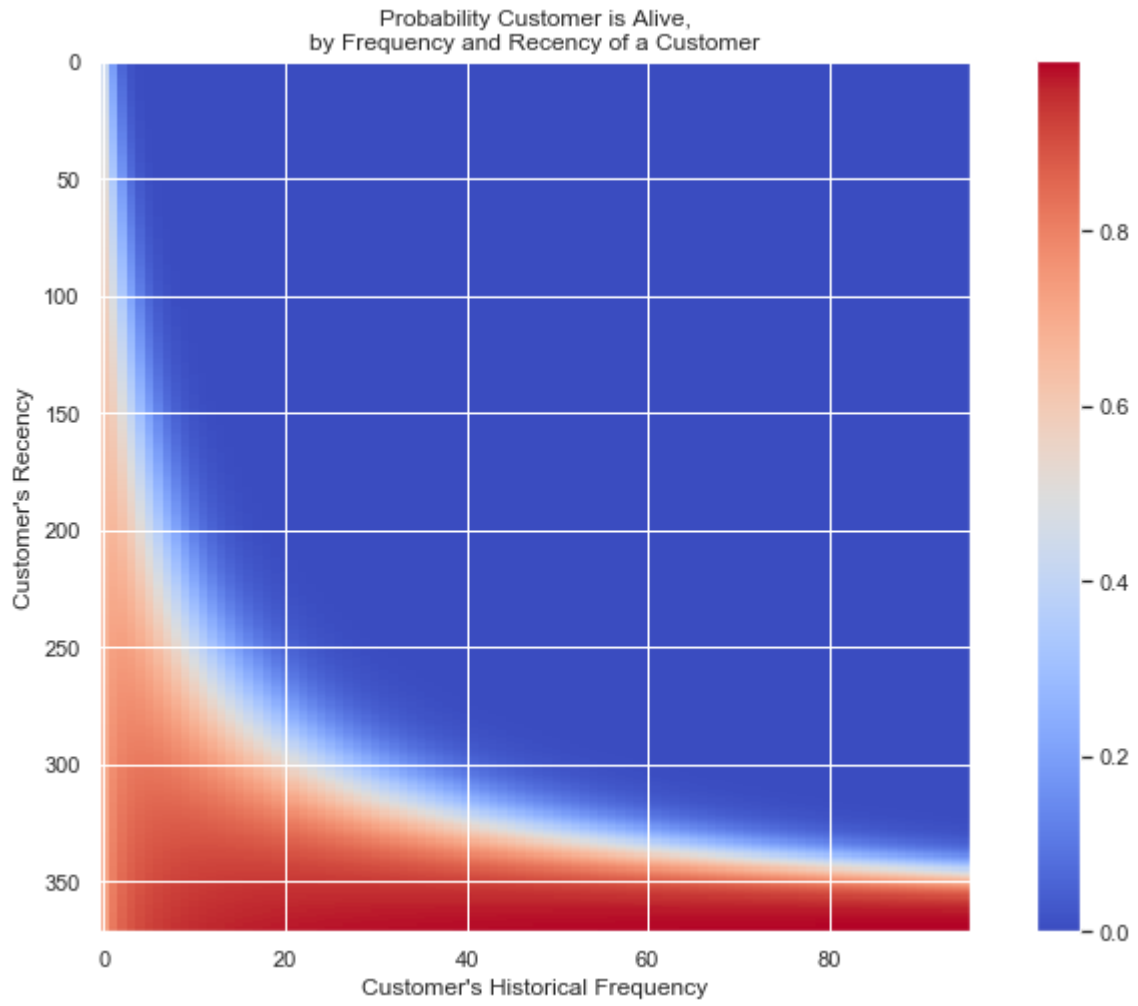
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x2403a876710>



The above graph tells us that our best customer will be the one who has made more than 80-85 purchases and has a recency of more than 350 days. Customers who have purchased a lot but their recency is low, will probably have no future purchases, while the customers who have around 50 to 60 purchases and a recency of 325 to 350 days have a medium chance of buying in the future

In [14]:
```
fig = plt.figure(figsize=(12,8))
plot_probability_alive_matrix(mbg)
```

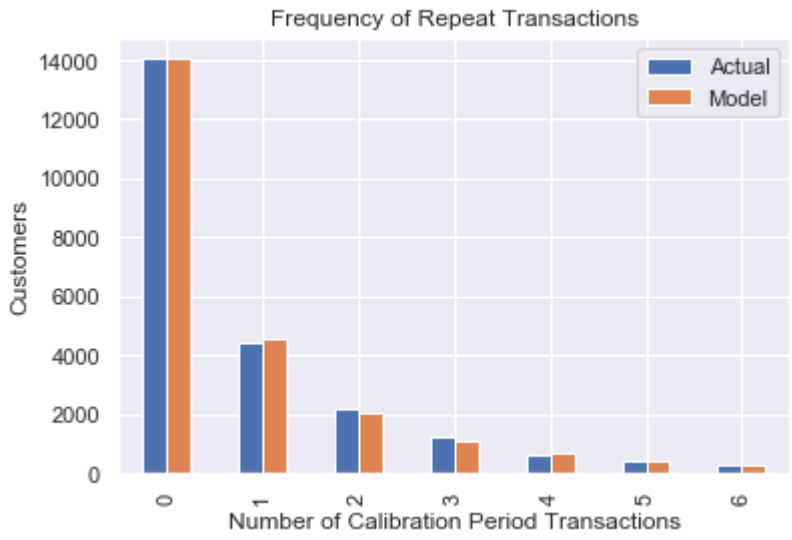
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x2403a7f9400>



The above graph tells us that the customers who have a recency of more than 350 days have a higher chance of being alive. As the recency goes on decreasing, the higher is the chance that the customer is dead or has dropped out

In [15]:
```
plot_period_transactions(mbg)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2403a97fb38>
```



The above graph shows that our model fits well, predicting closely to the actual observed data

List the 100 customers predicted to make the most purchases over the next 12 months.

As we have a granularity of days, we take 365 days as the time period. An important thing to note here is that if we take 12 months instead of 365 days, the results will differ as the in-built factor of converting months to days is of 30, which will make the number of days equal to 360 instead of 365

In [16]:
```
t = 365
predicted_purchases = mbg.conditional_expected_number_of_purchases_up_to_time(
t, data['frequency'], data['recency'], data['T'])
predicted_purchases = pd.DataFrame(predicted_purchases, columns = ['PredictedV
alues'])
```

Displaying the top 100 customers who are predicted to have the most purchases

In [17]:
```
pd.set_option('display.max_rows', None)
predicted_purchases.nlargest(100, 'PredictedValues')
```

Out[17]:

CustomerID	PredictedValues
a5fadc51b1ae844ad9a70ad3cfa46a64	64.4729
48a503edbaded96a3be27deee11967a1	30.9204
9f447f9415a380ac2eeee7df49c6ee7e	25.9509
5f01420f0edda6555df5ce1cc62b986c	25.0565
8d2ce54737dd404d20cadf1405d46dc8	22.9183
a62a17bb46864da2c6da691d838971b3	20.7534
3b11478939967e896ae2619615650f97	20.7343
2ad9a83ee23110d8c2f4c01600b94f20	19.9430
75fda9ea22086bf3814ff8c3f53de8ca	19.6025
30aa99d3357244cf38ca04eadef1473a	18.8912
5ac5ed64cd99ed2a8403b7a927e644ef	17.9184
a719d6643a7832535de9aded2f467825	17.7488
a1c8d419a97af1f7152e21c0dddfcbce	16.5157
cdf642859ae8d9a1e489c3a655cba827	16.3224
ed2b4332b3ca253cfbb0ffb54d3f5ae0	16.2586
880474a82e33389ae7f377b2ea35bbdd	15.6784
ca2202a96c2de6ca6b8a37a4a73fa730	15.6187
16134915d822fe17588ae585935e1e81	15.2935
f6a4f156c817ecb376d678f2aafdc570	15.2557
742d5a52d4df7cb14246d7f390de5d8a	14.4854
e675875db3648ab4dbbd52768296425c	14.3099
663b3df249f614305792ec7bf1dead30	14.2266
bb5d927e9f1aefa4db1881579e055a3e	14.0999
61d2370d7ccfe63adf6ab388d7415dc2	13.8604
9cb19c3fc7311aae01cf16571b528001	13.8139
4f672b7c5d7a4214b08ec6906163c980	13.6632
ca5cc4f526fcff10e85ce2685999e2c1	13.5957
2b75f007b50d6b21e1501f47dc8d632b	13.5395
cd1751286a561d51655ee47246b3019d	13.4713
7108b2bb5964a79ee1e6c8a9acde8107	13.3198
d6415e9e368a32ba39f756716c94e754	13.2148
4dfc88995d296a1fc700130826eb7fec	13.2085
08cf1040da9bd693c2373478bc984dac	13.0380
9690217261010ad90c43c1dd1d058e9c	12.9218

CustomerID	PredictedValues
2b5b01bb4d4c38be8af78817bc6746d7	12.8030
46454d378a1332750c086fb1101c07ce	12.7213
dca76db00cc59dfbcdcc97c8bbc7f9f1	12.6252
725d0a526feea495ab917ad7f8262765	12.4616
a6d16066cc225139ddf01aa9fd8723aa	12.4207
db0bd3b5ec2a35041de7436c699b215a	12.3825
0425ae0d99a8e31c2758c25cca4ac2f7	12.3684
1351b2fa6837a95abcf62a86054bdd12	12.3175
d6cef078357c829aed16490e7fee5aa2	12.1044
6f76cb699421405a52ad7c265af18d53	12.1002
63d51858691a5b69e063632e054a085c	11.9465
f11a653807d6bde53e4757f53e098c96	11.8195
6bd3cf2b2bf52ee22bef14f87d1f9f5b	11.8078
2ae416321f35d9ee6670ba973a7027e0	11.7557
29505921638c03d201f08fd602dee9ab	11.7514
1d9705cdc81922b119643d926e8b11de	11.6728
42f0d47cbd705d7da473b5d3e5cabc6c	11.6604
580a57e214a9d4df715accd2fe7065ba	11.5283
8118d20dc12ff07a429b76488bbf8d16	11.5097
765428c94ccfd7f17127a1bb12da3230	11.4700
c3cbbdb19ce41c5c3df16cbfb39b1ec8	11.3277
7863835a6120bfa52ccd665bfe03e9c8	11.3170
d24ac002f1ba9f9d4869c9a1a07620a2	11.2956
1a89258151f74319fd50778b0ed2f4f5	11.1299
e3bd9c3b75a87afe59a7d235693bd35a	11.0210
a434093fbf07b7f274ff64b472513cc3	10.9718
5c7eec1661af8e5b46f1e227fae6e1ee	10.7930
bef6f77f9664bf346550495098cd5796	10.7863
68f0db4c3018522a7c24607851afd762	10.7567
5afd521e378b4546fa52576528b9c1cc	10.6519
c84aa8c190410b584f7d830f41ac681a	10.5904
91b813ad50e0974adbb1461ed083237d	10.5581
23d45be9a8057f64e1103185e53e4f2a	10.5287
5cde4c0e001f042b081bf18070c53f4e	10.5061
cff00f3b67e59b8c9a89972c6cbdabf5	10.3902

CustomerID	PredictedValues
22101eef0531a7087dc997fac3d6a7c5	10.3786
c505f6e87ff9782ba4ca791ae088f329	10.3314
14f57af72e3d2a16b7b4c088cbe3196f	10.2485
13cb5c06f8d3c1c18e14539c16088e21	10.2401
1200a4df001756f0fc97e0142c069442	10.2095
922bdaa119d44065b17b0447255646ae	10.1919
bcf0ea738c6bb798b7d05f2603f8fccf	10.1298
6d61ac78f193cf9237f3ebfeede61704	10.1268
d0ad6b624dfc37784a755144be8c76a6	10.1225
15e20f36220dd72101f937433465f328	10.1174
25e5fe3494dcb7d0de25fcd6f6b499d9	10.0985
47810b7c7fc24d3cf8287d03de62448b	10.0869
80ed49605a244a6b8df7a0185125075d	10.0850
b32f83615c36a983c147d57d68f20804	10.0610
a92534133444b5028d12a129b1b128f7	10.0496
d88c16f53bf1f538e5dad005530611a9	9.9705
52558a8c536649d21e274da275ce1c8e	9.9388
326ede7ae03f35a1c596fd5e9029586f	9.9185
2da10b67a4882e14d257c728eda763c3	9.9111
0b937997e6f55b43f95a14c1c6fe94e2	9.8886
51b3b64735d01118ff09427f8082277a	9.8500
feeb6791f303fd34059c06d368f218e3	9.7969
6116c18166249fed2e39ee8f3a0733cf	9.7784
052a3c248db5edab01d32871d5acdbfa	9.7601
9459310958c7741a71359ef3a151b9ed	9.7561
4278f21f39173ebb616faa118c22384a	9.7521
bcf1f968bf4bf2b9241f77366a4aee86	9.7415
42059a7ede026d409ff0f255635d7a08	9.7382
ea623f715c09492b8a565e2b8dfff305	9.6899
e3a48ef6277d4ee1f0dffe0b50cc9a99	9.6665
51bf88a0aaf6da3d94ab246c9c857017	9.6104

List the 100 customers predicted to spend the most over the next 12 months.

To calculate this we use the Gamma-Gamma model. We filter out the customer who have no repeat purchases i.e. frequency is 0

In [18]:
```
frequency_filtered_customers = data[data['frequency'] > 0]

ggf = GammaGammaFitter(penalizer_coef = 0)
ggf.fit(frequency_filtered_customers['frequency'],
        frequency_filtered_customers['monetary_value'])
```

Out[18]:
```
<lifetimes.GammaGammaFitter: fitted with 9647 subjects, p: 4.57, q: 3.65, v: 135.88>
```

In [19]:
```
predicted_spend = ggf.customer_lifetime_value(
    mbg,
    data['frequency'],
    data['recency'],
    data['T'],
    data['monetary_value'],
    time = 365,
    freq = 'D'
    )
```

In [20]:
```
predicted_spend = pd.DataFrame(predicted_spend)
```

In [21]:
```
pd.set_option('display.max_rows', None)
predicted_spend.nlargest(100, 'clv')
```

Out[21]:

CustomerID	clv
a5fadc51b1ae844ad9a70ad3cfa46a64	119961.2341
ca2202a96c2de6ca6b8a37a4a73fa730	68488.2378
dca76db00cc59dfbcdcc97c8bbc7f9f1	51609.8739
5ac5ed64cd99ed2a8403b7a927e644ef	44989.8911
60c19a709e3ced2d16d7100eb1069df5	41079.6040
5f01420f0edda6555df5ce1cc62b986c	32223.4828
742d5a52d4df7cb14246d7f390de5d8a	31082.3446
a719d6643a7832535de9aded2f467825	30322.9973
ed2b4332b3ca253cfbb0ffb54d3f5ae0	29955.1282
eba458987dc67827871c1d4d92e646e1	29438.2539
48a503edbaded96a3be27deee11967a1	29412.7252
089ecc49200cfe79584d0bec2a3cf8c0	28668.1032
9cb19c3fc7311aae01cf16571b528001	28176.0523
98f8e41f45721cbe49a3147f6cf62432	27698.9294
66162981fc95e268e45bbfc738059687	27480.9650
a62a17bb46864da2c6da691d838971b3	27438.0849
cd4cb9ec252a085ed4d2d3af7c18280a	27427.4436
a92534133444b5028d12a129b1b128f7	26261.6716
42059a7ede026d409ff0f255635d7a08	26224.9198
2b75f007b50d6b21e1501f47dc8d632b	26051.1835
922bdaa119d44065b17b0447255646ae	25911.2186
30aa99d3357244cf38ca04eadef1473a	25317.8974
24f05bfab01fef56ec049a828ebe20ab	24181.1942
f09ff1c6c4ac8ea95d8621a94bb325fd	23864.8641
ede2a476c3894cf65d1619987d148422	23735.9051
f11a653807d6bde53e4757f53e098c96	23054.3424
2f486887c2edb2571d32c8cd15301711	22504.8950
4f672b7c5d7a4214b08ec6906163c980	21094.9354
7863835a6120bfa52ccd665bfe03e9c8	20935.4573
a63d0d4fe5bc662678bbbe6fbde1d900	20703.5008
741bbe09f8795badb5292473bff42ead	20426.1799
880474a82e33389ae7f377b2ea35bbdd	20118.4329
1200a4df001756f0fc97e0142c069442	20074.8604
8b6aa340953dad198a9d60bd12abfcb9	19994.0068

CustomerID	clv
0b9f48aef3295165238b3b14fffb981a	19714.5905
fe403ffcf47b4efdf39874d181ae6da4	19378.5352
0b937997e6f55b43f95a14c1c6fe94e2	19211.6912
80b4fe892813996f469f44c28a0d1c10	18710.1501
ca84dc4660f651a11a82041300367fae	18676.6699
56a3c0b28932e67c65c14efd9214f732	18655.7179
60b884e115b971ee52f550a8b64750f4	17929.4366
79d35a58b719d48ca15d7a573f8a2379	17862.7458
a033783169cd08fea7bcc7189b4859a5	17669.3284
ea6bd67206a400c56a02ef8f2e4bf01e	17079.6923
fc9447df681b99d9d9b74ea0056aa39a	16914.4762
5911b41fe8c5cd07a67571724f073e20	16851.7750
de70325370400abbce654bb35511812d	16785.3625
51b3b64735d01118ff09427f8082277a	16619.2817
29505921638c03d201f08fd602dee9ab	16605.2753
75fda9ea22086bf3814ff8c3f53de8ca	16560.9703
725d0a526feea495ab917ad7f8262765	16482.6283
d6cef078357c829aed16490e7fee5aa2	16301.9765
e4d423cc47a17c9ca7cea0c12e8e22be	16238.0574
f2114d783824ed6c7c2658be58579e3e	16117.0897
801266226172319918cca5963b492beb	16063.8086
a8c6dd0d4f1b33878dfc2f3559e88f87	16053.9779
a15d0cac4e30d80cc140913d01671552	15931.4906
a48de7bf6243a47341438a0e50bb2434	15910.8234
51ff794d843ce28fb7ca8885094979a7	15743.7362
e06bb6237a574b7216bf557a6477b10d	15534.2473
63d51858691a5b69e063632e054a085c	15379.9075
7b054e89ebafd6033a234ec267bc60b5	15238.5026
8291cd5d95841c98839e8f58d5816cf8	15138.7916
25e5fe3494dcb7d0de25fcd6f6b499d9	15117.8760
e0d591d2bbc656a278b5675d9bcae6c3	15094.8493
52558a8c536649d21e274da275ce1c8e	15083.7351
ba5cf27a059fb5e6357dc3e7f70dbab2	15019.2305
23a5e733f6dfdb2ebde078d407c37031	15016.3922
af77cebc5448b68b026556858d60c8cc	14983.8186

CustomerID	clv
6dce8b4af8ca65025bc7f84457ae13a1	14964.7076
52cc702cd8c70995cf6ba1c762e341d4	14952.8448
cda380d6b9e87ecf02a85e994622131c	14918.4439
cff00f3b67e59b8c9a89972c6cbdabf5	14566.8626
ffe8da9e101a7d6c33b8c3be8eb705bc	14483.3126
ab30a46cc9f7d41f0ee382ff9dd48ad0	14446.4637
bfd2558c6ca88693b5721952422b5b40	14332.6182
4278f21f39173ebb616faa118c22384a	14294.7594
b783899d4ba9b328769e55d592fa8deb	14283.9350
6061e84cb60705e4a2a378538353ba4d	14216.6047
607b2c3d2eb689b96ebf551fe735e203	14199.1653
2143bad4fa8805114a00a73853ac6ace	14154.8406
6f9ebbe87978c734ba71c0032e1f3e45	14153.6232
2e80c8b5d073f8310276f3ddfa2a1130	14003.3271
d80a3467a4f6fa760b9b1d2769f03295	13979.4718
6df09da9103ec49551e93232f684c323	13908.5783
323f6b3b7a7b0d114ea4bf79b6014241	13774.1038
16134915d822fe17588ae585935e1e81	13764.3010
8ab18aff369ee1c0ca9a7abeefd81219	13709.1922
970c6dfdd3bcc297b26100a96a9a06fe	13698.8870
68f0db4c3018522a7c24607851afd762	13690.2303
0cb0b9356e0a85830bef4f4fda74f92b	13597.7288
d6415e9e368a32ba39f756716c94e754	13482.2621
34bff14a781e6eabd03f59ac8f610ed7	13412.1863
0ae482faaa8baa024c5fcb1346c5f97e	13392.3146
9b306f73da56027310069b44074ef765	13255.8436
35bd91d013d04ffe65a66c2864be2c63	13252.2235
e74b7b4af8d956ed39d60c35fac0402c	13149.1734
580a57e214a9d4df715accd2fe7065ba	13017.4569
361251645f911412c958777a7068f8f4	12966.0215
0368729f6f064e2f961cc22bfe5d60a1	12930.6708

Analysis

Assumptions for Modified Beta Gemetric Model:

Assumption 1 - While active, the number of transactions made by a customer follows a poisson process with transaction rate λ. This is equivalent to assuming that the time between transactions is distributed exponential with transaction rate λ, i.e.,

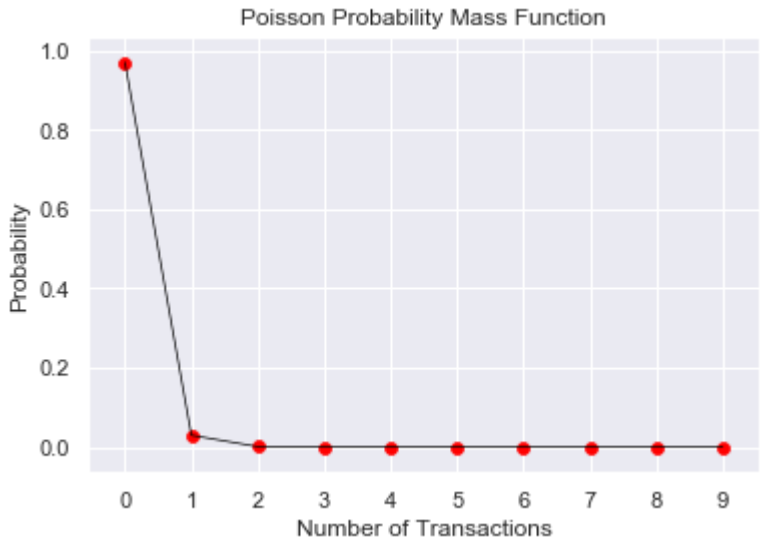
$$f(tj|tj-1;\lambda) = \lambda e - \lambda(tj - tj - 1), tj > tj - 1 \geq 0$$

Explanation - At every sub-period of a specific time interval, each customer tosses his buy coin and depending on the result, he purchases or not. The number of transactions (heads) we observe in the period depends on each customer's probability distribution around λ and not on his recent buying behavior.

In [22]:
```
from scipy.stats import poisson
poisson_lambda = 0.03
p_arr = []

distribution = poisson(poisson_lambda)
for transactions in range(0,10):
     p_arr.append(distribution.pmf(transactions))

plt.ylabel('Probability')
plt.xlabel('Number of Transactions')
plt.xticks(range(0, 10))
plt.title('Poisson Probability Mass Function')
plt.plot(p_arr, color='black', linewidth=0.7, zorder=1)
plt.scatter(range(0, 10), p_arr, color='red', linewidth=0.7)
plt.show()
```



Assumption 2 - The transaction rate λ and the dropout probability p vary independently across customers.

Explanation - This assumption captures an individual customer's chances of not purchasing again and the customer's rate of transaction. It states that it is different for each customer and does not depend on any other individual.

Assumption 3 - Heterogeneity in p follows a beta distribution, with parameters a and b.

$$f(p|a,b) = pa - 1(1-p)b - 1B(a,b), 0 \leq p \leq 1.$$

Heterogeneity in transaction rates among customers follows a Gamma distribution

$$f(\lambda|r,\alpha) = \alpha r \lambda r - 1e - \lambda\alpha\Gamma(r), \lambda > 0.$$

Explanation - To capture each customer's individual behaviour, we assume that each individual's probability of dropping out at the end of a purchase period can take on any one of an infinite number of possible values between 0 and 1, this is achieved by assuming that variability in these probabilities across customers is captured by a continuous probability distribution or a beta distribution.

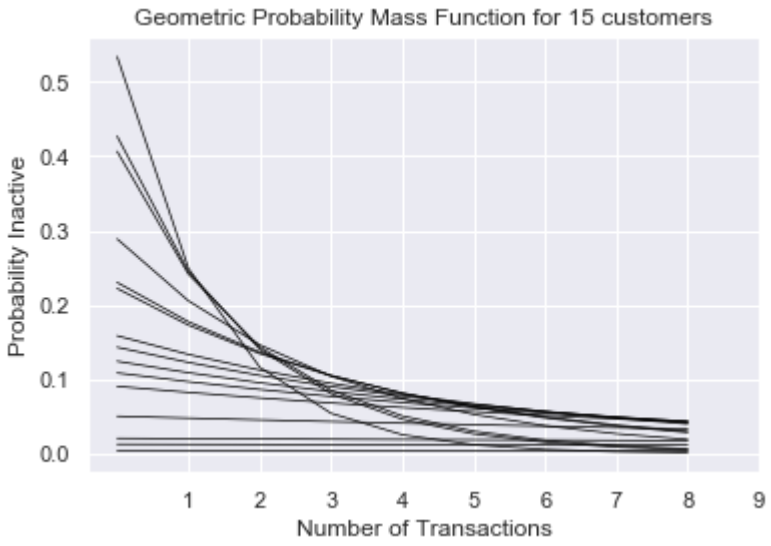
As Gamma distribution can take different shapes, it can capture the transactions for a customer, which can occur at any point in time, in a much better sense. The r controls the shape and skewness of distribution, while larger α leads to wider spread of the distribution.

In [23]:
```
beta_a = 0.54
beta_b = 1.45

for customer in range(0, 15):
    p_arr = []
    beta  = np.random.beta(a=beta_a, b=beta_b)
    for transaction in range(1,10):
        proba_inactive = beta*(1-beta)**(transaction-1)
        p_arr.append(proba_inactive)
    p_arr = np.array(p_arr)
    plt.plot(p_arr, color='black', linewidth=0.7)

plt.ylabel('Probability Inactive')
plt.xlabel('Number of Transactions')
plt.xticks(range(1, 10))
plt.title('Geometric Probability Mass Function for 15 customers')
plt.show
```

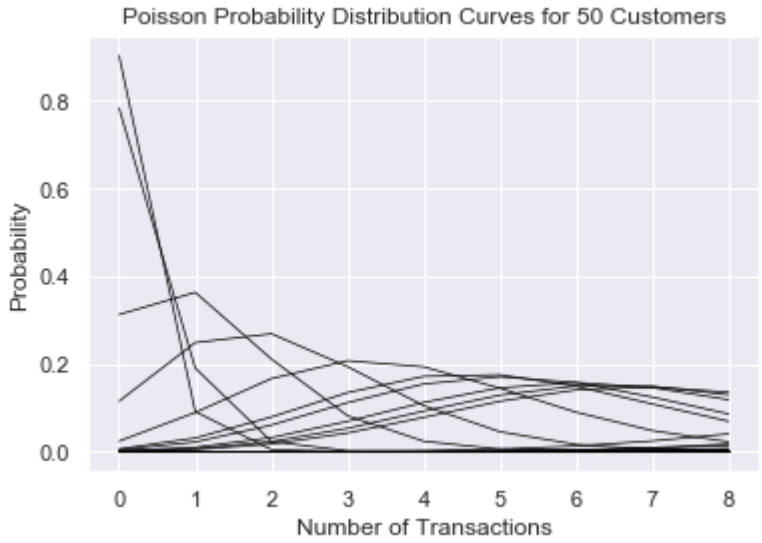
Out[23]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [24]:
```
gamma_shape = 0.79
gamma_scale = 85.27

for customer in range(0, 50):
    distribution = poisson(np.random.gamma(shape=gamma_shape, scale=gamma_scal
e))
    p_arr = []
    for transactions in range(0,9):
        p_arr.append(distribution.pmf(transactions))
    plt.plot(p_arr, color='black', linewidth=0.7)

plt.ylabel('Probability')
plt.xlabel('Number of Transactions')
plt.xticks(range(0,9))
plt.title('Poisson Probability Distribution Curves for 50 Customers')
plt.show()
```



Assumption 4 - After any transaction, a customer becomes inactive with probability p. Therefore the point at which the customer drops out is distributed across transactions according to a (shifted) geometric distribution.

P(inactive immediately after j^{th} transaction) $= p(1-p)j-1, j = 1, 2, 3, \ldots.$

Explanation - At the end of each period, a customer decides whether or not to drop-out. It is like saying that the customer tosses a coin to decide this, where T is the result when the customer will stop purchasing. The coin is not a fair coin and the customer continues his purchases till he gets a T as the result of coin toss.

Assumption 5 - A customer can drop out immediately after the first purchase

Explanation - An improvement over BG model, this assumption helps to make the model flexible. This assumption helps to identify the customers who just buy once and never return.

Assumptions for Gamma - Gamma Model:

Assumption 1 - For each individual customer, the profit per transaction is independent of the number of transactions

Assumption 2 - Expected profit per transaction does not vary over time

Assumption 3 - The monetary value of a customer's given transaction varies randomly around their average transaction value.

The application of the MBG/NBD model in an ecommerce store environment shows that the model captures a customer's purchase and dropping out process in an accurate way. These assumptions appear to be valid given that they are able to forecast the future purchase patterns of each customer. Also by using modified beta geometric model, we are able to capture the additional chance of dropout at time zero which particularly matches the ecommerce purchase behavior.

Simulation:

a. Generate a random sample of 100 customers.

Creating a sample data of 100 unique customers

In [25]:
```
random_list_customers = random.sample(list(df['CustomerID'].unique()), 100)
```

In [26]:
```
sample_100_data = df[df['CustomerID'].isin(random_list_customers)]
sample_100_data['CustomerID'].nunique()
```

Out[26]: 100

In [27]:
```
sample_100_data.head()
```

Out[27]:

	CustomerID	Timestamp	PurchaseValue
174	11976e2d4bec7421a1c106eec5a2c1e2	2017-08-15	23.7900
237	11976e2d4bec7421a1c106eec5a2c1e2	2017-03-10	26.7900
569	5970775279f05a38346cf4b7ea569d49	2017-07-15	33.6800
617	8ab18aff369ee1c0ca9a7abeefd81219	2017-06-22	34.5100
732	8118d20dc12ff07a429b76488bbf8d16	2017-05-06	36.1200

In [28]:
```
sample_100_data.shape
```

Out[28]:
```
(245, 3)
```

b. Simulate how each customer makes purchases over time.

Creating a summary data from the given transaction data

In [29]:
```
data_simulated = summary_data_from_transaction_data(sample_100_data, 'Customer ID', 'Timestamp',
                                                    monetary_value_col='PurchaseValue', observation_period_end = '2017-12-6')
```

In [30]:
```
data_simulated.head()
```

Out[30]:

CustomerID	frequency	recency	T	monetary_value
0cd591e7ed264e869d847cf61c81072b	2.0000	158.0000	187.0000	117.6750
0df73e46e0d5cbfcd1b85f1ab884d466	2.0000	162.0000	279.0000	170.5600
0f21342cc3c25d73c61bcde5db07a2c9	0.0000	0.0000	367.0000	0.0000
11106c61eb61a87a8bfc93482d8be237	0.0000	0.0000	51.0000	0.0000
116625aa406b86e5762811937275b0dd	0.0000	0.0000	46.0000	0.0000

In [31]:
```
mbgs = ModifiedBetaGeoFitter(penalizer_coef=0.0)

mbgs.fit(data_simulated['frequency'], data_simulated['recency'], data_simulated['T'])
```

Out[31]:
```
<lifetimes.ModifiedBetaGeoFitter: fitted with 100 subjects, a: 0.00, alpha: 118.46, b: 0.00, r: 1.26>
```

c. Count how many purchases the customers have made.

Creating a function to estimate the the purchases a customer would have made in a future time 't':

In [32]:
```
def future_purchases(t):

    predicted_purchases = mbgs.conditional_expected_number_of_purchases_up_to_
time(t, data_simulated['frequency'],

data_simulated['recency'], data_simulated['T'])

    predicted_purchases = pd.DataFrame(predicted_purchases, columns = ['Predic
tedValues'])

    return predicted_purchases
```

d. Count how many customers are alive after 10 days.

From *Batislam, E. M., Denizel, M., & Filiztekin, A. (2007)*, we get the formula for predicting the probability that the customer is alive at time 't' as:

$$P(Active|x, t_x, t, r, \alpha, a, b) = \frac{1}{1 + \frac{\Gamma(a+1)*\Gamma(b+x)}{\Gamma(a)*\Gamma(b+x+1)}(\frac{\alpha+t}{\alpha+t_x})^{r+x}}$$

We get the values for r, alpha, a, b from the following:

In [33]:
```
mbgs.params_
```

Out[33]:
```
r         1.2598
alpha   118.4593
a         0.0000
b         0.0000
dtype: float64
```

Creating a function to estimate the probability of the customers alive at time 't'

In [34]:
```
def future_probability_alive(t):

    future_probability = 1 / (1 + ((gamma(mbgs.params_['a'] + 1) * gamma(mbgs.
params_['b'] + data_simulated['frequency'])) /
                (gamma(mbgs.params_['a']) * gamma(mbgs.params_['b'] + data_simula
ted['frequency'] + 1))) *
                (((mbgs.params_['alpha'] + t) / (mbgs.params_['alpha'] + data_sim
ulated['recency'])) ** (mbgs.params_['r'] + data_simulated['frequency'])))

    future_probability = pd.DataFrame(future_probability, columns = ['Probabil
ityAlive'])

    return future_probability
```

For 10 Davs

The purchases the customers would make in 10 days are:

In [35]:
```
future_purchases(10).sum()
```

Out[35]:
```
PredictedValues   5.7839
dtype: float64
```

The number of customers alive after 10 days with alive probability greater than 0.5 are:

In [36]:
```
future_10 = future_probability_alive(10)
```

In [37]:
```
future_10.loc[future_10['ProbabilityAlive'] >= 0.5].count()
```

Out[37]:
```
ProbabilityAlive    100
dtype: int64
```

e. Repeat b-d for 1 year, 10 years, 100 years.

For 1 year

The purchases the customers would make in 365 days or 1 year are:

In [38]:
```
future_purchases(365).sum()
```

Out[38]:
```
PredictedValues   211.1140
dtype: float64
```

The number of customers alive after 365 days or 1 year with alive probability greater than 0.5 are:

In [39]:
```
future_365 = future_probability_alive(365)
```

In [40]:
```
future_365.loc[future_365['ProbabilityAlive'] >= 0.5].count()
```

Out[40]:
```
ProbabilityAlive    36
dtype: int64
```

For 10 years

The purchases the customers would make in 3652 days or 10 years are:

In [41]:
```
future_purchases(3652).sum()
```

Out[41]:
```
PredictedValues   2112.2964
dtype: float64
```

The number of customers alive after 3652 days or 10 years with alive probability greater than 0.5 are:

In [42]:
```
future_3652 = future_probability_alive(3652)
```

In [43]:
```
future_3652.loc[future_3652['ProbabilityAlive'] >= 0.5].count()
```

Out[43]:
```
ProbabilityAlive    33
dtype: int64
```

The number of customers alive after 3652 days or 10 years with alive probability greater than 0.1 are:

In [44]:
```
future_3652.loc[future_3652['ProbabilityAlive'] >= 0.1].count()
```

Out[44]:
```
ProbabilityAlive    35
dtype: int64
```

For 100 years

The purchases the customers would make in 36524 days or 100 years are:

In [45]:
```
future_purchases(36524).sum()
```

Out[45]:
```
PredictedValues   21125.2779
dtype: float64
```

The number of customers alive after 36524 days or 100 years with alive probability greater than 0.5 are:

In [46]:
```
future_36524 = future_probability_alive(36524)
```

In [47]:
```
future_36524.loc[future_36524['ProbabilityAlive'] >= 0.5].count()
```

Out[47]:
```
ProbabilityAlive    25
dtype: int64
```

The number of customers alive after 36524 days or 100 years with alive probability greater than 0.05 are:

In [48]:
```
future_36524.loc[future_36524['ProbabilityAlive'] >= 0.05].count()
```

Out[48]:
```
ProbabilityAlive    25
dtype: int64
```

Some strengths of the model -

1) The model performs pretty well to estimate a customer's characteristics like future purchases and drop out probability

2) The model does well to predict what a group of customers, having the same characteristics will do collectively with respect to the number of purchases.

3) Due to the assumption that the customer can drop out after first purchase, the modified beta geometric model tends to fit better than beta geometric model

Some deficiencies of the model -

1) The model struggles to understand that the customer's behaviour may change overtime. For example, if a customer builds a trust in a particular product or starts disliking a particular product, it may affect his purchasing patterns.

2) The model struggles to take into account a customer's purchase pattern during some promotional events. The promotional events may give a rise to a customer's rate of purchases disturbing the pattern of recency and frequency of purchases. Also the model struggles to capture the seasonality of sales. For example, increased sales during Christmas.

3) The customers who have a high probability of being active in their last purchase remain with an active status if no other purchase is observed. A major misclassification of active status might take place for customers with a very low number of repeat purchases in the observation period which might not be sufficient to specify their purchase pattern.

How can we address the deficiencies of the model?

If we observe the seasonality problem in the model, we can address this issue by bringing in seasonality covariates, which would help to reduce the prediction gaps in the seasonal purchases. If we split the data by taking out the parts of data where seasonality occurs and fit it separately it might help to predict the seasonality in a much better way. Bringing in cyclicality related to business, customers and macroeconomy will also help to address this issue.