In [1]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import pyreadr
```

In [2]:
```
dataset1 = pyreadr.read_r('promotions.rds')
print(dataset1.keys())
```

```
odict_keys([None])
```

In [3]:
```
promo = dataset1[None]
dataset2 = pyreadr.read_r('products.rda')
print(dataset2.keys())
```

```
odict_keys(['products'])
```

In [4]:
```
prod = dataset2['products']
dataset3 = pyreadr.read_r('transactions.rds')
print(dataset3.keys())
```

```
odict_keys([None])
```

In [5]:
```
tran = dataset3[None]
dataset4 = pyreadr.read_r('demographics.rda')
print(dataset4.keys())
```

```
odict_keys(['demographics'])
```

In [6]:
```
demo = dataset4['demographics']
dataset5 = pyreadr.read_r('coupons.rda')
print(dataset5.keys())
```

```
odict_keys(['coupons'])
```

In [7]:
```
cou = dataset5['coupons']
dataset6 = pyreadr.read_r('coupon_redemptions.rda')
print(dataset6.keys())
```

```
odict_keys(['coupon_redemptions'])
```

In [8]:
```
cou_redem = dataset6['coupon_redemptions']
dataset7 = pyreadr.read_r('campaigns.rda')
print(dataset7.keys())
```

```
odict_keys(['campaigns'])
```

In [334…
```
camp = dataset7['campaigns']
dataset8 = pyreadr.read_r('campaign_descriptions.rda')
print(dataset8.keys())
```

```
odict_keys(['campaign_descriptions'])
```

In [335…
```
camp_desc = dataset8['campaign_descriptions']
```

Trying to find our which product type and category have generated highest sales

In [395…
```
# Group transactions by product_id and calculate sales
group_transactions = tran.groupby('product_id')['sales_value'].sum().reset_index()
prod_sales = pd.merge(prod, group_transactions, on='product_id')

# Group product sales by product_category
category_sales = prod_sales.groupby('product_category')['sales_value'].sum().reset_index()
```

In [13]:
```
group_transactions['sales_value'].max()
```

Out[13]:
```
303116.02
```

In [14]:
```
# Product Id generating highest revenue
group_transactions[group_transactions['sales_value'] == group_transactions['sales_value'].max()]
```

Out[14]:

	product_id	sales_value
40917	6534178	303116.02

In [15]:
```
tran['basket_id'].nunique()
```

Out[15]:
```
155848
```

In [16]:
```
tran['transaction_timestamp'].nunique()
```

Out[16]:
```
155036
```

In [17]:
```
prod['product_id'].nunique()
```

Out[17]:
```
92331
```

In [18]:
```
prod['product_type'].nunique()
```

Out[18]:
```
2378
```

In [19]:
```
prod['product_category'].nunique()
```

Out[19]:
```
303
```

In [20]:
```
prod.isnull().sum()
```

Out[20]:
```
product_id              0
manufacturer_id         0
department              0
brand                   0
product_category      540
product_type          528
package_size        30586
dtype: int64
```

Dropping rows where both product type and category are vaccant, as we cant deduce simply by product id that which product went sold or unsold later on or which product generates highest or l;owest revenue

In [21]:
```
prod1 = prod.dropna(subset=['product_category', 'product_type'], how='all')
prod1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 91902 entries, 0 to 92330
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype   
---  ------            --------------  -----   
 0   product_id        91902 non-null  object  
 1   manufacturer_id   91902 non-null  object  
 2   department        91902 non-null  object  
 3   brand             91902 non-null  category
 4   product_category  91791 non-null  object  
 5   product_type      91803 non-null  object  
 6   package_size      61719 non-null  object  
dtypes: category(1), object(6)
memory usage: 5.0+ MB
```

In [24]:
```
prod1.isnull().sum()
```

Out[24]:
```
product_id              0
manufacturer_id         0
department              0
brand                   0
product_category      111
product_type           99
package_size        30183
dtype: int64
```

Defining 2nd and 3rd mode to figure out which product a particular household is more interested in and what are the household's 2nd and 3rd preferred items

In [40]:
```
# Define a custom function to calculate the second highest occurrence
def second_mode(series):
    counts = series.value_counts()
    if len(counts) >= 2:
        return counts.index[1]
    else:
        return None
# Define a custom function to calculate the third highest occurrence
def third_mode(series):
    counts = series.value_counts()
    if len(counts) >= 3:
        return counts.index[2]
    else:
        return None

time_grp = tran.groupby('transaction_timestamp').agg({
    "product_id": [lambda x: x.mode().iloc[0],lambda x: second_mode(x),lambda x: third_mode(x)],
}).reset_index()
time_grp.columns = ['Transaction Group','Highest Purchased','2nd Highest Purchased','3rd Highest Purchased']
```

In [41]:
```
time_grp
```

Out[41]:

	Transaction Group	Highest Purchased	2nd Highest Purchased	3rd Highest Purchased
0	2017-01-01 11:53:26	1095275	None	None
1	2017-01-01 12:10:28	9878513	None	None
2	2017-01-01 12:26:30	1041453	None	None
3	2017-01-01 12:30:27	1020156	1053875	1060312
4	2017-01-01 12:56:33	9297106	988791	9297106
...	...	...	...	...
155031	2018-01-01 03:43:30	898496	None	None
155032	2018-01-01 03:44:38	6534178	None	None
155033	2018-01-01 03:47:38	10457447	6396292	6396339
155034	2018-01-01 03:50:03	1071713	921744	1071713
155035	2018-01-01 04:01:20	1029743	933067	1029743

155036 rows × 4 columns

In [42]:
```
bask_grp = tran.groupby('basket_id').agg({
    "product_id": [lambda x: x.mode().iloc[0],lambda x: second_mode(x),lambda x: third_mode(x)],
}).reset_index()
bask_grp.columns = ['Basket Group','Highest Purchased','2nd Highest Purchased','3rd Highest Purchased']
bask_grp
```

Out[42]:

	Basket Group	Highest Purchased	2nd Highest Purchased	3rd Highest Purchased
0	31198437603	1009200	5591154	1044805
1	31198445400	5591538	None	None
2	31198445429	12757398	12757398	None
3	31198445465	1025611	858478	863447
4	31198452527	1053690	1053690	1065538
...	...	...	...	...
155843	41480008448	1020682	897954	1020682
155844	41480013048	1016286	1016286	1022956
155845	41480018446	1013641	1013641	12812474
155846	41481252623	1190075	1228634	1243632
155847	41481282915	1315320	1369595	None

155848 rows × 4 columns

Defining which basket group or order ids are owned by which household, we can either use the "first" or "mode" function to be more precise

In [44]:
```
bask_house_grp = tran.groupby('basket_id').agg({
    "household_id": [lambda x: x.mode().iloc[0]]
}).reset_index()
bask_house_grp.columns = ['Basket Group','Purchased by Household']
bask_house_grp
```

Out[44]:

	Basket Group	Purchased by Household
0	31198437603	1085
1	31198445400	1068
2	31198445429	1068
3	31198445465	1068
4	31198452527	1230
...	...	...
155843	41480008448	85
155844	41480013048	1776
155845	41480018446	1384
155846	41481252623	131
155847	41481282915	1433

155848 rows × 2 columns

The below suggests how many orders were placed with which household is appearing most number of times indicating maximum number of purchases by which household

In [146…
```
bask_house_grp.describe()
```

Out[146]:

	Basket Group	Purchased by Household
count	155848	155848
unique	155848	2469
top	31198437603	2337
freq	1	780

It looks like Household Id No. 2337 is doing 780 purchases per year which is unusual, so let's plot a boxplot for Frequency of purchases by different households.
Also lets find the number of purchases by other households before plotting a boxplot.

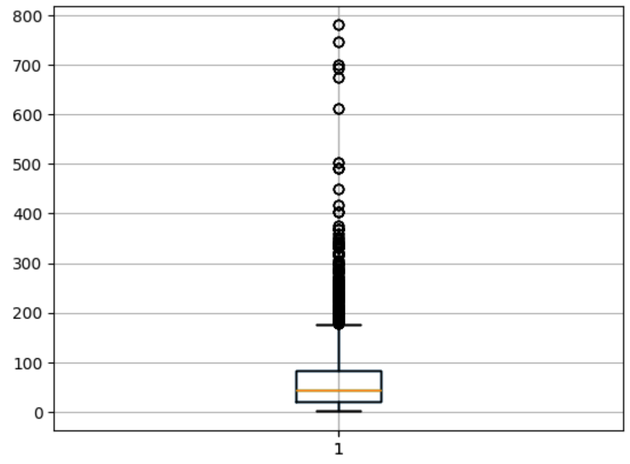
In [47]:
```
household_pur = bask_house_grp['Purchased by Household'].value_counts().reset_index()
household_pur.columns = ['Household Id', 'No. of Purchases']
household_pur
```

Out[47]:

	Household Id	No. of Purchases
0	2337	780
1	1510	747
2	1467	701
3	1795	694
4	900	674
...	...	...
2464	240	1
2465	1309	1
2466	2249	1
2467	2409	1
2468	272	1

2469 rows × 2 columns

In [53]:
```
import matplotlib.pyplot as plt
plt.boxplot(household_pur['No. of Purchases'])
plt.show()
```



Looks like many households purchased almost twice a day or may be different members of a household went for purchases separately.

Total Revenue Per Household

In [54]:
```
trph = tran.groupby("household_id").agg({
    "sales_value": ["max","sum"],
    "retail_disc": ["sum"],
    "coupon_disc": ["sum"],
}).reset_index()
trph.columns = ['Household Id','Costliest Item','Tot Rev','Tot Ret Disc','Tot Coupons']
trph
```

Out[54]:

	Household Id	Costliest Item	Tot Rev	Tot Ret Disc	Tot Coupons
0	1	12.50	2415.56	371.66	55.56
1	10	20.00	29.96	5.78	0.00
2	100	46.00	1336.87	179.28	2.50
3	1000	26.73	2509.38	312.77	3.75
4	1001	24.00	2558.82	264.33	13.24
...	...	...	...	...	...
2464	995	9.99	456.93	81.73	1.00
2465	996	33.33	1172.57	241.04	25.00
2466	997	42.18	2476.95	225.41	1.00
2467	998	45.00	4840.55	1052.66	26.35
2468	999	29.99	957.67	143.21	0.55

2469 rows × 5 columns

There are total 2469 unique households as opposed to 801 as reflected in household dataset. The total unique household_ids who purchased atleast one item can be found from transaction dataset

In [61]:
```
trph_freq = pd.merge(trph, household_pur, on='Household Id', how='inner')
trph_freq
```

Out[61]:

	Household Id	Costliest Item	Tot Rev	Tot Ret Disc	Tot Coupons	No. of Purchases
0	1	12.50	2415.56	371.66	55.56	51
1	10	20.00	29.96	5.78	0.00	1
2	100	46.00	1336.87	179.28	2.50	21
3	1000	26.73	2509.38	312.77	3.75	85
4	1001	24.00	2558.82	264.33	13.24	46
...	...	...	...	...	...	...
2464	995	9.99	456.93	81.73	1.00	6
2465	996	33.33	1172.57	241.04	25.00	74
2466	997	42.18	2476.95	225.41	1.00	47
2467	998	45.00	4840.55	1052.66	26.35	152
2468	999	29.99	957.67	143.21	0.55	26

2469 rows × 6 columns

Now let us find the top 20 shoppers (irrespective of revenue generated by them) & top 20 households who have generated maximum revenue

Top 20 number of purchases by household

In [151…
```
top_20_times_pur = trph_freq.sort_values(by='No. of Purchases', ascending=False)
top_20_times_pur.head(20)
```

Out[151]:

	Household Id	Costliest Item	Tot Rev	Tot Ret Disc	Tot Coupons	No. of Purchases
1464	2337	35.91	6921.86	1066.10	12.65	780
561	1510	128.78	6901.27	849.80	4.50	747
513	1467	60.00	2088.14	340.78	6.49	701
873	1795	43.69	5891.11	1730.66	159.42	694
2360	900	45.67	9549.60	810.04	9.38	674
2206	762	59.99	5999.99	761.80	1.55	612
499	1453	38.97	13376.23	1765.53	74.32	502
1596	2459	40.61	9947.03	1913.38	102.41	491
1345	2226	184.99	7067.38	733.82	51.25	491
989	1901	60.00	5750.40	816.75	56.90	449
526	1479	32.00	5610.23	666.77	32.58	417
1174	2070	59.99	5358.67	1289.83	36.10	404
249	1228	139.80	7998.24	1277.31	189.15	403
27	1023	247.47	24879.75	1405.78	48.47	375
2369	909	48.00	7061.14	1192.29	107.43	369
1876	463	27.70	3217.43	743.52	20.00	369
2216	771	25.98	5646.19	1402.85	17.19	368
2114	68	37.34	3372.52	656.68	27.42	358
2157	718	58.49	10687.43	1883.82	168.44	353
537	1489	31.00	10963.69	2328.54	8.65	348

Top 20 Revenue generators(households)

In [152…

```
top_20_rev = trph_freq.sort_values(by='Tot Rev', ascending=False)
top_20_rev.head(20)
```

Out[152]:

	Household Id	Costliest Item	Tot Rev	Tot Ret Disc	Tot Coupons	No. of Purchases
27	1023	247.47	24879.75	1405.78	48.47	375
669	1609	840.00	16581.36	1317.29	45.00	245
499	1453	38.97	13376.23	1765.53	74.32	502
1449	2322	95.89	13187.54	1277.69	33.45	178
474	1430	50.84	12849.31	1544.01	52.85	202
123	1111	23.96	12368.36	1677.12	6.49	162
1807	400	59.82	11801.64	1135.18	95.33	191
2145	707	106.17	11491.56	1245.04	7.25	295
717	1653	103.67	11190.39	2564.01	29.24	338
537	1489	31.00	10963.69	2328.54	8.65	348
1070	1975	56.29	10843.51	1097.54	127.78	217
2157	718	58.49	10687.43	1883.82	168.44	353
1407	2284	87.98	10176.41	1263.39	20.04	168
1120	2019	50.01	10013.96	1764.88	99.83	225
813	1740	50.82	9994.91	1126.43	6.50	320
1596	2459	40.61	9947.03	1913.38	102.41	491
1479	2351	70.03	9934.21	2089.39	30.24	190
949	1864	46.63	9922.95	856.81	23.80	98
579	1527	75.00	9909.42	1086.00	1.25	185
1386	2264	107.88	9890.74	1506.31	6.45	188

1st, 2nd and 3rd most product_ids purchased by different households

In [66]:

```
house_prod = tran.groupby('household_id').agg({
    "product_id": [lambda x: x.mode().iloc[0],lambda x: second_mode(x),lambda x: third_mode(x)]
}).reset_index()
house_prod.columns = ['Household Id','Top Purchased','2nd most Purchased','3rd most Purchased']
house_prod
```

Out[66]:

	Household Id	Top Purchased	2nd most Purchased	3rd most Purchased
0	1	856942	1082185	995242
1	10	10182810	868764	900370
2	100	6534178	849843	831536
3	1000	1082185	12427353	986912
4	1001	1104475	997987	9526254
...	...	...	...	...
2464	995	1086001	1086001	1092937
2465	996	1133018	896938	883404
2466	997	42346	41332	51716
2467	998	6534178	8090521	1037840
2468	999	1029743	6704207	983584

2469 rows × 4 columns

Separating the Top, 2nd and 3rd most purchased product_ids by different households to later find out which product types and categories they belong to

In [67]:
```
house_prod1 = house_prod[['Household Id','Top Purchased']].rename(columns={'Household Id':'Household Id', 'Top Purchased': 'product_id'})
house_prod2 = house_prod[['Household Id','2nd most Purchased']].rename(columns={'Household Id':'Household Id', '2nd most Purchased': 'product_id'})
house_prod3 = house_prod[['Household Id','3rd most Purchased']].rename(columns={'Household Id':'Household Id', '3rd most Purchased': 'product_id'})
```

In [71]:
```
most_pur_house = pd.merge(house_prod1, prod, on='product_id', how='inner')
second_pur_house = pd.merge(house_prod2, prod, on='product_id', how='inner')
third_pur_house = pd.merge(house_prod3, prod, on='product_id', how='inner')
```

In [76]:
```
most_pur_house.describe()
```

Out[76]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	2466	2466	2466	2466	2466	2466	2466	1829
unique	2466	923	273	18	2	169	391	235
top	1	1082185	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	302	1134	1419	1328	517	477	366

In [77]:
```
second_pur_house.describe()
```

Out[77]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	2455	2455	2455	2455	2455	2452	2452	1880
unique	2455	1179	322	17	2	183	473	278
top	1	1082185	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	184	1038	1562	1412	456	408	266

In [78]:
```
third_pur_house.describe()
```

Out[78]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	2449	2449	2449	2449	2449	2449	2449	1905
unique	2449	1289	352	17	2	193	491	300
top	1	1082185	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	109	974	1604	1470	399	344	191

We see that "Fluid White Milk" is being purchased most number of times by many household. It stays as the mode of household's 1st 2nd and 3rd preferences.

Let us find out which product is generating how much revenue

In [396...
```
prod_sales = tran.groupby('product_id')['sales_value'].sum().reset_index()
prod_id_sales = pd.merge(prod_sales,prod,on='product_id')
prod_id_sales
```

Out[396]:

	product_id	sales_value	manufacturer_id	department	brand	product_category	product_type	package_size
0	1000002	422.43	4373	DELI	National	DELI MEATS	MEAT: BEEF BULK	NaN
1	1000050	442.31	1046	GROCERY	National	COLD CEREAL	KIDS CEREAL	14.5 OZ
2	1000057	10.87	1046	GROCERY	National	DRY MIX DESSERTS	TOPPING MIXES WHIP TOPPING	5.2 OZ
3	1000059	16.47	4974	DRUG GM	National	SINUS AND ALLERGY	NASAL SPRAY AND DROPS	NaN
4	1000092	6.42	2350	DRUG GM	National	HAIR CARE PRODUCTS	HAIR CONDITIONERS AND RINSES	6.8 OZ
...	...	...	...	...	...	...	...	...
68487	999973	98.36	317	GROCERY	National	CHEESE	CREAM CHEESE	8 OZ
68488	999982	45.46	1422	GROCERY	National	CONDIMENTS/SAUCES	SALAD MUSTARD	10.5 OZ
68489	999987	23.96	1010	GROCERY	National	DOMESTIC WINE	FIGHTING VARIETAL WINES	1.5 L
68490	999992	42.61	69	DRUG GM	Private	KITCHEN GADGETS	GADGETS/TOOLS	NaN
68491	9e+05	69.00	2393	COSMETICS	National	FRAGRANCES	DESIGNER FRAGRANCES	.8 OZ

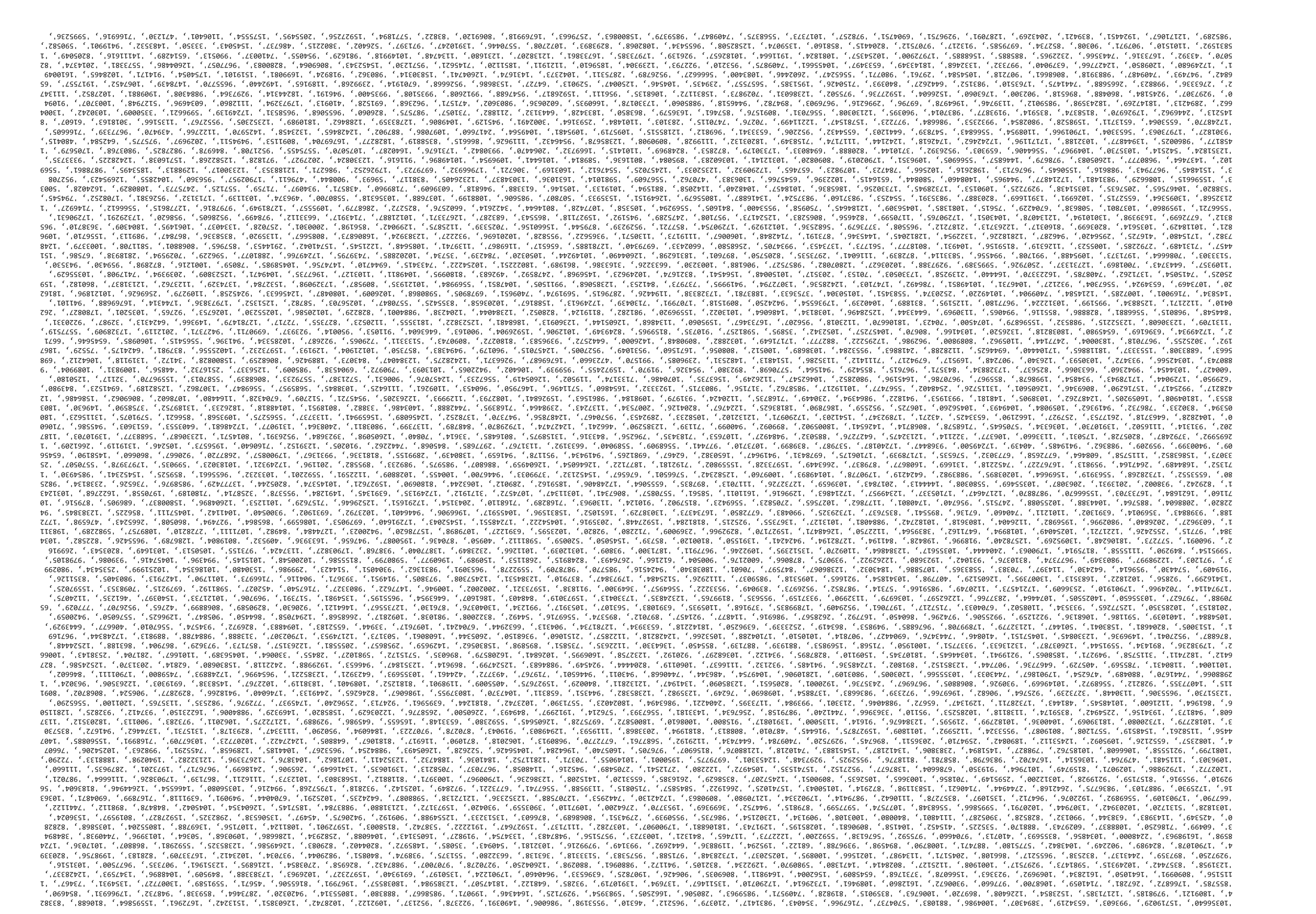
68492 rows × 8 columns

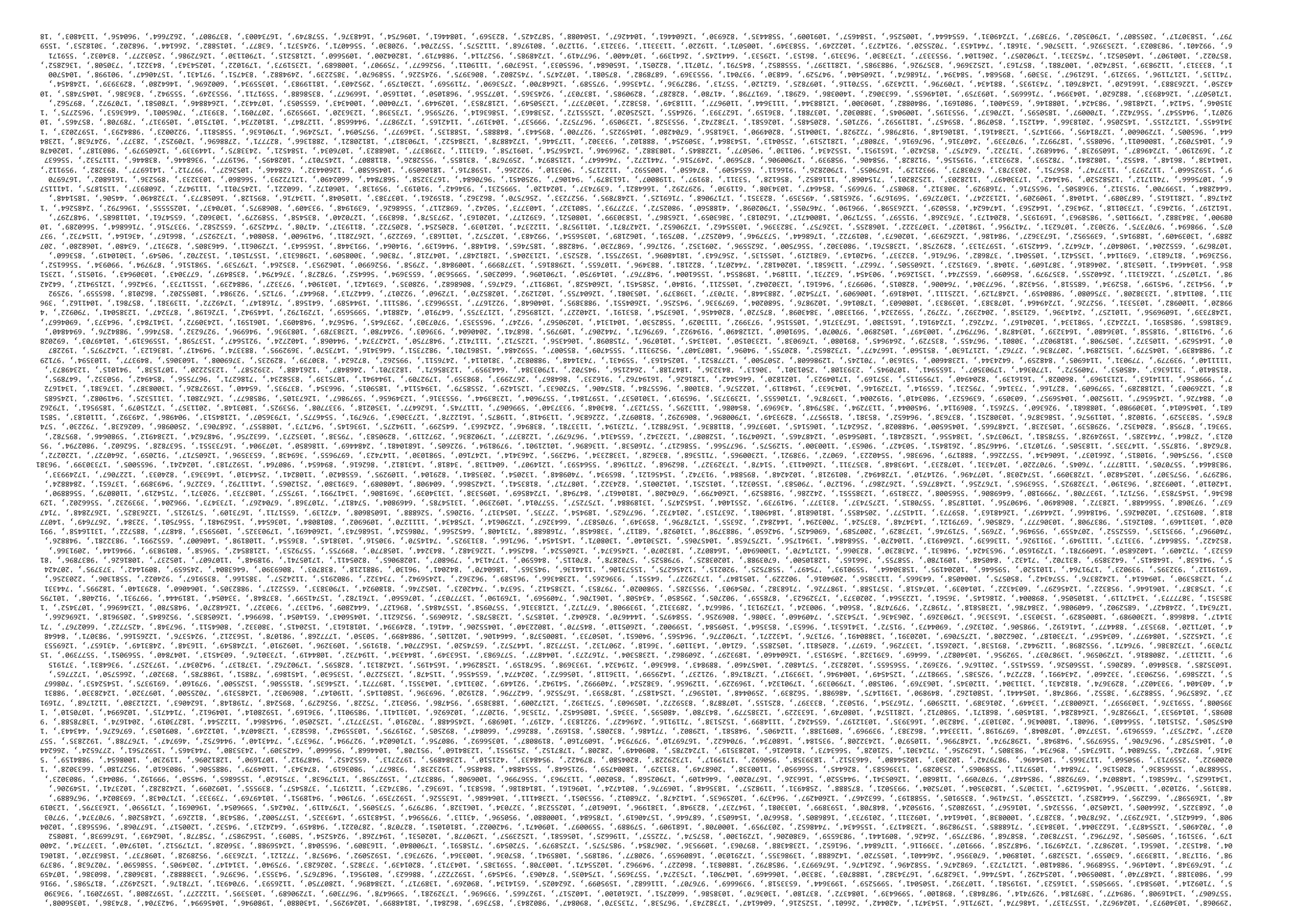
The below is to find out how many products went unsold and the number of products which were sold but their details are not present under products dataset and there are 17 such products, all with manufacturer id = 1 (delete this manufacturer is = 1 details before publishing)

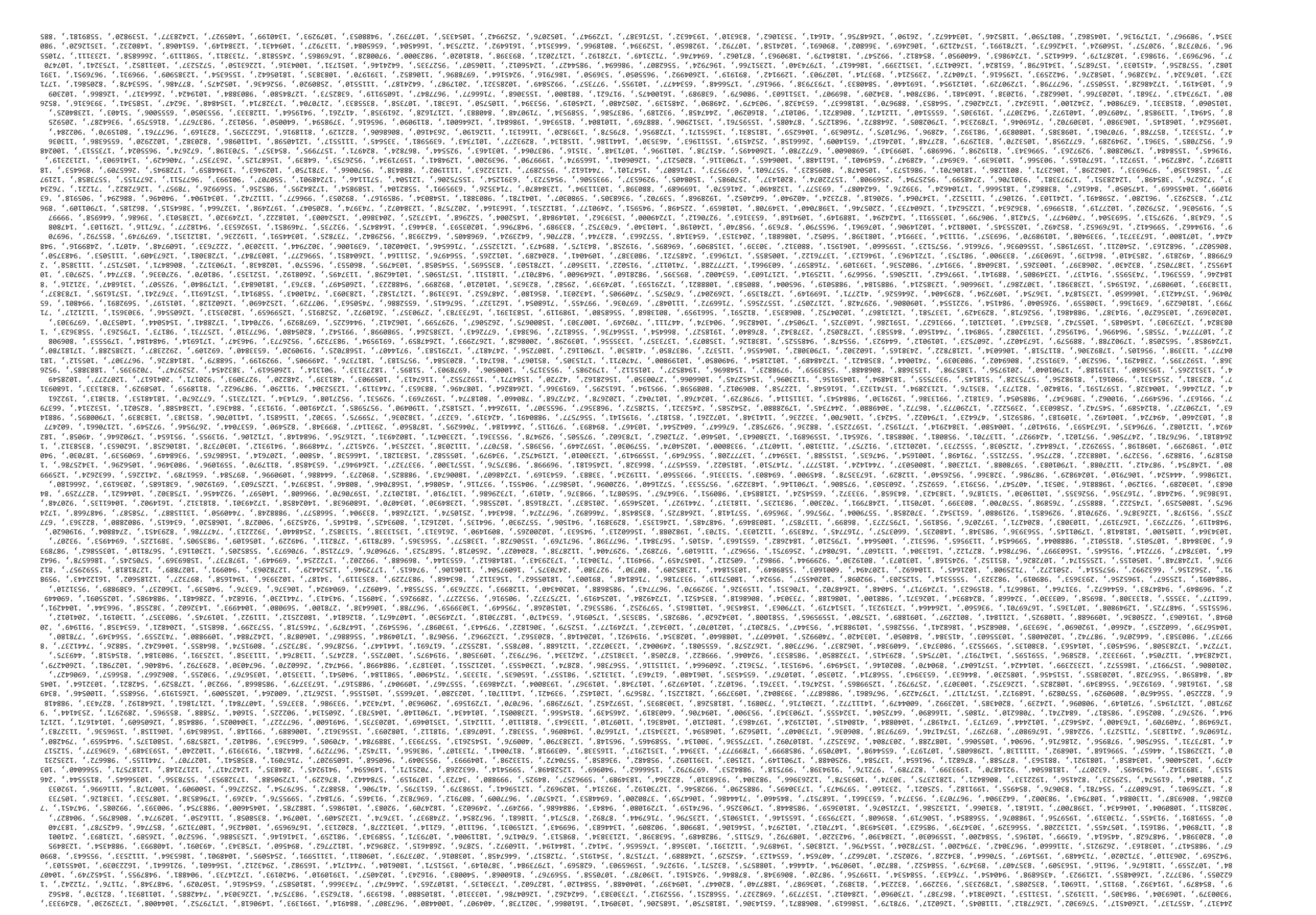
In [397…

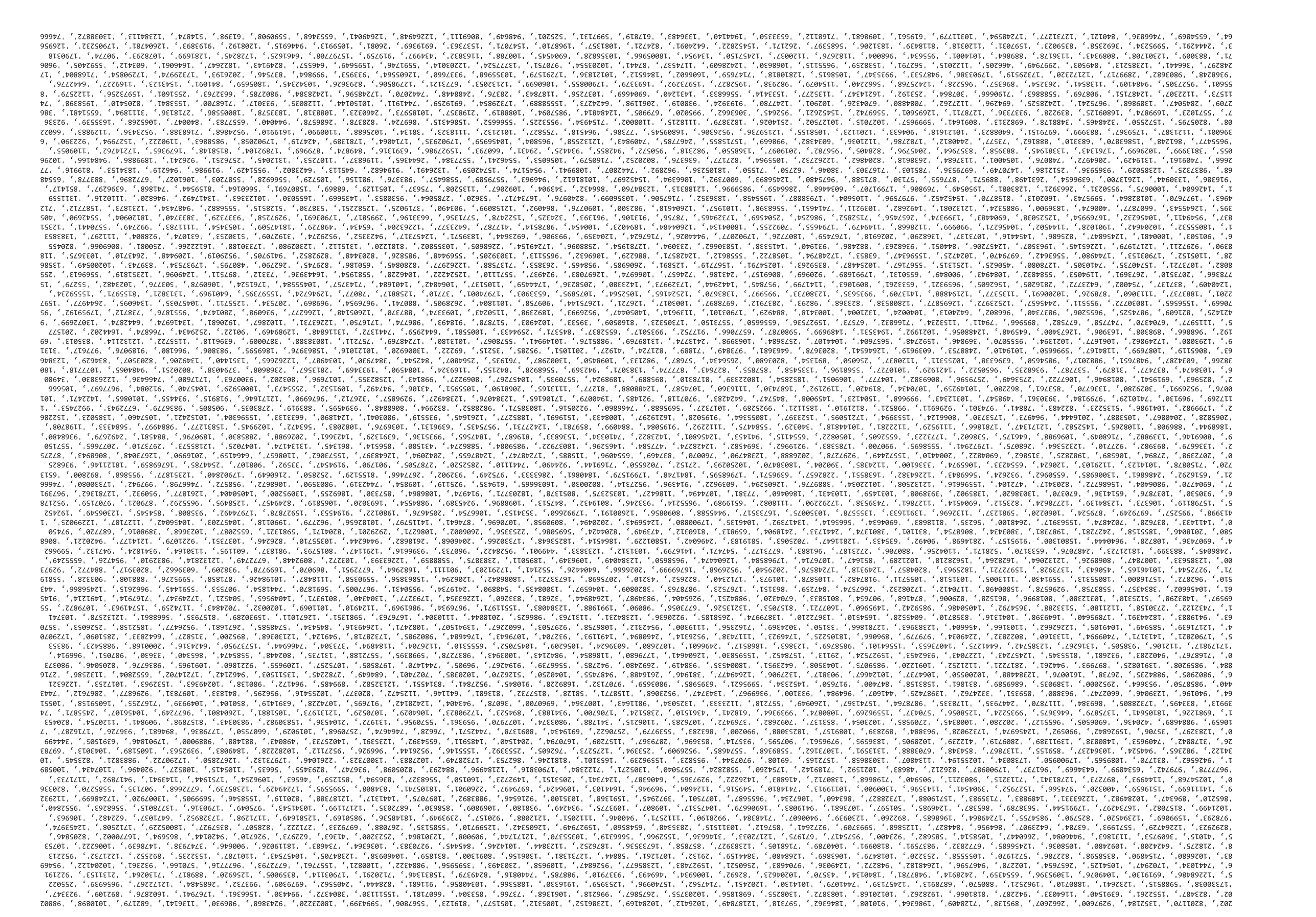
```
# Column to compare
column_to_compare = 'product_id'

# Get unique values in each DataFrame
unique_values_prod = set(prod[column_to_compare])
unique_values_prod_sales = set(prod_sales[column_to_compare])

# Find the differences
differences_in_prod = unique_values_prod.difference(unique_values_prod_sales)
differences_in_prod_sales = unique_values_prod_sales.difference(unique_values_prod)

# Count the number of mismatches
num_mismatches = len(differences_in_prod) + len(differences_in_prod_sales)

# List the values unique to each DataFrame
values_unique_to_prod = list(differences_in_prod)
values_unique_to_prod_sales = list(differences_in_prod_sales)

# Print the results
print(f'Number of Mismatches: {num_mismatches}')
print(f'Values Unique to DataFrame 1: {values_unique_to_prod}')
print(f'Values Unique to DataFrame 2: {values_unique_to_prod_sales}')
```

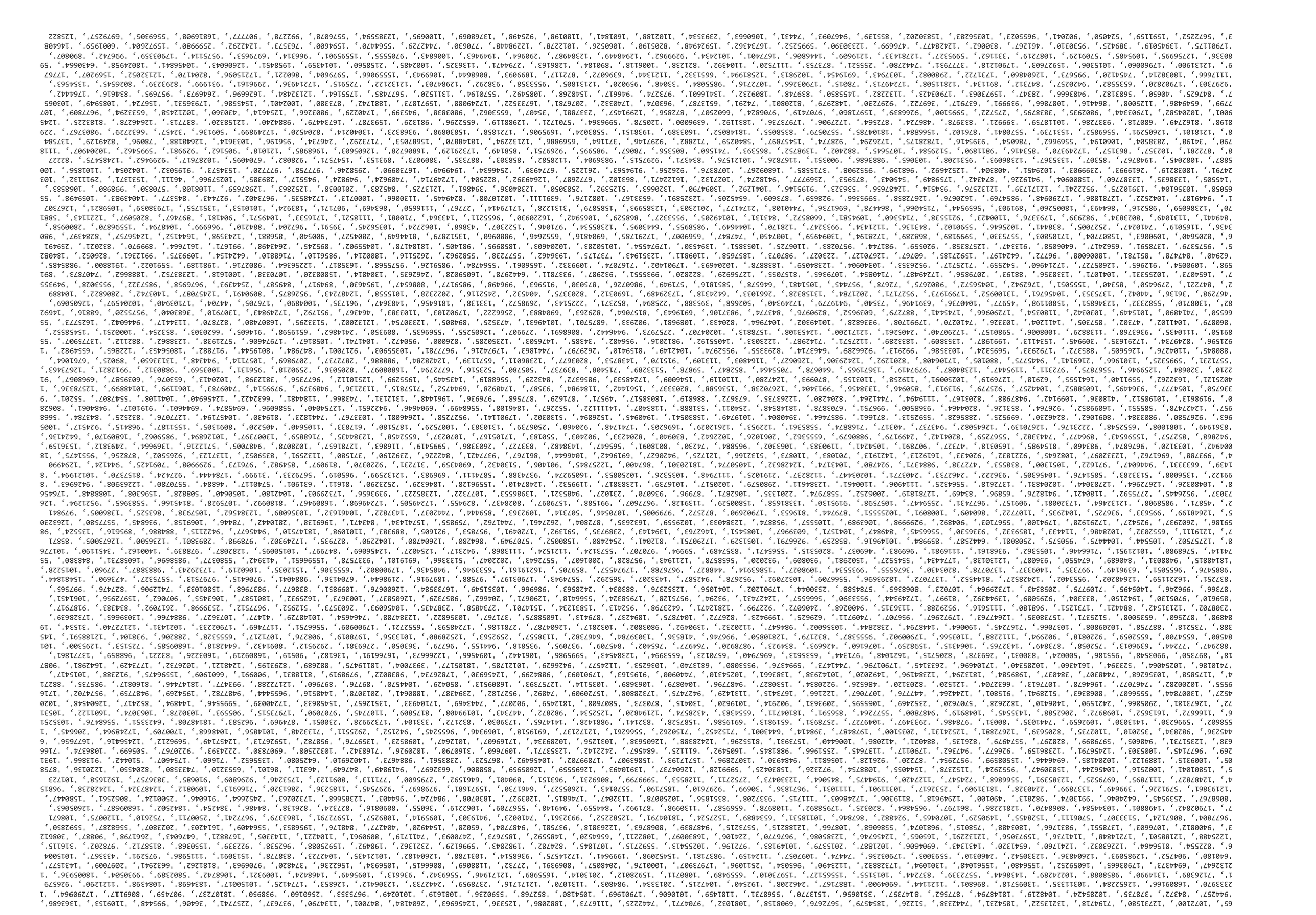
Number of Mismatches: 23856
Values Unique to DataFrame 1: ['7413002', '1129879', '859240', '1840524', '10356352', '730776', '1962166', '6423822', '808299', '1051279', '2343066', '2391270', '7432797', '9553362', '1022361', '9357596', '1935278', '2673360', '6633259', '2060019', '1044992', '325486', '749318', '9652098', '12487379', '664229', '1700247', '1799848', '976973', '1040973', '10312176', '976593', '8293802', '10150307', '989097', '9686158', '8293696', '6602619', '852377', '921383', '673995', '977327', '10355620', '1682003', '8091637', '1813964', '7025308', '1642893', '16219529', '615822', '1372706', '12608181', '1054095', '943128', '37756', '879741', '2051181', '956275', '932453', '61079', '27030', '2467669', '13419089', '1070483', '9489214', '535498', '5566974', '6536789', '6039678', '1303983', '2454767', '15830162', '906854', '978069', '143294', '837453', '8019369', '880701', '2285730', '7442952', '1188565', '1103274', '3293947', '9889226', '1020486', '12731237', '14077952', '332325', '1538543', '9670430', '1030258', '2859919', '9297606', '836297', '1061944', '1109746', '9487542', '113949', '1646304', '1823037', '9526379', '2381488', '1022019', '1212108', '974281', '12456941', '8292985', '10344716', '5800011', '1048312', '361520', '8159556', '7125398', '1609367', '990243', '1547254', '17902801', '779462', '945455', '1217879', '2616037', '9221202', '1322314', '1540944', '8359156', '9337908', '12427853', '17290824', '1113112', '8354334', '1051604', '822705', '1367483', '7441157', '1064821', '5718469', '6704201', '7441342', '833183', '10253982', '12605200', '12811565', '6463855', '6607334', '8020202', '1008343', '1988114', '5567220', '9487645', '5582944', '18206211', '12352173', '12384999', '1091654', '1131457', '8020190', '9654940', '8019303', '1045505', '7168460', '40938', '997535', '917325', '17328852', '10251773', '838992', '449781', '1321035', '1223896', '1485672', '12131929', '9446229', '2029502', '1142934', '7202040', '1989927', '12604746', '1094610', '2399068', '1736396', '1832307', '1033798', '845836', '17104918', '10121633', '1281721', '990461', '12695480', '12263414', '17329056', '679697', '15453554', '18148676', '451089', '185712', '1174859', '9829093', '18025052', '12605003', '7441721', '947584', '2596756', '10149693', '6979429', '9305129', '983411', '1264402', '17214943', '1020339', '7470099', '1740162', '126338', '12352174', '9419924', '1402945', '2548175', '1657446', '13417999', '1010715', '3133282', '6396384', '12256909', '1052904', '2145412', '9526268', '90602', '1576709', '2648323', '1255027', '9405078', '1506178', '1551738', '909550', '9885346', '765196', '1471712', '59073', '924563', '12385983', '1017664', '3247986', '1924706', '555748', '909080', '1058970', '1504223', '1375741', '1979997', '58612', '6552889', '952550', '12947252', '15659991', '10356065', '841581', '46252', '410281', '723303', '18037898', '1078124', '12648529', '5567252', '1015846', '9913437', '12132646', '1207191', '1059586', '9368504', '919732', '970351', '6978992', '5590473', '639695', '12132574', '5981426', '13007945', '505149', '12693352', '5582264', '8019525', '867037', '12456310', '17291676', '995258', '17248638', '898007', '17291636', '698044', '9214515', '9553027', '72374', '2843015', '8124101', '18147457', '1077822', '9445551', '6919440', '6396098', '718149', '1099169', '984185', '229541', '819768', '1272077', '826814', '12487526', '7409939', '956446', '1127634', '1528277', '914734', '96100', '5716552', '17973526', '9577384', '1007277', '2617997', '842937', '30179', '2418215', '854473', '2746796', '1030750', '12352506', '7409688', '656465', '2480159', '487940', '9523220', '2651875', '995712', '17249101', '8015541', '10457347', '9521799', '944639', '12388249', '10355437', '1085426', '1624711', '17249751', '905253', '2671436', '919332', '9420158', '1806006', '869932', '920951', '1146356', '1349525', '40382', '1689958', '10457013', '889909', '5582540', '1423448', '872453', '17902846', '46764', '8358017', '8299359', '504143', '1510914', '858640', '1245862', '200029', '1126694', '842115', '30954', '669453', '17974269', '5592503', '1326313', '12385714', '10285345', '6632287', '262363', '69435', '938638', '9707273', '998726', '8068775', '3204483', '986192', '12812068', '8203605', '940139', '9527227', '12604635', '1829596', '12812046', '12194552', '2448092', '6772572', '9913579', '1164401', '2070733', '1592985', '12189694', '1037153', '1352472', '499889', '6983506', '1130752', '462727', '105408', '5570276', '105511', '9677838', '863075', '18107338', '1076874', '9704766', '7166545', '17330251', '1951501', '9487256', '13943769', '1082004', '17901229', '12386017', '9546328', '2538362', '9194572', '1030786', '10311407', '2814466', '1197704', '13010600', '2025784', '6547322', '12384997', '9479490', '1925092', '994273', '7441211', '12456458', '12462296', '1244622', '987399', '5583487', '1031930', '1306422', '12456384', '2594056', '899653', '999026', '9588935', '7112044', '886591', '6979786', '106945', '1750760', '3029647', '8204050', '861257', '954843', '1062294', '9420104', '969343', '1117944', '17208675', '12385422', '1033078', '12171600', '1132107', '1518879', '15863747', '879271', '899182', '2182786', '6552334', '1656449', '6552447', '1780149', '970483', '1108470', '15927354', '15921011', '837917', '545219', '6704408', '12757322', '12383763', '34776', '8249108', '9220817', '13877223', '6034887', '1410142', '17901869', '26691', '12171674', '1519526', '1081745', '577739', '5995543', '963037', '508777', '16769730', '8019803', '1316746', '1486823', '842412', '9859384', '919520', '1079493', '872427', '2321881', '1030090', '893014', '5588502', '10284852', '1615007', '2022949', '1113665', '483258', '628694', '2807802', '884911', '882732', '9707138', '844637', '17902837', '9220977', '52786', '6537551', '12330827', '7412116', '9396011', '1377402', '5590708', '1266128', '803926', '1045644', '15972870', '1276967', '979309', '9337855', '15863683', '13157800', '17248625', '667518', '6553852', '1331961', '160776', '988292', '17902844', '17106514', '1505057', '13911622', '381497', '2416183', '13944627', '13418006', '1029455', '2664008', '15801129', '16769588', '7143013', '2081654', '217731', '6380761', '1344521', '8066904', '16769811', '1248832', '1017834', '17901319', '999183', '4519357', '12383833', '2540205', '2687567', '7154747', '68595', '2560101', '7443567', '1006738', '7024954', '1198022', '12648606', '12456966', '907411', '3655778', '15778240', '9245520', '1021090', '1132904', '45364', '880344', '12171215', '1299245', '13986034', '17249515', '267991', '46102', '2506283', '17829210', '1462356', '13095357', '1265435', '1053206', '1211740', '10356223', '2033391', '12987822', '12987933', '5995484', '946637', '1110316', '1689989', '477956', '1109464', '1030249', '838156', '12384901', '5569935', '852651', '118843', '9830337', '12675661', '1555683', '103857', '12384097', '6552389', '1073893', '6554407', '1027338', '849166', '984307', '7409943', '906299', '819468', '1188444', '12604188', '9526791', '5549851', '844043', '6039753', '10250988', '1080961', '1015855', '1081613', '931975', '2064119', '8065638', '944414', '10148779', '9708767', '2744813', '2524534', '982512', '1432620', '1014745', '924181', '787189', '1114786', '491933', '10355589', '850459', '9196920', '947504', '1784742', '10207148', '17248890', '8019177', '8310056', '872668', '9487729', '15830637', '873201', '1018879', '1801060', '1446574', '5569285', '5584950', '1025920', '701697', '10121955', '9300081', '5643335', '917848', '17214937', '9487401', '3889774', '848905', '882103', '1801785', '9374526', '1754098', '1023658', '1113355', '975418', '13513038', '44186', '2594995', '935611', '2463155', '5583564', '5707702', '15797466', '967131', '897635', '12171377', '17330408', '959210', '843087', '6919376', '1562074', '3037386', '5980103', '742125', '6552336', '2411599', '1807630', '940834', '995334', '2078883', '3802088', '917862', '8203231', '6431241', '9707433', '8069053', '9803248', '1030522', '1066185', '945595', '1269157', '1424031', '3703549', '1133659', '8013777', '862407', '2076672', '17974084', '1040705', '9243219', '5582595', '6979664', '10254376', '15678839', '9798469', '12457152', '12352145', '1094322', '10260295', '17330185', '6602605', '12171661', '1987650', '12605691', '1119086', '44802', '5573719', '17291075', '1695306', '1111236', '5979257', '7143886', '5133310', '9220902', '51076', '1048732', '1402844', '1167263', '12301872', '91119', '880890', '1208604', '566870', '879298', '6534913', '8293880', '764499', '6892256', '2527108', '9885415', '9450764', '2820332', '1379234', '607934', '5568792', '10311243', '119451', '15971784', '1027766', '927625', '1078314', '10154492', '1095847', '7413121', '5566187', '9880287', '12604491', '1509220', '9803183', '40394', '1183520', '9807328', '10312345', '1616507', '913729', '16198217', '1250276', '18121252', '12485332', '1110836', '12525042', '863452', '9440105', '17248306', '953087', '939121', '197525', '15927251', '5567615', '1621091', '12201823', '1883791', '1008923', '13007333', '6034455', '1922555', '1815931', '15489806', '986793', '8063107', '110450', '580463', '161158', '913017', '9836357', '7168400', '1411451', '5566508', '12810487', '1613496', '9527163', '9296940', '14024901', '8085916', '6637229', '12332283', '824784', '9712837', '5133239', '5569042', '558191', '919326', '978550', '5560928', '2973511', '851015', '17903403', '15567566', '68499', '6979119', '833307', '1117416', '1830400', '1882615', '6602307', '1130025', '12205904', '1037190', '864603', '928246', '1078902', '842934', '9677499', '1024387', '1261151', '1177591', '9931906', '1020057', '2616939', '2602313', '908216', '7168253', '925956', '18148208', '17899836', '13945093', '566591', '839101', '914648', '5571051', '9885532', '113570', '18147773', '9803197', '691343', '2482283', '1111091', '5757604', '90213', '7414828', '1886755', '1053666', '5570338', '1646237', '930864', '12385706', '843260', '1069475', '1250020', '8155505', '46370', '670522', '8118582', '1084594', '7453992', '1098940', '5590557', '938363', '1065944', '6822873', '1406375', '17290931', '1692670', '1934764', '9829563', '881318', '889502', '1469767', '816637', '978356', '1867877', '1631064', '1758471', '937690', '1028480', '2592701', '28157', '8236004', '1312143', '851623', '1474198', '9859045', '5133371', '5578752', '15679762', '833064', '954465', '15863838', '10150108', '1870851', '1337628', '8121909', '6703833', '9527463', '1754440', '9677105', '2688043', '6547705', '5700155', '1871210', '2072259', '18106286', '7138116', '1198038', '1481266', '9445733', '1006398', '1866855', '2820778', '731807', '694130', '17904900', '2592807', '12131992', '3025516', '12812009', '541329', '10258820', '12425070', '924833', '6569871', '9639587', '1126918', '12427829', '5581116', '1056116', '1129195', '1107497', '116799', '12330306', '12604685', '1019330', '2552137', '1029386', '1109997', '7164711', '6534292', '12457097', '836539', '9301204', '2630154', '1025594', '17902375', '10455948', '18293439', '17249397', '1479873', '6908745', '876974', '69267', '12386099', '2013559', '6634398', '1007079', '1173776', '2537417', '6754452', '12430568', '848172', '1109781', '8292782', '5589029', '93445', '9553121', '1056082', '839502', '955578', '6534720', '121067', '9553139', '9837578', '373990', '9959585', '10290232', '9552803', '446852', '2368544', '10204350', '62904', '17252607', '1370524', '866015', '17990988', '1057033', '1433317', '5576399', '1107382', '307003', '2062633', '1240755', '1410195', '7409690', '861634', '823354', '9884072', '1547826', '878909', '1053248', '8068808', '9857257', '12384979', '6979696', '1413199', '13416527', '7441057', '29977', '969105', '1010282', '6535995', '10312077', '874727', '866480', '12457033', '5726172', '1098785', '10285352', '8160046', '832140', '5589210', '1158024', '6030276', '1554998', '931732', '9552420', '1064853', '1510255', '1331810', '1933030', '1021438', '1668889', '863091', '1151454', '9301716', '1824453', '28346', '8156746', '8293782', '17902790', '3316834', '12185030', '10369057', '1273010', '928389', '1768973', '959469', '12487598', '1090408', '823129', '12578000', '2630867', '1423595', '8183452', '6461741', '1081628', '1939427', '7177346', '1854631', '12386142', '16771791', '272910', '591391', '15724794', '359526', '1093626', '1907823', '17249418', '15934346', '10254436', '17901793', '14029047', '962132', '956867', '993824', '12301157', '18243471', '1137125', '1575209', '1086583', '3909433', '1085460', '40722', '1006371', '6534017', '1068423', '2041847', '916584', '9677891', '12427654', '984317', '17249507', '6533528', '630348', '12524052', '924584', '309997', '7409702', '1115947', '9837117', '1874586', '28424', '831696', '1503908', '863523', '1363834', '9477612', '2672157', '5742427', '916597', '2011729', '9885180', '3832174', '75738
3', '12604723', '7167275', '17937077', '2636819', '12777653', '1943897', '6633232', '923008', '672839', '2846160', '5979563', '9879419', '5592686', '12605394', '1048346', '1026506', '12757681', '8069112', '12193085', '1328041', '13416265', '5566681', '308939', '9677744', '1024433', '1009692', '92984', '17974595', '64543', '2539728', '992380', '1205055', '994503', '34197', '17903399', '8019423', '12385748', '2588528', '32608', '1089629', '545564', '8156094', '1082679', '219996', '1365202', '1101208', '10249982', '15863849', '16223274', '1012387', '17291600', '744343030', '8203652', '1071936', '1565297', '1830785', '12385854', '12132392', '1822301', '12384167', '450736', '8158208', '9674028', '963487', '2679218', '131409', '1033263', '890425', '847577', '4004755', '500828', '1082127', '5591517', '6979190', '48893', '116169', '7162824', '1868373', '507957', '10203985', '1541865', '2021544', '3514213', '887669', '2047155', '921912', '1475430', '8155934', '1890350', '15596603', '9485620', '8145246', '829487', '410204', '1049536', '18148439', '826920', '6845611', '860854', '1112451', '17329773', '7199608', '6555327', '16057147', '17179711', '906402', '12606028', '406820', '5566451', '10457552', '1269901', '855512', '886616', '1905523', '10285025', '1303272', '1192620', '15927110', '77280', '2413713', '985886', '13417865', '50658', '1097223', '1535159', '17106513', '7167683', '862473', '7441763', '2281854', '10255464', '40504', '873772', '5979872', '2499126', '1567673', '1763944', '17329806', '5573261', '1136620', '599258', '7466348', '1057843', '81587', '1192510', '2814731', '15927258', '12781389', '942982', '1004114', '825122', '891040', '991415', '17897739', '83153', '9527050', '12273045', '12647296', '2294654', '859926', '9803138', '3758238', '2616282', '885172', '1053449', '414803', '6879088', '1381708', '9487592', '945488', '9575601', '9337892', '1103841', '15484460', '518484', '936439', '284946', '962533', '1930020', '7157378', '6982429', '5130756', '1467266', '885762', '114102', '13417371', '1898457', '120919', '866525', '727224', '6470031', '2616569', '2512150', '1771152', '990049', '17285598', '6395998', '997818', '9245294', '2227650', '7138001', '1222612', '1460216', '5587214', '17903154', '2047752', '12330877', '17973382', '13382278', '1632988', '1919959', '821859', '6544680', '911519', '723354', '828870', '1095919', '12183251', '1116990', '8203488', '8090564', '12729530', '1028205', '1059346', '17179924', '880604', '930713', '10356061', '1448835', '12352333', '9839775', '419764', '8068818', '9529723', '8243828', '822690', '6634442', '5875574', '712213', '1013809', '2025754', '16730332', '79318', '1124021', '876159', '1447726', '1009086', '1927784', '1998861', '221474', '903309', '10461566', '1850278', '848536', '12427828', '12384229', '6442630', '17903744', '78888', '6039054', '9553396', '3015587', '908036', '2062838', '1483968', '17249765', '755522', '1253364', '1171484', '12757343', '1454842', '12395611', '1111215', '709840', '10356801', '996822', '17285551', '1032383', '1009054', '702733', '33128', '432984', '1355022', '6023352', '12487487', '692765', '8310969', '78094', '119118', '2642679', '15971549', '1448188', '8243534', '9885473', '1214889', '18038459', '1639231', '428815', '1071163', '1001421', '962495', '895356', '14050891', '1412497', '1558125', '3841423', '15830919', '10204605', '1137841', '667982', '9654795', '1202652', '17902376', '934022', '3812041', '1132677', '12385307', '15830663', '12185466', '8175871', '909148', '1859441', '1103981', '9682847', '2577736', '17248200', '6961620', '13911207', '922128', '860367', '57157', '9302594', '509837', '5583317', '928024', '9803285', '10456116', '1413664', '939280', '1003503', '2528097', '12487145', '1547085', '928358', '6919313', '2609046', '2628250', '1018847', '7452711', '12428307', '8090552', '12457145', '7194717', '17240848', '835301', '1136215', '5570881', '4216812', '2440370', '12263968', '1084966', '1306970', '17903551', '6570240', '1611104', '9368175', '1747015', '1868079', '902280', '9187169', '1054109', '5565320', '9483881', '1419663', '10457248', '427308', '1061517', '1052700', '8068710', '9673403', '1631897', '1051909', '7024977', '1043107', '1014247', '5995220', '6864545', '2002696', '919917', '1460905', '9837159', '939547', '12262594', '1627221', '9883931', '9422911', '1087225', '926586', '1049066', '6410433', '18183744', '2360149', '16058819', '1051734', '821980', '1052558', '5709098', '10285433', '9860237', '2484265', '151245', '13013322', '132725', '1632441', '5590975', '132684', '9829494', '5565378', '15628057', '15927507', '6772974', '8293123', '17329304', '9704577', '2439351', '12457216', '1076723', '1134558', '884962', '1283486', '941097', '6978839', '9554726', '5995329', '1083639', '1118329', '17249553', '1106993', '1044227', '12301710', '1106714', '5591406', '9552872', '258948', '830321', '5773473', '3720104', '12188011', '940435', '1066642', '1733566', '7466194', '557061', '17974309', '10355868', '2047043', '1013678', '8159106', '834666', '6534234', '1133421', '9803004', '1136642', '137420', '9527642', '18148576', '1868765', '12731323', '995661', '1031441', '17179431', '2068608', '1027795', '682910', '12605982', '5565889', '1583853', '902479', '636801', '9653207', '9858741', '948505', '2663600', '7413008', '5571791', '17249512', '1042494', '6632390', '15452826', '9828745', '7167256', '12456302', '962497', '10355806', '819573', '12695716', '10457291', '8157140', '1653930', '10456222', '1064262', '2014133', '790627', '2624924', '17972400', '9194856', '10204517', '13672227', '9219910', '168266', '6396302', '894173', '15927390', '7168596', '889525', '959113', '12158849', '66750', '2076030', '1121588', '251672', '938392', '1053625', '1537376', '1863891', '10355965', '1796706', '12383807', '969263', '1049295', '1851060', '8203846', '903070', '5576554', '9707158', '1104931', '1230546', '17105328', '12649584', '1816907', '877134', '1133705', '6839180', '5576522', '12523692', '12427950', '2348799', '1405371', '904048', '945762', '2470670', '5582357', '2645150', '1007996', '5568613', '17929071', '12387395', '9796734', '17974351', '6462595', '7410275', '1522509', '1060424', '254309', '5568520', '1924807', '17973821', '100043', '10463041', '12132657', '970163', '1023829', '1000394', '12538549', '1119783', '888215', '1188488', '17329702', '2009513', '290420', '12262891', '67254', '17900994', '927020', '1039482', '153304', '17283349', '1015245', '429364', '5995341', '1069508', '12268265', '10123375', '9677441', '1029235', '286128', '909709', '9575331', '6447448', '1183474', '5980633', '867880', '10153963', '829741', '18148441', '6633022', '1131622', '2227654', '930381', '8292982', '1178788', '12554203', '9832594', '5129569', '1443691', '1226357', '12456359', '12171872', '8160126', '1087913', '2042988', '711135', '8046920', '9423802', '10119158', '1849775', '874983', '17992899', '1994074', '2487749', '8354206', '948662', '1035148', '1057512', '12385253', '42150', '978860', '7149725', '79001', '845676', '1466112', '9527259', '831224', '1699543', '1212380', '1319024', '8293966', '9297287', '8204414', '805785', '17329104', '1138367', '7166566', '742873', '6919522', '1020751', '1408027', '18056430', '996557', '935576', '946266', '6443026', '17249835', '1761365', '16769916', '828956', '863186', '1105817', '36413', '6545277', '2487020', '8358051', '489939', '1135452', '9839106', '8091026', '2624045', '1022316', '910856', '1486331', '817040', '5981147', '9796964', '17900943', '987459', '10355560', '1094444', '88964', '389959', '13877023', '9882362', '13039403', '774499', '9836446', '2029622', '15630647', '9884647', '12487257', '870408', '96670', '17900914', '7463741', '1018984', '6533614', '1734754', '820656', '12384613', '921870', '6632989', '9858918', '1600582', '17249474', '18055205', '985921', '1710359', '9657596', '1367945', '17240079', '12487380', '2448366', '9334100', '1486159', '1901418', '1026910', '1101810', '1075179', '9885459', '1087101', '9575536', '45308', '12330291', '919608', '10150219', '61751', '967530', '10122879', '9803632', '9677634', '9655738', '9861639', '7441392', '8292849', '9884358', '15573144', '37427', '1100813', '989268', '27158', '822377', '1136720', '12385595', '5589712', '923080', '816986', '1010011', '10206695', '6979616', '405930', '1191218', '2007872', '17829834', '730947', '7093322', '905649', '6514229', '1172271', '9858972', '12132774', '946009', '1865362', '2015537', '1835111', '8157117', '1959474', '907871', '546869', '1306813', '2768166', '985199', '910179', '6463495', '10255496', '435391', '1032053', '50293', '2359600', '10150008', '1218459', '9837268', '6442611', '1023269', '15926619', '1913120', '101345', '12383915', '1096743', '2556783', '12457158', '5576306', '1628504', '15594233', '1127658', '9884910', '1250403', '871416', '1787537', '17903111', '914939', '433499', '9881502', '12696085', '1135396', '718478', '9934856', '5581370', '18055329', '65751', '9570790', '1463721', '1003435', '1425609', '914459', '495238', '17285377', '1005905', '1021897', '1167371', '1923790', '15452174', '889194', '5572979', '9839942', '13944997', '9213534', '734426', '2660052', '2426107', '8022119', '991900', '1211446', '12456921', '821893', '496555', '898948', '17179792', '50776', '1542246', '389229', '9829966', '1102831', '17329334', '17291590', '64583', '1096005', '1118641', '1118662', '13777546', '2002287', '987259', '707657', '2670431', '1390253', '855231', '1476612', '1151415', '3791298', '466155', '957052', '822685', '961759', '7410115', '783480', '1795086', '1614922', '15564572', '852469', '79450', '17901911', '5133399', '6772724', '1024450', '9420106', '8068380', '9677611', '1115056', '1380910', '5561059', '10356879', '16099957', '1729165', '6633385', '1169565', '8019113', '16733610', '17208519', '6569773', '2022283', '982807', '5571128', '6443250', '935566', '82526', '17214857', '8358720', '9296994', '13512765', '8205746', '624419', '855110', '2563513', '5995598', '8181186', '826467', '1020129', '3932774', '9932138', '9858830', '892420', '1613722', '1026264', '17900905', '8205839', '984619', '1031777', '9796237', '878104', '6400825', '999286', '9220116', '10150072', '2674499', '970526', '8019875', '9527407', '10354562', '12456103', '8019658', '1076551', '9884958', '488782', '500898', '93737', '1354554', '358373', '133974', '3905843', '17901205', '821748', '6632892', '18106138', '524165', '7410350', '12352209', '9884349', '912024', '1931264', '890913', '9527039', '2520265', '2799279', '6919549', '849319', '949579', '2043948', '2555124', '1055193', '1051835', '17291627', '883701', '915053', '10309284', '17936675', '1428032', '9883765', '209999', '12171706', '12263769', '6555345', '5125711', '7152652', '1034654', '1095904', '1028611', '18031553', '1218440', '908505', '744223', '9925873', '10462531', '859319', '1055806', '929326', '6463608', '12301047', '1096980', '839002', '1631363', '72302', '1069682', '980890', '6424383', '929268', '860081', '1093943', '1010942', '1743644', '1119283', '2587240', '1142047', '8013854', '12672820', '12578251', '8032443', '7166744', '5685694', '9270718', '1640515', '981941', '8019187', '1329854', '18148128', '17901912', '9363396', '10457325', '2427103', '5996261', '994579', '9532208', '10356486', '13417200', '5581193', '865725', '9677209', '833362', '17903217', '2013957', '965756', '1047799', '1160597', '106627', '572062', '9676829', '9492337', '12385224', '1005941', '9654880', '563013', '5978721', '1102775', '5125214', '2099173', '1693497', '15973195', '1620538', '578899', '1665981', '1058908', '7119117', '5665168', '6633017', '8091122', '9446459', '877508', '1857800', '17936506', '298145', '12194882', '1063310', '12757456', '29618', '977862', '1737682', '893108', '10255332', '928870', '759115', '1061107', '1347110', '10121458', '2385704', '1942920', '2594648', '5567653', '1049330', '1059198', '10119611', '962164', '1123699', '8154606', '1959023', '366758', '984944', '886239', '986985', '1396882', '1127030', '31161', '9677749', '1288421', '2323272', '6753826', '7113135', '10121805', '1403730', '8068616', '18106047', '18119460', '9858968', '125348', '1125182', '12570246', '28889', '886723', '1033835', '524374', '902187', '10255004', '1020921', '5670299', '986871', '9526361', '1263632', '993151', '12263469', '1195119', '9525128', '9677794', '957350', '847454', '914272', '9707272', '8090446', '1027802', '1062795', '1668695', '15925181', '593705', '9424327', '6904468', '1539953', '820771', '1050387', '9522731', '9803361', '953884', '16734328', '5750524', '922875', '1058503', '634199', '1057374', '997831', '2756336', '9423981', '10250749', '983194', '7443204', '1728528', '963378', '1596321', '882070', '1074690', '15927420', '9336532', '776506', '10184732', '18148632', '10121602', '988845', '10355644', '2591199', '36599', '431608', '15512052', '17902374', '6978316', '9422993', '16769686', '884895', '15571575', '9885058', '10285166', '15716377', '1033826', '12384623', '997949', '1131642', '822667', '2935351', '9803068', '12171655', '1116053', '7159602', '1136288', '1117860', '1150996', '18183739', '10180358', '900251',

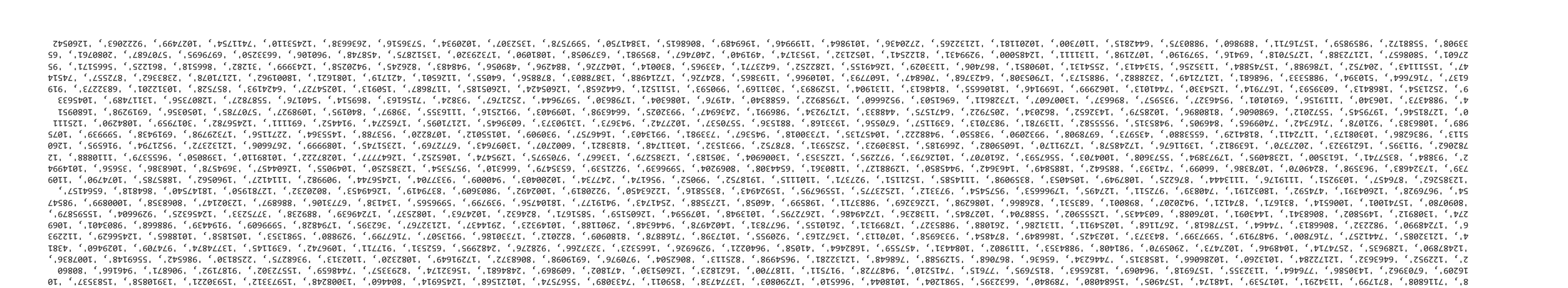


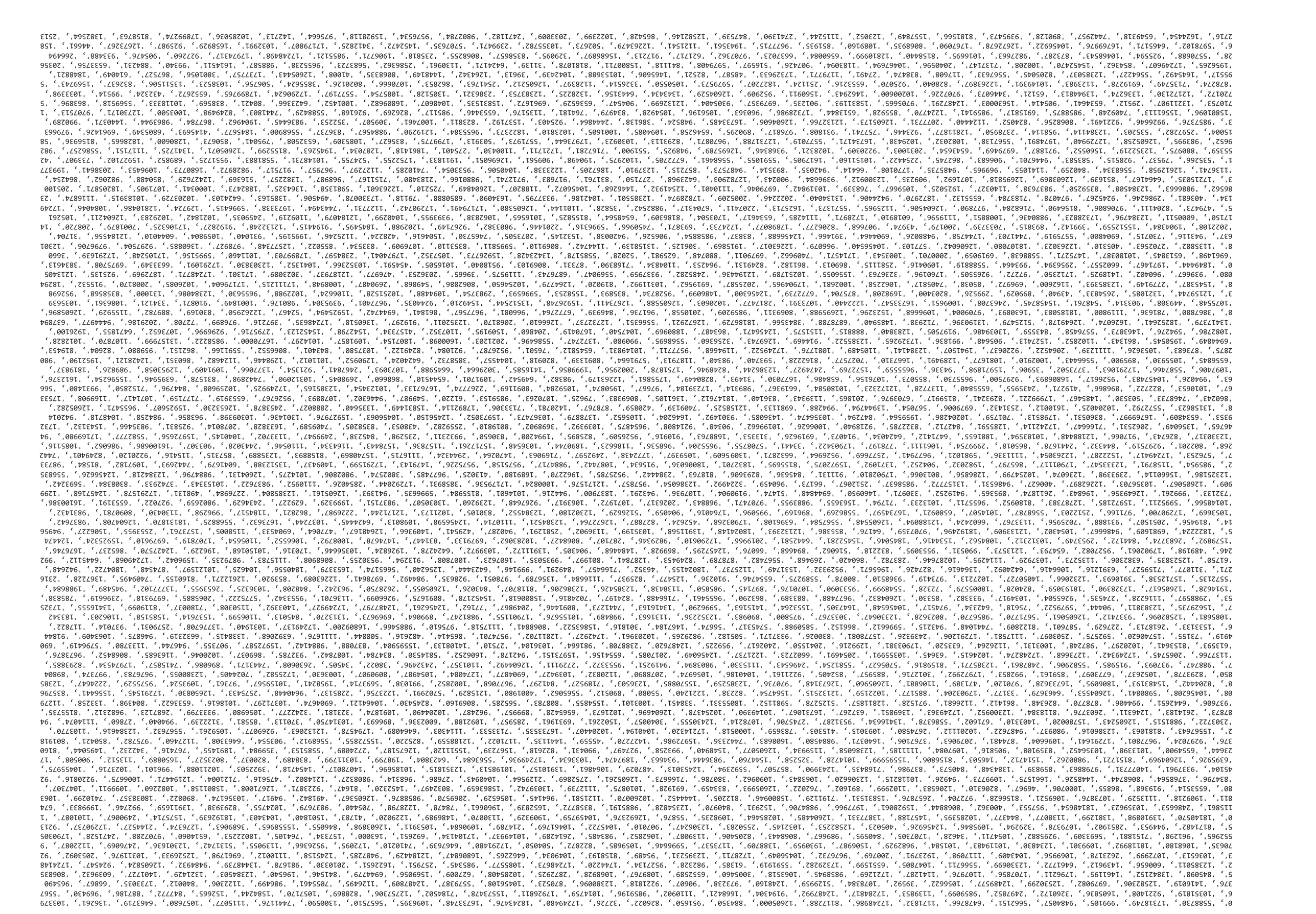


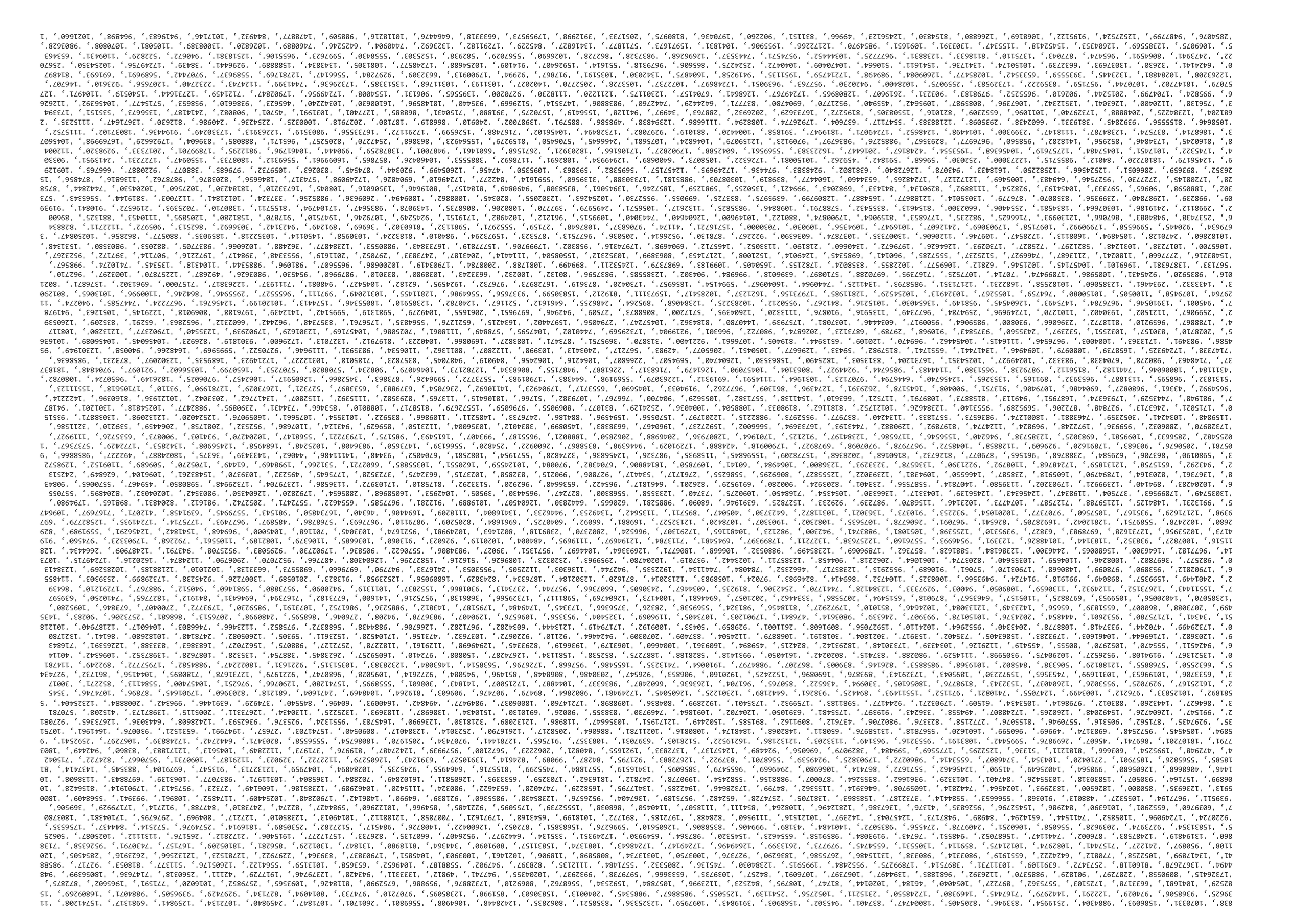












0255118', '1999338', '1047514', '1000441', '15626895', '1630294', '17179096', '1762097', '9803440', '967413', '13072822', '953477', '7409673', '856424', '1111820', '842402', '645551', '1213060', '924096', '13117952', '10285167', '5569196', '126715', '912328', '16767863', '1837966', '12757088', '871158', '937872', '17899707', '9655374', '1525350', '54283', '6553940', '10150229', '1187149', '15831420', '988899', '571596', '2066751', '9802680', '906182', '1173531', '53337', '5978156', '9526901', '9798999', '1270785', '9884864', '16100307', '17248718', '948116', '987848', '12187830', '923355', '8156116', '10285150', '104803', '1988937', '332873', '85393', '1950736', '1133370', '99846', '5995085', '1111094', '15830203', '8293283', '12672919', '940895', '9445761', '9526712', '2732764', '81586', '653330', '1842184', '1573412', '2675584', '1112229', '899200', '9836393', '10144031', '9959004', '6978981', '7412877', '7441976', '1083878', '1079253', '853930', '17249716', '17105034', '1126995', '5981023', '5569240', '936640', '549097', '13910883', '968432', '8014022', '1536244', '1008552', '2972235', '9884623', '10254199', '12188560', '832098', '5567553', '10355761', '16809531', '1017300', '6519584', '7147829', '15972228', '2614147', '13512030', '12385696', '845397', '1329768', '9527412', '982513', '7441573', '10150028', '1073249', '1360022', '13039590', '480105', '17328663', '9683758', '1040280', '359698', '5773760', '112295', '1734795', '395989', '10355411', '9487587', '12346850', '1100276', '9487441', '8068773', '939654', '9490977', '6513533', '2505748', '819989', '8181537', '6704348', '17248528', '849508', '2066533', '677973', '9194491', '989182', '994102', '10284457', '1103360', '2828227', '2440371', '565679', '995469', '429988', '1509794', '123983', '12428231', '1621185', '7441925', '1440256', '5682821', '928269', '934168', '13007843', '9245353', '9487591', '10355935', '1331624', '1876826', '1011328', '5996216', '593012', '850187', '9419531', '928537', '9487322', '9546240', '1104095', '1800620', '1039519', '9526793', '9707472', '1017318', '9836574', '17903021', '12427600', '45971', '12677764', '1145464', '1320372', '6919472', '1849577', '12673059', '12147899', '1530673', '2176860', '852319', '2499998', '1094028', '2065971', '2022762', '2488540', '377804', '8162764', '12301312', '10285392', '6005965', '8381293', '9551111', '9859942', '3784796', '6525354', '853709', '1207681', '1531901', '47272', '854667', '8068809', '1125553', '907886', '10456573', '5996434', '584208', '1003993', '1120200', '9518043', '97935', '836315', '12672832', '1087904', '8293457', '1755643', '9887232', '17903267', '2439191', '2456700', '946458', '1092645', '12428084', '9425312', '84593', '968790', '10254765', '839412', '18273051', '1021170', '6772650', '977898', '5131719', '859172', '2799701', '9553124', '18185766', '1731674', '195896', '2420142', '8204826', '820423', '1018009', '918468', '7409924', '1013587', '12330878', '1060152', '1114741', '12301378', '12385384', '1048210', '9483866', '5584947', '1000499', '7442659', '12184225', '57707', '1426152', '9250325', '1014757', '828932', '7466847', '12330940', '17249416', '64569', '16769828', '45907', '622318', '374453', '822016', '7151959', '9245921', '1813707', '12487673', '234687', '15741347', '8205573', '2032765', '1829930', '970982', '6424118', '838617', '9552988', '9859145', '9883959', '12171485', '36722', '1096449', '1678577', '1819634', '444071', '848361', '15863557', '5586142', '435678', '9803639', '9193045', '10455976', '821512', '1119684', '12158905', '997040', '887912', '793114', '1095626', '9839319', '358107', '14050549', '1006766', '7167833', '1100758', '103133', '9401385', '228847', '13986664', '824068', '6389894', '5589006', '10457526', '6980078', '1552971', '1276042', '2073559', '915399', '566963', '15863815', '95025', '689420', '15973204', '10457227', '13008410', '6633236', '7419529', '10201212', '253490', '263040', '2072059', '1603274', '6034378', '6533885', '2478675', '828290', '777328', '1040251', '12456788', '1864347', '1036043', '1034754', '1227556', '12384483', '934197', '122372', '27925', '7149904', '12383943', '948437', '1373192', '12673369', '17171877', '1563339', '897702', '12132022', '10286463', '1003611', '884836', '938767', '1282670', '173003', '834581', '17901150', '7166456', '1557055', '972340', '12757001', '5573811', '10121830', '2532981', '110469', '3702413', '16124885', '10282473', '1143187', '6633315', '13094981', '401725', '5571718', '1644387', '1448401', '1117051', '18005477', '1021794', '1061355', '8248932', '1138201', '1511281', '1400708', '958180', '2012756', '1017639', '9368032', '10211969', '12132046', '380545', '1413798', '1008163', '9903575', '1290532', '1008806', '10122953', '1125439', '14027861', '2557978', '1108093', '384821', '6978895', '8357894', '1069875', '1829466', '1517862', '1064721', '1230050', '17291653', '1915282', '872073', '963654', '9858736', '1979887', '1005397', '50193', '5566620', '419165', '7409540', '17902815', '841249', '10285483', '873532', '9655065', '1886283', '17898587', '296178', '10200499', '112446', '1144244', '12132289', '1973037', '1037872', '1000222', '13006885', '329555', '915822', '9682728', '12673175', '12607563', '13877021', '1002588', '9333577', '8181039', '7202791', '12132320', '857871', '2611284', '9802592', '2026984', '9802722', '923672', '5651341', '12648309', '9532795', '297383', '9446206', '1034757', '8179366', '1838691', '9677385', '990282', '231629', '8373105', '826406', '9575570', '975016', '2409343', '1339163', '6463867', '12777788', '5677564', '1869644', '1459810', '1125230', '17969732', '1214976', '6396172', '12696016', '16059182', '2074634', '12384348', '6632420', '2006910', '17248267', '2141641', '17329562', '16733549', '9677429', '404120', '895861', '9532678', '9832746', '1347667', '7190731', '1856437', '1083468', '10255255', '9576074', '951640', '1229538', '9859046', '6942901', '6396242', '895122', '6391555', '2008406', '17903155', '5591658', '871989', '894823', '202272', '1149233', '1809880', '1064782', '16730761', '100384', '12524926', '2649362', '986314', '9655703', '1403531', '5131880', '2761041', '545410', '1255334', '50233', '3036341', '969436', '1299319', '962394', '10145968', '959861', '10456351', '8249439', '5568660', '7024875', '954527', '182402', '1168287', '2418093', '1649130', '703189', '884291', '60466', '1757013', '13511941', '12171232', '1073874', '17291956', '1330747', '10456421', '8359131', '17248998', '10457118', '91677', '865260', '1032211', '990588', '12605679', '9681203', '2986129', '1126384', '9333466', '841811', '1133161', '8086218', '9471011', '323246', '9446433', '1835138', '10341510', '9446850', '1903958', '3566906', '15798101', '10312331', '6034732', '9337745', '1785732', '918192', '1674824', '2424043', '9487331', '934913', '18243940', '15719620', '1663843', '12263060', '6626827', '1129462', '17903195', '1137847', '12263813', '2014700', '2475101', '923144', '435294', '6391378', '7443496', '13417463', '17292048', '9832854', '1850428', '9857982', '1147613', '5980754', '6772861', '5583391', '1069948', '883148', '2132977', '17249611', '2615836', '12605708', '1569002', '1439608', '5570430', '855447', '9798739', '8068415', '16769679', '1017899', '12383748', '1205726', '1136352', '858256', '348386', '1031902', '5568231', '54705', '2416154', '1994084', '5548673', '2979297', '955230', '826877', '581975', '12427932', '1085765', '1319520', '12781959', '12262975', '878894', '1125105', '9883622', '1120303', '17248683', '1056531', '10457221', '16733502', '826167', '943537', '17926256', '17290929', '930939', '5586171', '10465868', '938663', '5568337', '12487368', '844316', '6556516', '9424247', '5722204', '423151', '1333629', '2599406', '18147781', '1755772', '15783532', '1009347', '1851698', '16769658', '1287471', '9487287', '12697382', '1102982', '1032294', '8293280', '827954', '5567504', '1393176', '15925104', '1055950', '831710', '8199410', '10203959', '6961742', '859713', '1089691', '5682633', '5554469', '6904256', '1215866', '5982202', '8292934', '1123584', '970195', '9337979', '1400663', '1030539', '354984', '15928037', '940737', '7148658', '860881', '16830379', '15627944', '10356648', '5721077', '6979626', '14027916', '1974147', '9527409', '9553291', '7025033', '6978626', '3401009', '5585335', '5564702', '917988', '5126541', '5590182', '12158128', '5568470', '6558518', '5683617', '1059781', '977994', '6391340', '964065', '6877021', '54205', '17903398', '12604697', '1991251', '424426', '2456277', '862588', '8168577', '1016899', '17903886', '1110458', '8357405', '1600405', '928459', '1337272', '7409915', '406740', '5569809', '1266110', '1875368', '137461', '932730', '297803', '18107424', '10456522', '1105929', '5568748', '1078375', '6534132', '861028', '8275078', '8204140', '1086741', '2339091', '469298', '5588714', '828969', '14110920', '6553211', '9858916', '9710595', '96069', '9888234', '1279127', '18185791', '876859', '6995067', '3864002', '895821', '1116938', '2020047', '7433026', '16734408', '1324794', '6603362', '12132131', '1046502', '8364862', '10202900', '1220599', '247085', '1067605', '13217063', '5848538', '12604684', '2081333', '56240', '6602256', '5569980', '1109162', '43421', '983544', '1239261', '1729312', '941152', '12949599', '10204842', '885675', '1512901', '12188519', '17248911', '15452221', '1277693', '949775', '925394', '8290987', '30656', '540131', '6619421', '5979932', '18104790', '5586195', '6979307', '938439', '12487459', '8249088', '7128086', '1100816', '15863983', '9676869', '450670', '323922', '1946295', '2566244', '9487802', '2411742', '5564922', '12132259', '1095005', '1128568', '16729349', '1376789', '106018', '9830335', '8155157', '15573581', '741468', '12424877', '8293370', '8090895', '6919620', '927915', '931113', '1480464', '982941', '923514', '5574437', '906320', '17214945', '3030455', '12384625', '418697', '8155120', '2679144', '986842', '9829003', '961488', '887856', '1072487', '9803431', '1051175', '17903823', '9883935', '870449', '900398', '12781303', '1022828', '8159391', '17249686', '12132758', '3195606', '7442784', '12731297', '10457517', '821011', '1916776', '43869', '9858717', '12757338', '6391066', '474327', '924638', '7189662', '18104973', '895449', '8203518', '1013245', '964857', '1082000', '996209', '1883860', '17902236', '12456286', '467130', '30317', '7413766', '954837', '393693', '1048510', '2506601', '1005222', '2068363', '915830', '2588940', '1096437', '824151', '747770', '9705232', '6533205', '12256274', '908399', '10311757', '415550', '913907', '12470244', '7441715', '2058885', '30012', '12386015', '972801', '1108923', '8144459', '1973703', '9884901', '2696637', '9884555', '989737', '9487370', '6554207', '12133486', '9270468', '12604868', '1256340', '1063357', '12604705', '1562948', '1764686', '8091230', '1125652', '975603', '8118542', '999716', '12604653', '1019583', '1020278', '847605', '7102714', '955123', '545936', '1277030', '91957', '8357744', '8205766', '6602411', '1024395', '5582821', '48425', '968384', '1216025', '998150', '957674', '1720472', '5566939', '950084', '2620851', '3689647', '1213264', '1079706', '10182720', '17903364', '1667707', '1591091', '15567459', '822934', '7443414', '1834330', '1951928', '6604259', '10287524', '2572578', '15830653', '1032929', '916361', '930720', '7460448', '6757524', '9803216', '167162', '16769685', '931781', '8020113', '974459', '7167075', '9677065', '15553073', '17337336', '1068192', '2575934', '2563603', '9878803', '6396404', '1833370', '666969', '1622526', '12171526', '2575073', '843176', '952614', '862735', '536482', '936320', '1120565', '2054329', '17900917', '12811505', '921983', '6018431', '1718811', '821557', '5996326', '838082', '1730646', '12188190', '94548', '1132907', '10250579', '930043', '122894', '1517832', '12171340', '727603', '2000699', '8118116', '5663873', '925748', '6639027', '625788', '12386137', '9527365', '652898', '847706', '18005961', '899494', '1748959', '9858943', '937262', '7442981', '1034421', '12383938', '1788784', '18022633', '86447', '851774', '310705', '40023', '964916', '9374013', '8181569', '828537', '8063190', '1178407', '910783', '12577245', '1044305', '1088284', '18185306', '875689', '9296956', '1849292', '1549991', '112609', '9552834', '1014906', '2446824', '2712657', '8090777', '251241', '13417343', '2944006', '753319', '1104686', '12949826', '1733764', '1383468', '6903760', '1996176', '8205258', '12132097', '6473203', '12263622', '12757246', '5574614', '9884838', '12946988', '10455459', '994976', '5805762', '1071374', '10312145', '1505572', '8121226', '842330', '10120669', '917269', '12604793', '57901', '1391919', '2902444', '1026598', '1071925', '28919', '6602570', '17901129', '1957435', '15686645', '7168924', '9420079', '13416328', '9553177', '9553180', '5995492', '956905', '867874', '29674', '82732', '12988034', '6602360', '10204723', '1512580', '10344956', '1222155', '6021951', '5997589', '6979665', '8354559', '189758', '12132039', '1034138', '771241', '9883995', '14077733', '12132704', '559631', '452211', '892879', '9883937', '96896', '9297210', '6039910', '5571749', '832650', '5590411', '3727037', '7119658', '17105628', '9487196', '1315855', '2060701', '10254994', '7167962', '1013726', '900327', '1668538', '6423981', '9655653', '1140005', '12352329', '13006949', '827576', '908306', '1055914', '1997835', '1992587', '17903133', '18119016', '9365504', '916884', '862935', '1073445', '6944758', '1035987', '15511727', '1009099', '5749406', '920585', '17249282', '8091195', '71127', '7441336', '1360872', '8158192', '917302', '13210463', '902969', '988015', '1029320', '12735644', '9400839', '1868629', '5672234', '945691', '5981218', '853437', '16036666', '848078', '1141884', '1334044', '15714223', '1055746', '1276257', '824666', '9337478', '1024574', '9527400', '2807368', '1023895', '5586783', '997747', '9803329', '12673439', '101785', '12648396', '8181241', '2888080', '12457146', '9399901', '3135986', '1101690', '16058563', '1848083', '8204248', '9858481', '15828522', '3777147', '839383', '2381049', '1293898', '13513341', '17901792', '5540724', '1113638', '882840', '1896112', '8203414', '8249037', '17899706', '958839', '15563361', '8205843', '1978074', '930037', '1076189', '3302339', '1138540', '17900900', '961608', '992200', '949454', '222584', '13034948', '17903416', '17901795', '5591473', '7409993', '1780270', '412400', '15927632', '12351964', '911843', '17966987', '997189', '1025057', '6553194', '10453941', '1408631', '9706116', '971074', '833469', '1192704', '1040013', '2409151', '948058', '129733', '10312140', '15566489', '2386089', '114413', '987738', '744099', '17249704', '8276615', '16769684', '12427685', '1130266', '1035245', '847628', '3387313', '8069238', '6408616', '7441969', '6557083', '10204133', '1285059', '1836452', '1037079', '106667', '7443316', '18272784', '906314', '17106402', '908159', '837804', '961619', '842614', '820914', '885061', '899372', '4590490', '6602749', '825601', '6533476', '5580451', '1665246', '15778107', '1301476', '656961', '9527318', '402443', '750961', '10457125', '777563', '1085202', '1090975', '1709542', '6978978', '973074', '288248', '89409', '2619205', '61239', '9837322', '5570861', '9338253', '1033376', '12384154', '1107074', '2060942', '973506', '1070481', '9294722', '471693', '6552870', '762924', '16734753', '10281900', '1566064', '1058906', '1122650', '9884572', '7172838', '12129539', '9267970', '988958', '1129013', '17249861', '37447', '1014503', '9194549', '7443318', '953381', '951223', '873658', '12582020', '1019767', '982052', '17903501', '15557802', '6554473', '451553', '8157097', '9423333', '2821523', '1116193', '1284741', '2391208', '1951532', '1850213', '5661978', '908861', '15573904', '35511', '8068764', '985027', '12306467', '9529351', '2058243', '1567286', '1325911', '541596', '1950457', '1379661', '992267', '2483909', '9879236', '5579408', '2081026', '1070167', '1141944', '16733557', '12428027', '898821', '961604', '1800550', '1133437', '12385770', '948672', '1032522', '1029907', '921236', '1264566', '7131260', '7166466', '3917282', '2240299', '956227', '1939869', '12812288', '8355433', '9880291', '9861750', '9270525', '411695', '17903074', '12604619', '2446923', '485675', '10254265', '40694', '9860257', '1803228', '9188904', '8066609', '6608907', '9553405', '5589494', '1115289', '1530121', '850105', '75237', '1119083', '7440884', '13006897', '5574807', '48081', '911293', '1359940', '12648391', '997177', '1198704', '866129', '12947925', '940608', '8090787', '827740', '1649044', '986113', '7168812', '8014646', '1006104', '703181', '1657963', '84804', '903229', '820320', '1585187', '6979648', '9194124', '18148533', '12386112', '8069290', '9520758', '41481', '9802917', '49650', '5565794', '1498276', '15660194', '12263706', '12352218', '14073108', '907384', '1129703', '1117203', '10144077', '1657113', '1022238', '12351887', '9885374', '5570423', '10182774', '863682', '6925015', '5995635', '1129098', '9836214', '8159026', '6472068', '54137', '117029', '3055461', '6423956', '3024141', '888645', '10356819', '1108346', '17284818', '831968', '930733', '965606', '901526', '2015060', '9859590', '3306167', '886625', '1357067', '866848', '932810', '1047651', '1061429', '9884379', '12692147', '6602539', '8293259', '2586103', '10144250', '9677122', '18104798', '2285160', '45033', '1037346', '5580725', '5682642', '17902060', '857259', '8064782', '1053340', '854587', '9296632', '991141', '1023605', '105910', '856247', '1767191', '878790', '18203189', '1683411', '7029671', '1028094', '603679', '1122359', '9864192', '2052679', '6632302', '847421', '1014605', '8068465', '834105', '871074', '1751070', '7189022', '9677409', '114354', '10201139', '123650', '2061413', '9420105', '2058889', '5590074', '983154', '843205', '9553034', '15971899', '6554573', '3965154', '2729379', '1551144', '8204063', '856486', '2003805', '17973398', '8158002', '5129665', '1540051', '960720', '935230', '747948', '1106613', '836425', '1525087', '8181505', '1361698', '10254202', '715889', '900196', '1060455', '9527244', '12604311', '7136033', '768334', '61748', '970135', '2878423', '567012', '1059375', '2579427', '1063639', '8065972', '1162461', '348834', '916525', '7127001', '9707021', '7441955', '6925351', '1068075', '12132517', '1086899', '6551635', '8204091', '6602739', '1301054', '12756932', '1763442', '2249597', '16809426', '5738311', '13986499', '1112526', '1955254', '5592457', '5574336', '90234', '1020336', '12331803', '1776777', '1846565', '12605409', '9297306', '7477626', '5566647', '851176', '5998721', '1801871', '12731625', '9885390', '18037793', '1770105', '1982709', '3441345', '12811135', '9552830', '2672631', '979053', '8067125', '6704161', '857596', '6919565', '8069130', '2612247', '951232', '8129170', '8063812', '2770330', '13416938', '1797108', '839544', '1281562', '6925037', '945389', '482597', '1000112', '5694590', '17968481', '13945250', '7410291', '18148540', '587583', '1595969', '12605547', '1026069', '404498', '961927', '1014577', '2457715', '669892', '12648275', '5567187', '968206', '8251333', '1012464', '1413191', '2491085', '880235', '17291582', '825093', '910929', '12810564', '1005437', '17902580', '1539347', '1103013', '15925531', '15927265', '6391285', '12456073', '6395862', '58253', '927445', '2880021', '930671', '6872577', '901983', '1044064', '1288712', '7133532', '1289447', '1115097', '1063544', '10286208', '1775263', '8068874', '1071779', '1008057', '2028517', '6569768', '5567425', '56136', '5705303', '93662', '8068743', '12731325', '2444055', '1104446', '10455922', '1057853', '869034', '8205641', '3933114', '2485573', '2929806', '948809', '6979162', '932135', '2022006', '888328', '12171156', '2942027', '954350', '948608', '1037819', '1057678', '2583257', '1116745', '9883803', '17248491', '854892', '899512', '3398428', '5586238', '1894285', '5557456', '1314533', '1085858', '5569186', '31126', '9392936', '803581', '13910639', '2855968', '5980957', '2009534', '102969', '8203474', '5577764', '890143', '10289609', '443999', '1786478', '12757326', '5589984', '5996509', '18203516', '12810652', '6601564', '6462002', '10454239', '1096506', '8091573', '15721326', '9681767', '10122012', '8181386', '1012381', '2009708', '7122995', '6570756', '896443', '3863754', '12430684', '2413077', '18104750', '5692418', '437011', '6877927', '5591644', '18005341', '901668', '8069134', '10144499', '80783', '1960583', '877768', '932428', '1561565', '9500809', '1078399', '2575900', '2777271', '9297719', '937375', '1779667', '17329684', '701381', '10285354', '12384287', '40349', '1051795', '12487309', '955920', '1052236', '6391543', '12991432', '1081524', '6949116', '1417232', '9526732', '1951635', '2543746', '1127452', '1291982', '1040365', '1845670', '923620', '6552386', '911493', '15927689', '6005896', '1499438', '8185733', '1776384', '240608', '1086874', '937252', '17899758', '1778473', '1068760', '9911491', '981427', '12301824', '12171114', '12605401', '6039700', '10206993', '1960089', '9420003', '14020992', '13417183', '9446152', '824489', '1797672', '8037621', '979554', '594966', '1149536', '1896751', '2017560', '17169762', '10456115', '969536', '1957271', '17971670', '1104363', '2014498', '13417240', '1479160', '9802535', '1894289', '12988286', '1517541', '9220491', '6565718', '962651', '47467', '1063506', '10455392', '8203466', '17215061', '917241', '45720', '9885274', '9707204', '993880', '853630', '7410202', '8091526', '1234759', '12301765', '873073', '12605970', '1684084', '1956421', '12188298', '34214', '1956534', '1155426', '9337170', '1036232', '6703937', '112478', '922391', '5564785', '851916', '5563857', '18183776', '926944', '94446', '921679', '1001934', '1353810', '982542', '915190', '544011', '30491', '2458513', '114477', '1090428', '1125949', '850808', '12949422', '9655218', '9671895', '60776', '9422375', '866354', '1134021', '9883970', '8288091', '3903722', '17178958', '8293533', '18119014', '1502922', '8184528', '1108456', '5568107', '17291347', '7441081', '768215', '9419176', '6703965', '17328384', '910828', '7410033', '923895', '15863854', '907876', '17112935', '18118989', '93166', '6462557', '2048073', '1000029', '13003462', '15715898', '12333684', '653278', '727525', '2952844', '1129265', '12326703', '7167178', '472059', '966485', '18024956', '7409681', '13512467', '5583663', '118120', '6772641', '1177920', '12648459', '12605915', '12487481', '8068669', '8176091', '10118331', '6442429', '830815', '823959', '18004970', '487910', '6981246', '121045', '456020', '8352928', '1039118', '8241970', '438084', '8094575', '981289', '1598645', '1095162', '6703680', '1360293', '12605134', '882158', '6986383', '1592745', '2623001', '2044657', '949609', '2599533', '16053231', '462294', '1753292', '907579', '12524838', '1034986', '1136103', '52614', '7167840', '101885', '5589120', '5980393', '12523837', '17926817', '18104762', '844298', '9396744', '17104265', '9221995', '1691755', '8205290', '10456398', '1825205', '1545149', '10457574', '10355485', '12384038', '8118596', '8158149', '5979298', '10460517', '1518472', '17903347', '6562291', '472364', '953298', '15741088', '17249638', '949168', '9446473', '2224682', '897321', '2649068', '848076', '12605114', '18131707', '1067241', '38975', '2258669', '982782', '7441201', '1807134', '199832', '10204087', '1097479', '1523921', '17903421', '12301213', '287724', '17249768', '12456615', '12810808', '7132931', '1299251', '1745109', '489412', '1295810', '499517', '53854', '1473620', '1522900', '9803196', '3841063', '9655393', '1562428', '1266438', '1029613', '248170', '898792', '118005', '287929', '1086660', '93289', '658224', '2849210', '9884933', '83617', '3831877', '9573826', '1018287', '1117088', '12692505', '17898826', '826154', '8359058', '13775327', '1998643', '2183033', '2579327', '12632775', '2620924', '5127021', '1133263', '1027525', '827139', '8154873', '894283', '291887', '55148', '9855342', '980123', '2225459', '946783', '1892369', '12264722', '1352965', '923151', '103842', '848297', '931061', '1078919', '855591', '17901131', '12778002', '1105705', '986403', '1430684', '5981410', '2473128', '1109859', '5564557', '7412656', '1061030', '10314856', '2386561', '12781653', '1338135', '12132552', '9297181', '16809562', '5568782', '987385', '9855072', '2988659', '1025147', '1024151', '645553', '821043', '6553761', '839353', '8012079', '9884851', '8126623', '1869603', '854616', '1288030', '6463980', '1078714', '8354537', '978975', '12731645', '1673891', '885988', '941139', '7167027', '1128711', '18107212', '1793488', '9879679', '9655230', '1053012', '1164689', '934186', '843818', '1051759', '9489215', '5708977', '892423', '7462511', '17290617', '1047814', '963123', '841366', '2598417', '867405', '1110325', '1004317', '1165468']
Values Unique to DataFrame 2: ['5993055', '5126087', '5993054', '5978659', '1093587', '926775', '5977100', '5978650', '5978657', '5978648', '5978649', '5978656', '316011', '5993051', '5126106', '5126088', '5126107']

In the above printed output, values Unique to DataFrame1 are product IDs not purchased throughout the year. Recommended to the store to avoid buying these products(23839 in numbers) from wholesale outlet for sale to customers. Values Unique to DataFrame2 are product_ids whose details are not found in product dataset(17 in number) downloaded from R library, but these are available if downloaded from python library.

In [97]:
```
prod[prod['manufacturer_id'] == '2']
```

Out[97]:

	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
0	25671	2	GROCERY	National	FRZN ICE	ICE - CRUSHED/CUBED	22 LB
1	26081	2	MISCELLANEOUS	National	NaN	NaN	NaN
13	27021	2	GROCERY	National	AIR CARE	AIR CARE - AEROSOLS	NaN
772	49930	2	PRODUCE	National	STONE FRUIT	PEACHES YELLOW FLESH	NaN
775	49969	2	PRODUCE	National	MELONS	CANTALOUPE WHOLE	NaN
...	...	...	...	...	...	...	...
85042	13944371	2	MISCELLANEOUS	National	NaN	NaN	NaN
88302	15777906	2	PRODUCE	National	ORGANICS FRUIT & VEGETABLES	ORGANIC ONIONS	40 LB
90464	16769403	2	MISCELLANEOUS	National	NaN	NaN	NaN
91166	17240256	2	PRODUCE	National	ORGANICS FRUIT & VEGETABLES	BLUEBERRIES	72 CT
92228	18131707	2	FROZEN GROCERY	National	NaN	ORGANIC PEARS	NaN

1410 rows × 7 columns

If I replace 2 by 1, I find nothing, as manufacturer_id 1 is not found in R library but available in python library, but I havd taken R library since we were given R library only.

Sorting product id wise sales in descending order of revenue generated by them

In [192…
```
sorted_prod_id_sales = prod_id_sales.sort_values(by='sales_value', ascending=False)
sorted_prod_id_sales.head(20)
```

Out[192]:

	product_id	sales_value	manufacturer_id	department	brand	product_category	product_type	package_size
40901	6534178	303116.02	69	FUEL	Private	COUPON/MISC ITEMS	GASOLINE-REG UNLEADED	NaN
40871	6533889	27467.61	69	MISCELLANEOUS	Private	COUPON/MISC ITEMS	GASOLINE-REG UNLEADED	NaN
3875	1029743	22729.71	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
40897	6534166	20477.54	69	MISCELLANEOUS	Private	COUPON/MISC ITEMS	GASOLINE-REG UNLEADED	NaN
40864	6533765	19451.66	69	FUEL	Private	FUEL	GASOLINE-REG UNLEADED	NaN
9789	1082185	17219.59	2	PRODUCE	National	TROPICAL FRUIT	BANANAS	40 LB
56021	916122	16120.01	4314	MEAT	National	CHICKEN	CHICKEN BREAST BONELESS	NaN
12038	1106523	15629.95	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
68003	995242	15602.59	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	NaN
37642	5569230	13410.46	1208	GROCERY	National	SOFT DRINKS	SOFT DRINKS 12/18&15PK CAN CAR	12 OZ
13956	1127831	11500.01	5937	PRODUCE	National	BERRIES	STRAWBERRIES	16 OZ
5891	1044078	11445.70	2845	MEAT	National	BEEF	LEAN	NaN
49148	844179	11270.46	2852	MEAT	National	BEEF	PRIMAL	NaN
14440	1133018	8436.41	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	NaN
52134	874972	7905.59	2808	MEAT	National	BEEF	SELECT BEEF	NaN
13882	1126899	7805.95	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
493	1005186	7362.37	2	SALAD BAR	National	SALAD BAR	SALAD BAR FRESH FRUIT	NaN
51264	866211	7301.12	2	PRODUCE	National	GRAPES	GRAPES WHITE	18 LB
37673	5569471	7129.66	1208	GROCERY	National	SOFT DRINKS	SOFT DRINKS 12/18&15PK CAN CAR	12 OZ
8681	1070820	6952.38	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA

Lets focus on the product_ids of Gasoline, Milk and Bananas, which have generated most revenues

As we have already found out a list of 1st, 2nd and 3rdmost preferred products of different households, lets have a look on "for which household the above three products are most preferred"

In [113…
```
gasoline1 = house_prod1[house_prod1['product_id'] == '6534178']
gasoline2 = house_prod1[house_prod1['product_id'] == '6533889']
milk1 = house_prod1[house_prod1['product_id'] == '1029743']
bananas = house_prod1[house_prod1['product_id'] == '1082185']

gas_1 = gasoline1['Household Id']
gas_2 = gasoline2['Household Id']
milk_1 = milk1['Household Id']
bananas_1 = bananas['Household Id']
```

In [110…
```
gasoline1
```

Out[110]:

	Household Id	product_id
2	100	6534178
7	1004	6534178
11	1008	6534178
13	101	6534178
14	1010	6534178
...	...	...
2436	97	6534178
2450	982	6534178
2456	988	6534178
2459	990	6534178
2467	998	6534178

222 rows × 2 columns

So for 222 households, 6534178 or GASOLINE-REG UNLEADED is the most prefered product(number of purchases)

In [106…

```
gasoline2
```

Out[106]:

	Household Id	product_id
27	1023	6533889
124	1112	6533889
459	1417	6533889
1026	1935	6533889
1685	289	6533889

Only 5 households prefered to buy the 2nd Gasoline(2nd most revenue), the most number of times

In [107…

```
milk1
```

Out[107]:

	Household Id	product_id
10	1007	1029743
60	1054	1029743
89	1080	1029743
98	1089	1029743
128	1116	1029743
...	...	...
2435	969	1029743
2451	983	1029743
2453	985	1029743
2457	989	1029743
2468	999	1029743

140 rows × 2 columns

140 households prefered to buy Milk

In [108…

```
bananas
```

Out[108]:

	Household Id	product_id
3	1000	1082185
5	1002	1082185
18	1014	1082185
21	1017	1082185
33	1029	1082185
...	...	...
2446	979	1082185
2449	981	1082185
2452	984	1082185
2458	99	1082185
2462	993	1082185

302 rows × 2 columns

302 households prefered to buy bananas

Now let us find the 2nd and 3rd most purchased products from these household.

In [116…

```
second_by_gasoline1 = pd.merge(gas_1, house_prod2, on='Household Id')
third_by_gasoline1 = pd.merge(gas_1, house_prod3, on='Household Id')

second_by_gasoline2 = pd.merge(gas_2, house_prod2, on='Household Id')
third_by_gasoline2 = pd.merge(gas_2, house_prod3, on='Household Id')

second_by_milk1 = pd.merge(milk_1, house_prod2, on='Household Id')
third_by_milk1 = pd.merge(milk_1, house_prod3, on='Household Id')

second_by_bananas = pd.merge(bananas_1, house_prod2, on='Household Id')
third_by_bananas = pd.merge(bananas_1, house_prod3, on='Household Id')

secondgas1 = pd.merge(second_by_gasoline1, prod, on='product_id')
thirdgas1 = pd.merge(third_by_gasoline1, prod, on='product_id')
secondgas2 = pd.merge(second_by_gasoline2, prod, on='product_id')
thirdgas2 = pd.merge(third_by_gasoline2, prod, on='product_id')
secondmilk1 = pd.merge(second_by_milk1, prod, on='product_id')
thirdmilk1 = pd.merge(third_by_milk1, prod, on='product_id')
secondbananas = pd.merge(second_by_bananas, prod, on='product_id')
thirdbananas = pd.merge(third_by_bananas, prod, on='product_id')
```

In [118…

```
secondgas1.describe()
```

Out[118]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	221	221	221	221	221	221	221	157
unique	221	121	51	10	2	60	83	45
top	100	1082185	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	19	112	145	112	60	53	37

So most households who prefered to buy Gasoline 1 the most, bought 1082185 (FLUID MILK WHITE ONLY), the second most

In [120…

```
thirdgas1.describe()
```

Out[120]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	218	218	218	218	218	218	218	168
unique	218	147	59	14	2	73	101	66
top	100	1082185	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	18	92	140	125	52	44	31

So most households who prefered to buy Gasoline 1 the most, bought 1082185 (FLUID MILK WHITE ONLY), the third most

In [121…

```
secondgas2.describe()
```

Out[121]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	5	5	5	5	5	5	5	5
unique	5	5	4	3	2	5	5	5
top	1023	9859182	69	GROCERY	National	FACIAL TISS/DNR NAPKIN	FACIAL TISSUE & PAPER HANDKE	80 CT
freq	1	1	2	3	3	1	1	1

The 5 households who got gasoline2 bought different products in their second list but bought most of these second prefered list from grocery department

In [122…
```
thirdgas2.describe()
```

Out[122]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	5	5	5	5	5	5	5	5
unique	5	5	3	1	2	5	5	5
top	1023	963835	69	GROCERY	Private	PAPER TOWELS	PAPER TOWELS & HOLDERS	70.5 SQ FT
freq	1	1	3	5	3	1	1	1

Even they selected their thirdmost purchased item from the Grocery Department

In [123…
```
secondmilk1.describe()
```

Out[123]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	140	140	140	140	140	140	140	105
unique	140	81	43	10	2	43	58	35
top	1007	1029743	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	16	69	91	71	32	30	21

In [124…
```
thirdmilk1.describe()
```

Out[124]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	140	140	140	140	140	140	140	110
unique	140	96	48	8	2	53	76	47
top	1007	1082185	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	20 OZ
freq	1	11	59	94	81	22	17	14

In [125…
```
secondbananas.describe()
```

Out[125]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	302	302	302	302	302	302	302	209
unique	302	166	83	9	2	67	115	60
top	1000	1082185	69	GROCERY	National	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	26	148	193	153	93	86	43

In [126…
```
thirdbananas.describe()
```

Out[126]:

	Household Id	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	301	301	301	301	301	301	301	222
unique	301	181	72	14	2	73	124	65
top	1000	1029743	69	GROCERY	Private	FLUID MILK PRODUCTS	FLUID MILK WHITE ONLY	1 GA
freq	1	20	163	211	163	84	80	44

People who bought bananas the most bought fliud milk white the second and third most

This is to note that there are multiple product ids under a particular product_type. So lets group by product type instead of product_id as we are unable to deduce the second most prefered product if the product type is same.

In [129…
```
prod.describe()
```

Out[129]:

	product_id	manufacturer_id	department	brand	product_category	product_type	package_size
count	92331	92331	92331	92331	91791	91803	61745
unique	92331	6471	32	2	303	2378	3705
top	25671	69	GROCERY	National	GREETING CARDS/WRAP/PARTY SPLY	CARDS EVERYDAY	16 OZ
freq	1	12676	39023	78516	2785	1005	4054

In [193…
```
prod_type_sales = prod_id_sales.dropna(subset=['product_category','product_type'], how='all')
prod_type_sales.isnull().sum()
```

Out[193]:
```
product_id              0
sales_value             0
manufacturer_id         0
department              0
brand                   0
product_category       58
product_type           63
package_size        21712
dtype: int64
```

In [194…
```
# Group transactions by product_type and calculate sales
prod_typ = tran.groupby('product_id').agg({
    "household_id": [lambda x: x.mode().iloc[0]],
    "sales_value": ['sum'],
    "quantity": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum']
}).reset_index()
prod_typ.columns = ['product_id','household_id(mode)','Tot Rev','Tot Qty','Tot Ret Disc','Tot Coup Disc','Tot Coup Match Disc']

prod_typ_sales = pd.merge(prod_typ,prod1,on='product_id')
prod_typ_sales
```

Out[194]:

	product_id	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc	manufacturer_id	department	brand	product_category	product_type	package_size
0	1000002	2412	422.43	93.0	55.55	0.0	0.0	4373	DELI	National	DELI MEATS	MEAT: BEEF BULK	NaN
1	1000050	1802	442.31	173.0	144.16	6.0	0.0	1046	GROCERY	National	COLD CEREAL	KIDS CEREAL	14.5 OZ
2	1000057	1470	10.87	3.0	0.20	0.0	0.0	1046	GROCERY	National	DRY MIX DESSERTS	TOPPING MIXES WHIP TOPPING	5.2 OZ
3	1000059	1542	16.47	3.0	0.00	0.0	0.0	4974	DRUG GM	National	SINUS AND ALLERGY	NASAL SPRAY AND DROPS	NaN
4	1000092	1376	6.42	2.0	0.00	2.0	0.0	2350	DRUG GM	National	HAIR CARE PRODUCTS	HAIR CONDITIONERS AND RINSES	6.8 OZ
...	...	...	...	...	...	...	...	...	...	...	...	...	...
68203	999973	2199	98.36	57.0	16.63	1.3	0.3	317	GROCERY	National	CHEESE	CREAM CHEESE	8 OZ
68204	999982	2369	45.46	44.0	11.00	0.0	0.0	1422	GROCERY	National	CONDIMENTS/SAUCES	SALAD MUSTARD	10.5 OZ
68205	999987	692	23.96	4.0	0.00	0.0	0.0	1010	GROCERY	National	DOMESTIC WINE	FIGHTING VARIETAL WINES	1.5 L
68206	999992	1260	42.61	27.0	0.32	0.0	0.0	69	DRUG GM	Private	KITCHEN GADGETS	GADGETS/TOOLS	NaN
68207	9e+05	1935	69.00	1.0	0.00	0.0	0.0	2393	COSMETICS	National	FRAGRANCES	DESIGNER FRAGRANCES	.8 OZ

68208 rows × 13 columns

In [168…
```
col_drop1 = ['manufacturer_id', 'brand', 'package_size']
prod_typ_sales = prod_typ_sales.drop(columns=col_drop1, axis=1)
prod_typ_sales
```

Out[168]:

	product_id	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc	department	product_category	product_type
0	1000002	2412	422.43	93.0	55.55	0.0	0.0	DELI	DELI MEATS	MEAT: BEEF BULK
1	1000050	1802	442.31	173.0	144.16	6.0	0.0	GROCERY	COLD CEREAL	KIDS CEREAL
2	1000057	1470	10.87	3.0	0.20	0.0	0.0	GROCERY	DRY MIX DESSERTS	TOPPING MIXES WHIP TOPPING
3	1000059	1542	16.47	3.0	0.00	0.0	0.0	DRUG GM	SINUS AND ALLERGY	NASAL SPRAY AND DROPS
4	1000092	1376	6.42	2.0	0.00	2.0	0.0	DRUG GM	HAIR CARE PRODUCTS	HAIR CONDITIONERS AND RINSES
...	...	...	...	...	...	...	...	...	...	...
68203	999973	2199	98.36	57.0	16.63	1.3	0.3	GROCERY	CHEESE	CREAM CHEESE
68204	999982	2369	45.46	44.0	11.00	0.0	0.0	GROCERY	CONDIMENTS/SAUCES	SALAD MUSTARD
68205	999987	692	23.96	4.0	0.00	0.0	0.0	GROCERY	DOMESTIC WINE	FIGHTING VARIETAL WINES
68206	999992	1260	42.61	27.0	0.32	0.0	0.0	DRUG GM	KITCHEN GADGETS	GADGETS/TOOLS
68207	9e+05	1935	69.00	1.0	0.00	0.0	0.0	COSMETICS	FRAGRANCES	DESIGNER FRAGRANCES

68208 rows × 10 columns

In [169…
```
prod_typ_sales['product_type'].nunique()
```

Out[169]:
```
2242
```

In [172…
```
prod_type_sales = prod_typ_sales.groupby('product_type').agg({
    "household_id(mode)": [lambda x: x.mode().iloc[0]],
    "Tot Rev": ['sum'],
    "Tot Qty": ['sum'],
    "Tot Ret Disc": ['sum'],
    "Tot Coup Disc": ['sum'],
    "Tot Coup Match Disc": ['sum'],
    "department": ['first'],
    "product_category": ['first']
}).reset_index()
```

```
prod_type_sales.columns = ['product_type','household_id(mode)','Tot Rev','Tot Qty','Tot Ret Desc','Tot Coup Disc','Tot Coup Match Disc','department','product_category']
prod_type_sales.head(20)
```

Out[172]:

	product_type	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	department	product_category
0	4-20 BOYS TEAM SPORT	1097	46.93	3.0	0.00	0.00	0.00	DRUG GM	APPAREL
1	ABRASIVES	1032	624.95	383.0	67.49	7.50	4.50	GROCERY	HOUSEHOLD CLEANG NEEDS
2	ACCENT FURNITURE	18	19.99	1.0	0.00	0.00	0.00	DRUG GM	HOME FURNISHINGS
3	ACCESSORIES	1139	46.88	12.0	0.00	0.00	0.00	DRUG GM	IRONING AND CHEMICALS
4	ACNE MEDICATIONS	1845	1454.81	280.0	127.08	12.00	0.00	DRUG GM	HAND/BODY/FACIAL PRODUCTS
5	ACTIVITY	1111	323.64	99.0	7.30	0.00	0.00	DRUG GM	TOYS AND GAMES
6	ADDITIVES/FLUIDS	1023	247.14	82.0	1.64	0.00	0.00	DRUG GM	AUTOMOTIVE PRODUCTS
7	ADHESIVES/CAULK	1103	278.56	113.0	3.01	0.00	0.00	DRUG GM	HARDWARE SUPPLIES
8	ADULT ANALGESICS	1103	10725.85	2153.0	1882.45	5.39	0.00	DRUG GM	ANALGESICS
9	ADULT CEREAL	157	15838.88	5518.0	1917.60	110.69	7.65	GROCERY	COLD CEREAL
10	ADULT INCONTINENCE BRIEFS	1000	385.31	33.0	26.48	4.00	0.00	DRUG GM	ADULT INCONTINENCE
11	ADULT INCONTINENCE GUARDS	909	9.99	1.0	1.00	0.00	0.00	DRUG GM	ADULT INCONTINENCE
12	ADULT INCONTINENCE MISC PRODUC	1000	410.20	106.0	38.66	0.00	0.00	DRUG GM	ADULT INCONTINENCE
13	ADULT INCONTINENCE PADS	1726	439.08	67.0	15.34	10.50	0.00	DRUG GM	ADULT INCONTINENCE
14	ADULT INCONTINENCE UNDERGARMEN	1098	1936.87	182.0	112.84	11.90	0.40	DRUG GM	ADULT INCONTINENCE
15	ADULT PREMIUM	1864	2014.00	592.0	405.08	25.14	5.50	GROCERY	FRZN NOVELTIES/WTR ICE
16	AEROSOL DEODORANTS	119	993.25	279.0	92.80	7.50	1.00	DRUG GM	DEODORANTS
17	AEROSOL TOPPINGS	1043	1809.52	806.0	330.76	8.95	8.35	GROCERY	MILK BY-PRODUCTS
18	AGE RESTRICTED DVD S	1001	619.40	62.0	23.98	0.00	0.00	DRUG GM	AUDIO/VIDEO PRODUCTS
19	AGE RESTRICTED FIREWORKS	2312	711.09	108.0	0.00	0.00	0.00	DRUG GM	FIREWORKS

In [173…

```
sorted_prod_type_sales = prod_type_sales.sort_values(by='Tot Rev', ascending=False)
sorted_prod_type_sales.head(20)
```

Out[173]:

	product_type	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	department	product_category
849	GASOLINE-REG UNLEADED	1227	384044.07	151020580.0	12201.91	0.00	0.00	MISCELLANEOUS	COUPON/MISC ITEMS
737	FLUID MILK WHITE ONLY	1227	91834.49	52443.0	22638.47	83.54	0.00	GROCERY	FLUID MILK PRODUCTS
1925	SOFT DRINKS 12/18&15PK CAN CAR	1227	89656.71	31759.0	42191.83	2029.54	6.30	GROCERY	SOFT DRINKS
189	BEERALEMALT LIQUORS	27	82691.86	11186.0	69.00	0.00	0.00	GROCERY	BEERS/ALES
425	CIGARETTES	27	54349.68	9327.0	2.29	938.84	10.75	DRUG GM	CIGARETTES
417	CHOICE BEEF	2364	45684.40	7921.0	15072.96	0.00	0.00	MEAT	BEEF
1873	SHREDDED CHEESE	1420	38213.73	20256.0	10480.50	66.32	16.85	GROCERY	CHEESE
1637	PRIMAL	2373	36653.42	9727.0	5349.61	0.00	0.00	MEAT	BEEF
1599	PREMIUM	1032	36017.61	11494.0	17459.99	236.74	20.65	MEAT-PCKGD	BACON
115	BABY DIAPERS	1227	33840.33	2969.0	2210.50	531.23	1.75	DRUG GM	DIAPERS & DISPOSABLES
2081	TOILET TISSUE	1032	33618.23	9235.0	7384.04	351.20	59.05	GROCERY	BATH TISSUES
657	ENHANCED	2364	31782.37	6038.0	16069.45	0.00	0.00	MEAT	PORK
1578	POTATO CHIPS	2364	29907.96	13531.0	3592.82	115.11	22.20	GROCERY	BAG SNACKS
1853	SFT DRNK 2 LITER BTL CARB INCL	2144	29293.79	29753.0	9536.30	85.26	16.80	GROCERY	SOFT DRINKS
1910	SNACKS/APPETIZERS	1420	26997.55	9726.0	5327.77	138.93	72.50	GROCERY	FROZEN PIZZA
541	DAIRY CASE 100% PURE JUICE - O	1987	26758.95	11038.0	7754.92	89.86	17.00	GROCERY	REFRGRATD JUICES/DRNKS
831	FRZN SS PREMIUM ENTREES/DNRS/N	1420	26644.09	12479.0	10947.27	232.20	96.80	GROCERY	FRZN MEAT/MEAT DINNERS
1122	LIQUID LAUNDRY DETERGENTS	1023	26558.68	4641.0	5790.68	204.00	74.60	GROCERY	LAUNDRY DETERGENTS
1072	KIDS CEREAL	2373	25912.96	10348.0	7002.26	148.94	14.20	GROCERY	COLD CEREAL
1147	MAINSTREAM WHITE BREAD	1227	24729.61	19273.0	8693.25	40.46	15.80	GROCERY	BAKED BREAD/BUNS/ROLLS

In [187…

```
prod_typ_sales['product_type'].nunique()
```

Out[187]:

```
2242
```

In [188…

```
prod_typ_sales['product_category'].nunique()
```

Out[188]:

```
302
```

In [189…

```
prod_typ_sales['department'].nunique()
```

Out[189]: 30

In [178…
```
# Calculate the total amount
tot_amt = sorted_prod_type_sales['Tot Rev'].sum()

# Create a new DataFrame with the percentage column
rev_perc = sorted_prod_type_sales.copy()
rev_perc['% of Rev'] = (rev_perc['Tot Rev']/tot_amt)*100

rev_perc.head(20)
```

Out[178]:

	product_type	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	department	product_category	% of Rev
849	GASOLINE-REG UNLEADED	1227	384044.07	151020580.0	12201.91	0.00	0.00	MISCELLANEOUS	COUPON/MISC ITEMS	8.366049
737	FLUID MILK WHITE ONLY	1227	91834.49	52443.0	22638.47	83.54	0.00	GROCERY	FLUID MILK PRODUCTS	2.000530
1925	SOFT DRINKS 12/18&15PK CAN CAR	1227	89656.71	31759.0	42191.83	2029.54	6.30	GROCERY	SOFT DRINKS	1.953089
189	BEERALEMALT LIQUORS	27	82691.86	11186.0	69.00	0.00	0.00	GROCERY	BEERS/ALES	1.801367
425	CIGARETTES	27	54349.68	9327.0	2.29	938.84	10.75	DRUG GM	CIGARETTES	1.183958
417	CHOICE BEEF	2364	45684.40	7921.0	15072.96	0.00	0.00	MEAT	BEEF	0.995193
1873	SHREDDED CHEESE	1420	38213.73	20256.0	10480.50	66.32	16.85	GROCERY	CHEESE	0.832451
1637	PRIMAL	2373	36653.42	9727.0	5349.61	0.00	0.00	MEAT	BEEF	0.798461
1599	PREMIUM	1032	36017.61	11494.0	17459.99	236.74	20.65	MEAT-PCKGD	BACON	0.784611
115	BABY DIAPERS	1227	33840.33	2969.0	2210.50	531.23	1.75	DRUG GM	DIAPERS & DISPOSABLES	0.737181
2081	TOILET TISSUE	1032	33618.23	9235.0	7384.04	351.20	59.05	GROCERY	BATH TISSUES	0.732342
657	ENHANCED	2364	31782.37	6038.0	16069.45	0.00	0.00	MEAT	PORK	0.692350
1578	POTATO CHIPS	2364	29907.96	13531.0	3592.82	115.11	22.20	GROCERY	BAG SNACKS	0.651518
1853	SFT DRNK 2 LITER BTL CARB INCL	2144	29293.79	29753.0	9536.30	85.26	16.80	GROCERY	SOFT DRINKS	0.638138
1910	SNACKS/APPETIZERS	1420	26997.55	9726.0	5327.77	138.93	72.50	GROCERY	FROZEN PIZZA	0.588117
541	DAIRY CASE 100% PURE JUICE - O	1987	26758.95	11038.0	7754.92	89.86	17.00	GROCERY	REFRGRATD JUICES/DRNKS	0.582919
831	FRZN SS PREMIUM ENTREES/DNRS/N	1420	26644.09	12479.0	10947.27	232.20	96.80	GROCERY	FRZN MEAT/MEAT DINNERS	0.580417
1122	LIQUID LAUNDRY DETERGENTS	1023	26558.68	4641.0	5790.68	204.00	74.60	GROCERY	LAUNDRY DETERGENTS	0.578557
1072	KIDS CEREAL	2373	25912.96	10348.0	7002.26	148.94	14.20	GROCERY	COLD CEREAL	0.564490
1147	MAINSTREAM WHITE BREAD	1227	24729.61	19273.0	8693.25	40.46	15.80	GROCERY	BAKED BREAD/BUNS/ROLLS	0.538712

In [186…
```
sum_perc20 = rev_perc['% of Rev'].iloc[:20].sum()
rounded = round(sum_perc20, 2)
print(f"The top 20 product types have generated {rounded}% of the total revenue")
```

```
The top 20 product types have generated 25.6% of the total revenue
```

But there are 2242 categories in the list, so let's now filter category wise as they are 302 in number and later department wise which are 30 in number

In [199…
```
prod_cat_sales = prod_typ_sales.groupby('product_category').agg({
    "household_id(mode)": [lambda x: x.mode().iloc[0]],
    "Tot Rev": ['sum'],
    "Tot Qty": ['sum'],
    "Tot Ret Disc": ['sum'],
    "Tot Coup Disc": ['sum'],
    "Tot Coup Match Disc": ['sum'],
    "department": ['first']
}).reset_index()
prod_cat_sales.columns = ['product_category','household_id(mode)','Tot Rev','Tot Qty','Tot Ret Desc','Tot Coup Disc','Tot Coup Match Disc','department']
prod_cat_sales.head(20)
```

Out[199]:

	product_category	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	department
0	ADULT INCONTINENCE	1726	3220.61	398.0	195.07	28.90	0.90	DRUG GM
1	AIR CARE	1430	10291.26	3961.0	1197.05	564.33	13.44	GROCERY
2	ANALGESICS	1103	14899.53	3228.0	2627.12	6.94	0.45	DRUG GM
3	ANTACIDS	1489	6731.27	1293.0	943.03	7.80	0.80	DRUG GM
4	APPAREL	1707	2101.32	315.0	366.67	1.00	0.00	DRUG GM
5	APPLES	1823	22297.33	10278.0	2287.92	0.00	0.00	PRODUCE
6	AUDIO/VIDEO PRODUCTS	1023	7477.77	581.0	343.37	6.00	0.00	DRUG GM
7	AUTOMOTIVE PRODUCTS	1510	1757.69	577.0	20.56	1.00	0.00	DRUG GM
8	BABY FOODS	1227	13621.32	17136.0	545.21	182.18	45.00	DRUG GM
9	BABY HBC	1710	8897.77	2884.0	584.19	152.68	7.15	DRUG GM
10	BABYFOOD	1676	363.95	451.0	15.69	4.75	1.25	NUTRITION
11	BACON	1420	25213.36	8925.0	9565.21	35.25	8.90	MEAT-PCKGD
12	BAG SNACKS	2364	84943.85	45751.0	11760.71	331.00	52.10	GROCERY
13	BAKED BREAD/BUNS/ROLLS	2364	82429.70	56709.0	23205.06	263.86	94.86	GROCERY
14	BAKED SWEET GOODS	2337	31689.84	18112.0	4918.86	36.43	21.82	GROCERY
15	BAKERY PARTY TRAYS	1782	613.87	97.0	2.50	0.00	0.00	PASTRY
16	BAKING	973	986.39	311.0	57.30	0.00	0.00	NUTRITION
17	BAKING MIXES	1675	17506.41	13541.0	4013.43	165.31	88.67	GROCERY
18	BAKING NEEDS	1227	16815.91	9696.0	2087.86	38.72	20.40	GROCERY
19	BATH	1740	2559.18	808.0	235.93	4.50	1.50	COSMETICS

In [200…

```
sorted_prod_cat_sales = prod_cat_sales.sort_values(by='Tot Rev', ascending=False)
sorted_prod_cat_sales.head(20)
```

Out[200]:

	product_category	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	department
76	COUPON/MISC ITEMS	1227	385971.85	150922290.0	12202.81	0.00	0.00	MISCELLANEOUS
268	SOFT DRINKS	1227	182126.30	89496.0	60931.79	2270.08	32.28	GROCERY
23	BEEF	2364	176614.54	37343.0	39135.14	0.00	0.00	MEAT
122	FLUID MILK PRODUCTS	1814	116360.68	66440.0	27903.98	225.66	88.60	GROCERY
50	CHEESE	2364	107011.61	55019.0	26644.58	276.79	106.34	GROCERY
137	FRZN MEAT/MEAT DINNERS	1420	93023.74	45835.0	26495.40	463.87	168.00	GROCERY
12	BAG SNACKS	2364	84943.85	45751.0	11760.71	331.00	52.10	GROCERY
13	BAKED BREAD/BUNS/ROLLS	2364	82429.70	56709.0	23205.06	263.86	94.86	GROCERY
24	BEERS/ALES	27	82039.45	10983.0	70.90	0.00	0.00	GROCERY
131	FROZEN PIZZA	1420	81175.04	37404.0	16828.57	491.21	172.19	GROCERY
65	COLD CEREAL	1420	63009.35	23407.0	14380.85	497.15	54.40	GROCERY
79	DELI MEATS	2364	60159.68	14681.0	5137.01	3.65	1.35	DELI
56	CIGARETTES	27	54349.68	9327.0	2.29	938.84	10.75	DRUG GM
52	CHICKEN	1053	52703.51	14844.0	34375.62	2.00	0.00	MEAT
182	LUNCHMEAT	2364	51858.04	25795.0	12867.91	99.30	39.70	MEAT-PCKGD
227	PORK	1228	50809.31	10131.0	22066.76	3.65	1.15	MEAT
269	SOUP	2364	49027.16	48605.0	6403.90	279.67	185.45	GROCERY
164	ICE CREAM/MILK/SHERBTS	1032	46876.51	17790.0	22649.84	241.97	16.15	GROCERY
42	CANDY - PACKAGED	2364	43680.94	25348.0	6841.32	117.60	20.25	DRUG GM
43	CANNED JUICES	2120	41924.84	19703.0	5412.56	121.11	24.30	GROCERY

In [201…

```
# Calculate the total amount
tot_amt_cat = sorted_prod_cat_sales['Tot Rev'].sum()

# Create a new DataFrame with the percentage column
rev_perc_cat = sorted_prod_cat_sales.copy()
rev_perc_cat['% of Rev'] = (rev_perc_cat['Tot Rev']/tot_amt_cat)*100

rev_perc_cat.head(20)
```

Out[201]:

	product_category	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	department	% of Rev
76	COUPON/MISC ITEMS	1227	385971.85	150922290.0	12202.81	0.00	0.00	MISCELLANEOUS	8.411936
268	SOFT DRINKS	1227	182126.30	89496.0	60931.79	2270.08	32.28	GROCERY	3.969291
23	BEEF	2364	176614.54	37343.0	39135.14	0.00	0.00	MEAT	3.849167
122	FLUID MILK PRODUCTS	1814	116360.68	66440.0	27903.98	225.66	88.60	GROCERY	2.535984
50	CHEESE	2364	107011.61	55019.0	26644.58	276.79	106.34	GROCERY	2.332229
137	FRZN MEAT/MEAT DINNERS	1420	93023.74	45835.0	26495.40	463.87	168.00	GROCERY	2.027375
12	BAG SNACKS	2364	84943.85	45751.0	11760.71	331.00	52.10	GROCERY	1.851281
13	BAKED BREAD/BUNS/ROLLS	2364	82429.70	56709.0	23205.06	263.86	94.86	GROCERY	1.796487
24	BEERS/ALES	27	82039.45	10983.0	70.90	0.00	0.00	GROCERY	1.787982
131	FROZEN PIZZA	1420	81175.04	37404.0	16828.57	491.21	172.19	GROCERY	1.769143
65	COLD CEREAL	1420	63009.35	23407.0	14380.85	497.15	54.40	GROCERY	1.373236
79	DELI MEATS	2364	60159.68	14681.0	5137.01	3.65	1.35	DELI	1.311130
56	CIGARETTES	27	54349.68	9327.0	2.29	938.84	10.75	DRUG GM	1.184506
52	CHICKEN	1053	52703.51	14844.0	34375.62	2.00	0.00	MEAT	1.148629
182	LUNCHMEAT	2364	51858.04	25795.0	12867.91	99.30	39.70	MEAT-PCKGD	1.130203
227	PORK	1228	50809.31	10131.0	22066.76	3.65	1.15	MEAT	1.107347
269	SOUP	2364	49027.16	48605.0	6403.90	279.67	185.45	GROCERY	1.068506
164	ICE CREAM/MILK/SHERBTS	1032	46876.51	17790.0	22649.84	241.97	16.15	GROCERY	1.021635
42	CANDY - PACKAGED	2364	43680.94	25348.0	6841.32	117.60	20.25	DRUG GM	0.951990
43	CANNED JUICES	2120	41924.84	19703.0	5412.56	121.11	24.30	GROCERY	0.913717

In [202…
```
sum_perc20_cat = rev_perc_cat['% of Rev'].iloc[:20].sum()
rounded_cat = round(sum_perc20_cat, 2)
print(f"The top 20 among 302 product categories have generated {rounded_cat}% of the total revenue")
```

```
The top 20 among 302 product categories have generated 41.54% of the total revenue
```

In [203…
```
sum_perc40_cat = rev_perc_cat['% of Rev'].iloc[:40].sum()
rounded_cat40 = round(sum_perc40_cat, 2)
print(f"The top 40 among 302 product categories have generated {rounded_cat40}% of the total revenue")
```

```
The top 40 among 302 product categories have generated 56.06% of the total revenue
```

In [204…
```
sum_perc60_cat = rev_perc_cat['% of Rev'].iloc[:60].sum()
rounded_cat60 = round(sum_perc60_cat, 2)
print(f"The top 60 among 302 product categories have generated {rounded_cat60}% of the total revenue")
```

```
The top 60 among 302 product categories have generated 66.9% of the total revenue
```

In [205…
```
sum_perc80_cat = rev_perc_cat['% of Rev'].iloc[:80].sum()
rounded_cat80 = round(sum_perc80_cat, 2)
print(f"The top 80 among 302 product categories have generated {rounded_cat80}% of the total revenue")
```

```
The top 80 among 302 product categories have generated 75.26% of the total revenue
```

In [206…
```
sum_perc100_cat = rev_perc_cat['% of Rev'].iloc[:100].sum()
rounded_cat100 = round(sum_perc100_cat, 2)
print(f"The top 100 among 302 product categories have generated {rounded_cat100}% of the total revenue")
```

```
The top 100 among 302 product categories have generated 81.6% of the total revenue
```

Lets sort by Department wise

In [207…
```
dept_sales = prod_typ_sales.groupby('department').agg({
    "household_id(mode)": [lambda x: x.mode().iloc[0]],
    "Tot Rev": ['sum'],
    "Tot Qty": ['sum'],
    "Tot Ret Disc": ['sum'],
    "Tot Coup Disc": ['sum'],
    "Tot Coup Match Disc": ['sum']
}).reset_index()
dept_sales.columns = ['product_category','household_id(mode)','Tot Rev','Tot Qty','Tot Ret Desc','Tot Coup Disc','Tot Coup Match Disc']
dept_sales.head(20)
```

Out[207]:

	product_category	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc
0	AUTOMOTIVE	1379	301.96	40.0	0.00	0.00	0.00
1	CHEF SHOPPE	1072	1701.98	681.0	63.08	0.00	0.00
2	CNTRL/STORE SUP	125	44.95	17.0	0.00	0.00	0.00
3	COSMETICS	2085	18987.44	4640.0	4422.86	125.54	3.65
4	COUPON	1795	652.51	529.0	139.35	16.62	3.75
5	DELI	1023	148344.06	37954.0	13761.45	102.93	16.35
6	DRUG GM	1023	596827.45	198635.0	58778.14	4159.05	315.47
7	ELECT &PLUMBING	1613	1.00	1.0	0.00	0.00	0.00
8	FLORAL	1023	22303.18	3160.0	675.28	62.93	0.00
9	FROZEN GROCERY	104	419.73	154.0	22.65	0.00	0.00
10	FUEL	1192	329594.45	129662940.0	10313.11	0.00	0.00
11	GARDEN CENTER	1023	5351.79	939.0	731.33	0.00	0.00
12	GM MERCH EXP	19	67.02	40.0	0.00	0.00	0.00
13	GROCERY	2364	2316393.89	1242944.0	453642.78	14051.45	3866.59
14	MEAT	2364	308575.33	67526.0	102859.16	54.10	6.70
15	MEAT-PCKGD	2364	232282.53	83423.0	69009.01	701.77	209.60
16	MISCELLANEOUS	1023	73999.35	21359814.0	1897.53	0.00	0.00
17	NUTRITION	973	57261.22	25024.0	7229.68	110.82	30.39
18	PASTRY	1023	69116.68	28132.0	9633.67	31.12	1.35
19	PHOTO & VIDEO	1024	51.70	19.0	8.66	1.75	0.75

In [208…

```
sorted_dept_sales = dept_sales.sort_values(by='Tot Rev', ascending=False)
sorted_dept_sales.head(20)
```

Out[208]:

	product_category	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc
13	GROCERY	2364	2316393.89	1242944.0	453642.78	14051.45	3866.59
6	DRUG GM	1023	596827.45	198635.0	58778.14	4159.05	315.47
10	FUEL	1192	329594.45	129662940.0	10313.11	0.00	0.00
22	PRODUCE	1227	322858.82	185444.0	41984.71	274.54	76.80
14	MEAT	2364	308575.33	67526.0	102859.16	54.10	6.70
15	MEAT-PCKGD	2364	232282.53	83423.0	69009.01	701.77	209.60
5	DELI	1023	148344.06	37954.0	13761.45	102.93	16.35
16	MISCELLANEOUS	1023	73999.35	21359814.0	1897.53	0.00	0.00
18	PASTRY	1023	69116.68	28132.0	9633.67	31.12	1.35
17	NUTRITION	973	57261.22	25024.0	7229.68	110.82	30.39
26	SEAFOOD-PCKGD	1675	35977.46	8028.0	12074.49	136.22	11.00
8	FLORAL	1023	22303.18	3160.0	675.28	62.93	0.00
3	COSMETICS	2085	18987.44	4640.0	4422.86	125.54	3.65
24	SALAD BAR	1057	17812.40	6185.0	675.34	0.00	0.00
25	SEAFOOD	1228	17081.36	2589.0	3371.92	0.00	0.00
27	SPIRITS	1172	12299.52	1276.0	0.00	0.00	0.00
11	GARDEN CENTER	1023	5351.79	939.0	731.33	0.00	0.00
1	CHEF SHOPPE	1072	1701.98	681.0	63.08	0.00	0.00
29	TRAVEL & LEISURE	1987	1425.68	506.0	154.16	0.00	0.00
23	RESTAURANT	1907	1260.97	347.0	52.75	0.00	0.00

In [235…

```
# Calculate the total amount
tot_amt_dept = sorted_dept_sales['Tot Rev'].sum()

# Create a new DataFrame with the percentage column
rev_perc_dept = sorted_dept_sales.copy()
rev_perc_dept['% of Rev'] = (rev_perc_dept['Tot Rev']/tot_amt_dept)*100

dep = rev_perc_dept.reset_index()
dep
```

Out[235]:

	index	product_category	household_id(mode)	Tot Rev	Tot Qty	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	% of Rev
0	13	GROCERY	2364	2316393.89	1242944.0	453642.78	14051.45	3866.59	50.455021
1	6	DRUG GM	1023	596827.45	198635.0	58778.14	4159.05	315.47	12.999923
2	10	FUEL	1192	329594.45	129662940.0	10313.11	0.00	0.00	7.179131
3	22	PRODUCE	1227	322858.82	185444.0	41984.71	274.54	76.80	7.032417
4	14	MEAT	2364	308575.33	67526.0	102859.16	54.10	6.70	6.721298
5	15	MEAT-PCKGD	2364	232282.53	83423.0	69009.01	701.77	209.60	5.059511
6	5	DELI	1023	148344.06	37954.0	13761.45	102.93	16.35	3.231187
7	16	MISCELLANEOUS	1023	73999.35	21359814.0	1897.53	0.00	0.00	1.611832
8	18	PASTRY	1023	69116.68	28132.0	9633.67	31.12	1.35	1.505480
9	17	NUTRITION	973	57261.22	25024.0	7229.68	110.82	30.39	1.247247
10	26	SEAFOOD-PCKGD	1675	35977.46	8028.0	12074.49	136.22	11.00	0.783651
11	8	FLORAL	1023	22303.18	3160.0	675.28	62.93	0.00	0.485801
12	3	COSMETICS	2085	18987.44	4640.0	4422.86	125.54	3.65	0.413579
13	24	SALAD BAR	1057	17812.40	6185.0	675.34	0.00	0.00	0.387985
14	25	SEAFOOD	1228	17081.36	2589.0	3371.92	0.00	0.00	0.372061
15	27	SPIRITS	1172	12299.52	1276.0	0.00	0.00	0.00	0.267905
16	11	GARDEN CENTER	1023	5351.79	939.0	731.33	0.00	0.00	0.116571
17	1	CHEF SHOPPE	1072	1701.98	681.0	63.08	0.00	0.00	0.037072
18	29	TRAVEL & LEISURE	1987	1425.68	506.0	154.16	0.00	0.00	0.031054
19	23	RESTAURANT	1907	1260.97	347.0	52.75	0.00	0.00	0.027466
20	4	COUPON	1795	652.51	529.0	139.35	16.62	3.75	0.014213
21	9	FROZEN GROCERY	104	419.73	154.0	22.65	0.00	0.00	0.009142
22	0	AUTOMOTIVE	1379	301.96	40.0	0.00	0.00	0.00	0.006577
23	12	GM MERCH EXP	19	67.02	40.0	0.00	0.00	0.00	0.001460
24	19	PHOTO & VIDEO	1024	51.70	19.0	8.66	1.75	0.75	0.001126
25	2	CNTRL/STORE SUP	125	44.95	17.0	0.00	0.00	0.00	0.000979
26	20	POSTAL CENTER	2142	6.57	6.0	0.39	0.00	0.00	0.000143
27	28	TOYS	1763	4.17	3.0	3.20	0.00	0.00	0.000091
28	21	PROD-WHS SALES	1247	2.52	3.0	0.00	0.00	0.00	0.000055
29	7	ELECT &PLUMBING	1613	1.00	1.0	0.00	0.00	0.00	0.000022

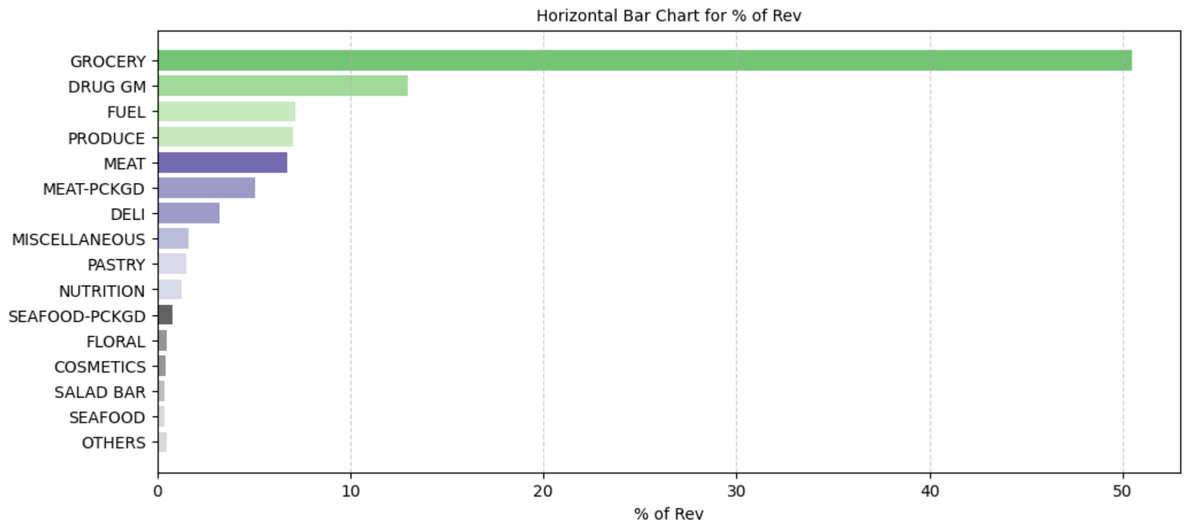
In [236…

```
sum_perc10_dep = dep['% of Rev'].iloc[:10].sum()
rounded_dep10 = round(sum_perc10_dep, 2)
print(f"The top 10 among 30 departments have generated {rounded_dep10}% of the total revenue")
```

```
The top 10 among 30 departments have generated 97.04% of the total revenue
```

In [250…

```
from matplotlib.cm import get_cmap
import warnings
warnings.filterwarnings('ignore')

column_to_plot = '% of Rev'

# Combine the last 15 rows into one row called 'Others'
last_15_sum = dep.iloc[-15:][column_to_plot].sum()
others_row = pd.DataFrame({'product_category': ['OTHERS'], column_to_plot: [last_15_sum]})

# Create a new DataFrame by concatenating the 'Others' row with the original DataFrame
dep_mod = pd.concat([dep.iloc[:-15], others_row])

dep_mod = dep_mod[::-1]

# Create a horizontal bar chart
plt.figure(figsize=(11.5, 5))
cmap = get_cmap('tab20c', len(dep))
cmap = cmap.reversed()
plt.barh(dep_mod['product_category'], dep_mod[column_to_plot], color=cmap(range(len(dep))))
plt.xlabel(column_to_plot)
plt.title(f'Horizontal Bar Chart for {column_to_plot}', size=10)

plt.grid(axis='x', linestyle='--', alpha=0.6)

# Show the chart
plt.show()
```



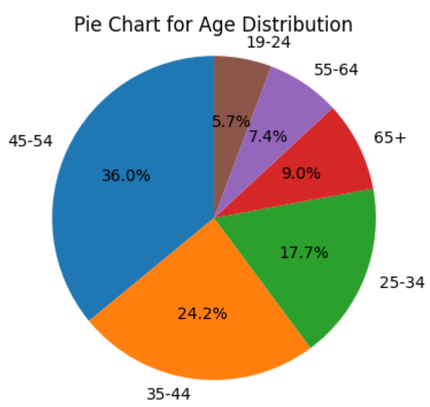
Now let us focus on which category of households generate more revenue. So let us study the demographics dataset

In [251…
```
demo.describe()
```

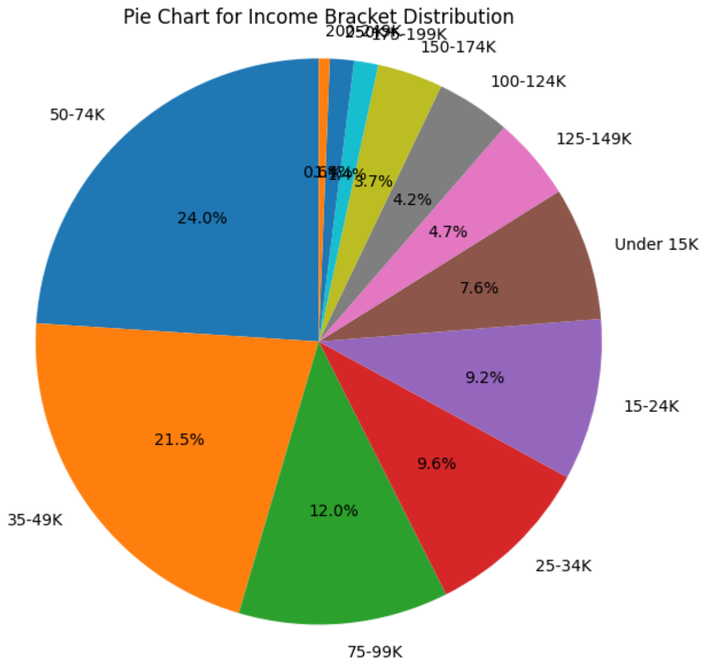
Out[251]:

	household_id	age	income	home_ownership	marital_status	household_size	household_comp	kids_count
count	801	801	801	568	664	801	801	801
unique	801	6	12	4	2	5	4	4
top	1	45-54	50-74K	Homeowner	Married	2	2 Adults No Kids	0
freq	1	288	192	504	340	318	258	513

In [412…
```
value_counts_age = demo['age'].value_counts()
percentage_age = round(((value_counts_age / len(demo)) * 100),2)
result_age = pd.DataFrame({'Age': value_counts.index, 'Count': value_counts_age, 'Percentage': percentage_age})
plt.figure(figsize=(4, 4))
plt.pie(value_counts_age, labels=value_counts_age.index, autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart for Age Distribution')
plt.axis('equal')
plt.show()
```

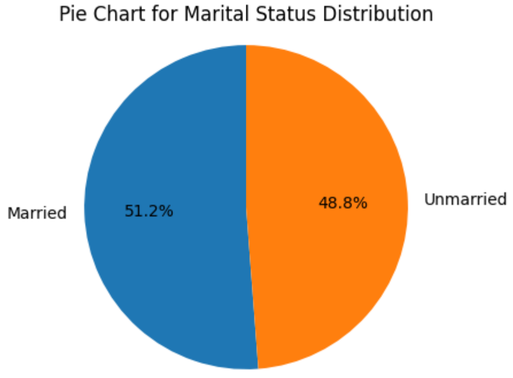


In [415…
```
value_counts_inc = demo['income'].value_counts()
percentage_inc = round(((value_counts_inc / len(demo)) * 100),2)
result_inc = pd.DataFrame({'Age': value_counts.index, 'Count': value_counts_age, 'Percentage': percentage_age})
plt.figure(figsize=(7, 7))
plt.pie(value_counts_inc, labels=value_counts_inc.index, autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart for Income Bracket Distribution')
plt.axis('equal')
plt.show()
```



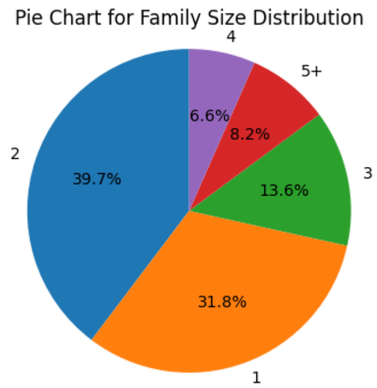
In [416…

```
value_counts_stat = demo['marital_status'].value_counts()
percentage_stat = round(((value_counts_stat / len(demo)) * 100),2)
result_stat = pd.DataFrame({'Marital Status': value_counts.index, 'Count': value_counts_age, 'Percentage': percentage_age})
plt.figure(figsize=(4, 4))
plt.pie(value_counts_stat, labels=value_counts_stat.index, autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart for Marital Status Distribution')
plt.axis('equal')
plt.show()
```



In [417…

```
value_counts_size = demo['household_size'].value_counts()
percentage_size = round(((value_counts_size / len(demo)) * 100),2)
result_size = pd.DataFrame({'Family Size': value_counts.index, 'Count': value_counts_age, 'Percentage': percentage_age})
result_inc
plt.figure(figsize=(4, 4))
plt.pie(value_counts_size, labels=value_counts_size.index, autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart for Family Size Distribution')
plt.axis('equal')
plt.show()
```



In [406…
```
result_age['Count'].sum()
```

Out[406]:
```
801
```

In [253…
```
demotran = pd.merge(demo, tran, on='household_id')
demotran
```

Out[253]:

	household_id	age	income	home_ownership	marital_status	household_size	household_comp	kids_count	store_id	basket_id	product_id	quantity	sales_value	retail_disc	coupon_disc	coupon_match_disc	week	transaction_timestamp
0	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	823721	1.0	2.99	0.00	0.0	0.0	2	2017-01-07 18:55:24
1	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	832990	2.0	5.98	0.00	0.0	0.0	2	2017-01-07 18:55:24
2	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	854920	1.0	1.49	0.00	0.0	0.0	2	2017-01-07 18:55:24
3	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	856942	1.0	2.99	0.00	0.0	0.0	2	2017-01-07 18:55:24
4	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	868401	1.0	0.59	0.00	0.0	0.0	2	2017-01-07 18:55:24
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
828845	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	5585635	1.0	0.40	0.00	0.0	0.0	51	2017-12-17 16:55:03
828846	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	5586942	1.0	0.40	0.00	0.0	0.0	51	2017-12-17 16:55:03
828847	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	5589787	1.0	3.49	0.00	0.0	0.0	51	2017-12-17 16:55:03
828848	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	8090536	1.0	4.00	1.29	0.0	0.0	51	2017-12-17 16:55:03
828849	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	12171886	1.0	12.99	1.30	0.0	0.0	51	2017-12-17 16:55:03

828850 rows × 18 columns

In [268…
```
demotran_id_rev = demotran.groupby('household_id').agg({
    "sales_value": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum'],
    "age": ['first'],
    "income": ['first'],
    "home_ownership": ['first'],
    "marital_status": ['first'],
    "household_size": ['first'],
    "household_comp": ['first']
}).reset_index()
demotran_id_rev.columns = ['household_id','Tot Rev','Tot Ret Desc','Tot Coup Disc','Tot Coup Match Disc','age','income','home_ownership','marital_status','household_size','household_comp']
demotran_id_rev
```

Out[268]:

	household_id	Tot Rev	Tot Ret Desc	Tot Coup Disc	Tot Coup Match Disc	age	income	home_ownership	marital_status	household_size	household_comp
0	1	2415.56	371.66	55.56	14.95	65+	35-49K	Homeowner	Married	2	2 Adults No Kids
1	1001	2558.82	264.33	13.24	4.80	45-54	50-74K	Homeowner	Unmarried	1	1 Adult No Kids
2	1003	1404.43	339.00	0.00	0.00	35-44	25-34K	NaN	Unmarried	1	1 Adult No Kids
3	1004	2486.93	303.73	0.00	0.00	25-34	15-24K	NaN	Unmarried	1	1 Adult No Kids
4	101	4483.35	1463.91	50.54	2.65	45-54	Under 15K	Homeowner	Married	4	2 Adults Kids
...	...	...	...	...	...	...	...	...	...	...	...
796	986	994.02	252.57	0.00	0.00	25-34	35-49K	NaN	Unmarried	1	1 Adult No Kids
797	992	431.94	36.28	0.00	0.00	45-54	35-49K	Homeowner	Married	3	2 Adults Kids
798	993	2177.92	424.52	9.10	0.90	55-64	50-74K	Homeowner	Married	5+	1 Adult Kids
799	996	1172.57	241.04	25.00	0.00	55-64	25-34K	Homeowner	Married	2	2 Adults No Kids
800	997	2476.95	225.41	1.00	0.00	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids

801 rows × 11 columns

In [266…
```
demotran['age'].nunique()
```

Out[266]:
```
6
```

In [267…
```
demotran['income'].nunique()
```

Out[267]:
```
12
```

Let us segregate by age

In [274…
```
demotran_age_rev = demotran.groupby('age').agg({
    "sales_value": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum']
}).reset_index()
demotran_age_rev.columns = ['Age Group','Tot Rev','Tot Ret Disc','Tot Coup Disc','Tot Coup Match Disc']
demotran_age_rev
```

Out[274]:

	Age Group	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc
0	19-24	125673.05	21756.06	469.44	95.39
1	25-34	453372.46	70751.87	2840.98	587.30
2	35-44	724357.40	122496.54	5239.58	963.63
3	45-54	971822.19	168343.09	6767.09	1160.81
4	55-64	173153.70	31136.86	1254.13	260.89
5	65+	176600.84	29201.92	833.55	137.57

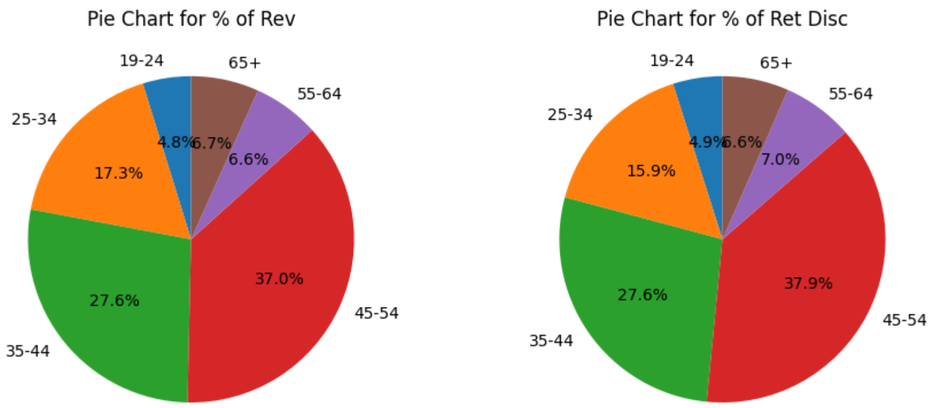
In [275…
```
tot_rev_age = demotran_age_rev['Tot Rev'].sum()
tot_ret_disc_age = demotran_age_rev['Tot Ret Disc'].sum()
tot_coup_disc_age = demotran_age_rev['Tot Coup Disc'].sum()
tot_coup_match_disc_age = demotran_age_rev['Tot Coup Match Disc'].sum()

# Create a new DataFrame with the percentage column
rev_perc_age = demotran_age_rev.copy()
rev_perc_age['% of Rev'] = (rev_perc_age['Tot Rev']/tot_rev_age)*100

ret_disc_perc_age = rev_perc_age.copy()
ret_disc_perc_age['% of Ret Disc'] = (ret_disc_perc_age['Tot Ret Disc']/tot_ret_disc_age)*100

coup_disc_age = ret_disc_perc_age.copy()
coup_disc_age['% of Coup Disc'] = (coup_disc_age['Tot Coup Disc']/tot_coup_disc_age)*100

coup_match_disc_age = coup_disc_age.copy()
coup_match_disc_age['% of Coup Match Disc'] = (coup_match_disc_age['Tot Coup Match Disc']/tot_coup_match_disc_age)*100

ages = coup_match_disc_age.reset_index()
ages
```

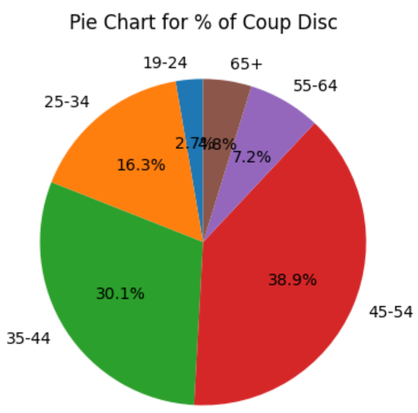
Out[275]:

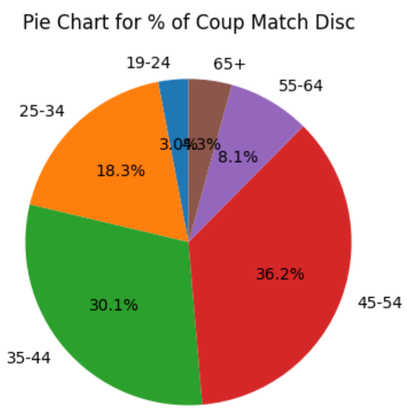
	index	Age Group	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc	% of Rev	% of Ret Disc	% of Coup Disc	% of Coup Match Disc
0	0	19-24	125673.05	21756.06	469.44	95.39	4.787582	4.903478	2.697192	2.975739
1	1	25-34	453372.46	70751.87	2840.98	587.30	17.271466	15.946371	16.322997	18.321120
2	2	35-44	724357.40	122496.54	5239.58	963.63	27.594782	27.608815	30.104276	30.060925
3	3	45-54	971822.19	168343.09	6767.09	1160.81	37.022085	37.941914	38.880663	36.212055
4	4	55-64	173153.70	31136.86	1254.13	260.89	6.596383	7.017764	7.205668	8.138595
5	5	65+	176600.84	29201.92	833.55	137.57	6.727703	6.581659	4.789204	4.291566

In [276…

```
categories = ages['Age Group']

# Create a subplot grid for multiple pie charts
fig, axes = plt.subplots(2, 2, figsize=(10, 8))

# Flatten the axes array to access each subplot
axes = axes.flatten()

# Plot a pie chart for each column
for i, col in enumerate(['% of Rev', '% of Ret Disc', '% of Coup Disc', '% of Coup Match Disc']):
    ax = axes[i]
    ax.pie(ages[col], labels=categories, autopct='%1.1f%%', startangle=90)
    ax.set_title(f'Pie Chart for {col}')

# Adjust layout
plt.tight_layout()

# Show the charts
plt.show()
```







Let us segregate by income criteria

In [277…

```
demotran_inc_rev = demotran.groupby('income').agg({
    "sales_value": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum']
}).reset_index()
```

```
demotran_inc_rev.columns = ['Income Group','Tot Rev','Tot Ret Disc','Tot Coup Disc','Tot Coup Match Disc']
demotran_inc_rev
```

Out[277]:

	Income Group	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc
0	100-124K	116225.09	18109.07	556.16	75.04
1	125-149K	172013.89	24666.83	1131.55	221.57
2	15-24K	180427.42	34284.12	1035.22	132.78
3	150-174K	150225.03	20323.27	1080.37	166.79
4	175-199K	50385.57	6074.10	145.47	26.25
5	200-249K	15368.37	1778.97	104.04	14.48
6	25-34K	221276.22	42113.24	1542.54	255.54
7	250K+	67575.91	8374.93	131.18	4.85
8	35-49K	485075.16	88570.03	2669.56	512.88
9	50-74K	639045.54	107364.04	4826.36	1029.87
10	75-99K	335833.52	53898.16	3245.31	591.35
11	Under 15K	191527.92	38129.58	937.01	174.19

In [278…

```
tot_rev_inc = demotran_inc_rev['Tot Rev'].sum()
tot_ret_disc_inc = demotran_inc_rev['Tot Ret Disc'].sum()
tot_coup_disc_inc = demotran_inc_rev['Tot Coup Disc'].sum()
tot_coup_match_disc_inc = demotran_inc_rev['Tot Coup Match Disc'].sum()

# Create a new DataFrame with the percentinc column
rev_perc_inc = demotran_inc_rev.copy()
rev_perc_inc['% of Rev'] = (rev_perc_inc['Tot Rev']/tot_rev_inc)*100

ret_disc_perc_inc = rev_perc_inc.copy()
ret_disc_perc_inc['% of Ret Disc'] = (ret_disc_perc_inc['Tot Ret Disc']/tot_ret_disc_inc)*100

coup_disc_inc = ret_disc_perc_inc.copy()
coup_disc_inc['% of Coup Disc'] = (coup_disc_inc['Tot Coup Disc']/tot_coup_disc_inc)*100

coup_match_disc_inc = coup_disc_inc.copy()
coup_match_disc_inc['% of Coup Match Disc'] = (coup_match_disc_inc['Tot Coup Match Disc']/tot_coup_match_disc_inc)*100

inc = coup_match_disc_inc.reset_index()
inc
```

Out[278]:

	index	Income Group	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc	% of Rev	% of Ret Disc	% of Coup Disc	% of Coup Match Disc
0	0	100-124K	116225.09	18109.07	556.16	75.04	4.427657	4.081503	3.195446	2.340911
1	1	125-149K	172013.89	24666.83	1131.55	221.57	6.552961	5.559520	6.501379	6.911988
2	2	15-24K	180427.42	34284.12	1035.22	132.78	6.873479	7.727107	5.947910	4.142139
3	3	150-174K	150225.03	20323.27	1080.37	166.79	5.722903	4.580549	6.207321	5.203098
4	4	175-199K	50385.57	6074.10	145.47	26.25	1.919465	1.369008	0.835805	0.818882
5	5	200-249K	15368.37	1778.97	104.04	14.48	0.585466	0.400952	0.597767	0.451711
6	6	25-34K	221276.22	42113.24	1542.54	255.54	8.429636	9.491669	8.862743	7.971699
7	7	250K+	67575.91	8374.93	131.18	4.85	2.574340	1.887579	0.753701	0.151298
8	8	35-49K	485075.16	88570.03	2669.56	512.88	18.479197	19.962307	15.338094	15.999551
9	9	50-74K	639045.54	107364.04	4826.36	1029.87	24.344781	24.198185	27.730099	32.127315
10	10	75-99K	335833.52	53898.16	3245.31	591.35	12.793757	12.147807	18.646095	18.447462
11	11	Under 15K	191527.92	38129.58	937.01	174.19	7.296358	8.593814	5.383639	5.433945

In [292…

```
inc['Tot Rev'].sum()
```

Out[292]:

```
2624979.64
```

In [293…

```
inc['Tot Ret Disc'].sum()
```

Out[293]:

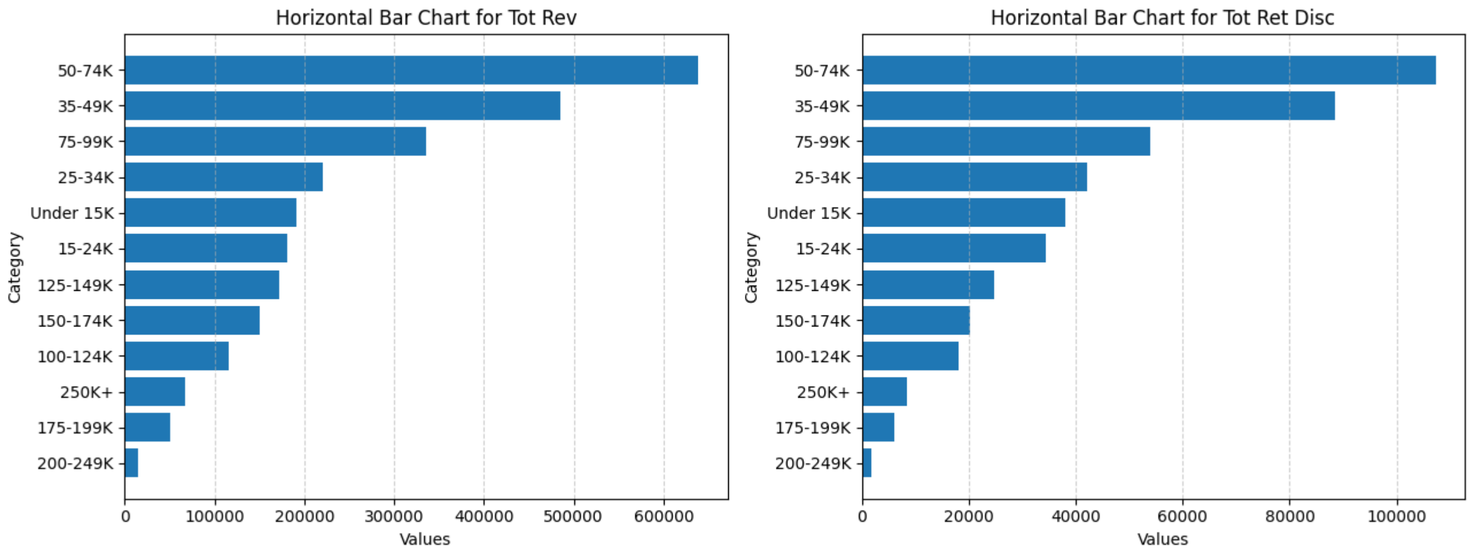
```
443686.34
```

In [ ]:

In [296…

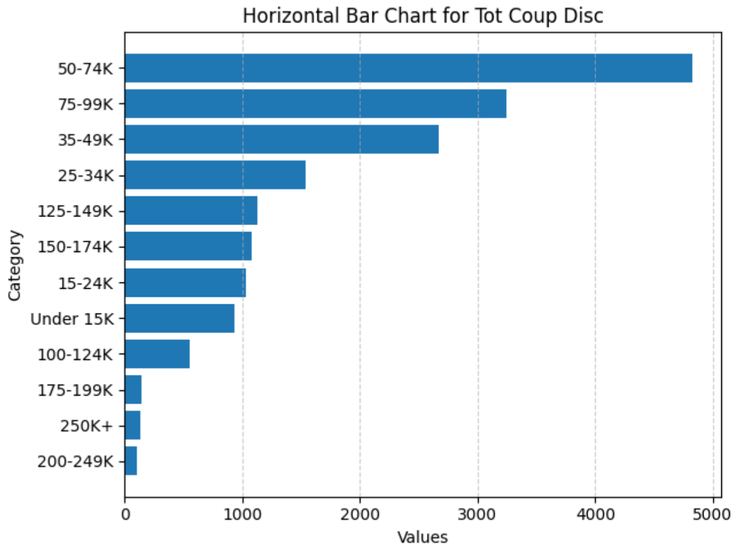
```
# Create subplots for each column
fig, axes = plt.subplots(1, 2, figsize=(13, 5))

# List of column names
columns_inc = ['Tot Rev', 'Tot Ret Disc']

# Sort and plot each column
for i, col in enumerate(columns_inc):
    ax = axes[i]
```

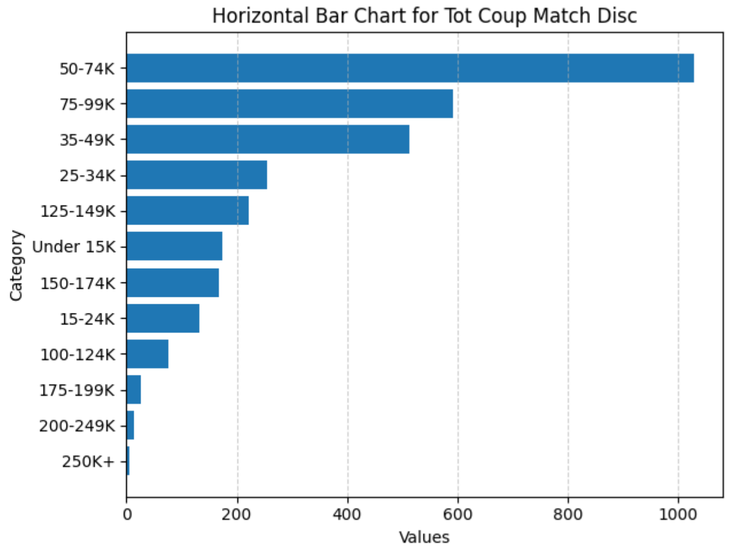
```
    # Sort the DataFrame by the current column in descending order
    inc_sorted = inc.sort_values(by=col, ascending=False)
    inc_sorted = inc_sorted[::-1]

    # Plot a horizontal bar chart for the current column
    ax.barh(inc_sorted['Income Group'], inc_sorted[col])

    # Set labels and title for the subplot
    ax.set_xlabel('Values')
    ax.set_ylabel('Category')
    ax.set_title(f'Horizontal Bar Chart for {col}')
    ax.grid(axis='x', linestyle='--', alpha=0.6)

# Adjust layout
plt.tight_layout()

# Show the charts
plt.show()
```



In [297…

```
# Create subplots for each column
fig, axes = plt.subplots(1, 2, figsize=(13, 5))

# List of column names
columns_inc1 = ['Tot Coup Disc', 'Tot Coup Match Disc']

# Sort and plot each column
for i, col in enumerate(columns_inc1):
    ax = axes[i]

    # Sort the DataFrame by the current column in descending order
    inc_sorted = inc.sort_values(by=col, ascending=False)
    inc_sorted = inc_sorted[::-1]

    # Plot a horizontal bar chart for the current column
    ax.barh(inc_sorted['Income Group'], inc_sorted[col])

    # Set labels and title for the subplot
    ax.set_xlabel('Values')
    ax.set_ylabel('Category')
    ax.set_title(f'Horizontal Bar Chart for {col}')
    ax.grid(axis='x', linestyle='--', alpha=0.6)

# Adjust layout
plt.tight_layout()

# Show the charts
plt.show()
```





In [294…
```
inc.to_csv('000my_data.csv', index=False)
```

In [298…
```
demotran['household_comp'].nunique()
```

Out[298]:
```
4
```

In [299…
```
demotran['household_size'].nunique()
```

Out[299]:
```
5
```

In [300…
```
demotran['marital_status'].nunique()
```

Out[300]:
```
2
```

In [329…
```
demotran['home_ownership'].nunique()
```

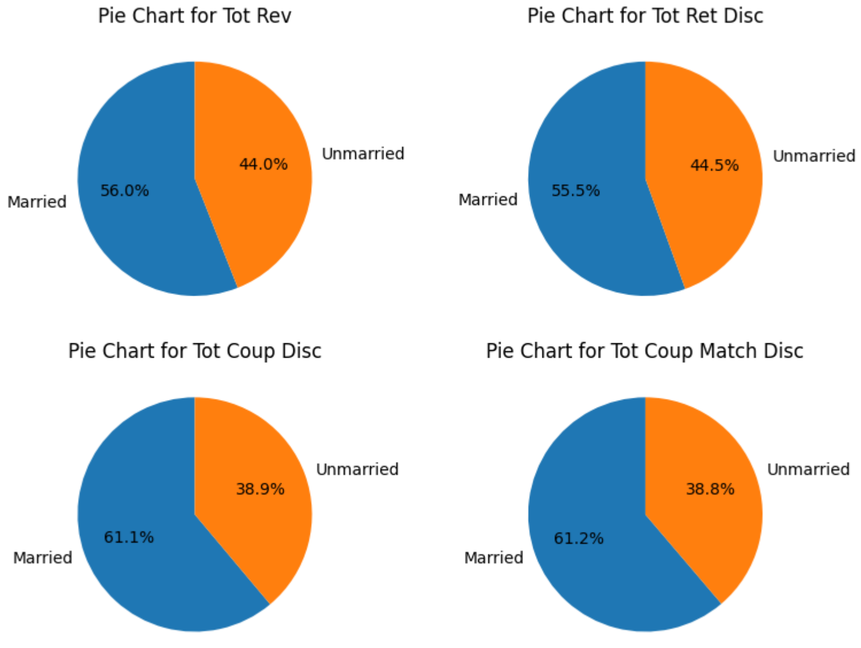
Out[329]:
```
4
```

In [301…
```
demotran_stat_rev = demotran.groupby('marital_status').agg({
    "sales_value": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum']
}).reset_index()
demotran_stat_rev.columns = ['Marital Status','Tot Rev','Tot Ret Disc','Tot Coup Disc','Tot Coup Match Disc']
demotran_stat_rev
```

Out[301]:

	Marital Status	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc
0	Married	1219736.00	204674.97	9061.00	1693.59
1	Unmarried	959318.66	163914.90	5765.55	1072.25

In [324…
```
# List of columns to plot
columns_to_plot_stat = ['Tot Rev', 'Tot Ret Disc', 'Tot Coup Disc', 'Tot Coup Match Disc']

fig, axes = plt.subplots(2, 2, figsize=(8, 6))
for i, col in enumerate(columns_to_plot_stat):
    ax = axes[i // 2, i % 2]
    values = demotran_stat_rev[col]
    percentages = (values / values.sum()) * 100
    labels = demotran_stat_rev['Marital Status']
    ax.pie(percentages, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.set_title(f'Pie Chart for {col}')

plt.tight_layout()

plt.show()
```



In [320…

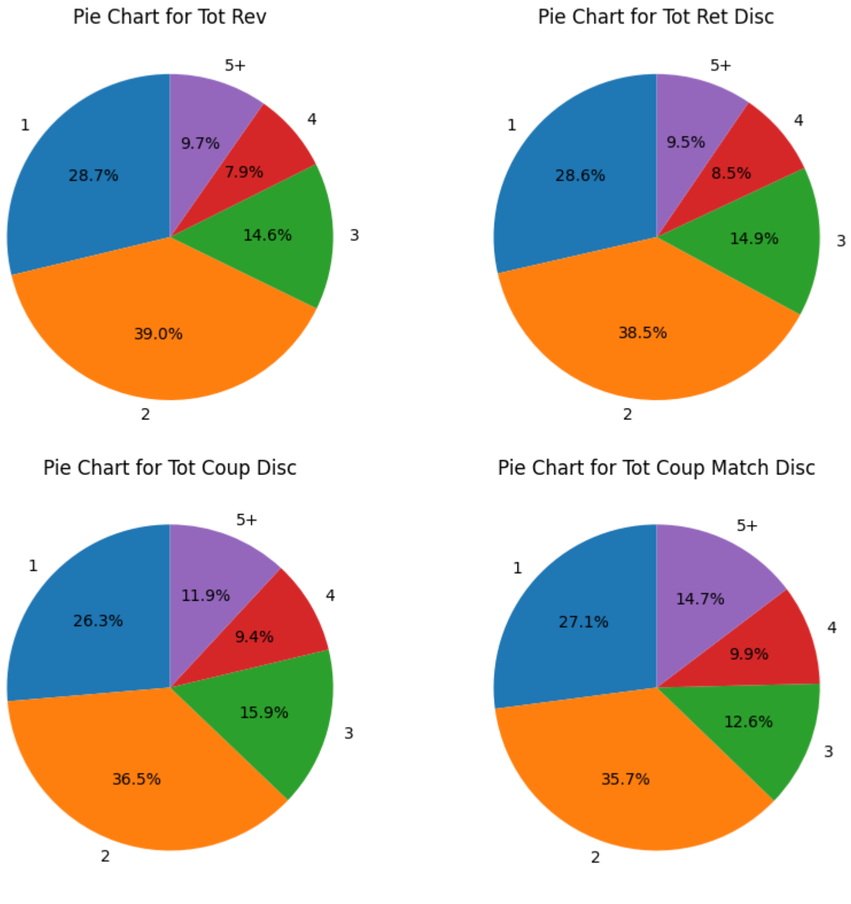
```
demotran_size_rev = demotran.groupby('household_size').agg({
    "sales_value": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum']
}).reset_index()
demotran_size_rev.columns = ['Household Size','Tot Rev','Tot Ret Disc','Tot Coup Disc','Tot Coup Match Disc']
demotran_size_rev
```

Out[320]:

	Household Size	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc
0	1	754648.96	126794.96	4585.25	868.17
1	2	1024643.15	170945.80	6356.82	1144.45
2	3	383256.15	65963.76	2762.74	403.89
3	4	207407.07	37714.95	1630.69	318.58
4	5+	255024.31	42266.87	2069.27	470.50

In [328…

```
# List of columns to plot
columns_to_plot_size = ['Tot Rev', 'Tot Ret Disc', 'Tot Coup Disc', 'Tot Coup Match Disc']

fig, axes = plt.subplots(2, 2, figsize=(9, 8))
for i, col in enumerate(columns_to_plot_size):
    ax = axes[i // 2, i % 2]
    values = demotran_size_rev[col]
    percentages = (values / values.sum()) * 100
    labels = demotran_size_rev['Household Size']
    ax.pie(percentages, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.set_title(f'Pie Chart for {col}')

plt.tight_layout()

plt.show()
```

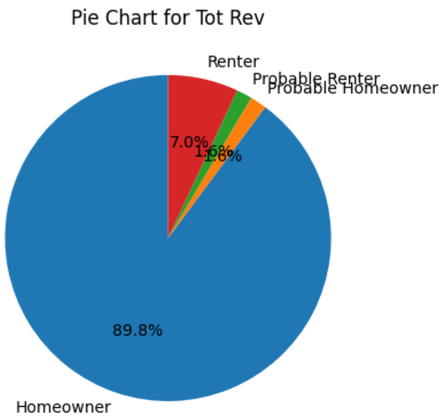


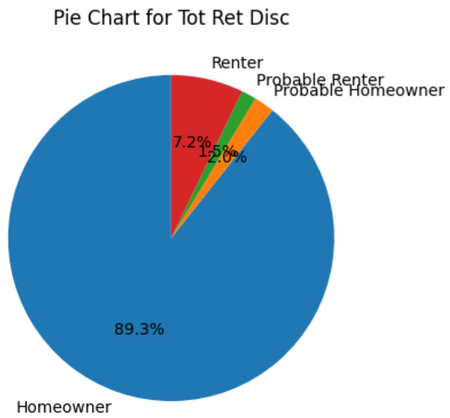
In [330…
```
demotran_own_rev = demotran.groupby('home_ownership').agg({
    "sales_value": ['sum'],
    "retail_disc": ['sum'],
    "coupon_disc": ['sum'],
    "coupon_match_disc": ['sum']
}).reset_index()
demotran_own_rev.columns = ['Ownership','Tot Rev','Tot Ret Disc','Tot Coup Disc','Tot Coup Match Disc']
demotran_own_rev
```

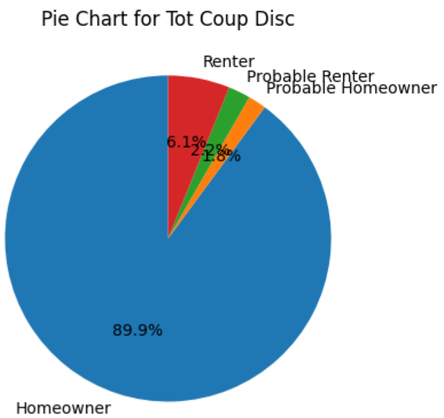
Out[330]:

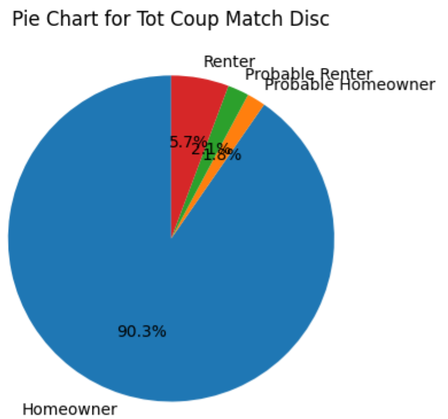
	Ownership	Tot Rev	Tot Ret Disc	Tot Coup Disc	Tot Coup Match Disc
0	Homeowner	1771333.14	293665.93	13218.09	2517.93
1	Probable Homeowner	32376.37	6638.59	264.21	51.19
2	Probable Renter	30639.99	4831.90	321.75	58.60
3	Renter	137301.68	23586.97	899.66	159.17

In [363…
```
# List of columns to plot
columns_to_plot_own = ['Tot Rev', 'Tot Ret Disc', 'Tot Coup Disc', 'Tot Coup Match Disc']

fig, axes = plt.subplots(2, 2, figsize=(9, 8))
for i, col in enumerate(columns_to_plot_own):
    ax = axes[i // 2, i % 2]
    values = demotran_own_rev[col]
    percentages = (values / values.sum()) * 100
    labels = demotran_own_rev['Ownership']
    ax.pie(percentages, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.set_title(f'Pie Chart for {col}')

plt.tight_layout()

plt.show()
```









In [336…
```
camp
```

Out[336]:

	campaign_id	household_id
0	1	105
1	1	1238
2	1	1258
3	1	1483
4	1	2200
…	…	…
6584	9	93
6585	9	941
6586	9	949
6587	9	993
6588	9	997

6589 rows × 2 columns

In [337…
```
camp.describe()
```

Out[337]:

	campaign_id	household_id
count	6589	6589
unique	27	1559
top	18	2317
freq	1133	15

In [339…
```
camp_desc.describe()
```

Out[339]:

	campaign_id	campaign_type	start_date	end_date
count	27	27	27	27
unique	27	3	25	25
top	1	Type B	2017-12-06	2017-04-09
freq	1	17	2	2

In [340…
```
camps = pd.merge(camp, camp_desc, on='campaign_id')
camps.describe()
```

Out[340]:

	campaign_id	household_id	campaign_type	start_date	end_date
count	6589	6589	6589	6589	6589
unique	27	1559	3	25	25
top	18	2317	Type A	2017-10-30	2017-12-24
freq	1133	15	3647	1133	1133

It looks like most purchases were done under campaign 18 for 1133 times.

In [341…
```
cou.describe()
```

Out[341]:

	coupon_upc	product_id	campaign_id
count	116204	116204	116204
unique	981	41857	27
top	10000085478	2182786	13
freq	14477	48	38248

In [342…
```
cou_redem.describe()
```

Out[342]:

	household_id	coupon_upc	campaign_id	redemption_date
count	2102	2102	2102	2102
unique	410	491	26	263
top	367	10000085475	18	2017-11-05
freq	30	63	653	47

In [354…
```
tran['household_id'].nunique()
```

Out[354]:
```
2469
```

In [345…
```
coup_used = pd.merge(cou_redem, cou, on='coupon_upc')
coup_used.describe()
```

Out[345]:

	household_id	coupon_upc	campaign_id_x	redemption_date	product_id	campaign_id_y
count	2265375	2265375	2265375	2265375	2265375	2265375
unique	410	491	26	263	39902	27
top	2007	10000085478	18	2017-11-05	1118641	18
freq	41545	289540	999428	86329	272	989217

In [346…
```
demotran
```

Out[346]:

	household_id	age	income	home_ownership	marital_status	household_size	household_comp	kids_count	store_id	basket_id	product_id	quantity	sales_value	retail_disc	coupon_disc	coupon_match_disc	week	transaction_timestamp
0	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	823721	1.0	2.99	0.00	0.0	0.0	2	2017-01-07 18:55:24
1	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	832990	2.0	5.98	0.00	0.0	0.0	2	2017-01-07 18:55:24
2	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	854920	1.0	1.49	0.00	0.0	0.0	2	2017-01-07 18:55:24
3	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	856942	1.0	2.99	0.00	0.0	0.0	2	2017-01-07 18:55:24
4	1	65+	35-49K	Homeowner	Married	2	2 Adults No Kids	0	436	31317046240	868401	1.0	0.59	0.00	0.0	0.0	2	2017-01-07 18:55:24
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
828845	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	5585635	1.0	0.40	0.00	0.0	0.0	51	2017-12-17 16:55:03
828846	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	5586942	1.0	0.40	0.00	0.0	0.0	51	2017-12-17 16:55:03
828847	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	5589787	1.0	3.49	0.00	0.0	0.0	51	2017-12-17 16:55:03
828848	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	8090536	1.0	4.00	1.29	0.0	0.0	51	2017-12-17 16:55:03
828849	997	45-54	75-99K	Homeowner	Unmarried	1	1 Adult No Kids	0	447	41260300605	12171886	1.0	12.99	1.30	0.0	0.0	51	2017-12-17 16:55:03

828850 rows × 18 columns

In [347…
```
demotran.describe()
```

Out[347]:

	quantity	sales_value	retail_disc	coupon_disc	coupon_match_disc	week	transaction_timestamp
count	828850.000000	828850.000000	828850.000000	828850.000000	828850.000000	828850.000000	828850
mean	118.228127	3.167014	0.535304	0.020999	0.003868	27.572110	2017-07-04 01:21:03.095836928
min	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	2017-01-01 11:53:26
25%	1.000000	1.290000	0.000000	0.000000	0.000000	15.000000	2017-04-03 20:13:28
50%	1.000000	2.090000	0.000000	0.000000	0.000000	28.000000	2017-07-04 17:43:49
75%	1.000000	3.490000	0.680000	0.000000	0.000000	41.000000	2017-10-02 21:09:06
max	89638.000000	840.000000	130.020000	55.930000	4.050000	53.000000	2018-01-01 04:01:20
std	1271.822028	4.410594	1.244723	0.249021	0.044956	15.051818	NaN

In [350…
```
import seaborn as sns

demotran.groupby(['age', 'household_id'])['sales_value'].sum().plot().bar
```

Out[350]:
```
<bound method Axes.bar of <Axes: xlabel='age,household_id'>>
```

In [364…
```
prod['department'].nunique()
```

Out[364]:
```
32
```

In [366…
```
tran['product_id'].nunique()
```

Out[366]:
```
68509
```

Let us find data for unsold products

In [373…
```
# Merge df1 and df2 using an outer join and indicator=True
unsold = pd.merge(prod, tran, on='product_id', how='outer', indicator=True)

# Filter rows where '_merge' is 'left_only' (not common with df2)
unsold_df = unsold[unsold['_merge'] == 'left_only']

# Drop the '_merge' column from the result
unsold_df = unsold_df.drop(columns=['_merge'])

# Display the resulting DataFrame
unsold_df
```

Out[373]:

	product_id	manufacturer_id	department	brand	product_category	product_type	package_size	household_id	store_id	basket_id	quantity	sales_value	retail_disc	coupon_disc	coupon_match_disc	week	transaction_timestamp
1	26081	2	MISCELLANEOUS	National	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
3	26190	69	GROCERY	Private	FRUIT - SHELF STABLE	APPLE SAUCE	50 OZ	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
4	26355	69	GROCERY	Private	COOKIES/CONES	SPECIALTY COOKIES	14 OZ	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
5	26426	69	GROCERY	Private	SPICES & EXTRACTS	SPICES & SEASONINGS	2.5 OZ	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
6	26540	69	GROCERY	Private	COOKIES/CONES	TRAY PACK/CHOC CHIP COOKIES	16 OZ	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1488305	18293142	6384	DRUG GM	National	BOOKSTORE	PAPERBACK BOOKS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
1488306	18293439	6393	DRUG GM	National	BOOKSTORE	CHILDRENS LOW END	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
1488307	18293696	6406	DRUG GM	National	BOOKSTORE	PAPERBACK BEST SELLER	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
1488308	18294080	6442	DRUG GM	National	BOOKSTORE	PAPERBACK BOOKS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT
1488309	18316298	764	GROCERY	National	PAPER TOWELS	PAPER TOWELS & HOLDERS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaT

23839 rows × 17 columns

In [370…
```
unsold_df['product_id'].nunique()
```

Out[370]:
```
23839
```

In [390…
```
unsold_df['department'].nunique()
```

Out[390]:
```
28
```

In [378…
```
unsold_df.isnull().sum()
```

Out[378]:
```
product_id                  0
manufacturer_id             0
department                  0
brand                       0
product_category          198
product_type              181
package_size             8615
household_id            23839
store_id                23839
basket_id               23839
quantity                23839
sales_value             23839
retail_disc             23839
coupon_disc             23839
coupon_match_disc       23839
week                    23839
transaction_timestamp   23839
dtype: int64
```

In [384…
```
select_col_unsold = ['product_id', 'department', 'product_category','product_type']
unsold_retain = unsold_df[select_col_unsold]
unsold_retain
```

Out[384]:

	product_id	department	product_category	product_type
1	26081	MISCELLANEOUS	NaN	NaN
3	26190	GROCERY	FRUIT - SHELF STABLE	APPLE SAUCE
4	26355	GROCERY	COOKIES/CONES	SPECIALTY COOKIES
5	26426	GROCERY	SPICES & EXTRACTS	SPICES & SEASONINGS
6	26540	GROCERY	COOKIES/CONES	TRAY PACK/CHOC CHIP COOKIES
...	...	...	...	...
1488305	18293142	DRUG GM	BOOKSTORE	PAPERBACK BOOKS
1488306	18293439	DRUG GM	BOOKSTORE	CHILDRENS LOW END
1488307	18293696	DRUG GM	BOOKSTORE	PAPERBACK BEST SELLER
1488308	18294080	DRUG GM	BOOKSTORE	PAPERBACK BOOKS
1488309	18316298	GROCERY	PAPER TOWELS	PAPER TOWELS & HOLDERS

23839 rows × 4 columns

In [385…
```
unsold_retain.isnull().sum()
```

Out[385]:
```
product_id            0
department            0
product_category    198
product_type        181
dtype: int64
```

In [387…
```
uns = unsold_retain.dropna(subset=['product_category', 'product_type'], how='all')
uns.isnull().sum()
```

Out[387]:
```
product_id          0
department          0
product_category    53
product_type        36
dtype: int64
```

In [388…
```
uns1 = uns.dropna(subset=['product_category', 'product_type'], how='any')
uns1.isnull().sum()
```

Out[388]:
```
product_id          0
department          0
product_category    0
product_type        0
dtype: int64
```

In [391…
```
uns1['department'].nunique()
```

Out[391]:
```
24
```

In [389…
```
unsold_det = uns1.groupby('department').agg({
    "product_category": [lambda x: x.mode().iloc[0],lambda x: second_mode(x),lambda x: third_mode(x)],
    "product_type": [lambda x: x.mode().iloc[0],lambda x: second_mode(x),lambda x: third_mode(x)]
}).reset_index()
unsold_det.columns = ['Department', 'Highest Catg Unsold', '2nd High Catg', '3rd High Catg',
                      'Highest Type Unsold', '2nd High Type', '3rd High Type']
unsold_det
```

Out[389]:

	Department	Highest Catg Unsold	2nd High Catg	3rd High Catg	Highest Type Unsold	2nd High Type	3rd High Type
0	COSMETICS	MAKEUP AND TREATMENT	BATH	FRAGRANCES	MAYBELLINE	LOREAL COSMETICS	REVLON
1	COUPON	COUPONS/STORE & MFG	None	None	COUPONS/STORE & MFG	None	None
2	DELI	CHEESES	DELI MEATS	SALADS/DIPS	CHEESE:SPECIALTY PREPACK	CHEESE: NATURAL BULK	MEAT: TURKEY BULK
3	DRUG GM	GREETING CARDS/WRAP/PARTY SPLY	CANDY - PACKAGED	BOOKSTORE	CARDS EVERYDAY	GIFT-WRAP EVERYDAY	CHILDRENS LOW END
4	FLORAL	FLORAL-FRESH CUT	FLORAL BALLOONS	FLORAL- HARD GOODS	BALLOONS 18IN PKG NON LICENSED	PREMIUM FLOWERING PLANTS	BALLOONS SHAPES/OTHER
5	FUEL	COUPON/MISC ITEMS	COUPON/MISC ITEMS	None	GASOLINE-REG UNLEADED	None	None
6	GARDEN CENTER	GARDEN CENTER	None	None	FLOWER SEEDS	BULB SETS	VEGETABLE SEEDS
7	GROCERY	SOFT DRINKS	BAG SNACKS	FRZN MEAT/MEAT DINNERS	YOGURT NOT MULTI-PACKS	BEERALEMALT LIQUORS	POURABLE SALAD DRESSINGS
8	MEAT	BEEF	PORK	CHICKEN	LOIN - STK/CHP/SLC	CHOICE BEEF	PRIMAL
9	MEAT-PCKGD	LUNCHMEAT	DINNER SAUSAGE	BREAKFAST SAUSAGE/SANDWICHES	POULTRY	HAM	SMOKED/COOKED
10	MISCELLANEOUS	COUPON/MISC ITEMS	None	None	GASOLINE-REG UNLEADED	MISC SALES TRANS	OUTSIDE VENDORS GIFT CARDS
11	NUTRITION	REFRIGERATED	FITNESS&DIET	SNKS/CKYS/CRKR/CNDY	FITNESS&DIET - BARS	JUICE	CANDY/CHOCOLATE
12	PASTRY	COOKIES	CAKES	BREAD	COOKIES: HOLIDAY/SPECIAL OCCAS	COOKIES: REGULAR	PIES: FRUIT/NUT
13	PHOTO & VIDEO	BEERS/ALES	None	None	BEERALEMALT LIQUORS	None	None
14	POSTAL CENTER	CANDY - CHECKLANE	CANDY - CHECKLANE	None	CANDY BARS (SINGLES)(INCLUDING	CANDY BARS (SINGLES)(INCLUDING	None
15	PROD-WHS SALES	PROD SUPPLIES	None	None	COUPON	None	None
16	PRODUCE	ORGANICS FRUIT & VEGETABLES	SALAD MIX	PROCESSED	BLENDS	POPCORN - MICROWAVE	POTATOES RUSSET (BULK&BAG)
17	RESTAURANT	COFFEE SHOP	COFFEE SHOP SWEET GOODS&RETAIL	None	SV BEV: COFFEE DINE IN	COFF SHOP: INEDIBLES	COFF SHOP: RETAIL PACK BEVERAG
18	SALAD BAR	SALAD BAR	None	None	SALAD BAR OTHER	CONDIMENTS/SUPPLIES	SOUPS
19	SEAFOOD	SEAFOOD-FRESH	SEAFOOD - FROZEN	SEAFOOD - MISC	SEAFOOD-FRE-RAW FINFISH-OTHER	SEAFOOD-FRZ-RW-ALL	SEAFOOD-FRE-SALMON
20	SEAFOOD-PCKGD	SEAFOOD - FROZEN	SEAFOOD - MISC	SEAFOOD-FRESH	FRZN BRD STICK/PORTON	SEAFOOD-FRZ-RAW FILLETS	SEAFOOD-FRE-NON RW-ALL
21	SPIRITS	LIQUOR	None	None	LIQUEURS/SPECIALTIES (42 UNDER	RTD COCKTAILS 750ML PLUS	BOURBON/TN WHISKEY
22	TOYS	TOYS	None	None	DIECAST MINI	None	None
23	TRAVEL & LEISURE	APPLES	TROPICAL FRUIT	BERRIES	APPLES OTHER (BULK&BAG)	APPLES GALA (BULK&BAG)	BANANAS

In [392…
```
unsold_det.to_csv('000unsold.csv', index=False)
```

In [399…
```
prod_sales
```

Out[399]:

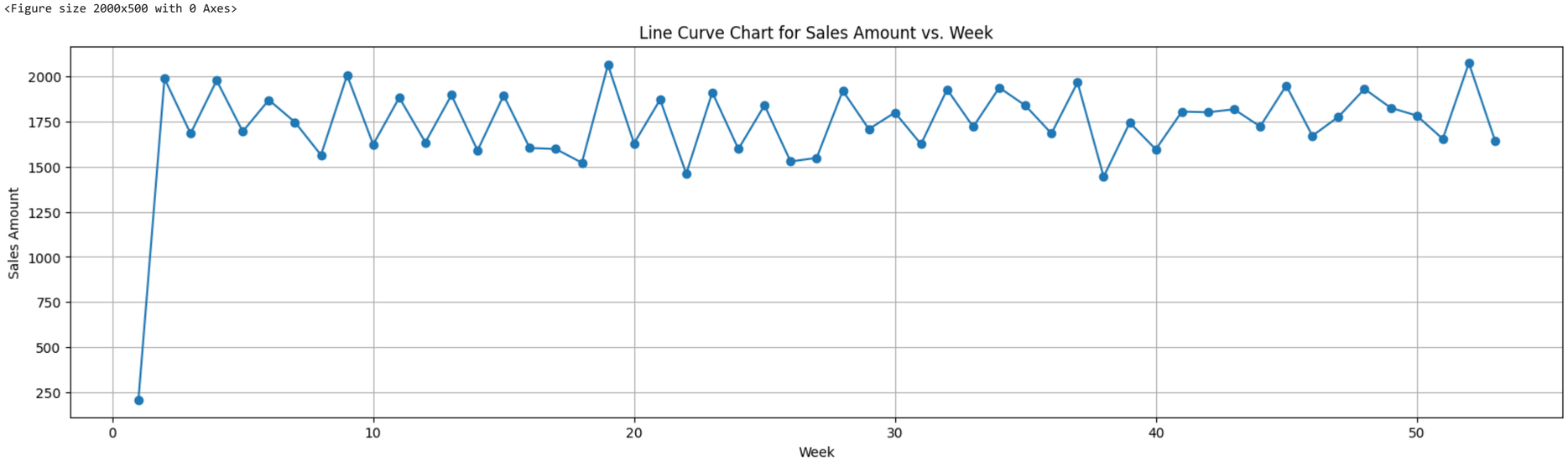
	product_id	sales_value
0	1000002	422.43
1	1000050	442.31
2	1000057	10.87
3	1000059	16.47
4	1000092	6.42
...	...	...
68504	999973	98.36
68505	999982	45.46
68506	999987	23.96
68507	999992	42.61
68508	9e+05	69.00

68509 rows × 2 columns

In [402…

```
prodid_sales = pd.merge(prod, tran, on='product_id')

milk_weekwise = prodid_sales[prodid_sales['product_type'] == 'FLUID MILK WHITE ONLY']

# Group by 'Week' and calculate the sum of 'sales_value' for Product Category A
milk_weekwise_sales = milk_weekwise.groupby('week')['sales_value'].sum().reset_index()

# Rename the columns for clarity
milk_weekwise_sales.columns = ['Week', 'Sales_Amount_Milk']

plt.figure(figsize=(20, 5))
plt.plot(milk_weekwise_sales['Week'], milk_weekwise_sales['Sales_Amount_Milk'], marker='o', linestyle='-')
plt.xlabel('Week')
plt.ylabel('Sales Amount')
plt.title('Line Curve Chart for Sales Amount vs. Week')
plt.grid(True)

# Show the chart
plt.show()
```

```
<Figure size 2000x500 with 0 Axes>
```



In [394…

```
prod_sales
```

Out[394]:

	product_id	sales_value
0	1000002	422.43
1	1000050	442.31
2	1000057	10.87
3	1000059	16.47
4	1000092	6.42
...	...	...
68504	999973	98.36
68505	999982	45.46
68506	999987	23.96
68507	999992	42.61
68508	9e+05	69.00

68509 rows × 2 columns

In [ ]: