In [1]:

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

import plotly.express as px
import xlrd
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import datetime
np.random.seed(42)
```

In [2]:

```
plt.style.use("ggplot")
```

In [3]:

```
df = pd.read_excel("online_retail_data.xlsx", sheet_name = ["Year 2009-2010", "Year 2010-20
```

In [4]:

```
df1 = df["Year 2009-2010"]
df2 = df["Year 2010-2011"]
```

In [5]:

```
df1.shape, df2.shape
```

Out[5]:

```
((525461, 8), (541910, 8))
```

In [6]:

```
sum([df1.shape[0], df2.shape[0]])
```

Out[6]:

```
1067371
```

In [7]:

```
data = df1.append(df2)
```

In [8]:

```
data.head(5)
```

Out[8]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [9]:

```
data.shape
```

Out[9]:

```
(1067371, 8)
```

In [10]:

```
data.isnull().sum()/data.shape[0]*100
```

Out[10]:

```
Invoice         0.000000
StockCode       0.000000
Description     0.410541
Quantity        0.000000
InvoiceDate     0.000000
Price           0.000000
Customer ID    22.766873
Country         0.000000
dtype: float64
```

In [11]:

```
data.dropna(axis = 0, subset = ["Description"], inplace = True) #drop the null discription
```

In [12]:

```
data.isnull().sum() #Now let's check the data before droping these customer ID's
```

Out[12]:

```
Invoice             0
StockCode           0
Description         0
Quantity            0
InvoiceDate         0
Price               0
Customer ID    238625
Country             0
dtype: int64
```

In [13]:

```
data[data["Customer ID"].isnull()]
```

Out[13]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
263	489464	21733	85123a mixed	-96	2009-12-01 10:52:00	0.00	NaN	United Kingdom
283	489463	71477	short	-240	2009-12-01 10:52:00	0.00	NaN	United Kingdom
284	489467	85123A	21733 mixed	-192	2009-12-01 10:53:00	0.00	NaN	United Kingdom
577	489525	85226C	BLUE PULL BACK RACING CAR	1	2009-12-01 11:49:00	0.55	NaN	United Kingdom
578	489525	85227	SET/6 3D KIT CARDS FOR KIDS	1	2009-12-01 11:49:00	0.85	NaN	United Kingdom
...	...	...	...	...	...	...	...	...
541536	581498	85099B	JUMBO BAG RED RETROSPOT	5	2011-12-09 10:26:00	4.13	NaN	United Kingdom
541537	581498	85099C	JUMBO BAG BAROQUE BLACK WHITE	4	2011-12-09 10:26:00	4.13	NaN	United Kingdom
541538	581498	85150	LADIES & GENTLEMEN METAL SIGN	1	2011-12-09 10:26:00	4.96	NaN	United Kingdom
541539	581498	85174	S/4 CACTI CANDLES	1	2011-12-09 10:26:00	10.79	NaN	United Kingdom
541540	581498	DOT	DOTCOM POSTAGE	1	2011-12-09 10:26:00	1714.17	NaN	United Kingdom

238625 rows × 8 columns

In [14]:

```
data.iloc[575:, :]
```

Out[14]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
576	C489524	21258	VICTORIAN SEWING BOX LARGE	-1	2009-12-01 11:48:00	12.75	15614.0	Unitec Kingdom
577	489525	85226C	BLUE PULL BACK RACING CAR	1	2009-12-01 11:49:00	0.55	NaN	Unitec Kingdom
578	489525	85227	SET/6 3D KIT CARDS FOR KIDS	1	2009-12-01 11:49:00	0.85	NaN	Unitec Kingdom
579	489526	85049E	SCANDINAVIAN REDS RIBBONS	12	2009-12-01 11:50:00	1.25	12533.0	Germany
580	489526	21976	PACK OF 60 MUSHROOM CAKE CASES	24	2009-12-01 11:50:00	0.55	12533.0	Germany
...	...	...	...	...	...	...	...	..
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France
541909	581587	POST	POSTAGE	1	2011-12-09 12:50:00	18.00	12680.0	France

1062414 rows × 8 columns

In [15]:

```
data.dropna(axis = 0, subset = ["Customer ID"], inplace = True)
# We have to drop the rows where customer ID is null because it's a unique customer ID of e
```

In [16]:

```
data.isnull().sum()
```

Out[16]:

```
Invoice        0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
Price          0
Customer ID    0
Country        0
dtype: int64
```

In [17]:

```
data.head(10)
```

Out[17]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom
5	489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45:00	1.65	13085.0	United Kingdom
6	489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom
7	489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45:00	5.95	13085.0	United Kingdom
8	489435	22350	CAT BOWL	12	2009-12-01 07:46:00	2.55	13085.0	United Kingdom
9	489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46:00	3.75	13085.0	United Kingdom

In [18]:

```
temp_df = pd.DataFrame(data["Country"].value_counts())
```

In [19]:

```
temp_df.head(10)
```

Out[19]:

	Country
United Kingdom	741301
Germany	17624
EIRE	16195
France	14202
Netherlands	5140
Spain	3811
Belgium	3123
Switzerland	3064
Portugal	2504
Australia	1913

In [20]:

```
names  = temp_df.index
```

In [21]:

```
px.scatter(temp_df, y = "Country", color = names, title = "Count of Countries")
```

In [22]:

```
data.groupby("Country").sum()["Quantity"].sort_values(ascending = False).head(10)
#Now let's check the total quantity and by country
```

Out[22]:

```
Country
United Kingdom    8353502
Netherlands        381951
EIRE               313373
Denmark            235218
Germany            224581
France             183339
Australia          103706
Sweden              87737
Switzerland         51831
Spain               45156
Name: Quantity, dtype: int64
```

In [23]:

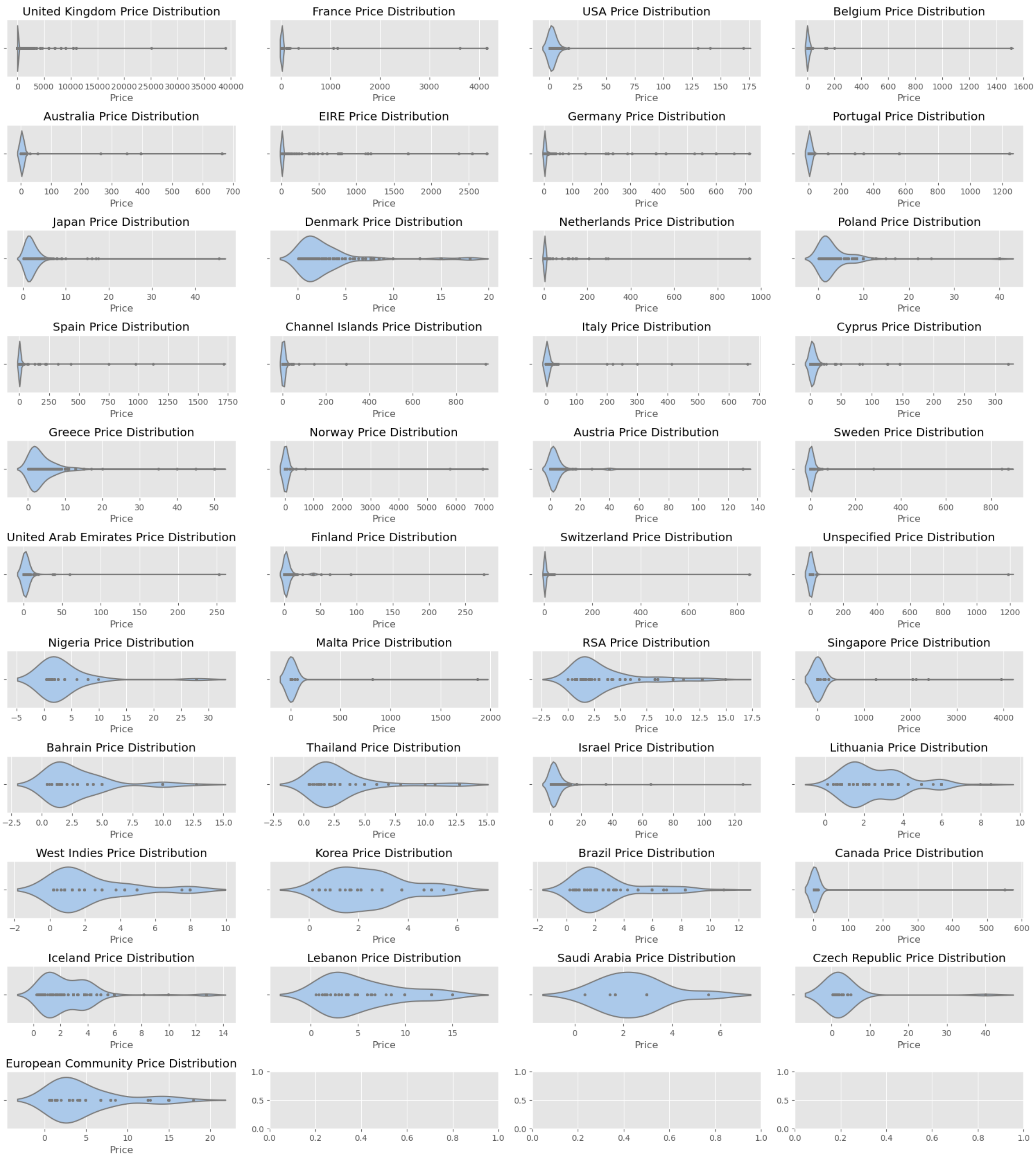
```
px.scatter(data.iloc[:, [5, 7]].groupby(["Country"]).mean(),
           y = "Price", size = "Price", title = "Average Price by Country", opacity = 0.48)
```

In [24]:

```
countries = data["Country"].unique()
```

In [25]:

```
fig, ax = plt.subplots(11, 4, figsize=(18,20))
axes_ = [axes_row for axes in ax for axes_row in axes]

for i, c in enumerate(countries):
    sns.violinplot(x = "Price", data = data[data["Country"] == c], ax = axes_[i], inner = "
    axes_[i].set_title(c + ' ' + "Price Distribution")
    plt.tight_layout()
```



In [26]:

```
#Total Number of Unique Invoices
len(data["Invoice"].unique())
```

Out[26]:

```
44876
```

In [27]:

```
temp_invoice_df = data.groupby("Invoice").sum()
```

In [28]:

```
temp_invoice_df.reset_index(inplace = True)
```

In [29]:

```
temp_invoice_df.sort_values(by = "Quantity", ascending = False).head(30).iloc[:,:2].style.b
#top 30 invoices with total quantity purchased
```

Out[29]:

	Invoice	Quantity
11080	518505	87167
13425	524174	87167
3064	497946	83774
36942	581483	80995
20348	541431	74215
4379	501534	63974
2096	495194	63302
4693	502269	40000
1604	493819	25018
1047	491812	20524
7529	509472	17766
246	490018	17520
16097	530715	15696
26551	556917	15049
29051	563076	14730
32810	572035	13392
12179	521315	13008
14557	526761	12954
2252	495591	12832
30858	567423	12572
35799	578841	12540
7239	508748	12500
24876	552883	12266
29282	563614	12196
18296	536009	12048
11136	518673	11904
28783	562439	11848
22980	548011	11116
21933	545475	10272
17933	535104	10014

In [30]:

```
data.groupby(["Invoice"]).mean().head(15).iloc[:, [1]].sort_values("Price", ascending = Fal
```

Out[30]:

	Price
Invoice	
489444	141.000000
489447	130.000000
489434	4.081250
489436	3.730526
489437	3.628261
489439	3.560000
489440	3.150000
489446	3.118519
489441	3.042500
489448	2.970000
489435	2.625000
489438	2.591176
489445	2.477895
489443	2.370000
489442	2.040870

In [31]:

```
data[(data["Invoice"] == 489444) | (data["Invoice"] == 489447)]
```

Out[31]:

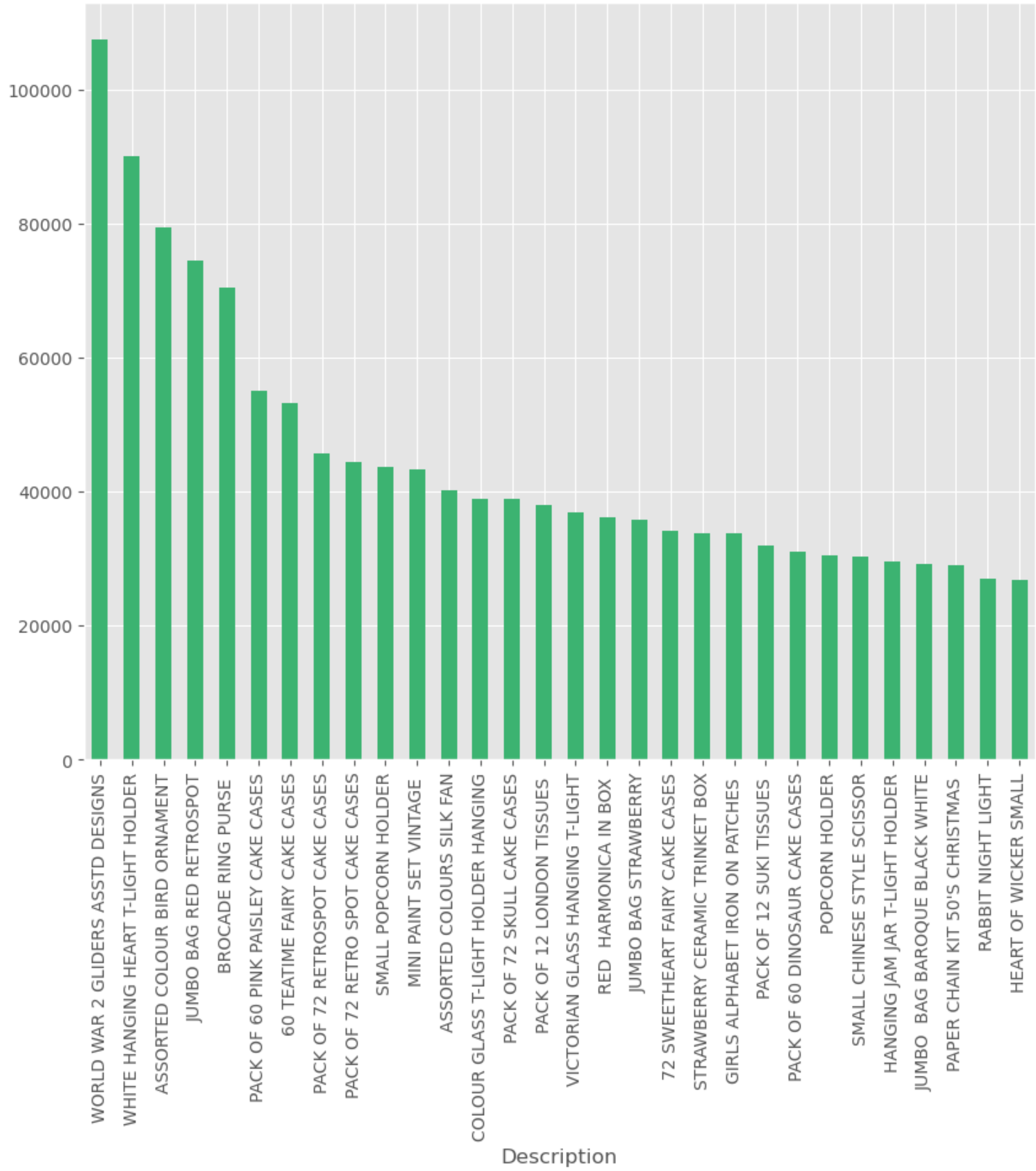
	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
126	489444	POST	POSTAGE	1	2009-12-01 09:55:00	141.0	12636.0	USA
173	489447	POST	POSTAGE	1	2009-12-01 10:10:00	130.0	12362.0	Belgium

In [32]:

```
#Let's check which product has been purchased more often so far
plt.figure(figsize=(10,8))
data.groupby("Description").sum().sort_values(by = "Quantity", ascending = False).head(30)[
```

Out[32]:

```
<AxesSubplot:xlabel='Description'>
```



In [33]:

```
temp_data = data.copy()
```

In [34]:

```
temp_data.head(5)
```

Out[34]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [35]:

```
#Date Time Analysis
temp_data.loc[:, "Month"] = data.InvoiceDate.dt.month
temp_data.loc[:, "Time"] = data.InvoiceDate.dt.time
temp_data.loc[:, "Year"] = data.InvoiceDate.dt.year
temp_data.loc[:, "Day"] = data.InvoiceDate.dt.day
temp_data.loc[:, "Quarter"] = data.InvoiceDate.dt.quarter
temp_data.loc[:, "Day of Week"] = data.InvoiceDate.dt.dayofweek
```

In [36]:

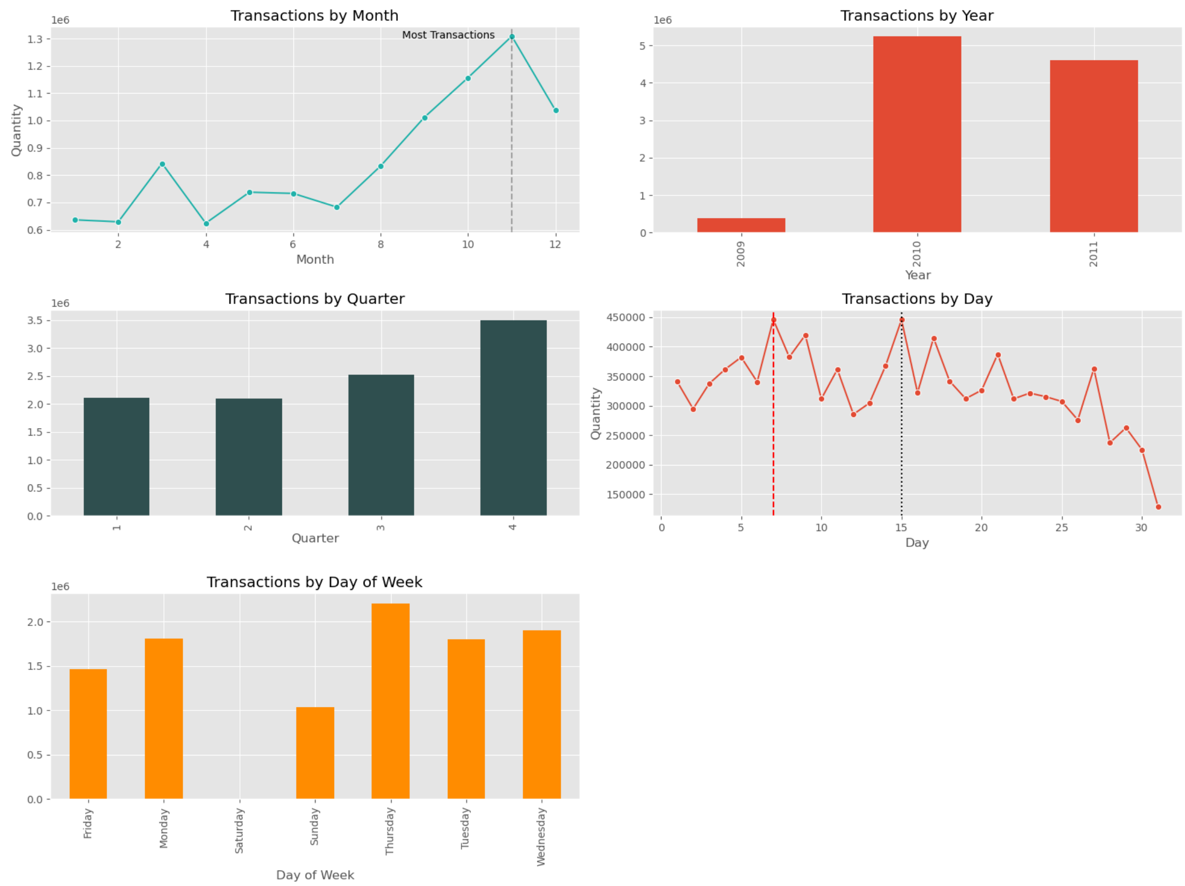
```
#Mapping day of week
dayofweek_mapping = dict({0: "Monday",
                         1: "Tuesday",
                         2: "Wednesday" ,
                         3: "Thursday",
                         4: "Friday",
                         5: "Saturday",
                         6: "Sunday"})
```

In [37]:

```
temp_data["Day of Week"] = temp_data["Day of Week"].map(dayofweek_mapping)
```

In [38]:

```
plt.figure(figsize=(16,12))
plt.subplot(3,2,1)
sns.lineplot(x = "Month", y = "Quantity", data = temp_data.groupby("Month").sum("Quantity")
plt.axvline(11, color = "k", linestyle = '--', alpha = 0.3)
plt.text(8.50, 1.3e6, "Most Transactions")
plt.title("Transactions by Month")
plt.subplot(3,2,2)
temp_data.groupby("Year").sum()["Quantity"].plot(kind = "bar")
plt.title("Transactions by Year")
plt.subplot(3,2,3)
temp_data.groupby("Quarter").sum()["Quantity"].plot(kind = "bar", color = "darkslategrey")
plt.title("Transactions by Quarter")
plt.subplot(3,2,4)
sns.lineplot(x = "Day", y = "Quantity", data = temp_data.groupby("Day").sum("Quantity"), ma
plt.axvline(7, color = 'r', linestyle = '--')
plt.axvline(15, color = 'k', linestyle = "dotted")
plt.title("Transactions by Day")
plt.subplot(3,2,5)
temp_data.groupby("Day of Week").sum()["Quantity"].plot(kind = "bar", color = "darkorange")
plt.title("Transactions by Day of Week")
plt.tight_layout()
plt.show()
```



In [39]:

```
print("Total Number of Countries in 2009: {}".format(len(temp_data[temp_data["Year"] == 200
print("Total Number of Transactions in 2009: {}".format(temp_data[temp_data["Year"] == 2009
print("---------------------------------------------")
print("Total Number of Countries in 2009: {}".format(len(temp_data[temp_data["Year"] == 201
print("Total Number of Transactions in 2009: {}".format(temp_data[temp_data["Year"] == 2010
print("---------------------------------------------")
print("Total Number of Countries in 2009: {}".format(len(temp_data[temp_data["Year"] == 201
print("Total Number of Transactions in 2009: {}".format(temp_data[temp_data["Year"] == 2011
```

```
Total Number of Countries in 2009: 23
Total Number of Transactions in 2009: 390286
---------------------------------------------
Total Number of Countries in 2009: 37
Total Number of Transactions in 2009: 5233315
---------------------------------------------
Total Number of Countries in 2009: 36
Total Number of Transactions in 2009: 4610527
```

In [40]:

```
_2009 = temp_data[temp_data["Year"] == 2009]["Country"].unique()
_2010 = temp_data[temp_data["Year"] == 2010]["Country"].unique()
_2011 = temp_data[temp_data["Year"] == 2011]["Country"].unique()
```

In [41]:

```
no_cols = []

for i in (_2010):
    if i not in _2009:
        no_cols.append(i)
print("These are the values which are not present in 2009: {}".format(no_cols))
```

```
These are the values which are not present in 2009: ['Unspecified', 'Nigeri
a', 'Malta', 'RSA', 'Singapore', 'Bahrain', 'Thailand', 'Israel', 'Lithuani
a', 'West Indies', 'Korea', 'Brazil', 'Canada', 'Iceland']
```

In [42]:

```
temp = data.groupby(["Country", "Description"]).sum()["Quantity"]
```

In [43]:

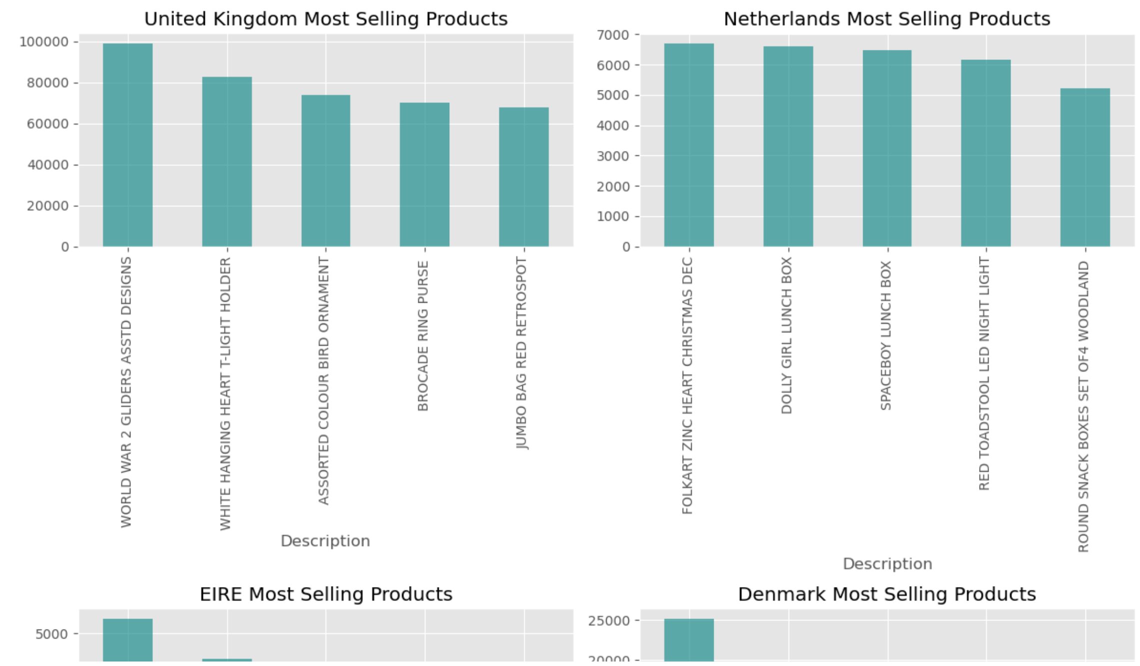
```
temp = pd.DataFrame(temp)
```

In [44]:

```
# top 8 countries with most transactions
top_8_countries = ["United Kingdom", "Netherlands", "EIRE", "Denmark", "Germany", "France",
```

In [45]:

```
x = 1
plt.figure(figsize=(12,24))
for x, c in enumerate(top_8_countries):
    ax = plt.subplot(4,2, x+1)
    plt.title(c + ' '+ "Most Selling Products")
    temp.loc[c].sort_values(by = "Quantity", ascending = False)["Quantity"].head(5).plot(ki
    plt.tight_layout()
```

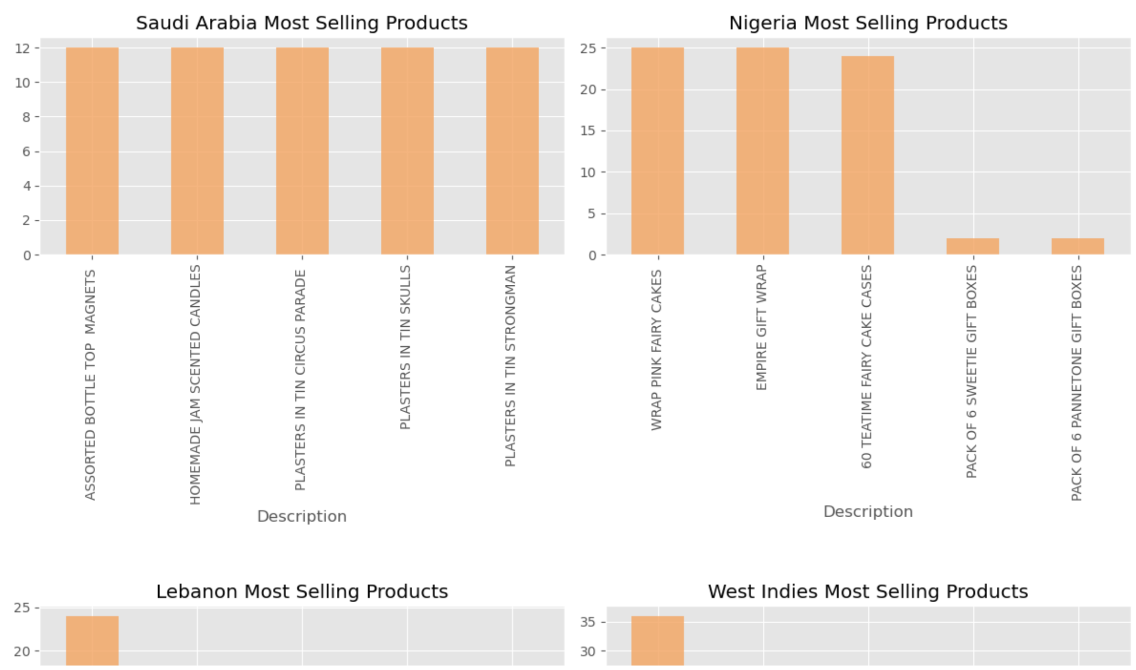


In [46]:

```
top 8 countries with least transactions
ast_8_country = ["Saudi Arabia", "Nigeria", "Lebanon", "West Indies", "European Community",
```

In [47]:

```
x = 1
plt.figure(figsize=(12,24))
for x, c in enumerate(least_8_country):
    ax = plt.subplot(4,2, x+1)
    plt.title(c + ' '+ "Most Selling Products")
    temp.loc[c].sort_values(by = "Quantity", ascending = False)["Quantity"].head(5).plot(ki

    plt.tight_layout()
```



In [48]:

```
# RFM Estimation - (Recency, Frequency, Monetary)
data["Total Amount"] = data["Quantity"]*data["Price"]
```

In [49]:

```
data.head()
```

Out[49]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	Tot Amou
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom	83
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom	100
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom	30

In [50]:

```
pip install lifetimes
```

```
Requirement already satisfied: lifetimes in c:\users\hp\anaconda3\lib\site-p
ackages (0.11.3)
Requirement already satisfied: pandas>=0.24.0 in c:\users\hp\anaconda3\lib\s
ite-packages (from lifetimes) (1.4.4)
Requirement already satisfied: numpy>=1.10.0 in c:\users\hp\anaconda3\lib\si
te-packages (from lifetimes) (1.21.5)
Requirement already satisfied: dill>=0.2.6 in c:\users\hp\anaconda3\lib\site
-packages (from lifetimes) (0.3.4)
Requirement already satisfied: scipy>=1.0.0 in c:\users\hp\anaconda3\lib\sit
e-packages (from lifetimes) (1.9.1)
Requirement already satisfied: autograd>=1.2.0 in c:\users\hp\anaconda3\lib
\site-packages (from lifetimes) (1.5)
Requirement already satisfied: future>=0.15.2 in c:\users\hp\anaconda3\lib\s
ite-packages (from autograd>=1.2.0->lifetimes) (0.18.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\hp\anacond
a3\lib\site-packages (from pandas>=0.24.0->lifetimes) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\anaconda3\lib\sit
e-packages (from pandas>=0.24.0->lifetimes) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.8.1->pandas>=0.24.0->lifetimes) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [51]:

```
import lifetimes
```

In [52]:

```
rfm_summary = lifetimes.utils.summary_data_from_transaction_data(data, "Customer ID", "Invo
```

In [53]:

```
rfm_summary.head()
```

Out[53]:

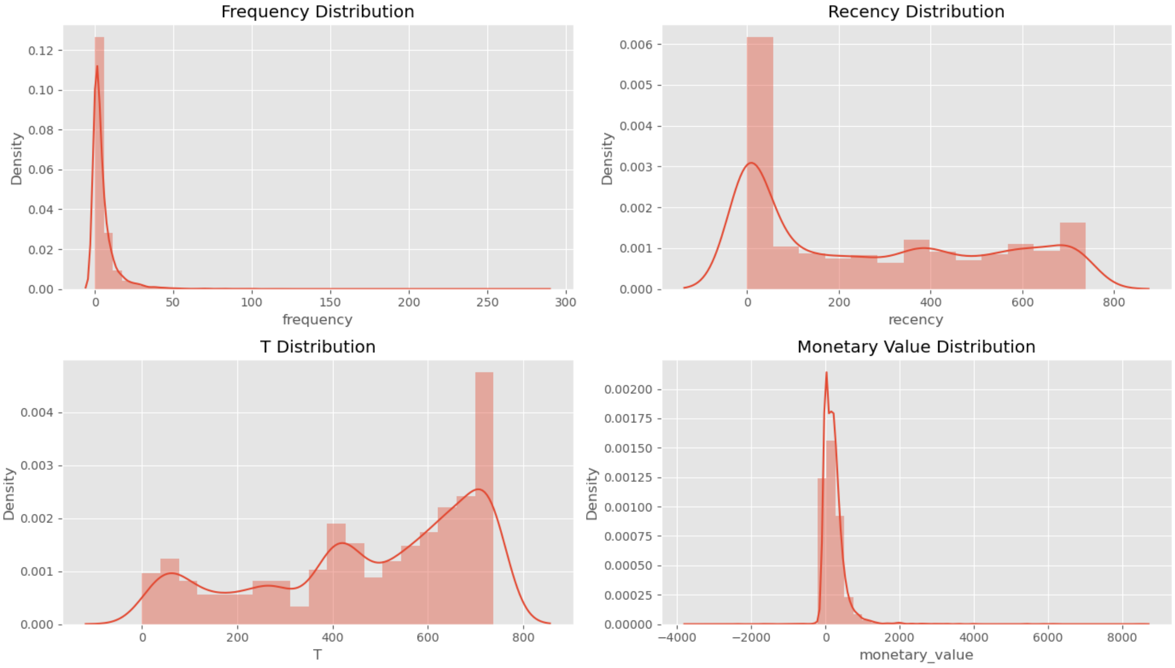
	frequency	recency	T	monetary_value
Customer ID				
12346.0	10.0	400.0	725.0	-15.468000
12347.0	7.0	402.0	404.0	717.398571
12348.0	4.0	363.0	438.0	449.310000
12349.0	4.0	717.0	735.0	1107.172500
12350.0	0.0	0.0	310.0	0.000000

In [54]:

```
rfm_summary.reset_index(inplace = True)
```

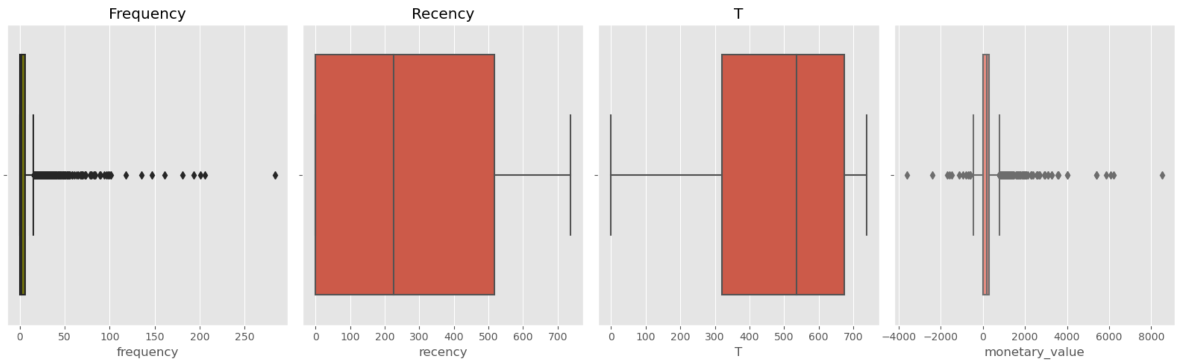
In [55]:

```
plt.figure(figsize=(14,8))
plt.subplot(221)
sns.distplot(rfm_summary["frequency"])
plt.title("Frequency Distribution")
plt.subplot(222)
sns.distplot(rfm_summary["recency"])
plt.title("Recency Distribution")
plt.subplot(223)
sns.distplot(rfm_summary["T"])
plt.title("T Distribution")
plt.subplot(224)
sns.distplot(rfm_summary["monetary_value"])
plt.title("Monetary Value Distribution")
plt.tight_layout()
```



In [56]:

```
plt.figure(figsize=(16,5))
plt.subplot(141)
sns.boxplot(rfm_summary["frequency"], color = "olive")
plt.title("Frequency")
plt.subplot(142)
sns.boxplot(rfm_summary["recency"])
plt.title("Recency")
plt.subplot(143)
sns.boxplot(rfm_summary["T"])
plt.title("T")
plt.subplot(144)
sns.boxplot(rfm_summary["monetary_value"], color = "salmon")

plt.tight_layout()
```



In [57]:

```
rfm_summary.describe(percentiles = [0.01,0.1,0.25,0.50,0.75,0.90,0.99])
```

Out[57]:

	Customer ID	frequency	recency	T	monetary_value
count	5942.000000	5942.000000	5942.000000	5942.000000	5942.000000
mean	15316.500000	5.479636	275.772299	478.229384	228.814496
std	1715.451981	11.293673	259.830840	223.879537	363.067124
min	12346.000000	0.000000	0.000000	0.000000	-3610.500000
1%	12405.410000	0.000000	0.000000	15.000000	-40.623900
10%	12940.100000	0.000000	0.000000	89.000000	0.000000
25%	13831.250000	0.000000	0.000000	320.500000	0.000000
50%	15316.500000	2.000000	225.000000	536.000000	174.900625
75%	16801.750000	6.000000	518.000000	674.000000	314.594375
90%	17692.900000	13.000000	672.900000	731.000000	502.095625
99%	18227.590000	44.590000	734.000000	738.000000	1328.480453
max	18287.000000	284.000000	738.000000	738.000000	8513.271143

In [58]:

```
#Pareto Model
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.utils import calibration_and_holdout_data
from lifetimes import ParetoNBDFitter
from lifetimes.plotting import plot_history_alive
from sklearn.metrics import mean_squared_error, r2_score
import math
from math import sqrt
```

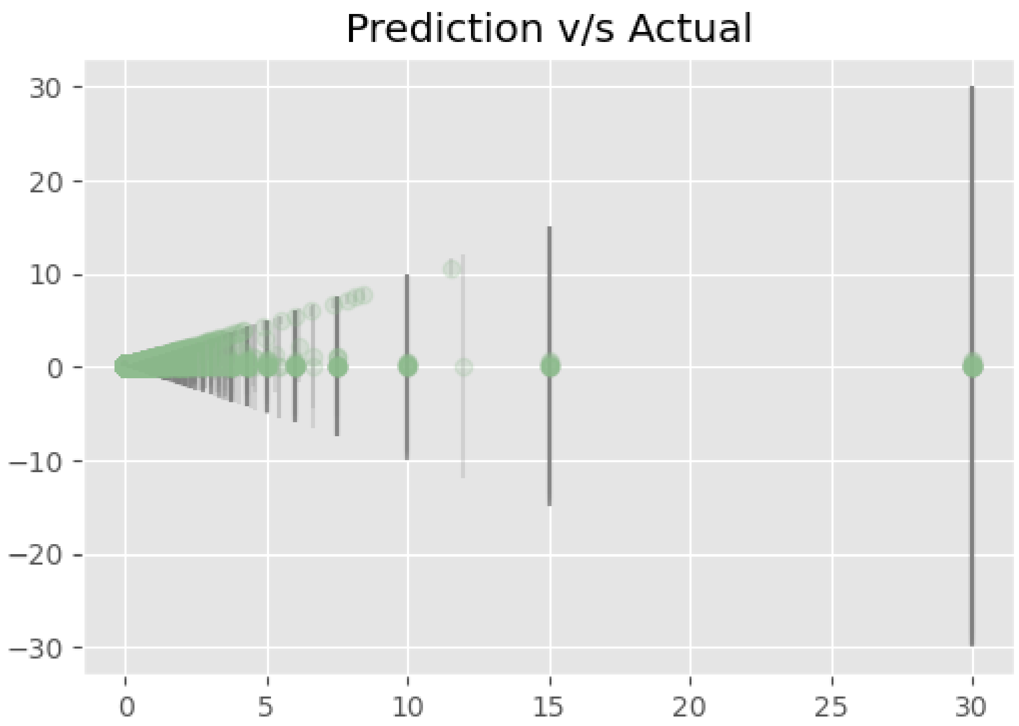
In [60]:

```
def get_model(data, penalizer_val, time):

    pareto_result = data.copy()

    pareto_model = ParetoNBDFitter(penalizer_coef = penalizer_val)
    pareto_model.fit(pareto_result["frequency"], pareto_result["recency"], pareto_result["T

    #calculating the predicted_purchases

    t = time

    pareto_result["predicted_purchases"] = pareto_model.conditional_expected_number_of_purc


    pareto_result["Actual_Purchases"] = pareto_result["frequency"]/pareto_result["recency"]

    #filling the null values
    pareto_result["Actual_Purchases"].fillna(0, inplace = True)


    #calculating the error
    pareto_result["Prediction_Error"] = pareto_result["Actual_Purchases"]-pareto_result["pr

    #calcuating the purchase prediction error
    pareto_mse_purchase = mean_squared_error(pareto_result["Actual_Purchases"], pareto_resu
    pareto_r2_purchase = r2_score(pareto_result["Actual_Purchases"], pareto_result["predict
    pareto_rmse_purchase = sqrt(mean_squared_error(pareto_result["Actual_Purchases"], paret
    pareto_avg_error_purchase = pareto_result["Prediction_Error"].mean()

    #printing the purchase prediction error
    print("Predicted Purchase Mean Squared Error: %s" %(pareto_mse_purchase))
    print("Predicted Purchase R2 Score: %s" %(pareto_r2_purchase))
    print("Predicted Purchase Root Mean Squared Error: %s" %(pareto_rmse_purchase))
    print("Predicted Purchase Average Purchases Error: %s" %(pareto_avg_error_purchase))


    #plotting the prediction v/s actual purchase plot
    plt.figure(figsize=(6,4))
    plt.errorbar(pareto_result["Actual_Purchases"], pareto_result["predicted_purchases"], y
                 ecolor='grey', elinewidth=1.5, capsize=0, alpha = 0.2);
    plt.title("Prediction v/s Actual")
```

In [61]:

```
get_model(rfm_summary, 0.001, 30)
```

```
Predicted Purchase Mean Squared Error: 4.335934568785449
Predicted Purchase R2 Score: 0.004258462996240442
Predicted Purchase Root Mean Squared Error: 2.0822907022760893
Predicted Purchase Average Purchases Error: 0.4123654498211612
```



In [62]:

```
pareto_model = lifetimes.ParetoNBDFitter(penalizer_coef = 0.1)
```

In [63]:

```
pareto_model.fit(rfm_summary["frequency"],rfm_summary["recency"],
                rfm_summary["T"])
```

Out[63]:

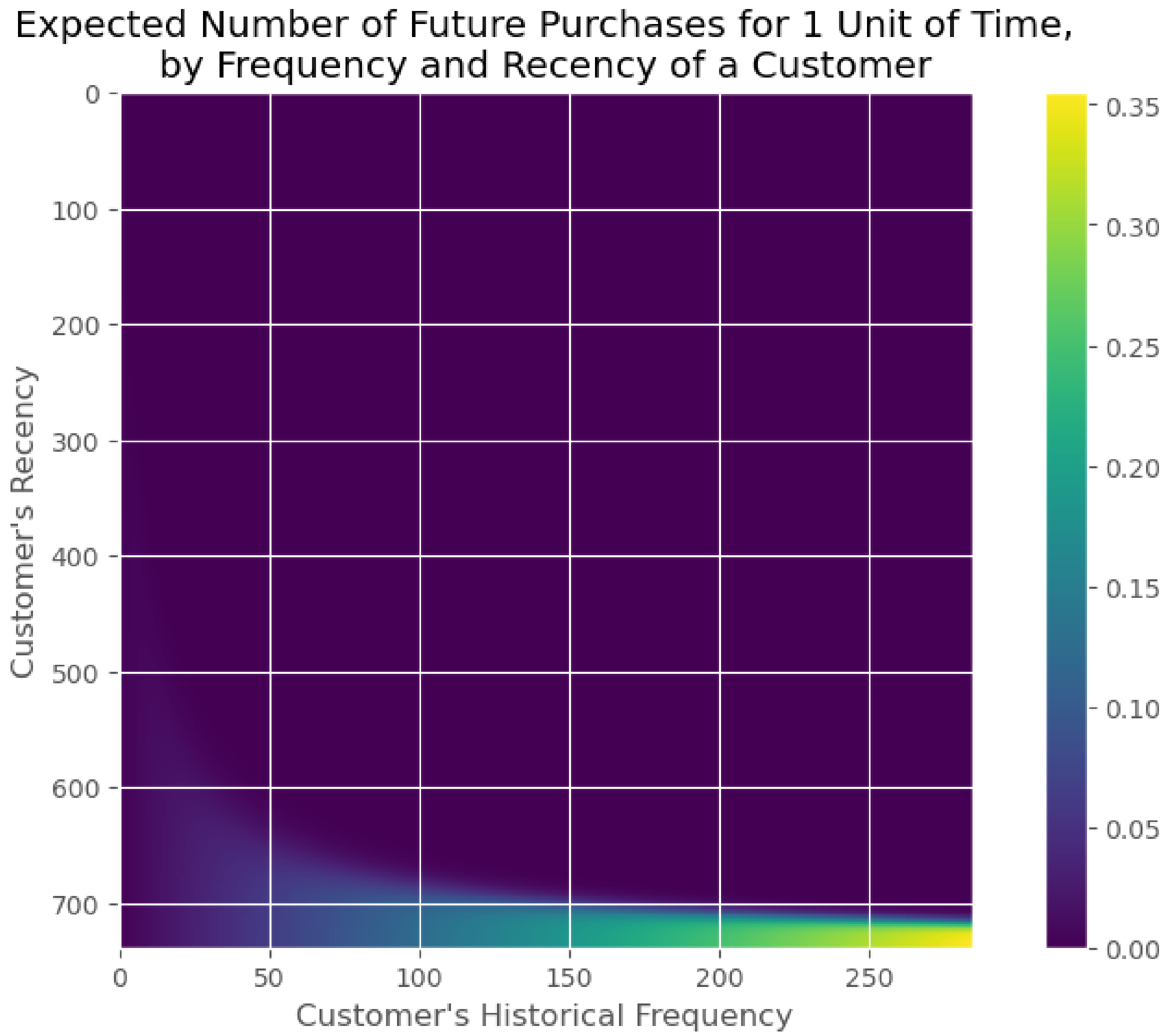
```
<lifetimes.ParetoNBDFitter: fitted with 5942 subjects, alpha: 63.87, beta: 1
24.21, r: 0.83, s: 0.16>
```

In [64]:

```
plt.figure(figsize=(10,6))
plot_frequency_recency_matrix(pareto_model)
```

Out[64]:

```
<AxesSubplot:title={'center':'Expected Number of Future Purchases for 1 Unit
of Time,\nby Frequency and Recency of a Customer'}, xlabel="Customer's Histo
rical Frequency", ylabel="Customer's Recency">
```

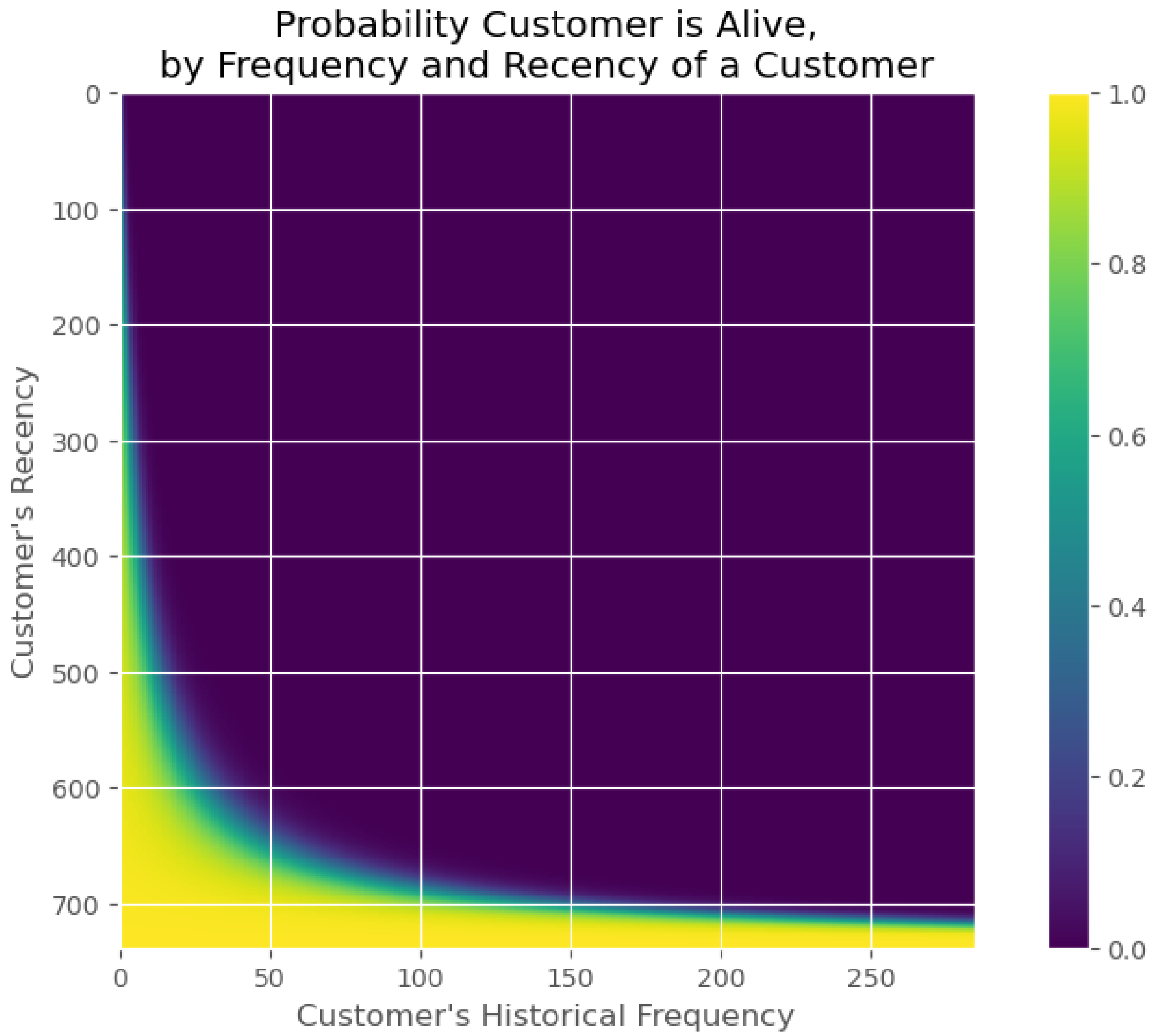


In [65]:

```
plt.figure(figsize=(10,6))
plot_probability_alive_matrix(pareto_model)
```

Out[65]:

```
<AxesSubplot:title={'center':'Probability Customer is Alive,\nby Frequency a
nd Recency of a Customer'}, xlabel="Customer's Historical Frequency", ylabel
="Customer's Recency">
```



In [66]:

```
pareto_result = rfm_summary.copy()
```

In [67]:

```
ive"] = 1-pareto_model.conditional_probability_alive(pareto_result["frequency"], pareto_resu
] = pareto_model.conditional_probability_alive(pareto_result["frequency"], pareto_result["re
```

In [68]:

```
pareto_result.head()
```

Out[68]:

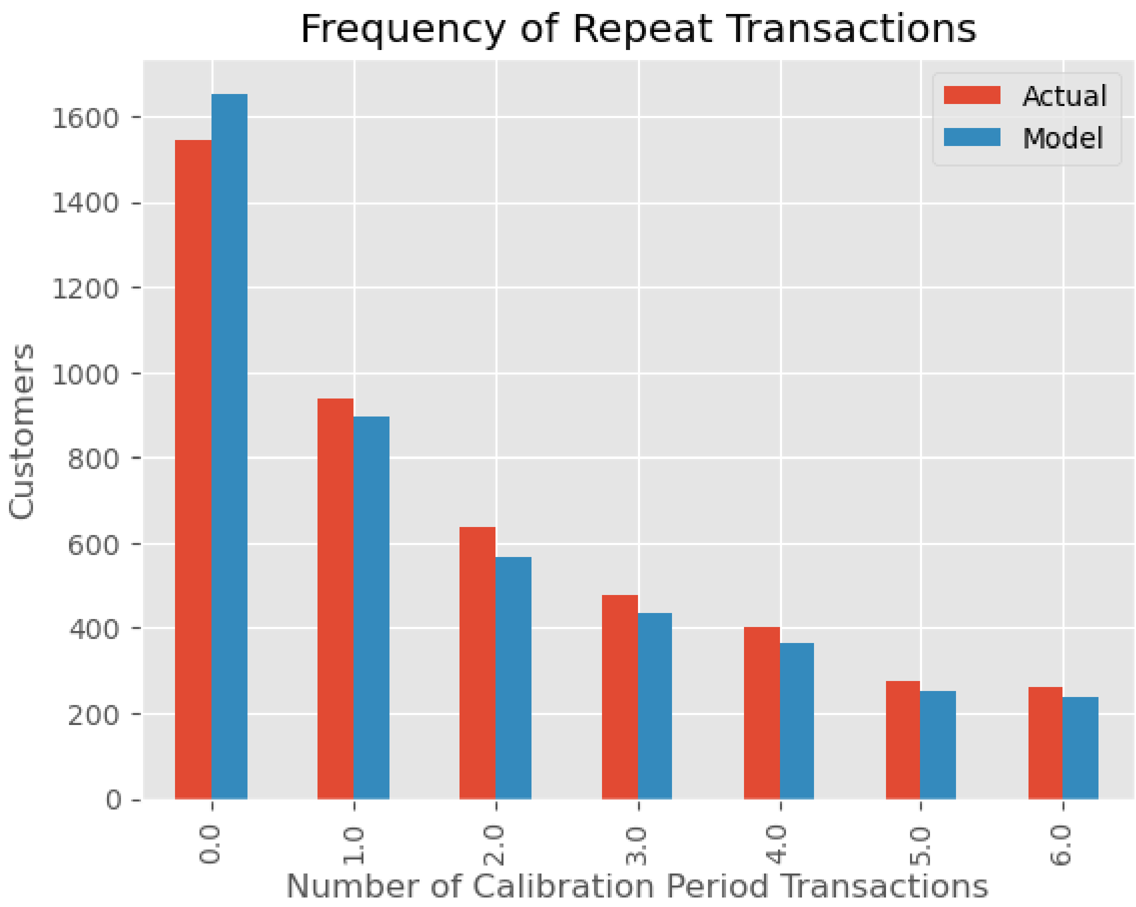
	Customer ID	frequency	recency	T	monetary_value	p_not_alive	p_alive
0	12346.0	10.0	400.0	725.0	-15.468000	0.819891	0.180109
1	12347.0	7.0	402.0	404.0	717.398571	0.000634	0.999366
2	12348.0	4.0	363.0	438.0	449.310000	0.034895	0.965105
3	12349.0	4.0	717.0	735.0	1107.172500	0.003674	0.996326
4	12350.0	0.0	0.0	310.0	0.000000	0.334744	0.665256

In [70]:

```
plot_period_transactions(pareto_model)
```

Out[70]:

```
<AxesSubplot:title={'center':'Frequency of Repeat Transactions'}, xlabel='Nu
mber of Calibration Period Transactions', ylabel='Customers'>
```



In [71]:

```
#dividing our dataset into training & holdout
pareto_summary_cal_holdout = calibration_and_holdout_data(data, "Customer ID", "InvoiceDate
                                                   calibration_period_end = '2011-06-08',
                                                  observation_period_end = '2011-12-09')
```

In [72]:

```
pareto_summary_cal_holdout.head()
```

Out[72]:

	frequency_cal	recency_cal	T_cal	frequency_holdout	duration_holdout
Customer ID					
12346.0	10.0	400.0	541.0	0.0	184.0
12347.0	3.0	158.0	220.0	4.0	184.0
12348.0	3.0	190.0	254.0	1.0	184.0
12349.0	3.0	328.0	551.0	1.0	184.0
12350.0	0.0	0.0	126.0	0.0	184.0

In [73]:

```
pareto_model.fit(pareto_summary_cal_holdout["frequency_cal"],
               pareto_summary_cal_holdout["recency_cal"],
               pareto_summary_cal_holdout["T_cal"])
```

Out[73]:

```
<lifetimes.ParetoNBDFitter: fitted with 5025 subjects, alpha: 63.81, beta: 8
01.34, r: 0.83, s: 0.76>
```

In [74]:

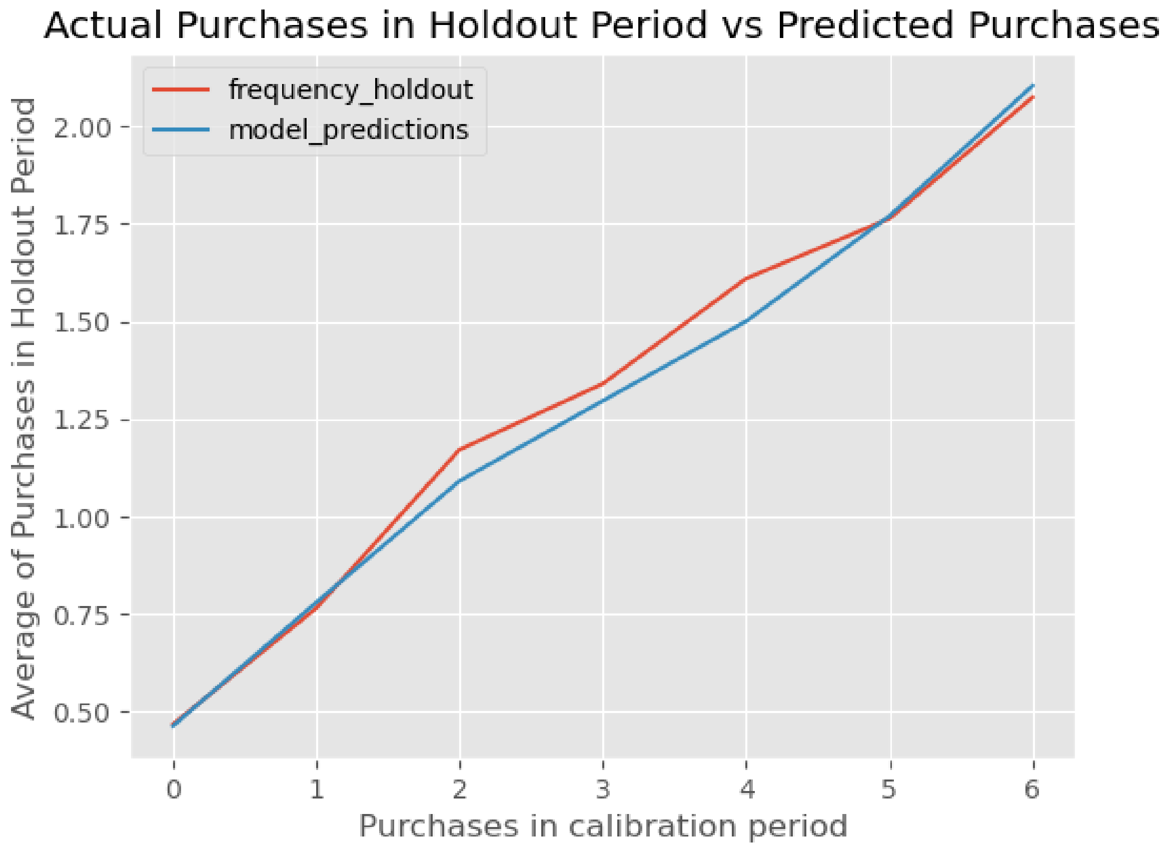
```
from lifetimes.plotting import plot_calibration_purchases_vs_holdout_purchases
```

In [75]:

```
plot_calibration_purchases_vs_holdout_purchases(pareto_model, pareto_summary_cal_holdout)
```

Out[75]:

```
<AxesSubplot:title={'center':'Actual Purchases in Holdout Period vs Predicte
d Purchases'}, xlabel='Purchases in calibration period', ylabel='Average of
Purchases in Holdout Period'>
```



In [82]:

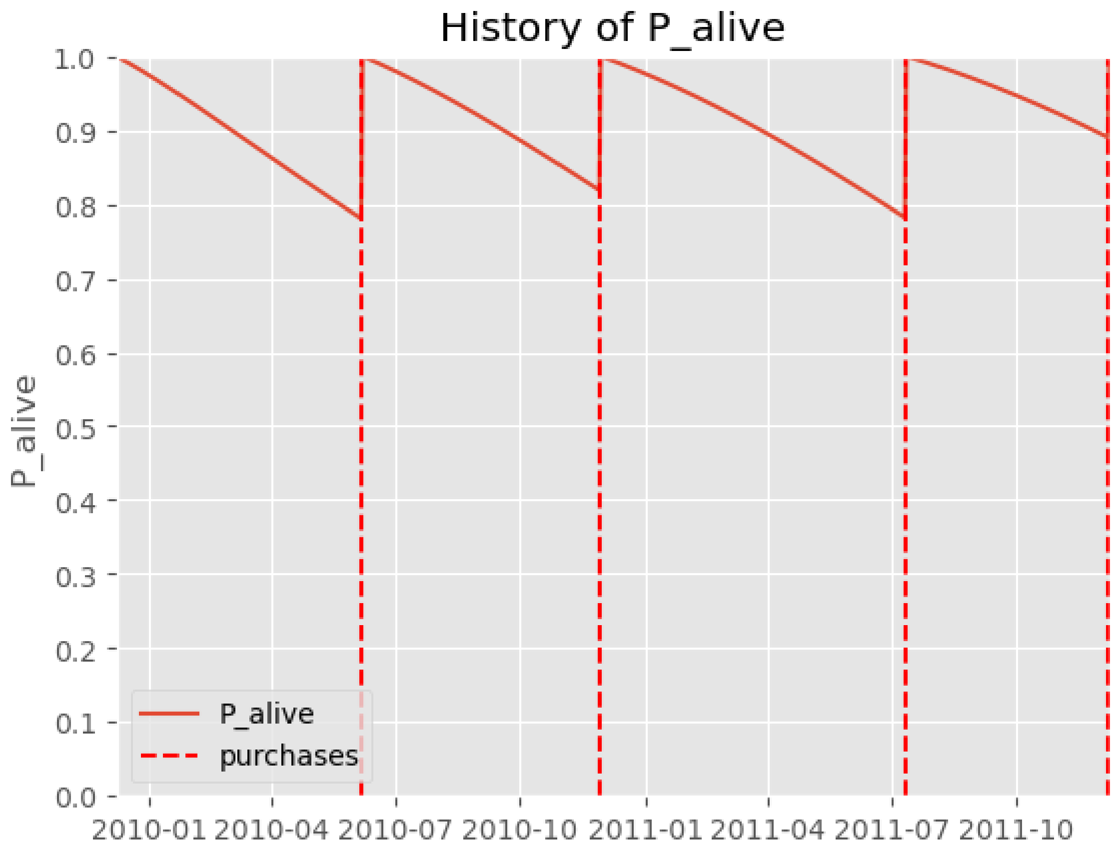
```
from lifetimes.plotting import plot_history_alive
```

In [83]:

```
#Get the history alive plot to check whether the customer is alive or not

def get_history_alive(t_, data_, id_):

    individual_alive = data[data["Customer ID"] == id_]
    plot_history_alive(pareto_model, t = t_, transactions = individual_alive, datetime_col
                 freq = "D")
```

In [84]:

```
get_history_alive(30, data, 12358.0)
```



In [85]:

```
#Before proceding with our Gamma Gamma Model,
#we have to first filter the data where we are going to remove the values with 0 frequency
idx = pareto_result[(pareto_result["frequency"] <= 0.0)]
```

In [86]:

```
idx = idx.index
```

In [87]:

```
ggf_filter = pareto_result.drop(idx, axis = 0)
```

In [88]:

```
m_idx = ggf_filter[(ggf_filter["monetary_value"] <= 0.0)].index
```

In [89]:

```
ggf_filter = ggf_filter.drop(m_idx, axis = 0)
```

In [90]:

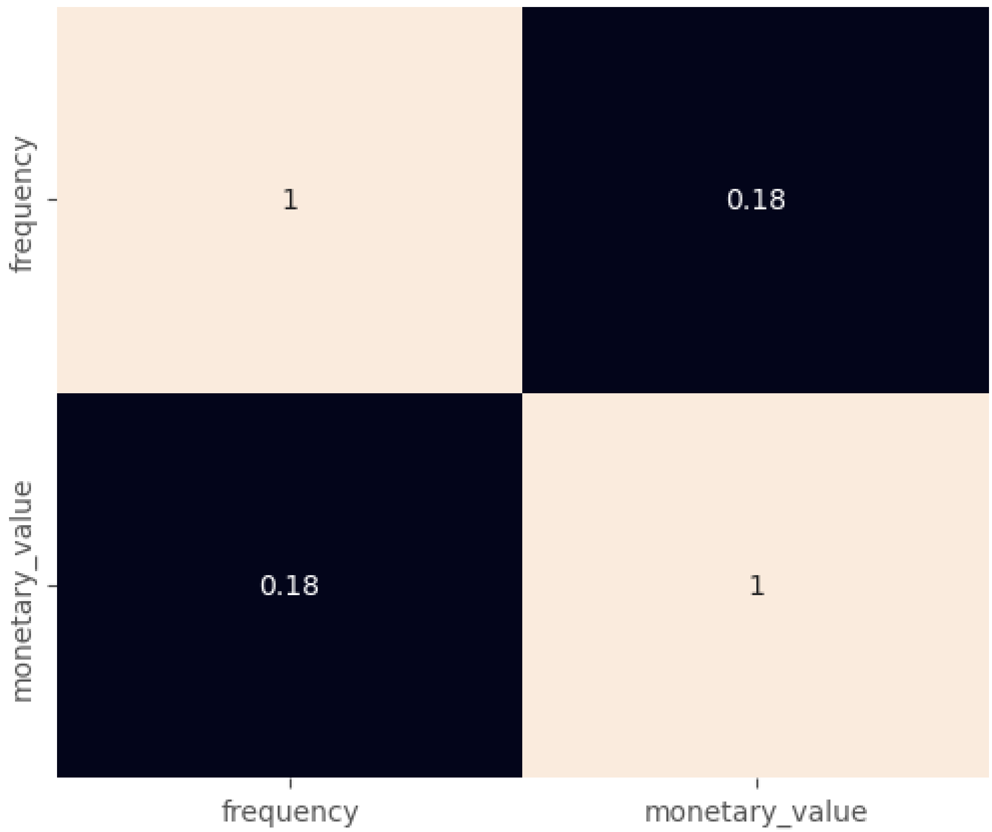
```
ggf_filter.reset_index().drop("index", axis = 1, inplace = True)
```

In [91]:

```
plt.figure(figsize=(6,5))
sns.heatmap(ggf_filter[["frequency", "monetary_value"]].corr(), annot = True, cbar = False)
```

Out[91]:

```
<AxesSubplot:>
```



In [ ]:

```

```