به نام خدا

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

درس مدیریت پایگاه دانش

تمرین سری .2...

نام و نام خانوادگی	ابوالفضل عابدینی
شماره دانشجویی	4023614026
تاریخ ارسال گزارش	1403/02/04

فهرست گزارش

مقدمه

در این تمرین ابتدا مراحل پیش پردازش را روی دیتاست مورد نظر خود انجام داده ایم سپس یک تحلیل EDA روی دیتاست صورت گرفته.دیتاست پیش پردازش شده را روی ان عمل قوانین انجمنی و الگو های مکرر انجام شده است.

پیش پردازش

اصلی ترین کار در این پروژه پیش پردازش است. در این مرحله4 کار انجام شده که به ترتیب انها را بیان میکنیم:

1.حذف سطر های تکراری:

میخواهیم سطرهای تکراری را حذف کنیم برای این منظور یک تابع به نام deduplication نوشته شده است.هرچند که داده ها سطر تکراری نداشته اند.

2.تصحیح مقادیر داده ایی ستون InvoiceNo وStockCode :

در این دو ستون از دیتاست باید مقادیر به صورت رشته ایی از عدد باشد در صورتی که در بعضی از ستون ها کاراکتر نیز پیدا شده است برای مثال نمونه ایی از داده اشتباه را در پایین مشاهده میکنید.

```
df
```

Out[1]:

	index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

به این منظور تابع ایی به نام remove_alpha_from_numberوdeleteAlphaCharacter نوشته شده تا مقادیری که کاراکتر الفبا دارند را تصحیح کند.

In [3]:
```
def remove_alpha_from_number(number_string):
    digits = [char for char in number_string if char.isdigit()]
    number_only = ''.join(digits)
    return number_only
```

In [4]:
```
def deleteAlphaCharacter(df):
    deep_copy = df.copy(deep=True)
    column_names = list(df.columns)
    for i in column_names:
        if i=='InvoiceNo' or i=='StockCode':
            row,col=deep_copy.shape
            for j in range(row):
                try:
                    int(deep_copy[i][j])
                except Exception as e:
                    deep_copy[i][j]= remove_alpha_from_number(deep_copy[i][j])
                    if deep_copy[i][j]=="" or int(deep_copy[i][j])<9999:
                        deep_copy[i][j]=None
    return deep_copy

```

3.پرکردن مقادیر خالی:

چون داده ها هم numerical و categoricalهستند پر کردن مقادیر خالی به دو صورت است است در داده های عددی تابعی به نام fillMisingValueWithMean نوشته شده که مقادیر خالی را با میانگین ان ستون پر میکند و همچنین داده های پرت و منفی را نیز را حذف و بجایش میانگین را میگذارد.در داده های categorical تابعی به نام fillNull نوشته شده که مقادیر خالی را با مد modeان ستون پر میکند. تصویر زیر دو تابع باال را نشان میدهد.

In [37]:
```
def fillNull(df):
    deep_copy = df.copy(deep=True)
    column_names = list(df.columns)
    for i in column_names:
        if i!="Description":
            most_frequent_category = deep_copy[i].mode()[0]
            deep_copy[i].fillna(most_frequent_category, inplace=True)
            row,col=deep_copy.shape
            for j in range(row):
                if i!="InvoiceDate" and i!="Country" and int(deep_copy[i][j])<0:
                    deep_copy[i][j]=most_frequent_category
    return deep_copy
```

In [38]:
```
def fillMisingValueWithMean(df):
    deep_copy = df.copy()
    column_names = list(deep_copy.columns)
    for i in column_names:
        if deep_copy[i].dtypes=="float64":
            deep_copy[i] = deep_copy[i].fillna(deep_copy[i].mean())
    return deep_copy
```

4.تبدیل داده های categorical بهnumerical :

بعد از انکه داده های خالی در ستون های categorical پر کردیم برای بهتر شدن دقت مدل وهمچنین تحلیل و بررسی در EDA داده ها را به صورت عددی تبدیل میکنیم برای اینکار تابع ایی به نام modifyCatToNumنوشته شده است.

در نهایت تمام این توابع در یک تابع مادر به نام preProcessing قرار میگیرند دیتا ست اولیه به عنوان ورودی به مرحله 1 داده میشود و خروجی ان ورودی مرحله بعدی است تا در نهایت دیتا پیش پردازش شده ایجاد شود خروجی دیتا ست پیش پردازش شده به صورت زیر است:

In [51]:
```
def preprocessing(df):
    df1=deduplication(df)
    df2=deleteAlphaCharacter(df1)
    df3=fillNull(df2)
    df4=fillMisingValueWithMean(df3)
    df5=modifyCatToNum(df4)
    df_no_nan_Description = df5.dropna(subset=['Description'])
    return df_no_nan_Description

```

In [*]:
```
df=preprocessing(df)
```

]:

	index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	0	536365	85123	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	36
1	1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	36
2	2	536365	84406	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	36
3	3	536365	84029	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	36
4	4	536365	84029	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	36
...	...	...	...	...	...	...	...	...	...
541904	541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0	13
541905	541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0	13
541906	541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0	13
541907	541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0	13
541908	541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0	13

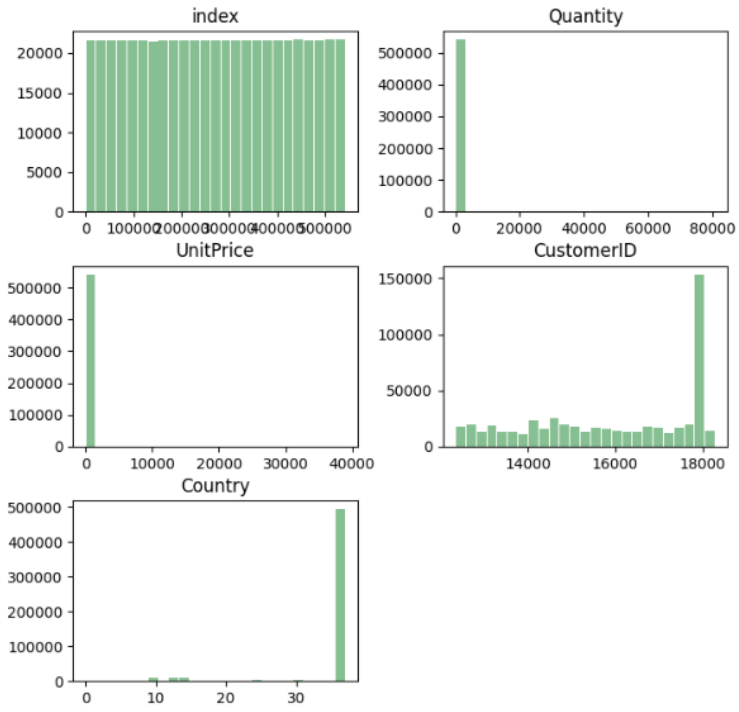
540455 rows × 9 columns

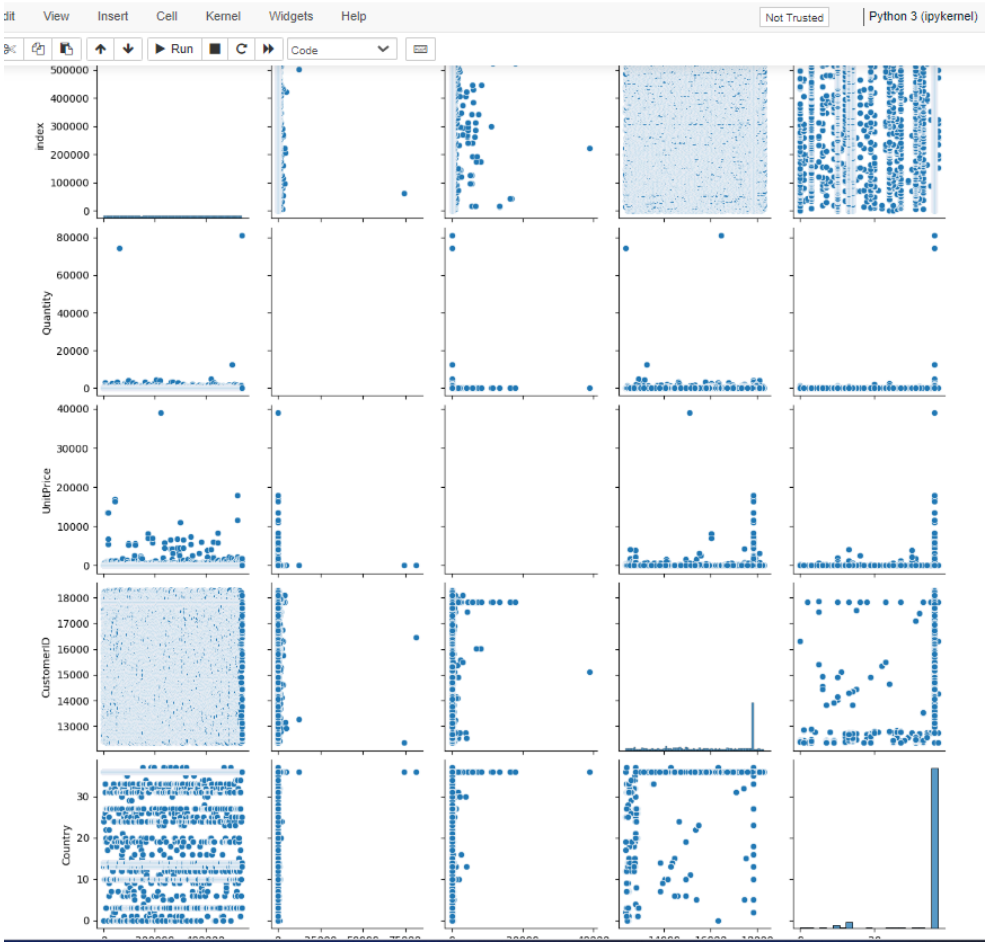
EDA

بعد از انجام پیش پردازش، میتوانیم به تحلیل EDA بپردازیم. در این مرحله، به طور مقدماتی داده ها را بررسی میکنیم و آمارها و الگوهای مختلفی را بررسی کنید، مانند میانگین، واریانس، توزیعها و نمودارهای مختلف در این بخش ثبت و انجام شدم و شامل مراحل زیر است.

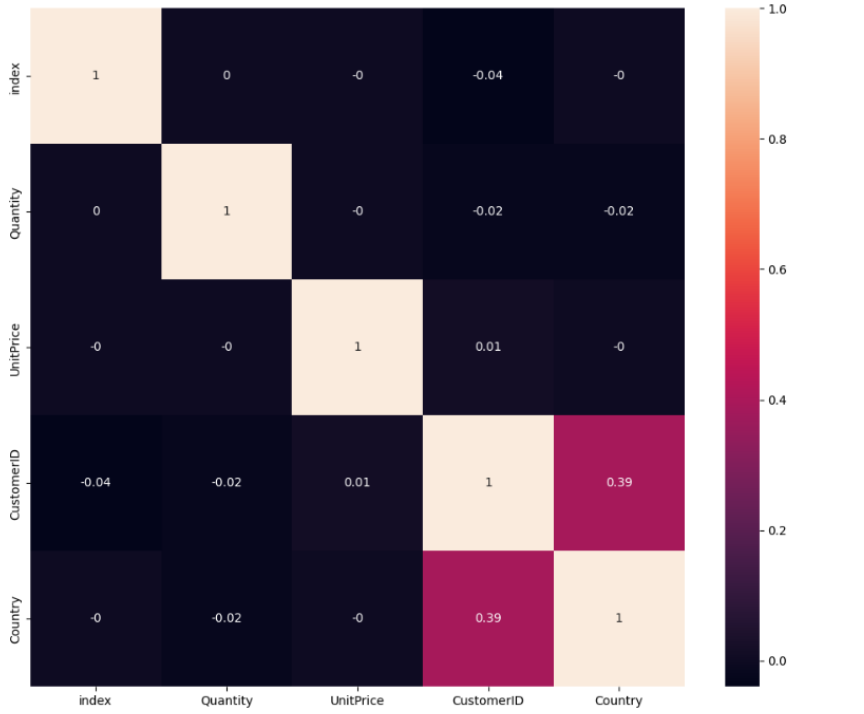
EDA

[54]:
```
1 import matplotlib.pyplot as plt
2 df.hist(bins=25, grid=False, figsize=(8,8), color='#86bf91', zorder=2, rwidth=0.9)
3 plt.show()
```





In [58]:
```
plt.figure(figsize=(12,10))
plt=sns.heatmap(correlation_matrix.round(2),annot=True)
```



سوال 1

نمایش 10 کالایی که بیشترین فروش و 10 کالایی که کمترین فروش داشته‌اند را (با در نظر گرفتن Quantity):

برای نمایش 10 کالایی که بیشترین فروش و 10 کالایی که کمترین فروش داشته‌اند تابع ایی به نام show_Height_lowest_selle نوشته شده است به طوری که ابتدا روی ستون StockCode گروه بندی میشود وسپس ستون Quantity جمع میشود بدنه تابع و خروجی به شکل زیر است.

```
In [10]:
def show_Height_lowest_selle(df):
    grouped = df.groupby('StockCode')
    df_grouped = grouped['Quantity'].sum()
    df_grouped=pd.DataFrame(df_grouped)
    df_grouped=df_grouped.sort_values(by='Quantity',ascending=False)
    heigth=df_grouped.head(10)
    low=df_grouped.tail(10)
    return heigth,low

In [11]:
heightesSell,lowestSell=show_Height_lowest_selle(df)

In [12]:
heightesSell
```

Out[12]:

StockCode	Quantity
85099	91783
23843	80996
23166	78043
22197	56971
84077	55052
85123	41999
84879	36474
21212	36434

سوال 2

تهیه دیتاست تراکنشی:

برای ساخت دیتاست تراکنشی دو تابع به نام های groupbyHandler وreturn_all_values نوشته شده به است به طوری که ابتدا روی ستون InvoiceNo گروه بندی میشود و سپس کالاهای تراکنش در لیستی قرار میگیرند.

```
In [17]: 1 df_grouped
```

Out[17]:

InvoiceNo	StockCode
536365	[85123, 71053, 84406, 84029, 84029, 22752, 21730]
536366	[22633, 22632]
536367	[84879, 22745, 22748, 22749, 22310, 84969, 226...
536368	[22960, 22913, 22912, 22914]
536369	[21756]
...	...
581583	[20725, 85038]
581584	[20832, 85038]
581585	[22481, 22915, 22178, 22460, 84832, 23084, 848...
581586	[22061, 23275, 21217, 20685]
581587	[22631, 22556, 22555, 22728, 22727, 22726, 227...

سوال 3

نمایش 10 کالایی که بیشترین فروش و 10 کالایی که کمترین فروش داشته‌اند را (بدون در نظر گرفتن Quantity):

برای نمایش 10 کالایی که بیشترین فروش و 10 کالایی که کمترین فروش داشته‌اند تابع ایی به نام show_Height_lowest_selle_without_quantity نوشته شده است.

```
In [49]: 1 heigth
```

Out[49]:

	stock	count
44	85099	6949
0	85123	2380
1187	47566	2208
508	22423	2203
130	85049	1802
61	84997	1736
63	20725	1639
573	84596	1589
8	84879	1502
161	22197	1476

```
In [50]: 1 low
```

Out[50]:

	stock	count
2632	84977	1
2628	84227	1
2626	85098	1
2622	35611	1

سوال 4

استفاده الگوریتم Apriori و FP-Growth در مجموعه مکرر:

برای اینکه بتوانیم از الگوریتم Apriori و FP-Growth استفاده کنیم باید ابتدا یک ماتریس bolean بر اساس تراکنش و کالا های موجود درست کنیم و بعد از ان الگوریتم Apriori و FP-Growth بر روی ماتریس به

وجود امده پیاده میکنیم.در صورت سوال 0.5 support ذکر شده در صورتی که با این عدد هیچ یک از مجموعه های مکرر حذف نمیشدند بر این اسا مقدار support را به 0.1 کاهش داده ایم.

```
In [23]: 1 frequent_itemsets_apriori = apriori(df_transaction, min_support=0.01, use_colnames=True)

In [24]: 1 frequent_itemsets_apriori
```

Out[24]:

	support	itemsets
0	0.021353	(15036)
1	0.032480	(15056)
2	0.015708	(16161)
3	0.011904	(16237)
4	0.013254	(20675)
...	...	...
1330	0.010677	(22355, 20724, 20723, 20719)
1331	0.010308	(22355, 22356, 20724, 20719)
1332	0.010104	(22356, 22355, 20724, 20723)
1333	0.012395	(85099, 22411, 21931, 22386)
1334	0.012395	(22699, 22423, 22698, 22697)

1335 rows × 2 columns

```
In [25]: 1 frequent_itemsets_fp_growth = fpmax(df_transaction, min_support=0.01, use_colnames=True)
         2 frequent_itemsets_fp_growth
```

Out[25]:

	support	itemsets
0	0.010022	(21716)
1	0.010022	(23310)
2	0.010022	(22634)
3	0.010022	(84792)
4	0.010022	(35961)
...	...	...
942	0.014808	(47566, 20725)
943	0.017467	(22423, 47566)
944	0.018367	(47566, 85123)
945	0.014767	(22423, 85123)
946	0.024585	(22423, 85099)

947 rows × 2 columns

حال با داشتن دو دیتاست الگوریتم Apriori و FP-Growth و ترکیب ان دو دیتاستی به نام combined_itemsets تولید میشود و از روی ان مجموعه های بیشتر از دو کالا قرار میگیرند .

ده ایتم برتر به صورت زیر خواهد بود:

```
In [27]:
1 top_10_itemsets = itemsets_with_2_or_more_items.sort_values(by='support', ascending=False).head(10)
2 top_10_itemsets
```

Out[27]:

	support	itemsets
971	0.041888	(85099, 22386)
859	0.039516	(85099, 21931)
978	0.036529	(22411, 85099)
725	0.033502	(85099, 20725)
1034	0.032071	(22699, 22697)
1119	0.031130	(85099, 23203)
1104	0.030271	(23199, 85099)
654	0.030230	(20712, 85099)
841	0.027653	(85099, 21929)
835	0.027448	(21928, 85099)

سوال 5

استفاده الگوریتم Apriori و FP-Growth در قوانین انجمنی:

با دیتاست ترکیب شده از مرحله قبل association ruls را پیاده میکنیم. در صورت سوال 0.5confidence ذکر شده در صورتی که با این عدد هیچ یک از مجموعه های مکرر حذف نمیشدند بر این اسا مقدار confidenceرا به 0.1 کاهش داده ایم.وسپس ده قانون برتر را لیست کرده ایم.

```
In [85]:
1 top_10_rules
```

Out[85]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
1593	(20712, 22386, 21931)	(85099)	0.011413	0.195533	0.011249	0.985663	5.040904	0.009018	56.111572	0.810877
1605	(20712, 22411, 21931)	(85099)	0.011249	0.195533	0.011045	0.981818	5.021240	0.008845	44.245684	0.809957
1675	(22411, 22386, 21931)	(85099)	0.012681	0.195533	0.012395	0.977419	4.998743	0.009915	35.626395	0.810224
1618	(20712, 22411, 22386)	(85099)	0.010390	0.195533	0.010145	0.976378	4.993417	0.008113	34.055769	0.808133
1414	(22386, 22379)	(85099)	0.012231	0.195533	0.011904	0.973244	4.977390	0.009512	30.066954	0.808986
1419	(22411, 22379)	(85099)	0.010554	0.195533	0.010186	0.965116	4.935823	0.008122	23.061387	0.805905
922	(20712, 22386)	(85099)	0.017631	0.195533	0.016976	0.962877	4.924371	0.013529	21.670330	0.811231
874	(20712, 21928)	(85099)	0.011290	0.195533	0.010799	0.956522	4.891868	0.008592	18.502741	0.804664
865	(22411, 20711)	(85099)	0.010881	0.195533	0.010390	0.954887	4.883509	0.008263	17.832352	0.803977
1349	(21930, 22411)	(85099)	0.012313	0.195533	0.011699	0.950166	4.859364	0.009292	16.142971	0.804113

```
In [82]:
1 combined_itemsets=association_rules(combined_itemsets, metric="confidence", min_threshold=0.2)
```

```
In [83]:
1 combined_itemsets
```

Out[83]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(15036)	(85099)	0.021353	0.195533	0.010022	0.469349	2.400355	0.005847	1.515999	0.596124
1	(15056)	(85099)	0.032480	0.195533	0.016567	0.510076	2.608642	0.010216	1.642023	0.637360
2	(20676)	(85099)	0.018449	0.195533	0.010513	0.569845	2.914315	0.006906	1.870178	0.669212
3	(20685)	(85099)	0.028798	0.195533	0.012272	0.426136	2.179358	0.006641	1.401843	0.557195
4	(20712)	(20711)	0.035670	0.021762	0.011822	0.331422	15.229216	0.011046	1.463162	0.968898
...	...	...	...	...	...	...	...	...	...	...
1693	(22699, 22423)	(22697, 22698)	0.021967	0.026344	0.012395	0.564246	21.418561	0.011816	2.234416	0.974723
1694	(22698, 22699)	(22697, 22423)	0.025117	0.021190	0.012395	0.493485	23.289079	0.011862	1.932442	0.981719
1695	(22697)	(22699, 22698, 22423)	0.043238	0.013785	0.012395	0.286660	20.794359	0.011799	1.382531	0.994929
1696	(22698)	(22697, 22699, 22423)	0.032766	0.016853	0.012395	0.378277	22.445057	0.011842	1.581326	0.987814
1697	(22699)	(22697, 22698, 22423)	0.045815	0.014154	0.012395	0.270536	19.114208	0.011746	1.351466	0.993186

1698 rows × 10 columns

سوال 6

باتوجه به قوانین انجمنی سوال قبل تابعی به نام ده ایتم برتر توسط تابعی به نام top_Lift_Items_Handler پیاده سازی شده که خروجی ان به صورت زیر است:

```
In [86]:
def top_Lift_Items_Handler(df):
    df=df[df["lift"]>10]
    top_10 = df.sort_values(by='confidence', ascending=False).head(10)
    return top_10
```

```
In [87]:
top_Lift_Items_Handler(combined_itemsets)
```

Out[87]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
1525	(23170, 23172)	(23171)	0.011127	0.015790	0.010554	0.948529	60.071891	0.010378	19.121796	0.994418
1524	(23171, 23172)	(23170)	0.011536	0.018694	0.010554	0.914894	48.939802	0.010338	11.530342	0.990999
1660	(22356, 20723, 22355)	(20724)	0.011167	0.042952	0.010104	0.904762	21.064580	0.009624	10.049006	0.963284
1688	(22699, 22698, 22423)	(22697)	0.013785	0.043238	0.012395	0.899110	20.794359	0.011799	9.483198	0.965216
724	(23172)	(23171)	0.012845	0.015790	0.011536	0.898089	56.877430	0.011333	9.657562	0.995201
1292	(21080, 21086)	(21094)	0.012108	0.021558	0.010840	0.895270	41.528989	0.010579	9.342546	0.987882
1492	(22698, 22699)	(22697)	0.025117	0.043238	0.022458	0.894137	20.679346	0.021372	9.037720	0.976160
1685	(22697, 22698, 22423)	(22699)	0.014154	0.045815	0.012395	0.875723	19.114208	0.011746	7.677859	0.961289
722	(23172)	(23170)	0.012845	0.018694	0.011127	0.866242	46.337315	0.010886	7.336429	0.991150
1472	(22698, 22423)	(22697)	0.016444	0.043238	0.014154	0.860697	19.905948	0.013443	6.868183	0.965643

سوال 7

در این سوال قوانین که در ستون antecedents دارای 3 کالا و در ستون consequents دارای یک کالا باشد را استخراج میکنیم برای اینکار تابعی به نام

show_association_rules_antecedents3item_consequents1item نوشته شده است خروجی به صورت زیر است.

```
In [88]:
def show_association_rules_antecedents3item_consequents1item(df):
    row,col=df.shape
    listOfInfoDf=[]
    for i in range(row):
        if (len(df["antecedents"][i])==3) and (len(df["consequents"][i])==1):
            listOfInfoDf.append(df.iloc[i].to_dict())
    new_df=pd.DataFrame(listOfInfoDf)
    new_df=new_df.sort_values(by='confidence', ascending=False)
    return new_df
```

```
In [89]:
show_association_rules_antecedents3item_consequents1item(combined_itemsets)
```

Out[89]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
2	(20712, 22386, 21931)	(85099)	0.011413	0.195533	0.011249	0.985663	5.040904	0.009018	56.111572	0.810877
5	(20712, 22411, 21931)	(85099)	0.011249	0.195533	0.011045	0.981818	5.021240	0.008845	44.245684	0.809957
27	(22411, 22386, 21931)	(85099)	0.012681	0.195533	0.012395	0.977419	4.998743	0.009915	35.626395	0.810224
9	(20712, 22411, 22386)	(85099)	0.010390	0.195533	0.010145	0.976378	4.993417	0.008113	34.055769	0.808133
22	(22356, 20723, 22355)	(20724)	0.011167	0.042952	0.010104	0.904762	21.064580	0.009624	10.049006	0.963284
31	(22699, 22698, 22423)	(22697)	0.013785	0.043238	0.012395	0.899110	20.794359	0.011799	9.483198	0.965216
28	(22697, 22698, 22423)	(22699)	0.014154	0.045815	0.012395	0.875723	19.114208	0.011746	7.677859	0.961289
15	(20719, 20723, 22355)	(20724)	0.012436	0.042952	0.010677	0.858553	19.988741	0.010142	6.766108	0.961934
18	(22356, 20719, 22355)	(20724)	0.012027	0.042952	0.010308	0.857143	19.955918	0.009792	6.699337	0.961452