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

df=pd.read_csv('/content/ifood_df.csv')

df

{"type":"dataframe","variable_name":"df"}

df.head()

{"type":"dataframe","variable_name":"df"}

df.head(10)

{"type":"dataframe","variable_name":"df"}

df.shape

(2205, 39)

df.columns

Index(['Income', 'Kidhome', 'Teenhome', 'Recency', 'MntWines',
'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases',
'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue',
'Response',
       'Age', 'Customer_Days', 'marital_Divorced', 'marital_Married',
       'marital_Single', 'marital_Together', 'marital_Widow',
       'education_2n Cycle', 'education_Basic',
'education_Graduation',
       'education_Master', 'education_PhD', 'MntTotal',
'MntRegularProds',
       'AcceptedCmpOverall'],
      dtype='object')

header=['Income', 'Num of kids', 'Num of teens', 'Recency',
'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases',
'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue',
'Response',
```

```
       'Age', 'Customer_Days', 'marital_Divorced', 'marital_Married',
       'marital_Single', 'marital_Together', 'marital_Widow',
       'education_2n Cycle', 'education_Basic',
'education_Graduation',
       'education_Master', 'education_PhD', 'MntTotal',
'MntRegularProds',
       'AcceptedCmpOverall']
df.columns = header

df.head(10)
```

```
{"type":"dataframe","variable_name":"df"}
```

```
num_of_kids = df.index[df['Num of kids'] >= 1].tolist()
num_of_kids
```

```
[1,
 3,
 4,
 7,
 8,
 9,
 12,
 15,
 18,
 20,
 25,
 26,
 33,
 35,
 39,
 41,
 44,
 45,
 48,
 50,
 53,
 56,
 60,
 61,
 64,
 67,
 69,
 74,
 75,
 76,
 79,
 80,
 84,
 85,
```

```
86,
88,
96,
97,
99,
105,
106,
108,
109,
110,
112,
113,
114,
118,
119,
121,
123,
125,
126,
127,
128,
133,
135,
136,
137,
142,
146,
149,
151,
153,
157,
158,
159,
160,
161,
165,
166,
169,
171,
173,
177,
178,
180,
181,
182,
186,
191,
193,
194,
```

```
199,
200,
202,
203,
206,
207,
210,
212,
216,
217,
219,
222,
223,
225,
228,
231,
232,
236,
237,
240,
241,
244,
245,
246,
248,
249,
250,
251,
253,
255,
257,
259,
260,
266,
270,
271,
273,
277,
279,
280,
282,
283,
286,
288,
289,
290,
296,
297,
299,
```

```
300,
301,
302,
303,
305,
306,
310,
311,
312,
313,
314,
317,
318,
321,
327,
329,
331,
332,
334,
338,
345,
346,
347,
350,
353,
354,
355,
357,
358,
360,
362,
363,
364,
366,
367,
368,
372,
374,
379,
380,
381,
382,
383,
385,
387,
388,
389,
390,
391,
```

```
393,
396,
399,
402,
403,
404,
405,
408,
410,
411,
414,
415,
417,
421,
422,
423,
426,
429,
431,
435,
436,
438,
440,
447,
450,
452,
454,
455,
456,
457,
459,
462,
463,
464,
465,
471,
474,
483,
486,
493,
499,
500,
503,
507,
508,
512,
515,
516,
517,
```

```
522,
523,
525,
528,
530,
531,
532,
533,
534,
538,
539,
543,
550,
552,
554,
555,
557,
559,
560,
561,
562,
563,
565,
567,
568,
573,
576,
578,
579,
580,
581,
582,
587,
588,
590,
595,
596,
598,
599,
600,
605,
607,
610,
611,
612,
617,
627,
629,
633,
```

```
640,
642,
648,
649,
650,
656,
659,
674,
677,
681,
684,
685,
686,
689,
690,
692,
694,
697,
699,
702,
704,
708,
710,
711,
712,
713,
714,
718,
721,
722,
723,
726,
728,
738,
740,
741,
743,
752,
753,
755,
758,
761,
762,
763,
764,
765,
766,
768,
773,
```

```
779,
781,
782,
788,
791,
796,
799,
801,
806,
807,
809,
810,
813,
814,
816,
820,
822,
823,
829,
833,
837,
838,
840,
841,
843,
844,
847,
849,
852,
853,
854,
856,
857,
859,
860,
862,
863,
866,
867,
868,
870,
871,
873,
878,
879,
884,
887,
888,
893,
```

```
900,
902,
903,
904,
911,
913,
915,
919,
920,
927,
928,
930,
931,
932,
933,
934,
935,
938,
942,
944,
947,
948,
949,
951,
952,
957,
958,
960,
962,
969,
971,
972,
973,
974,
977,
978,
982,
983,
988,
989,
992,
994,
998,
1000,
1001,
1002,
1004,
1006,
1007,
```

```
1013,
1016,
1017,
1018,
1021,
1024,
1028,
1031,
1033,
1034,
1040,
1043,
1045,
1050,
1057,
1058,
1059,
1061,
1063,
1066,
1068,
1071,
1073,
1074,
1078,
1082,
1083,
1084,
1088,
1089,
1092,
1094,
1096,
1101,
1102,
1103,
1105,
1106,
1107,
1111,
1112,
1115,
1116,
1121,
1122,
1125,
1126,
1127,
1129,
```

```
1135,
1136,
1141,
1147,
1153,
1154,
1156,
1158,
1163,
1164,
1165,
1166,
1174,
1177,
1180,
1183,
1194,
1195,
1197,
1200,
1201,
1203,
1204,
1207,
1209,
1210,
1211,
1213,
1214,
1215,
1218,
1219,
1220,
1221,
1224,
1226,
1227,
1228,
1235,
1238,
1240,
1243,
1244,
1246,
1247,
1252,
1253,
1254,
1256,
1257,
```

```
1262,
1265,
1267,
1271,
1272,
1273,
1274,
1276,
1279,
1282,
1285,
1293,
1294,
1296,
1297,
1298,
1302,
1303,
1306,
1309,
1310,
1313,
1318,
1321,
1323,
1324,
1331,
1335,
1338,
1339,
1340,
1341,
1346,
1347,
1350,
1351,
1355,
1356,
1359,
1360,
1362,
1363,
1365,
1366,
1367,
1368,
1369,
1371,
1372,
```

```
1374,
1375,
1379,
1380,
1384,
1386,
1387,
1389,
1390,
1393,
1394,
1395,
1400,
1401,
1402,
1403,
1405,
1406,
1408,
1409,
1410,
1411,
1414,
1415,
1416,
1417,
1425,
1428,
1432,
1437,
1438,
1439,
1443,
1445,
1446,
1457,
1458,
1462,
1464,
1465,
1466,
1468,
1469,
1471,
1475,
1476,
1477,
1478,
1481,
```

1486,
1490,
1492,
1493,
1494,
1497,
1498,
1499,
1500,
1502,
1506,
1508,
1509,
1510,
1511,
1514,
1515,
1516,
1518,
1520,
1524,
1530,
1531,
1532,
1533,
1535,
1537,
1539,
1543,
1544,
1546,
1548,
1549,
1551,
1555,
1560,
1561,
1563,
1567,
1571,
1572,
1573,
1574,
1578,
1579,
1580,
1582,
1588,
1589,

```
1590,
1591,
1592,
1596,
1598,
1599,
1603,
1604,
1606,
1607,
1608,
1613,
1614,
1615,
1617,
1618,
1619,
1624,
1625,
1627,
1629,
1634,
1635,
1637,
1639,
1641,
1642,
1650,
1651,
1652,
1653,
1655,
1656,
1657,
1659,
1666,
1667,
1670,
1672,
1675,
1676,
1678,
1679,
1682,
1688,
1689,
1691,
1692,
1694,
```

```
1698,
1699,
1700,
1702,
1704,
1706,
1710,
1713,
1715,
1726,
1728,
1729,
1730,
1731,
1732,
1733,
1735,
1738,
1740,
1744,
1749,
1750,
1752,
1754,
1755,
1758,
1759,
1761,
1762,
1764,
1765,
1767,
1768,
1769,
1771,
1773,
1776,
1777,
1778,
1781,
1784,
1789,
1793,
1795,
1796,
1799,
1803,
1804,
1807,
```

```
1808,
1809,
1814,
1816,
1818,
1820,
1823,
1825,
1830,
1833,
1837,
1839,
1840,
1841,
1842,
1844,
1846,
1853,
1855,
1859,
1860,
1862,
1863,
1866,
1870,
1874,
1877,
1881,
1883,
1891,
1893,
1898,
1899,
1903,
1904,
1906,
1907,
1909,
1913,
1916,
1918,
1922,
1930,
1933,
1936,
1937,
1945,
1946,
1948,
```

```
1951,
1952,
1953,
1954,
1955,
1958,
1960,
1962,
1968,
1972,
1974,
1975,
1976,
1978,
1985,
1989,
1991,
1992,
1993,
1994,
1995,
1996,
1999,
2000,
2003,
2006,
2008,
2010,
2012,
2015,
2016,
2017,
2018,
2019,
2022,
2024,
2025,
2027,
2028,
2030,
2033,
2035,
2040,
2041,
2044,
2047,
2048,
2050,
2056,
```

2058,
2063,
2065,
2066,
2068,
2070,
2073,
2078,
2080,
2084,
2087,
2088,
2089,
2092,
2097,
2098,
2100,
2103,
2104,
2106,
2107,
2108,
2109,
2113,
2114,
2115,
2117,
2119,
2120,
2121,
2122,
2123,
2128,
2131,
2132,
2136,
2147,
2150,
2156,
2158,
2162,
2165,
2166,
2167,
2172,
2174,
2176,
2179,
2182,

```
 2183,
 2185,
 2186,
 2189,
 2190,
 2192,
 2193,
 2195,
 2196,
 2199,
 2201,
 2204]
```

```
len(num_of_kids)
```

```
929
```

```
num_of_teens = df.index[df['Num of teens'] >= 1].tolist()
num_of_teens
```

```
[1,
 5,
 6,
 9,
 12,
 15,
 17,
 21,
 22,
 23,
 25,
 30,
 31,
 33,
 34,
 36,
 38,
 40,
 45,
 46,
 50,
 54,
 56,
 57,
 58,
 61,
 62,
 63,
 64,
 66,
```

```
67,
79,
80,
81,
83,
87,
88,
91,
92,
93,
97,
98,
101,
103,
106,
109,
111,
118,
119,
120,
121,
123,
126,
127,
128,
130,
132,
134,
137,
139,
140,
141,
143,
145,
146,
147,
149,
153,
157,
158,
161,
162,
163,
165,
166,
168,
174,
175,
177,
```

```
181,
182,
186,
187,
189,
191,
194,
195,
196,
197,
199,
200,
203,
205,
209,
211,
213,
214,
217,
218,
221,
224,
227,
230,
231,
233,
235,
240,
242,
243,
244,
245,
247,
250,
251,
253,
258,
259,
260,
261,
265,
267,
269,
270,
273,
280,
281,
285,
286,
```

```
287,
294,
297,
299,
300,
301,
303,
304,
306,
307,
308,
312,
316,
319,
325,
326,
329,
331,
333,
337,
338,
339,
342,
344,
347,
348,
352,
354,
356,
357,
358,
361,
366,
369,
370,
371,
372,
377,
379,
383,
385,
387,
388,
390,
392,
394,
395,
397,
402,
```

```
408,
412,
415,
416,
418,
419,
420,
424,
432,
433,
434,
436,
437,
438,
441,
443,
445,
451,
453,
456,
458,
459,
460,
461,
466,
469,
470,
473,
476,
478,
479,
480,
481,
483,
484,
487,
489,
492,
494,
495,
497,
499,
503,
512,
513,
514,
515,
516,
518,
```

```
519,
521,
524,
527,
531,
532,
533,
535,
538,
542,
545,
546,
547,
548,
549,
550,
551,
552,
553,
554,
556,
558,
559,
560,
561,
562,
563,
564,
565,
568,
570,
571,
580,
583,
584,
586,
587,
589,
590,
591,
596,
598,
599,
600,
601,
602,
608,
609,
610,
```

```
614,
617,
619,
620,
621,
622,
624,
625,
627,
630,
631,
634,
635,
636,
637,
639,
640,
641,
643,
647,
650,
651,
652,
653,
654,
655,
657,
661,
664,
666,
670,
672,
674,
675,
677,
680,
682,
687,
688,
691,
698,
699,
700,
705,
708,
710,
711,
712,
713,
```

```
714,
716,
717,
719,
720,
721,
722,
731,
732,
736,
737,
739,
741,
745,
748,
749,
750,
752,
753,
754,
755,
756,
758,
759,
761,
766,
767,
769,
773,
774,
776,
777,
780,
781,
782,
785,
786,
787,
788,
789,
793,
795,
796,
798,
804,
805,
806,
807,
808,
```

```
810,
812,
814,
815,
818,
819,
820,
821,
827,
828,
830,
831,
833,
834,
835,
836,
838,
841,
845,
846,
848,
851,
853,
856,
861,
863,
866,
867,
868,
871,
875,
876,
877,
881,
884,
892,
898,
900,
903,
909,
912,
917,
918,
920,
921,
926,
927,
929,
931,
```

```
932,
934,
935,
936,
937,
938,
939,
940,
941,
943,
944,
945,
951,
953,
957,
958,
959,
960,
963,
965,
969,
974,
977,
979,
980,
982,
984,
985,
986,
988,
989,
990,
991,
993,
994,
1002,
1003,
1005,
1008,
1009,
1016,
1017,
1019,
1020,
1021,
1025,
1026,
1027,
1029,
```

```
1031,
1033,
1036,
1040,
1041,
1043,
1044,
1045,
1047,
1051,
1052,
1053,
1057,
1065,
1066,
1073,
1074,
1075,
1076,
1079,
1081,
1086,
1089,
1092,
1095,
1099,
1100,
1102,
1104,
1105,
1106,
1107,
1110,
1111,
1113,
1116,
1119,
1120,
1123,
1127,
1128,
1131,
1132,
1134,
1135,
1136,
1139,
1140,
1141,
1142,
```

```
1144,
1145,
1146,
1147,
1153,
1155,
1156,
1157,
1158,
1160,
1161,
1162,
1163,
1166,
1167,
1168,
1169,
1171,
1172,
1173,
1174,
1175,
1176,
1179,
1180,
1181,
1182,
1185,
1188,
1189,
1190,
1191,
1192,
1194,
1195,
1196,
1197,
1199,
1203,
1204,
1205,
1208,
1209,
1210,
1212,
1216,
1219,
1221,
1222,
```

```
1227,
1229,
1231,
1232,
1233,
1234,
1235,
1236,
1237,
1238,
1240,
1243,
1244,
1246,
1248,
1251,
1253,
1255,
1257,
1259,
1262,
1266,
1267,
1273,
1274,
1275,
1277,
1279,
1282,
1284,
1285,
1288,
1289,
1291,
1292,
1294,
1295,
1296,
1297,
1300,
1301,
1304,
1310,
1311,
1314,
1317,
1319,
1320,
1321,
```

```
1325,
1327,
1332,
1334,
1338,
1339,
1341,
1343,
1344,
1345,
1348,
1349,
1350,
1351,
1352,
1353,
1354,
1355,
1357,
1358,
1359,
1362,
1363,
1365,
1368,
1370,
1371,
1373,
1374,
1375,
1376,
1377,
1378,
1380,
1381,
1382,
1383,
1384,
1385,
1387,
1393,
1394,
1395,
1397,
1398,
1403,
1408,
1409,
1410,
```

1411,
1415,
1416,
1422,
1424,
1431,
1432,
1435,
1436,
1437,
1440,
1441,
1443,
1444,
1445,
1447,
1448,
1449,
1450,
1452,
1454,
1455,
1456,
1459,
1460,
1462,
1463,
1465,
1466,
1472,
1473,
1475,
1478,
1481,
1482,
1484,
1486,
1487,
1491,
1500,
1502,
1504,
1505,
1506,
1507,
1508,
1513,
1516,
1518,

```
1519,
1523,
1524,
1525,
1527,
1529,
1530,
1531,
1533,
1534,
1536,
1538,
1540,
1541,
1542,
1543,
1544,
1546,
1550,
1554,
1555,
1558,
1560,
1561,
1565,
1568,
1569,
1570,
1571,
1575,
1578,
1579,
1580,
1581,
1583,
1584,
1589,
1593,
1594,
1595,
1598,
1599,
1600,
1601,
1605,
1607,
1609,
1610,
1611,
```

```
1613,
1615,
1616,
1617,
1621,
1624,
1627,
1628,
1630,
1631,
1632,
1634,
1636,
1637,
1640,
1641,
1643,
1648,
1655,
1656,
1658,
1659,
1661,
1662,
1664,
1669,
1670,
1672,
1673,
1674,
1675,
1677,
1680,
1681,
1683,
1685,
1686,
1690,
1691,
1692,
1693,
1700,
1701,
1702,
1703,
1704,
1706,
1708,
1709,
```

```
1710,
1712,
1714,
1718,
1722,
1724,
1725,
1727,
1728,
1729,
1730,
1732,
1734,
1736,
1737,
1739,
1743,
1748,
1749,
1755,
1757,
1758,
1759,
1760,
1762,
1763,
1764,
1765,
1768,
1769,
1770,
1772,
1774,
1775,
1776,
1777,
1778,
1779,
1780,
1781,
1784,
1785,
1788,
1790,
1792,
1793,
1794,
1795,
1796,
```

```
1797,
1800,
1805,
1806,
1808,
1809,
1811,
1812,
1813,
1818,
1820,
1822,
1829,
1830,
1834,
1836,
1839,
1840,
1845,
1846,
1847,
1848,
1852,
1856,
1857,
1860,
1862,
1867,
1869,
1873,
1874,
1878,
1880,
1885,
1887,
1888,
1889,
1893,
1894,
1895,
1898,
1900,
1903,
1904,
1905,
1908,
1909,
1910,
1911,
```

```
1912,
1915,
1921,
1922,
1923,
1925,
1931,
1933,
1934,
1936,
1937,
1938,
1940,
1941,
1944,
1946,
1947,
1948,
1949,
1951,
1957,
1958,
1959,
1960,
1963,
1965,
1967,
1968,
1971,
1973,
1978,
1979,
1980,
1981,
1982,
1983,
1989,
1991,
1992,
1993,
1996,
1997,
1998,
2000,
2002,
2004,
2005,
2007,
2009,
```

```
 2010,
 2011,
 2015,
 2020,
 2024,
 2025,
 2026,
 2027,
 2028,
 2031,
 2032,
 2035,
 2036,
 2037,
 2039,
 2042,
 2043,
 2051,
 2053,
 2055,
 2057,
 2058,
 2060,
 2062,
 2063,
 2065,
 2067,
 2068,
 2069,
 2071,
 2074,
 2075,
 2076,
 2078,
 2080,
 2081,
 2082,
 2083,
 ...]
```

```
len(num_of_teens)
```

```
1066
```

```
df.head(10)
```

{"type":"dataframe","variable_name":"df"}

```
df['CustomerID'] = df.index + 1
cols = ['CustomerID'] + [col for col in df.columns if col !=
```

```
'CustomerID']
df = df[cols]

df.head(10)
```


```
{"type":"dataframe","variable_name":"df"}
```


```
selected = ['Recency', 'MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds','MntTotal',
'MntRegularProds', 'NumDealsPurchases',
            'NumWebPurchases', 'NumCatalogPurchases',
'NumStorePurchases']
data=df[selected]

data.head(10)
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 2205,\n  \"fields\":
[\n    {\n      \"column\": \"Recency\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 28,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 100,\n
\"samples\": [\n          45,\n          48,\n          10\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntWines\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 337,\n        \"min\": 0,\n
\"max\": 1493,\n        \"num_unique_values\": 775,\n
\"samples\": [\n          840,\n          581,\n          465\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntFruits\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
39,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 158,\n        \"samples\": [\n          62,\n
81,\n          57\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntMeatProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 217,\n        \"min\": 0,\n
\"max\": 1725,\n        \"num_unique_values\": 551,\n
\"samples\": [\n          655,\n          21,\n          137\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntFishProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 54,\n        \"min\": 0,\n
\"max\": 259,\n        \"num_unique_values\": 182,\n
\"samples\": [\n          20,\n          75,\n          94\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntSweetProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 41,\n        \"min\": 0,\n
\"max\": 262,\n        \"num_unique_values\": 176,\n
\"samples\": [\n          2,\n          35,\n          55\n        ],\
```

```
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntGoldProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
51,\n        \"min\": 0,\n        \"max\": 321,\n
\"num_unique_values\": 212,\n        \"samples\": [\n          102,\n
166,\n          106\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntTotal\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 575,\n        \"min\": 4,\n
\"max\": 2491,\n        \"num_unique_values\": 897,\n
\"samples\": [\n          155,\n          1555,\n          656\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntRegularProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
553,\n        \"min\": -283,\n        \"max\": 2458,\n
\"num_unique_values\": 974,\n        \"samples\": [\n          749,\n
448,\n          919\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumDealsPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 15,\n        \"num_unique_values\": 15,\n        \"samples\":
[\n          0,\n          10,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumWebPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2,\n        \"min\": 0,\n        \"max\": 27,\n
\"num_unique_values\": 15,\n        \"samples\": [\n          0,\n
9,\n          8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumCatalogPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 0,\n
\"max\": 28,\n        \"num_unique_values\": 13,\n        \"samples\":
[\n          7,\n          5,\n          10\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumStorePurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 13,\n
\"num_unique_values\": 14,\n        \"samples\": [\n          12,\n
13,\n          4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```


```
data.shape
```

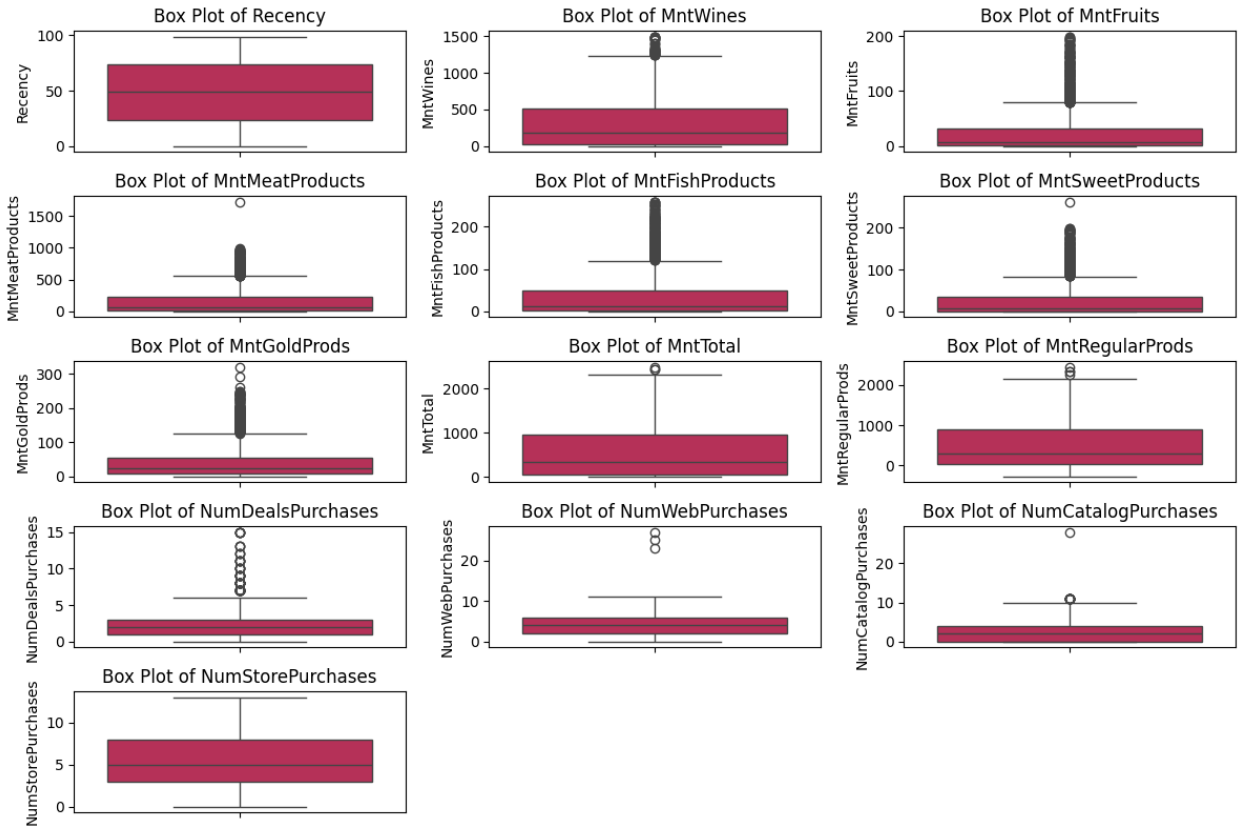
```
(2205, 13)
```

```
plt.figure(figsize=(12, 8))  # Set the figure size
num_rows = 5  # Increased num_rows to 5 to accommodate all 13 plots
num_cols = 3
```

```python
for i, column in enumerate(selected, 1):
    plt.subplot(num_rows, num_cols, i)  # Update subplot grid
    sns.boxplot(y=data[column],palette='rocket')
    plt.title(f'Box Plot of {column}')

plt.tight_layout()  # Adjust subplot parameters for a tight layout
plt.show()
```

```
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
<ipython-input-22-b97ef03e3e57>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y=data[column],palette='rocket')
```




```
data.describe()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 8,\n  \"fields\": [\n    {\n      \"column\": \"Recency\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 763.8319765217124,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          49.00907029478458,\n          49.0,\n          2205.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"MntWines\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 792.768927193356,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          306.16462585034014,\n          178.0,\n          2205.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"MntFruits\",\n        \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 766.7695599917939,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          26.4031746031746,\n          8.0,\n          2205.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"MntMeatProducts\",\n        \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 869.5274024418567,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
```

```
165.31201814058957,\n          68.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MntFishProducts\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
763.1798477256916,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
37.756462585034015,\n          12.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MntSweetProducts\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
765.6360893796907,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
27.128344671201813,\n          8.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MntGoldProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
761.0423085548417,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
44.05714285714286,\n          25.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MntTotal\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
947.3683264287661,\n        \"min\": 4.0,\n        \"max\": 2491.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
562.7646258503402,\n          343.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MntRegularProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
990.7051766526779,\n        \"min\": -283.0,\n        \"max\":
2458.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
518.7074829931972,\n          288.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumDealsPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
778.3266329327768,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
2.3183673469387753,\n          2.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumWebPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
777.3168399571634,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
4.100680272108844,\n          4.0,\n          2205.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumCatalogPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
777.647801525299,\n        \"min\": 0.0,\n        \"max\": 2205.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          2205.0,\n
2.6453514739229025,\n          4.0\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NumStorePurchases\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
777.6722258907135,\n   \"min\": 0.0,\n   \"max\": 2205.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
5.823582766439909,\n   5.0,\n   2205.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

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

{"summary":"{\n \"name\": \"data\",\n \"rows\": 2205,\n \"fields\":
[\n   {\n   \"column\": \"Recency\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 28,\n   \"min\": 0,\n
\"max\": 99,\n   \"num_unique_values\": 100,\n
\"samples\": [\n   45,\n   48,\n   10\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"MntWines\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 337,\n   \"min\": 0,\n
\"max\": 1493,\n   \"num_unique_values\": 775,\n
\"samples\": [\n   840,\n   581,\n   465\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"MntFruits\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
39,\n   \"min\": 0,\n   \"max\": 199,\n
\"num_unique_values\": 158,\n   \"samples\": [\n   62,\n
81,\n   57\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"MntMeatProducts\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 217,\n   \"min\": 0,\n
\"max\": 1725,\n   \"num_unique_values\": 551,\n
\"samples\": [\n   655,\n   21,\n   137\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"MntFishProducts\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 54,\n   \"min\": 0,\n
\"max\": 259,\n   \"num_unique_values\": 182,\n
\"samples\": [\n   20,\n   75,\n   94\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"MntSweetProducts\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 41,\n   \"min\": 0,\n
\"max\": 262,\n   \"num_unique_values\": 176,\n
\"samples\": [\n   2,\n   35,\n   55\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"MntGoldProds\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
51,\n   \"min\": 0,\n   \"max\": 321,\n
\"num_unique_values\": 212,\n   \"samples\": [\n   102,\n

```
166,\n          106\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntTotal\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 575,\n        \"min\": 4,\n
\"max\": 2491,\n        \"num_unique_values\": 897,\n
\"samples\": [\n          155,\n          1555,\n          656\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntRegularProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
553,\n        \"min\": -283,\n        \"max\": 2458,\n
\"num_unique_values\": 974,\n        \"samples\": [\n          749,\n
448,\n          919\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumDealsPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 15,\n        \"num_unique_values\": 15,\n        \"samples\":
[\n          0,\n          10,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumWebPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2,\n        \"min\": 0,\n        \"max\": 27,\n
\"num_unique_values\": 15,\n        \"samples\": [\n          0,\n
9,\n          8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumCatalogPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 0,\n
\"max\": 28,\n        \"num_unique_values\": 13,\n        \"samples\":
[\n          7,\n          5,\n          10\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumStorePurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 13,\n
\"num_unique_values\": 14,\n        \"samples\": [\n          12,\n
13,\n          4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```


```
frequency = data[['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].count()

frequency_data = frequency.reset_index()
frequency_data.columns = ['Product Category', 'Frequency of
Purchases']

frequency_data
```


```
{"summary":"{\n  \"name\": \"frequency_data\",\n  \"rows\": 6,\n
\"fields\": [\n    {\n      \"column\": \"Product Category\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n
```

```
\"MntWines\",\n          \"MntFruits\",\n          \"MntGoldProds\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Frequency of Purchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 2205,\n        \"max\": 2205,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          2205\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"frequency_data"}
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

sc_df=sc.fit_transform(data)

sc_df
```

```
array([[ 0.31083003,  0.97456584,  1.54861372, ...,  1.42477184,
         2.62852614, -0.5626499 ],
       [-0.38060021, -0.8747762 , -0.63866448, ..., -1.13295709,
        -0.58804297, -1.17973173],
       [-0.79545835,  0.3551548 ,  0.5681097 , ...,  1.42477184,
        -0.2306464 ,  1.2885956 ],
       ...,
       [ 1.45168993,  1.78365298,  0.54296857, ..., -0.76756724,
         0.12675017,  2.21421835],
       [-1.41774556,  0.36108218,  0.09042825, ...,  0.69399214,
         0.8415433 ,  1.2885956 ],
       [-0.31145718, -0.6584269 , -0.58838223, ..., -0.4021774 ,
        -0.58804297, -0.5626499 ]])
```

```
from sklearn.decomposition import PCA

pca=PCA(n_components=3)
pca
```

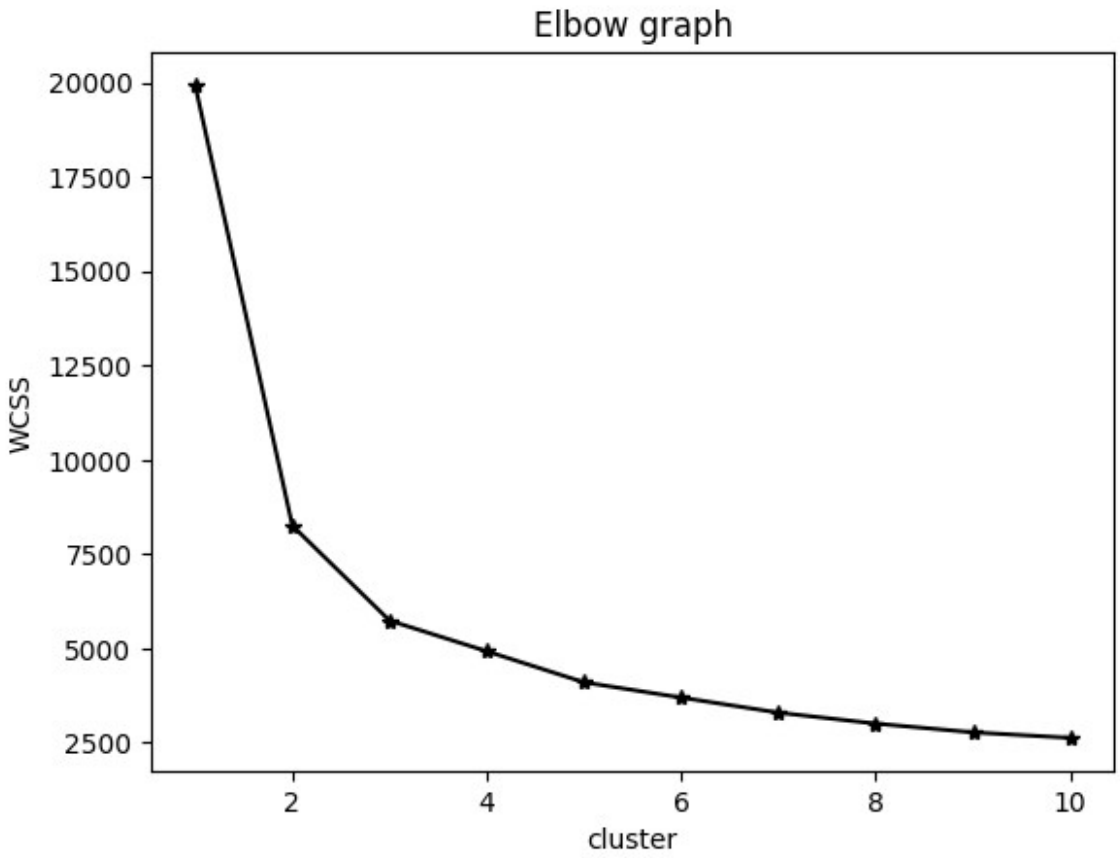
```
PCA(n_components=3)
```

```
p_df=pca.fit_transform(sc_df)

p_df
```

```
array([[ 4.82741325, -0.43851246,  0.23116695],
       [-2.68372663, -0.61513288, -0.32448308],
       [ 1.39691763,  0.15630771, -0.95850715],
       ...,
       [ 1.99136827, -0.18868646,  1.54201403],
       [ 1.59304685,  0.35317397, -1.44944077],
       [-1.96342372,  0.25825981, -0.27628074]])
```

```
from sklearn.cluster import KMeans
```

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(p_df)
    # Append inertia value to wcss within the loop
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss, marker='*', c='black')
plt.title('Elbow graph')
plt.xlabel('cluster')
plt.ylabel('WCSS')
plt.show()
```



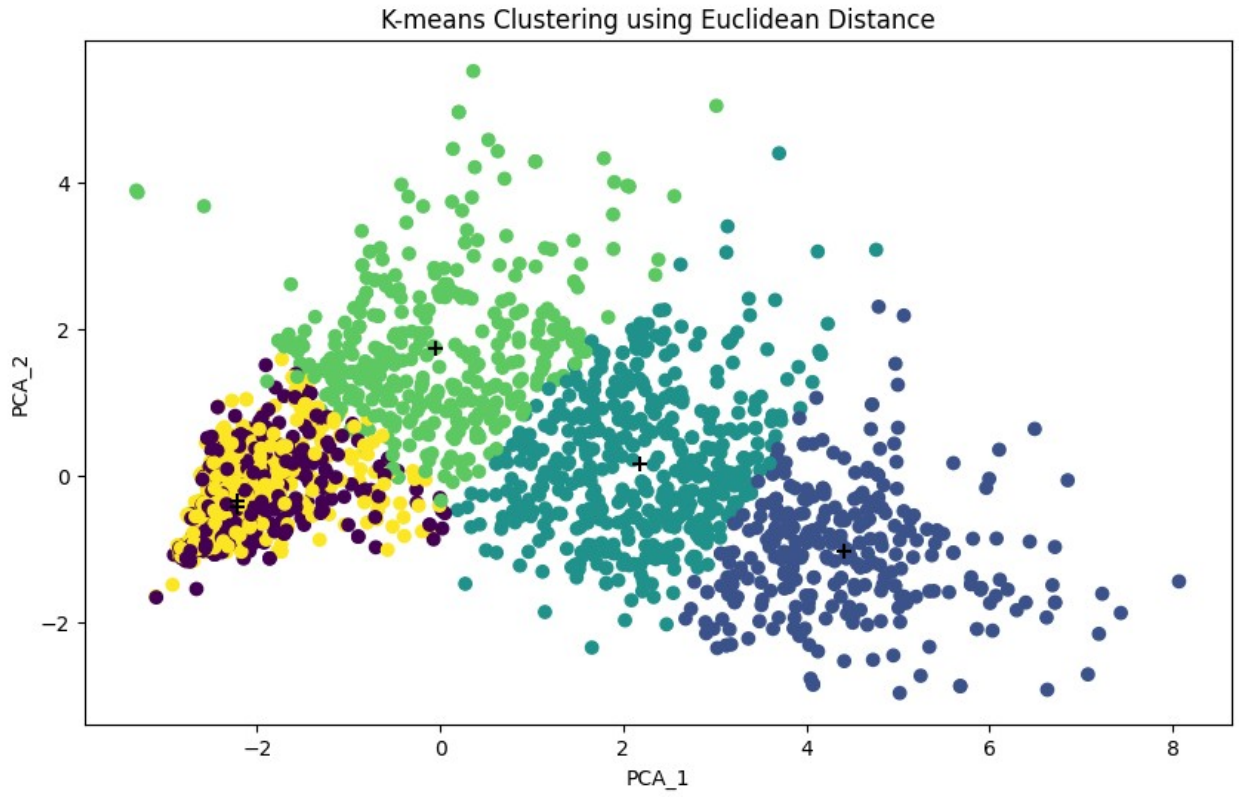

```
km1=KMeans(n_clusters=5,init='random',random_state=0)
km1.fit(p_df)
```

```
KMeans(init='random', n_clusters=5, random_state=0)
```

```
cluster_centers_km1 = km1.cluster_centers_
cluster_centers_km1
```

```
array([[-2.21852438, -0.43214454,  0.90089582],
       [ 4.41940644, -1.03584495,  0.07016189],
```

```
       [ 2.18940393,  0.16810662, -0.09184452],
       [-0.04657575,  1.7409655 ,  0.03601567],
       [-2.20863557, -0.35479209, -0.85278032]])

label1 = pd.Series(km1.labels_)

label1

0       1
1       4
2       2
3       4
4       3
       ..
2200    2
2201    3
2202    2
2203    2
2204    4
Length: 2205, dtype: int32

label1.value_counts()

4    540
0    520
2    467
3    375
1    303
Name: count, dtype: int64

plt.figure(figsize=(10, 6))
plt.scatter(p_df[:,0], p_df[:,1], c=label1, cmap='viridis')
plt.scatter(cluster_centers_km1[:, 0], cluster_centers_km1[:, 1],
s=50, c='black', marker='+', label='Cluster Centers')
plt.xlabel('PCA_1')
plt.ylabel('PCA_2')
plt.title('K-means Clustering using Euclidean Distance')
plt.show()
```




```
from sklearn.metrics import silhouette_score, calinski_harabasz_score

silhouette_man = silhouette_score(p_df, label1)
ch_man = calinski_harabasz_score(p_df, label1)

print("Silhouette Score:", silhouette_man)
print("Calinski-Harabasz Index:", ch_man)
```

```
Silhouette Score: 0.320231409107203
Calinski-Harabasz Index: 2122.1807642516346
```

```
data
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 2205,\n  \"fields\":
[\n    {\n      \"column\": \"Recency\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 28,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 100,\n
\"samples\": [\n          45,\n          48,\n          10\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntWines\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 337,\n        \"min\": 0,\n
\"max\": 1493,\n        \"num_unique_values\": 775,\n
\"samples\": [\n          840,\n          581,\n          465\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntFruits\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
39,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 158,\n        \"samples\": [\n          62,\n
81,\n          57\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntMeatProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 217,\n        \"min\": 0,\n
\"max\": 1725,\n        \"num_unique_values\": 551,\n
\"samples\": [\n          655,\n          21,\n          137\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntFishProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 54,\n        \"min\": 0,\n
\"max\": 259,\n        \"num_unique_values\": 182,\n
\"samples\": [\n          20,\n          75,\n          94\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntSweetProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 41,\n        \"min\": 0,\n
\"max\": 262,\n        \"num_unique_values\": 176,\n
\"samples\": [\n          2,\n          35,\n          55\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntGoldProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
51,\n        \"min\": 0,\n        \"max\": 321,\n
\"num_unique_values\": 212,\n        \"samples\": [\n          102,\n
166,\n          106\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntTotal\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 575,\n        \"min\": 4,\n
\"max\": 2491,\n        \"num_unique_values\": 897,\n
\"samples\": [\n          155,\n          1555,\n          656\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntRegularProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
553,\n        \"min\": -283,\n        \"max\": 2458,\n
\"num_unique_values\": 974,\n        \"samples\": [\n          749,\n
448,\n          919\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumDealsPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 15,\n        \"num_unique_values\": 15,\n        \"samples\":
[\n          0,\n          10,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumWebPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2,\n        \"min\": 0,\n        \"max\": 27,\n
\"num_unique_values\": 15,\n        \"samples\": [\n          0,\n
9,\n          8\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"NumCatalogPurchases\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 2,\n   \"min\": 0,\n
\"max\": 28,\n   \"num_unique_values\": 13,\n   \"samples\":
[\n   7,\n   5,\n   10\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NumStorePurchases\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3,\n   \"min\": 0,\n   \"max\": 13,\n
\"num_unique_values\": 14,\n   \"samples\": [\n   12,\n
13,\n   4\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data"}

```
labels1=pd.DataFrame(label1)
labels1.columns=['label']
labels1
```

{"summary":"{\n \"name\": \"labels1\",\n \"rows\": 2205,\n
\"fields\": [\n   {\n   \"column\": \"label\",\n
\"properties\": {\n   \"dtype\": \"int32\",\n
\"num_unique_values\": 5,\n   \"samples\": [\n   4,\n
0,\n   2\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"labels1"}

```
l1_df = pd.concat([labels1, data],axis=True)
```

```
l1_df
```

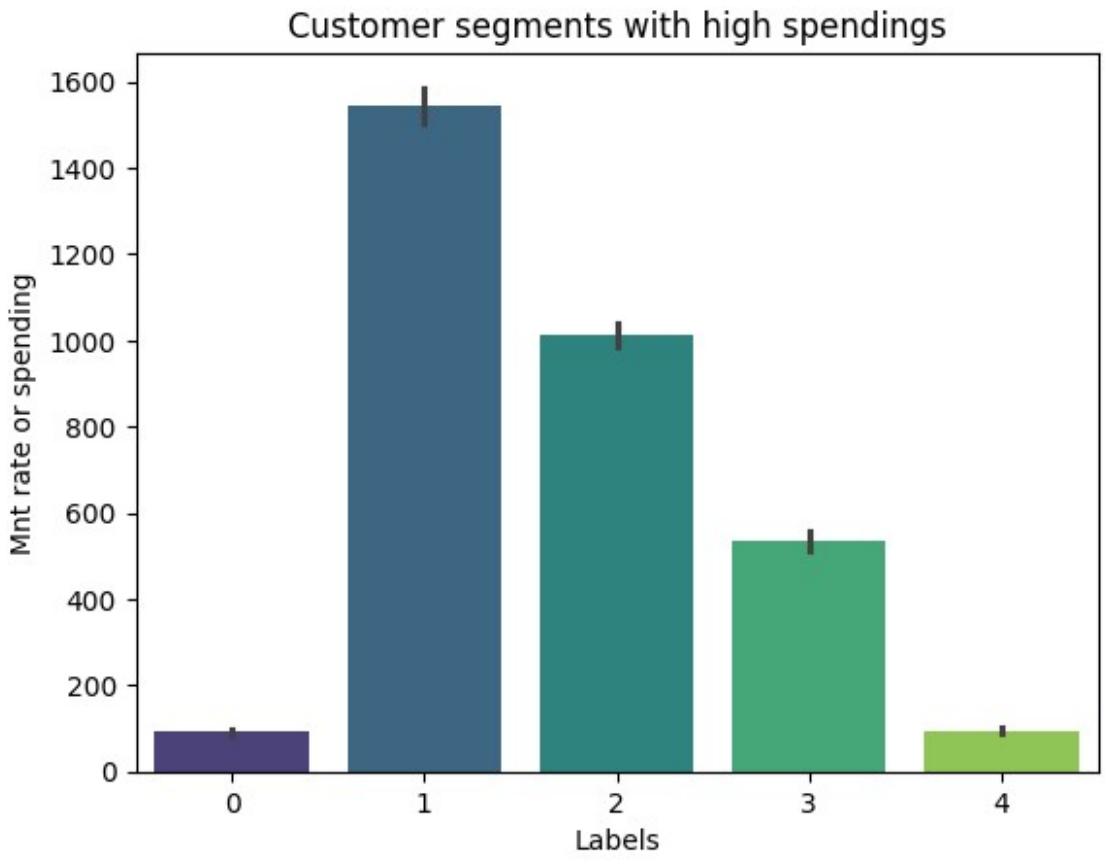
{"summary":"{\n \"name\": \"l1_df\",\n \"rows\": 2205,\n
\"fields\": [\n   {\n   \"column\": \"label\",\n
\"properties\": {\n   \"dtype\": \"int32\",\n
\"num_unique_values\": 5,\n   \"samples\": [\n   4,\n
0,\n   2\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Recency\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 28,\n   \"min\": 0,\n   \"max\": 99,\n
\"num_unique_values\": 100,\n   \"samples\": [\n   45,\n
48,\n   10\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"MntWines\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 337,\n   \"min\": 0,\n
\"max\": 1493,\n   \"num_unique_values\": 775,\n
\"samples\": [\n   840,\n   581,\n   465\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"MntFruits\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
39,\n   \"min\": 0,\n   \"max\": 199,\n
\"num_unique_values\": 158,\n   \"samples\": [\n   62,\n

```
81,\n          57\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntMeatProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 217,\n        \"min\": 0,\n
\"max\": 1725,\n        \"num_unique_values\": 551,\n
\"samples\": [\n          655,\n          21,\n          137\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntFishProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 54,\n        \"min\": 0,\n
\"max\": 259,\n        \"num_unique_values\": 182,\n
\"samples\": [\n          20,\n          75,\n          94\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntSweetProducts\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 41,\n        \"min\": 0,\n
\"max\": 262,\n        \"num_unique_values\": 176,\n
\"samples\": [\n          2,\n          35,\n          55\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntGoldProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
51,\n        \"min\": 0,\n        \"max\": 321,\n
\"num_unique_values\": 212,\n        \"samples\": [\n          102,\n
166,\n          106\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MntTotal\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 575,\n        \"min\": 4,\n
\"max\": 2491,\n        \"num_unique_values\": 897,\n
\"samples\": [\n          155,\n          1555,\n          656\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"MntRegularProds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
553,\n        \"min\": -283,\n        \"max\": 2458,\n
\"num_unique_values\": 974,\n        \"samples\": [\n          749,\n
448,\n          919\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumDealsPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 15,\n        \"num_unique_values\": 15,\n        \"samples\":
[\n          0,\n          10,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NumWebPurchases\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2,\n        \"min\": 0,\n        \"max\": 27,\n
\"num_unique_values\": 15,\n        \"samples\": [\n          0,\n
9,\n          8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NumCatalogPurchases\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 0,\n
```

\"max\": 28,\n   \"num_unique_values\": 13,\n   \"samples\": [\n   7,\n   5,\n   10\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"NumStorePurchases\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3,\n   \"min\": 0,\n   \"max\": 13,\n   \"num_unique_values\": 14,\n   \"samples\": [\n   12,\n   13,\n   4\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"l1_df"}

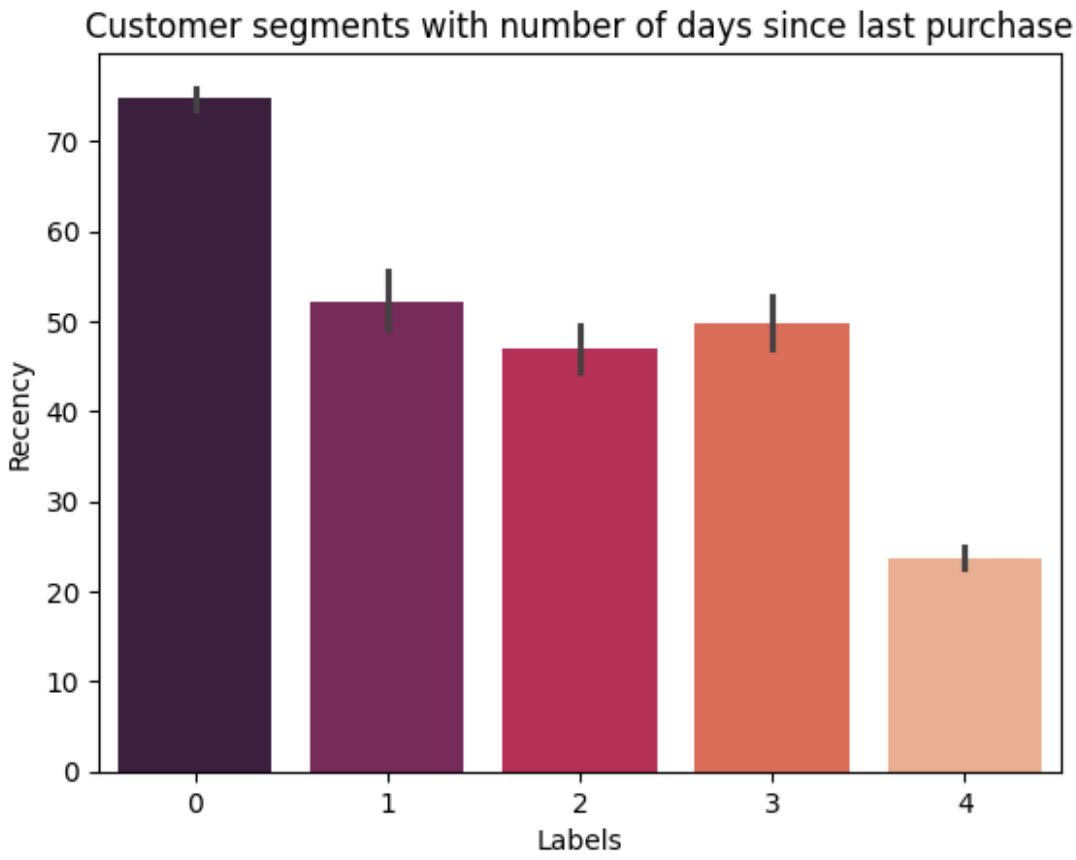
```
sns.barplot(x=l1_df['label'], y=l1_df['MntTotal'], palette='viridis')
plt.title('Customer segments with high spendings')
plt.xlabel('Labels')
plt.ylabel('Mnt rate or spending')
plt.show()
```

```
<ipython-input-47-ddabbf3f6367>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=l1_df['label'], y=l1_df['MntTotal'], palette='viridis')
```

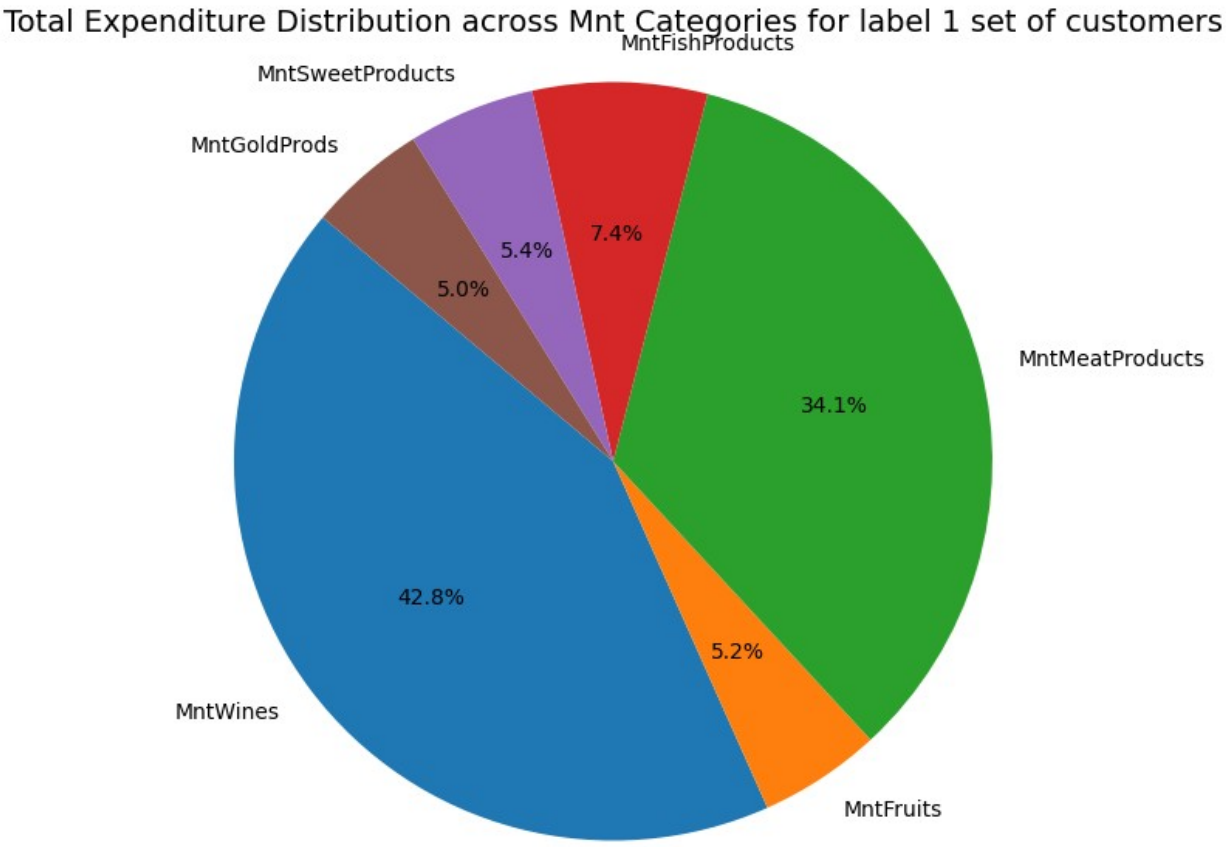



```
sns.barplot(x=l1_df['label'], y=l1_df['Recency'], palette='rocket')
plt.title('Customer segments with number of days since last purchase')
plt.xlabel('Labels')
plt.ylabel('Recency')
plt.show()
```

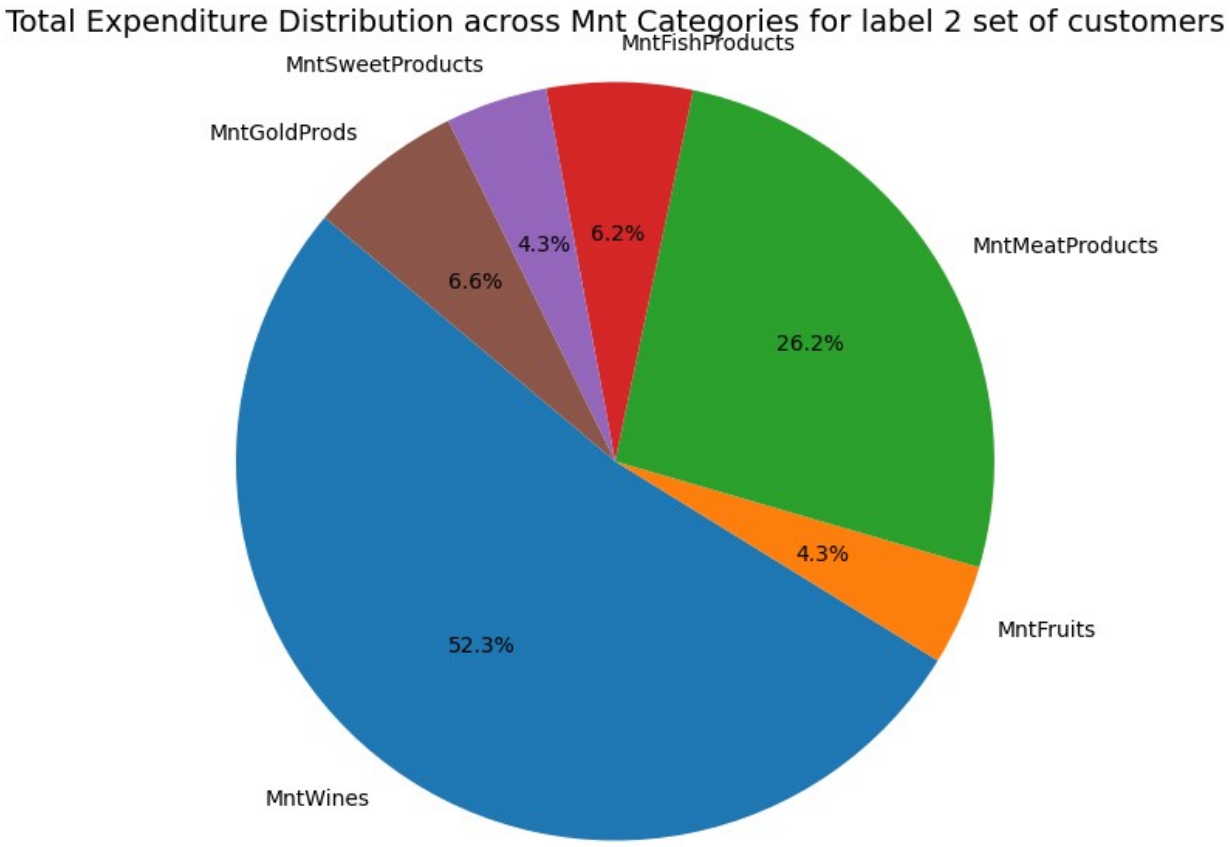
```
<ipython-input-48-b9cff904f928>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=l1_df['label'], y=l1_df['Recency'], palette='rocket')
```



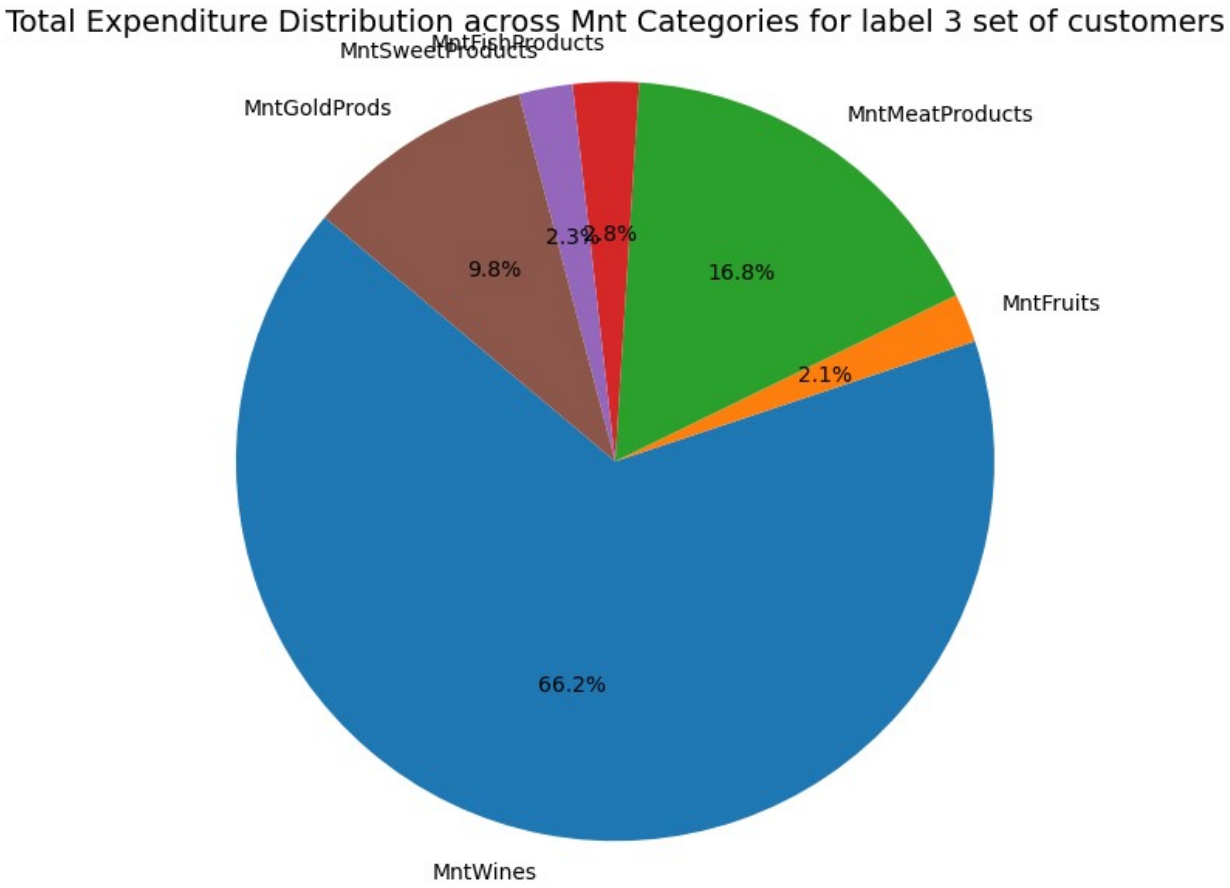

```
l1_data=l1_df[l1_df['label']==1]
mnt_terms = ['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
total_expenditure = l1_data[mnt_terms].sum()

plt.figure(figsize=(10, 7))
plt.pie(total_expenditure, labels=total_expenditure.index,
autopct='%1.1f%%', startangle=140)
plt.title('Total Expenditure Distribution across Mnt Categories for
label 1 set of customers', fontsize=14)
plt.axis('equal')
plt.show()
```



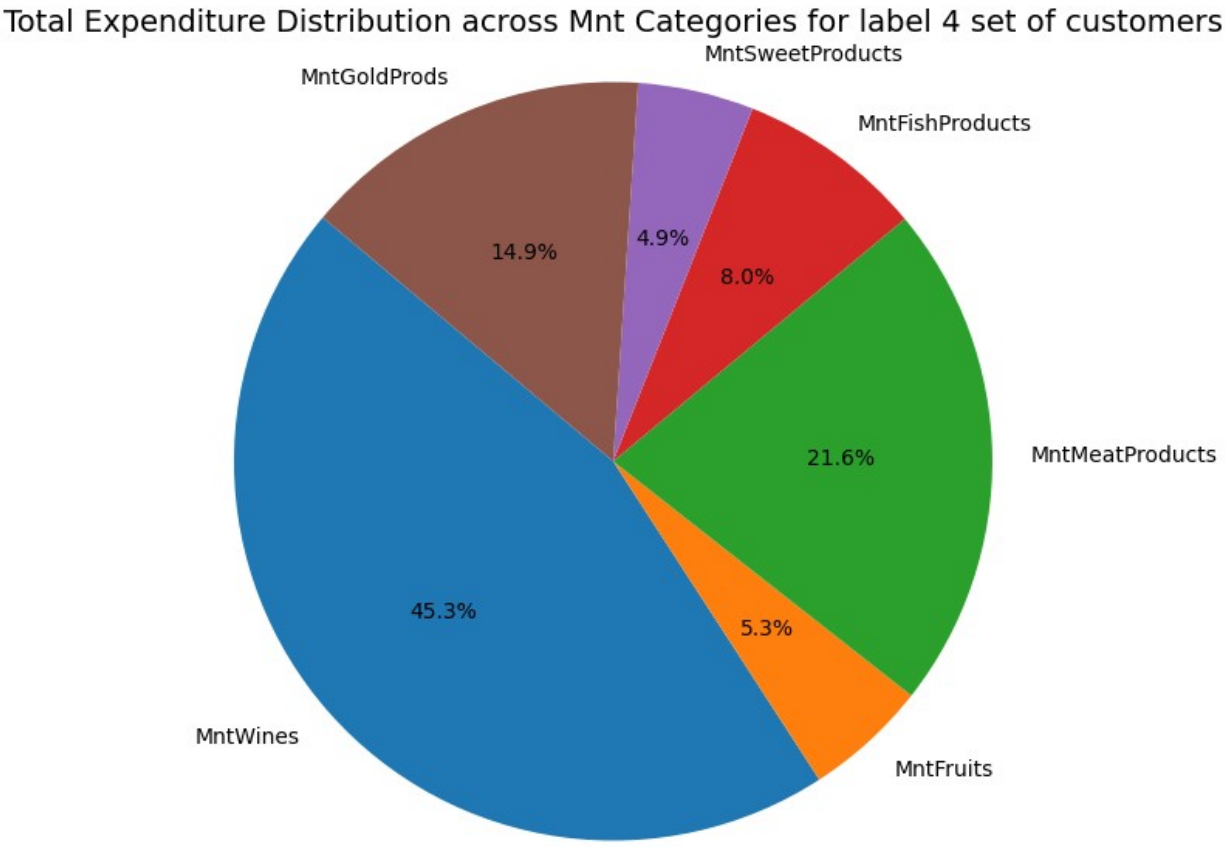

```
l2_data=l1_df[l1_df['label']==2]
mnt_terms = ['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
total_expenditure = l2_data[mnt_terms].sum()

plt.figure(figsize=(10, 7))
plt.pie(total_expenditure, labels=total_expenditure.index,
autopct='%1.1f%%', startangle=140)
plt.title('Total Expenditure Distribution across Mnt Categories for
label 2 set of customers', fontsize=14)
plt.axis('equal')
plt.show()
```



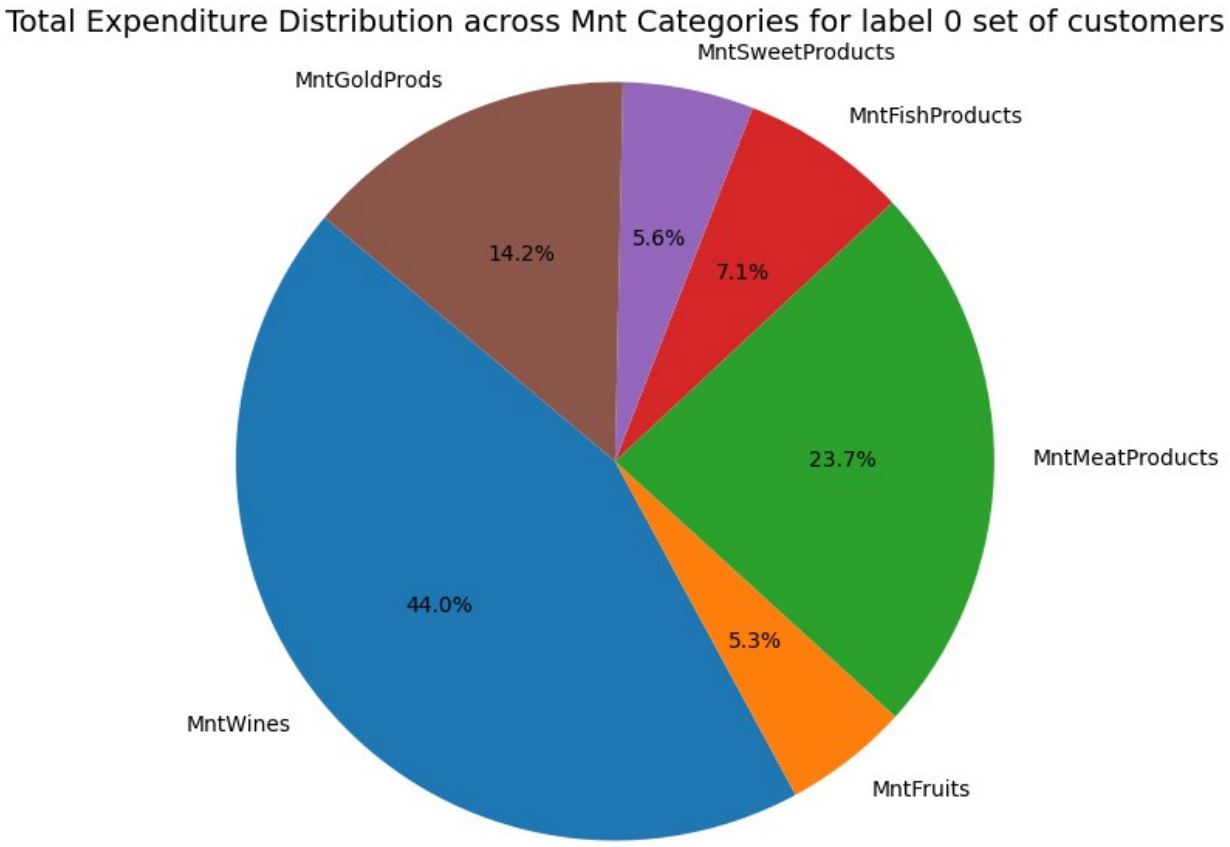

```
l3_data=l1_df[l1_df['label']==3]
mnt_terms = ['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
total_expenditure = l3_data[mnt_terms].sum()

plt.figure(figsize=(10, 7))
plt.pie(total_expenditure, labels=total_expenditure.index,
autopct='%1.1f%%', startangle=140)
plt.title('Total Expenditure Distribution across Mnt Categories for
label 3 set of customers', fontsize=14)
plt.axis('equal')
plt.show()
```




```
l4_data=l1_df[l1_df['label']==4]
mnt_terms = ['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
total_expenditure = l4_data[mnt_terms].sum()

plt.figure(figsize=(10, 7))
plt.pie(total_expenditure, labels=total_expenditure.index,
autopct='%1.1f%%', startangle=140)
plt.title('Total Expenditure Distribution across Mnt Categories for
label 4 set of customers', fontsize=14)
plt.axis('equal')
plt.show()
```

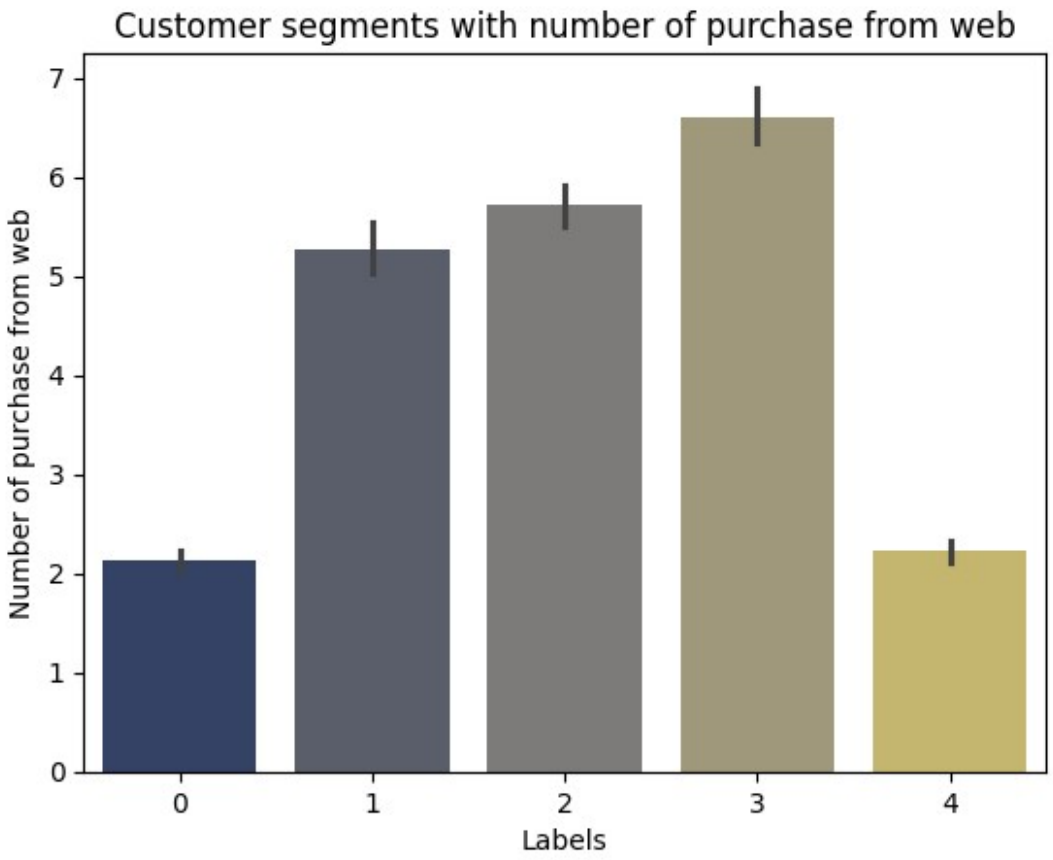



```
l0_data=l1_df[l1_df['label']==0]
mnt_terms = ['MntWines', 'MntFruits', 'MntMeatProducts',
'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
total_expenditure = l0_data[mnt_terms].sum()

plt.figure(figsize=(10, 7))
plt.pie(total_expenditure, labels=total_expenditure.index,
autopct='%1.1f%%', startangle=140)
plt.title('Total Expenditure Distribution across Mnt Categories for
label 0 set of customers', fontsize=14)
plt.axis('equal')
plt.show()
```



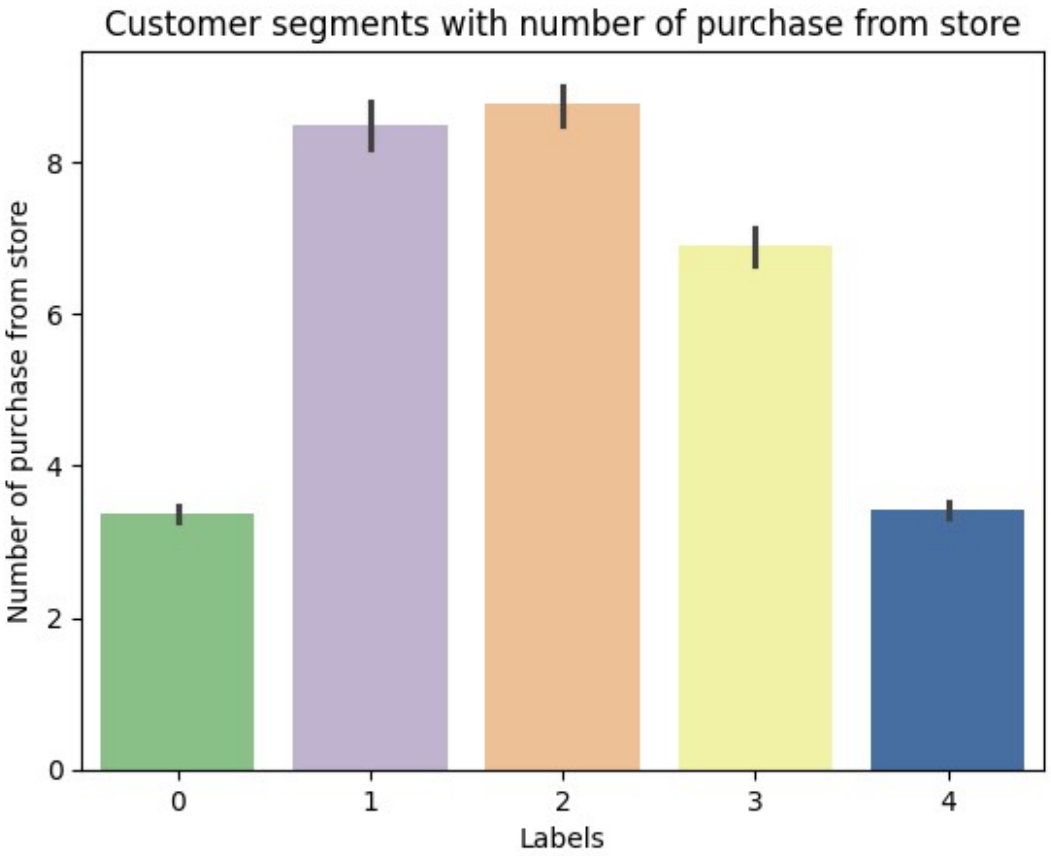

```
sns.barplot(x=l1_df['label'], y=l1_df['NumWebPurchases'],
palette='cividis')
plt.title('Customer segments with number of purchase from web')
plt.xlabel('Labels')
plt.ylabel('Number of purchase from web')
plt.show()
```

```
<ipython-input-60-1c6241ec4c9e>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=l1_df['label'], y=l1_df['NumWebPurchases'],
palette='cividis')
```




```
sns.barplot(x=l1_df['label'], y=l1_df['NumStorePurchases'],
palette='Accent')
plt.title('Customer segments with number of purchase from store')
plt.xlabel('Labels')
plt.ylabel('Number of purchase from store')
plt.show()
```

```
<ipython-input-62-caabf348b552>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=l1_df['label'], y=l1_df['NumStorePurchases'],
palette='Accent')
```



Insights and Recommendations:

- Label 1 set of customer have highest spending where as 0/4 has lowest and 2/3 have average spending.
- So customer with label 0 have high Recency also customer with label 4 have less recency that means there are more customer who purchased some stuff recently. So store should focus on attracting cuutomer with 0 label more.
- So most of the set of customer are focusing on wine and meat product. So we need to focus on different product to increae sell.
- So customer in label 1/2/4/0 are also interested in fish product so we can provide them with some offer on fish to just increase more sell on fish product too.
- If we took closer look at Gold product then customer with label 4 and 0 are interseted in gold than other so we can suppose that they are investor and are more into investing.
- Now sweet and fruit product are have very low purchase rate. So people are more toowards gold, wine product, fish and meat product. so suggesion is people need

some offer on sweet product and fruits. So its necessary to introduce some offer discount over such product.