Tugas EDA dan Klasifikasi (25 Oktober 2023)

Kelompok 2

1. Fauziyah Zayyan Nur
2. Muhamad Irfan
3. Muhammad Hanif Abdillah
4. Raffi Ardhi Naufal
5. Ridwan Abdul Jamil

Latar belakang

Fraud di kartu kredit adalah penggunaan kartu kredit oleh orang yang tidak berhak. Ini masa kartu kredit masih belum wajib menggunakan chip dan pin ya. Jadi misalnya kartu kredit kita dicuri/dicopet, maka dapat langsung digunakan. Ada juga tempat yang menggandakan kartu kredit (gampang sebelum menggunakan chip) yang clone-nya kemudian digunakan untuk belanja. Berbeda dengan kartu debit yang langsung autodebet, pembayaran kartu kredit "ditalangi" dulu oleh bank, sehingga bank sangat berkepentingan untuk mencegah fraud (dia yang rugi).

In [1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
from datetime import datetime
```

In [2]:
```python
# Load data
data = pd.read_csv("fraud_train.txt")
```

In [3]:
```python
data.head(10)
```

Out[3]:

	X	id_tanggal_transaksi_awal	tanggal_transaksi_awal	tipe_kartu	id_merchant	nama_merchant	tipe_mesin	tipe_transaksi	nama_transaksi	nilai_transaksi	...	nama_channel
0	4676	2457646	2457726	111	-2	1798	2605127	26	10	2200000.0	...	1
1	788	2457419	2457507	111	-2	1798	-3	156	12	2500000.0	...	5
2	1520	2457521	2457612	2	-2	1798	-3	156	12	1200000.0	...	5
3	9346	2457659	2457746	103	75336	249	2806174	58	6	320000.0	...	2
4	2914	2457311	2457385	0	-2	1798	2334932	26	10	150000.0	...	1
5	6947	2457413	2457453	111	-2	1798	16834	385	11	100000.0	...	1
6	14131	2457615	2457640	93	-2	1798	1850104	385	11	2000000.0	...	1
7	4811	2457429	2457463	0	-2	1798	2435842	385	11	150000.0	...	1
8	1458	2457371	2457453	93	-2	1798	1370214	159	19	50000.0	...	1
9	4123	2457320	2457365	93	-2	1798	1942704	26	10	1250000.0	...	1

10 rows × 28 columns

Karakteristik Dataset:

Dataset memiliki 13,125 entri (baris) dengan 28 kolom (atribut). Beberapa kolom penting dalam dataset termasuk informasi tentang tanggal transaksi, jenis kartu, merchant, tipe transaksi, nilai transaksi, negara, dan sebagainya.

In [4]:
```python
data.dtypes
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13125 entries, 0 to 13124
Data columns (total 28 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   X                          13125 non-null  int64
 1   id_tanggal_transaksi_awal  13125 non-null  int64
 2   tanggal_transaksi_awal     13125 non-null  int64
 3   tipe_kartu                 13125 non-null  int64
 4   id_merchant                13125 non-null  int64
 5   nama_merchant              13125 non-null  int64
 6   tipe_mesin                 13125 non-null  int64
 7   tipe_transaksi             13125 non-null  int64
 8   nama_transaksi             13125 non-null  int64
 9   nilai_transaksi            13125 non-null  float64
 10  id_negara                  13125 non-null  int64
 11  nama_negara                13125 non-null  int64
 12  nama_kota                  13125 non-null  int64
 13  lokasi_mesin               13125 non-null  int64
 14  pemilik_mesin              13125 non-null  int64
 15  waktu_transaksi            13125 non-null  int64
 16  kuartal_transaksi          13125 non-null  int64
 17  kepemilikan_kartu          13125 non-null  int64
 18  nama_channel               13125 non-null  int64
 19  id_channel                 13125 non-null  int64
 20  flag_transaksi_finansial   13125 non-null  bool
 21  status_transaksi           13125 non-null  int64
 22  bank_pemilik_kartu         13125 non-null  int64
 23  rata_rata_nilai_transaksi  13104 non-null  float64
 24  maksimum_nilai_transaksi   13104 non-null  float64
 25  minimum_nilai_transaksi    13104 non-null  float64
 26  rata_rata_jumlah_transaksi 13104 non-null  float64
 27  flag_transaksi_fraud       13125 non-null  int64
dtypes: bool(1), float64(5), int64(22)
memory usage: 2.7 MB
```

In [5]:
```python
print(data.isnull().sum())
```

```
X                              0
id_tanggal_transaksi_awal      0
tanggal_transaksi_awal         0
tipe_kartu                     0
id_merchant                    0
nama_merchant                  0
tipe_mesin                     0
tipe_transaksi                 0
nama_transaksi                 0
nilai_transaksi                0
id_negara                      0
nama_negara                    0
nama_kota                      0
lokasi_mesin                   0
pemilik_mesin                  0
waktu_transaksi                0
kuartal_transaksi              0
kepemilikan_kartu              0
nama_channel                   0
id_channel                     0
flag_transaksi_finansial       0
status_transaksi               0
bank_pemilik_kartu             0
rata_rata_nilai_transaksi     21
maksimum_nilai_transaksi      21
minimum_nilai_transaksi       21
rata_rata_jumlah_transaksi    21
flag_transaksi_fraud           0
dtype: int64
```

In [6]:
```python
print(data.describe())
```

```
                  X  id_tanggal_transaksi_awal  tanggal_transaksi_awal  \
count  13125.000000               1.312500e+04            1.312500e+04
mean    7507.821943               2.457490e+06            2.457541e+06
std     4326.959832               1.047995e+02            1.088401e+02
min        1.000000               2.457297e+06            2.457303e+06
25%     3784.000000               2.457404e+06            2.457451e+06
50%     7475.000000               2.457500e+06            2.457543e+06
75%    11265.000000               2.457581e+06            2.457632e+06
max    15000.000000               2.457662e+06            2.457754e+06

         tipe_kartu    id_merchant  nama_merchant    tipe_mesin  \
count  13125.000000   13125.000000   13125.000000  1.312500e+04
mean      85.340114   39300.980076    1678.483886  1.649037e+06
std       40.921347  124187.745686     361.236161  9.914309e+05
min        0.000000      -2.000000       2.000000 -4.000000e+00
25%       93.000000      -2.000000    1798.000000  1.130699e+06
50%      103.000000      -2.000000    1798.000000  1.836319e+06
75%      111.000000      -2.000000    1798.000000  2.419350e+06
max      138.000000  720990.000000    1859.000000  6.928943e+06

       tipe_transaksi  nama_transaksi  nilai_transaksi  ...  \
count    13125.000000    13125.000000     1.312500e+04  ...
mean       178.776362       10.729524     1.315219e+06  ...
std        142.504458        3.891783     2.838050e+06  ...
min         26.000000        1.000000     1.000000e+00  ...
25%         26.000000        9.000000     2.000000e+05  ...
50%        156.000000       10.000000     5.700000e+05  ...
75%        301.000000       11.000000     1.250000e+06  ...
max        640.000000       20.000000     7.500000e+07  ...

       kepemilikan_kartu  nama_channel   id_channel  status_transaksi  \
count       13125.000000  13125.000000  13125.000000           13125.0
mean            1.932267      1.404267      8.237410               3.0
std             0.251297      1.013583      1.721997               0.0
min             1.000000      1.000000      3.000000               3.0
25%             2.000000      1.000000      9.000000               3.0
50%             2.000000      1.000000      9.000000               3.0
75%             2.000000      1.000000      9.000000               3.0
max             2.000000      5.000000      9.000000               3.0

       bank_pemilik_kartu  rata_rata_nilai_transaksi  \
count             13125.0               1.310400e+04
mean                999.0               1.364132e+06
std                   0.0               1.448583e+06
min                 999.0               5.000000e+04
25%                 999.0               5.685634e+05
50%                 999.0               1.024239e+06
75%                 999.0               1.679778e+06
max                 999.0               2.466667e+07

       maksimum_nilai_transaksi  minimum_nilai_transaksi  \
count              1.310400e+04             1.310400e+04
mean               1.228760e+07             7.651933e+04
std                1.645905e+07             6.765391e+05
min                3.800000e+04             1.000000e+00
25%                2.500000e+06             2.000000e+00
50%                6.000000e+06             3.696400e+04
75%                1.500000e+07             6.320000e+04
max                1.000000e+08             7.500000e+07

       rata_rata_jumlah_transaksi  flag_transaksi_fraud
count                13104.000000          13125.000000
mean                     2.436182              0.069333
std                      1.389367              0.254030
min                      1.000000              0.000000
25%                      1.680000              0.000000
50%                      2.100000              0.000000
75%                      2.790000              0.000000
max                     19.780000              1.000000

[8 rows x 27 columns]
```

In [7]:
```python
# Temukan modus dari kolom 'rata_rata_nilai_transaksi'
modus = data['rata_rata_nilai_transaksi'].mode().iloc[0]

# Isi nilai yang hilang dengan modus
data['rata_rata_nilai_transaksi'].fillna(modus, inplace=True)
```

Cek representasi nilai dari data data kepemilikan_kartu, nama_merchant, nama_kota, dan nama_negara pada dataset

In [8]:
```python
kepemilikan_kartu = set()
for value in data['kepemilikan_kartu']:
    kepemilikan_kartu.add(value)
print(kepemilikan_kartu)
```

```
{1, 2}
```

In [9]:
```python
nama_merchant = set()
for value in data['nama_merchant']:
    nama_merchant.add(value)
print(nama_merchant)
```

```
{2, 5, 7, 8, 10, 11, 12, 13, 14, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 33, 35, 36, 38, 39, 40, 41, 45, 47, 52, 54, 57, 58, 60, 61, 64, 66, 6
8, 70, 72, 73, 74, 78, 81, 82, 85, 86, 87, 88, 89, 92, 95, 96, 97, 98, 99, 101, 102, 103, 106, 107, 108, 109, 110, 112, 113, 114, 118, 119, 120, 12
1, 122, 124, 125, 126, 127, 129, 130, 131, 133, 135, 136, 139, 142, 143, 144, 146, 147, 149, 151, 152, 154, 155, 159, 160, 164, 167, 169, 170, 171,
173, 174, 178, 179, 180, 182, 183, 185, 187, 189, 191, 192, 193, 195, 196, 199, 200, 201, 203, 205, 208, 210, 211, 212, 215, 216, 217, 219, 220, 22
1, 222, 224, 226, 229, 231, 235, 237, 239, 241, 244, 245, 246, 247, 248, 249, 251, 253, 254, 255, 256, 257, 258, 259, 262, 266, 269, 271, 272, 273,
276, 278, 280, 281, 282, 283, 284, 285, 286, 288, 289, 290, 292, 294, 296, 297, 298, 300, 301, 303, 304, 306, 308, 310, 313, 315, 316, 319, 320, 32
2, 323, 324, 325, 326, 327, 329, 330, 331, 332, 334, 337, 338, 339, 340, 341, 344, 345, 347, 350, 351, 352, 353, 354, 355, 356, 357, 362, 363, 366,
367, 368, 371, 373, 374, 375, 376, 377, 378, 380, 383, 385, 390, 391, 394, 395, 396, 397, 399, 402, 403, 404, 405, 409, 410, 411, 413, 415, 416, 41
7, 419, 420, 421, 422, 423, 424, 426, 427, 429, 430, 432, 433, 434, 435, 437, 438, 439, 440, 444, 446, 447, 450, 451, 452, 453, 454, 455, 457, 458,
460, 461, 465, 467, 470, 475, 476, 477, 478, 479, 481, 482, 483, 484, 488, 489, 492, 494, 495, 498, 499, 500, 501, 502, 503, 505, 506, 509, 510, 51
2, 513, 514, 515, 519, 522, 523, 525, 526, 527, 528, 529, 531, 534, 536, 537, 539, 540, 541, 543, 544, 546, 547, 548, 549, 552, 553, 554, 557, 558,
561, 562, 563, 566, 567, 569, 570, 572, 573, 575, 578, 579, 580, 582, 583, 584, 585, 586, 587, 588, 589, 590, 592, 593, 594, 595, 596, 598, 599, 60
0, 601, 602, 603, 604, 605, 607, 608, 611, 612, 616, 619, 621, 622, 624, 625, 626, 630, 631, 632, 634, 635, 639, 640, 642, 644, 646, 647, 648, 650,
652, 653, 654, 655, 656, 660, 661, 664, 666, 667, 670, 671, 673, 674, 675, 677, 682, 686, 687, 689, 691, 692, 693, 694, 696, 698, 700, 702, 705, 70
7, 708, 709, 710, 711, 712, 713, 714, 715, 717, 718, 720, 721, 723, 724, 726, 727, 728, 730, 731, 734, 736, 737, 740, 742, 743, 745, 747, 748, 749,
751, 754, 755, 757, 759, 760, 762, 763, 764, 765, 766, 767, 771, 772, 774, 775, 776, 779, 782, 783, 784, 785, 786, 787, 792, 793, 795, 796, 798, 79
9, 800, 801, 802, 804, 806, 807, 810, 811, 812, 815, 816, 817, 818, 820, 821, 823, 824, 825, 826, 827, 828, 829, 832, 835, 836, 837, 838, 839, 842,
845, 849, 850, 851, 852, 853, 854, 855, 856, 858, 859, 862, 863, 864, 866, 869, 870, 871, 873, 874, 875, 876, 880, 882, 883, 884, 885, 887, 889, 89
0, 893, 894, 897, 899, 903, 904, 905, 906, 908, 909, 913, 914, 916, 918, 919, 920, 921, 922, 924, 925, 927, 931, 932, 933, 935, 936, 938, 941, 944,
947, 948, 951, 952, 954, 955, 956, 957, 958, 959, 961, 962, 963, 967, 968, 969, 970, 972, 974, 977, 978, 979, 980, 981, 983, 984, 985, 986, 988, 98
9, 993, 994, 995, 1000, 1002, 1004, 1006, 1007, 1009, 1010, 1011, 1014, 1015, 1016, 1017, 1020, 1021, 1025, 1027, 1028, 1029, 1030, 1034, 1035, 103
6, 1037, 1039, 1040, 1041, 1042, 1044, 1045, 1046, 1047, 1049, 1050, 1051, 1052, 1054, 1055, 1056, 1061, 1064, 1066, 1068, 1069, 1073, 1075, 1078,
1080, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1093, 1096, 1097, 1101, 1103, 1104, 1107, 1111, 1112, 1114, 1115, 1118, 1119, 1121, 112
2, 1128, 1129, 1130, 1133, 1134, 1135, 1136, 1137, 1139, 1141, 1142, 1144, 1147, 1149, 1151, 1152, 1153, 1156, 1157, 1159, 1160, 1163, 1166, 1168,
1170, 1175, 1179, 1180, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1202, 1203, 1204, 1205, 1206, 1207, 1210, 1211, 1214, 1215, 121
6, 1218, 1220, 1221, 1222, 1223, 1225, 1227, 1229, 1231, 1234, 1235, 1236, 1238, 1242, 1245, 1247, 1248, 1250, 1251, 1252, 1253, 1254, 1258, 1259,
1261, 1262, 1264, 1265, 1267, 1268, 1270, 1273, 1274, 1275, 1278, 1279, 1280, 1281, 1283, 1284, 1286, 1287, 1289, 1292, 1294, 1295, 1304, 1305, 130
9, 1310, 1311, 1313, 1315, 1316, 1318, 1319, 1321, 1322, 1324, 1326, 1327, 1330, 1332, 1333, 1334, 1335, 1336, 1341, 1343, 1345, 1346, 1347, 1348,
1349, 1351, 1352, 1355, 1356, 1357, 1358, 1360, 1363, 1365, 1366, 1367, 1368, 1370, 1371, 1372, 1375, 1379, 1381, 1382, 1383, 1384, 1385, 1388, 138
9, 1390, 1391, 1393, 1394, 1396, 1397, 1398, 1400, 1403, 1404, 1408, 1409, 1412, 1413, 1414, 1416, 1422, 1423, 1427, 1428, 1429, 1430, 1432, 1433,
1437, 1438, 1440, 1443, 1444, 1445, 1447, 1449, 1453, 1455, 1456, 1457, 1465, 1466, 1469, 1471, 1475, 1476, 1478, 1479, 1480, 1482, 1483, 1485, 148
7, 1488, 1490, 1492, 1493, 1494, 1500, 1502, 1503, 1504, 1507, 1508, 1510, 1511, 1512, 1513, 1514, 1515, 1517, 1518, 1519, 1520, 1523, 1524, 1525,
1528, 1530, 1531, 1534, 1535, 1536, 1537, 1538, 1543, 1545, 1550, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1561, 1565, 1566, 1567, 1569, 1570, 157
1, 1574, 1576, 1579, 1580, 1584, 1585, 1587, 1588, 1592, 1596, 1597, 1598, 1599, 1601, 1602, 1604, 1605, 1607, 1610, 1612, 1613, 1614, 1615, 1616,
1619, 1620, 1621, 1624, 1625, 1626, 1627, 1629, 1630, 1632, 1634, 1637, 1638, 1639, 1642, 1643, 1644, 1645, 1646, 1648, 1650, 1651, 1653, 1654, 165
5, 1659, 1660, 1662, 1663, 1664, 1666, 1669, 1670, 1671, 1672, 1675, 1677, 1679, 1680, 1681, 1683, 1686, 1687, 1688, 1692, 1694, 1695, 1696, 1697,
1700, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1710, 1711, 1712, 1713, 1715, 1717, 1718, 1719, 1720, 1724, 1725, 1726, 1727, 1731, 1732, 1733, 173
5, 1736, 1737, 1738, 1741, 1742, 1746, 1747, 1748, 1750, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1762, 1763, 1765, 1766, 1767, 1768, 1769,
1770, 1771, 1772, 1773, 1775, 1776, 1777, 1778, 1779, 1783, 1784, 1785, 1789, 1790, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1802, 180
5, 1809, 1810, 1811, 1812, 1813, 1814, 1817, 1819, 1820, 1822, 1824, 1825, 1827, 1828, 1829, 1831, 1832, 1833, 1834, 1835, 1836, 1840, 1842, 1844,
1845, 1846, 1847, 1848, 1850, 1851, 1852, 1854, 1855, 1856, 1857, 1858, 1859}
```

In [10]:
```python
nama_kota = set()
for value in data['nama_merchant']:
    nama_kota.add(value)
print(nama_kota)
```

```
{2, 5, 7, 8, 10, 11, 12, 13, 14, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27, 28, 33, 35, 36, 38, 39, 40, 41, 45, 47, 52, 54, 57, 58, 60, 61, 64, 66, 6
8, 70, 72, 73, 74, 78, 81, 82, 85, 86, 87, 88, 89, 92, 95, 96, 97, 98, 99, 101, 102, 103, 106, 107, 108, 109, 110, 112, 113, 114, 118, 119, 120, 12
1, 122, 124, 125, 126, 127, 129, 130, 131, 133, 135, 136, 139, 142, 143, 144, 146, 147, 149, 151, 152, 154, 155, 159, 160, 164, 167, 169, 170, 171,
173, 174, 178, 179, 180, 182, 183, 185, 187, 189, 191, 192, 193, 195, 196, 199, 200, 201, 203, 205, 208, 210, 211, 212, 215, 216, 217, 219, 220, 22
1, 222, 224, 226, 229, 231, 235, 237, 239, 241, 244, 245, 246, 247, 248, 249, 251, 253, 254, 255, 256, 257, 258, 259, 262, 266, 269, 271, 272, 273,
276, 278, 280, 281, 282, 283, 284, 285, 286, 288, 289, 290, 292, 294, 296, 297, 298, 300, 301, 303, 304, 306, 308, 310, 313, 315, 316, 319, 320, 32
2, 323, 324, 325, 326, 327, 329, 330, 331, 332, 334, 337, 338, 339, 340, 341, 344, 345, 347, 350, 351, 352, 353, 354, 355, 356, 357, 362, 363, 366,
367, 368, 371, 373, 374, 375, 376, 377, 378, 380, 383, 385, 390, 391, 394, 395, 396, 397, 399, 402, 403, 404, 405, 409, 410, 411, 413, 415, 416, 41
7, 419, 420, 421, 422, 423, 424, 426, 427, 429, 430, 432, 433, 434, 435, 437, 438, 439, 440, 444, 446, 447, 450, 451, 452, 453, 454, 455, 457, 458,
460, 461, 465, 467, 470, 475, 476, 477, 478, 479, 481, 482, 483, 484, 488, 489, 492, 494, 495, 498, 499, 500, 501, 502, 503, 505, 506, 509, 510, 51
2, 513, 514, 515, 519, 522, 523, 525, 526, 527, 528, 529, 531, 534, 536, 537, 539, 540, 541, 543, 544, 546, 547, 548, 549, 552, 553, 554, 557, 558,
561, 562, 563, 566, 567, 569, 570, 572, 573, 575, 578, 579, 580, 582, 583, 584, 585, 586, 587, 588, 589, 590, 592, 593, 594, 595, 596, 598, 599, 60
0, 601, 602, 603, 604, 605, 607, 608, 611, 612, 616, 619, 621, 622, 624, 625, 626, 630, 631, 632, 634, 635, 639, 640, 642, 644, 646, 647, 648, 650,
652, 653, 654, 655, 656, 660, 661, 664, 666, 667, 670, 671, 673, 674, 675, 677, 682, 686, 687, 689, 691, 692, 693, 694, 696, 698, 700, 702, 705, 70
7, 708, 709, 710, 711, 712, 713, 714, 715, 717, 718, 720, 721, 723, 724, 726, 727, 728, 730, 731, 734, 736, 737, 740, 742, 743, 745, 747, 748, 749,
751, 754, 755, 757, 759, 760, 762, 763, 764, 765, 766, 767, 771, 772, 774, 775, 776, 779, 782, 783, 784, 785, 786, 787, 792, 793, 795, 796, 798, 79
9, 800, 801, 802, 804, 806, 807, 810, 811, 812, 815, 816, 817, 818, 820, 821, 823, 824, 825, 826, 827, 828, 829, 832, 835, 836, 837, 838, 839, 842,
845, 849, 850, 851, 852, 853, 854, 855, 856, 858, 859, 862, 863, 864, 866, 869, 870, 871, 873, 874, 875, 876, 880, 882, 883, 884, 885, 887, 889, 89
0, 893, 894, 897, 899, 903, 904, 905, 906, 908, 909, 913, 914, 916, 918, 919, 920, 921, 922, 924, 925, 927, 931, 932, 933, 935, 936, 938, 941, 944,
947, 948, 951, 952, 954, 955, 956, 957, 958, 959, 961, 962, 963, 967, 968, 969, 970, 972, 974, 977, 978, 979, 980, 981, 983, 984, 985, 986, 988, 98
9, 993, 994, 995, 1000, 1002, 1004, 1006, 1007, 1009, 1010, 1011, 1014, 1015, 1016, 1017, 1020, 1021, 1025, 1027, 1028, 1029, 1030, 1034, 1035, 103
6, 1037, 1039, 1040, 1041, 1042, 1044, 1045, 1046, 1047, 1049, 1050, 1051, 1052, 1054, 1055, 1056, 1061, 1064, 1066, 1068, 1069, 1073, 1075, 1078,
1080, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1093, 1096, 1097, 1101, 1103, 1104, 1107, 1111, 1112, 1114, 1115, 1118, 1119, 1121, 112
2, 1128, 1129, 1130, 1133, 1134, 1135, 1136, 1137, 1139, 1141, 1142, 1144, 1147, 1149, 1151, 1152, 1153, 1156, 1157, 1159, 1160, 1163, 1166, 1168,
1170, 1175, 1179, 1180, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1202, 1203, 1204, 1205, 1206, 1207, 1210, 1211, 1214, 1215, 121
6, 1218, 1220, 1221, 1222, 1223, 1225, 1227, 1229, 1231, 1234, 1235, 1236, 1238, 1242, 1245, 1247, 1248, 1250, 1251, 1252, 1253, 1254, 1258, 1259,
1261, 1262, 1264, 1265, 1267, 1268, 1270, 1273, 1274, 1275, 1278, 1279, 1280, 1281, 1283, 1284, 1286, 1287, 1289, 1292, 1294, 1295, 1304, 1305, 130
9, 1310, 1311, 1313, 1315, 1316, 1318, 1319, 1321, 1322, 1324, 1326, 1327, 1330, 1332, 1333, 1334, 1335, 1336, 1341, 1343, 1345, 1346, 1347, 1348,
1349, 1351, 1352, 1355, 1356, 1357, 1358, 1360, 1363, 1365, 1366, 1367, 1368, 1370, 1371, 1372, 1375, 1379, 1381, 1382, 1383, 1384, 1385, 1388, 138
9, 1390, 1391, 1393, 1394, 1396, 1397, 1398, 1400, 1403, 1404, 1408, 1409, 1412, 1413, 1414, 1416, 1422, 1423, 1427, 1428, 1429, 1430, 1432, 1433,
1437, 1438, 1440, 1443, 1444, 1445, 1447, 1449, 1453, 1455, 1456, 1457, 1465, 1466, 1469, 1471, 1475, 1476, 1478, 1479, 1480, 1482, 1483, 1485, 148
7, 1488, 1490, 1492, 1493, 1494, 1500, 1502, 1503, 1504, 1507, 1508, 1510, 1511, 1512, 1513, 1514, 1515, 1517, 1518, 1519, 1520, 1523, 1524, 1525,
1528, 1530, 1531, 1534, 1535, 1536, 1537, 1538, 1543, 1545, 1550, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1561, 1565, 1566, 1567, 1569, 1570, 157
1, 1574, 1576, 1579, 1580, 1584, 1585, 1587, 1588, 1592, 1596, 1597, 1598, 1599, 1601, 1602, 1604, 1605, 1607, 1610, 1612, 1613, 1614, 1615, 1616,
1619, 1620, 1621, 1624, 1625, 1626, 1627, 1629, 1630, 1632, 1634, 1637, 1638, 1639, 1642, 1643, 1644, 1645, 1646, 1648, 1650, 1651, 1653, 1654, 165
5, 1659, 1660, 1662, 1663, 1664, 1666, 1669, 1670, 1671, 1672, 1675, 1677, 1679, 1680, 1681, 1683, 1686, 1687, 1688, 1692, 1694, 1695, 1696, 1697,
1700, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1710, 1711, 1712, 1713, 1715, 1717, 1718, 1719, 1720, 1724, 1725, 1726, 1727, 1731, 1732, 1733, 173
5, 1736, 1737, 1738, 1741, 1742, 1746, 1747, 1748, 1750, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1762, 1763, 1765, 1766, 1767, 1768, 1769,
1770, 1771, 1772, 1773, 1775, 1776, 1777, 1778, 1779, 1783, 1784, 1785, 1789, 1790, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1802, 180
5, 1809, 1810, 1811, 1812, 1813, 1814, 1817, 1819, 1820, 1822, 1824, 1825, 1827, 1828, 1829, 1831, 1832, 1833, 1834, 1835, 1836, 1840, 1842, 1844,
1845, 1846, 1847, 1848, 1850, 1851, 1852, 1854, 1855, 1856, 1857, 1858, 1859}
```

In [11]:
```python
nama_negara = set()
for value in data['nama_negara']:
    nama_negara.add(value)
print(nama_negara)
```

```
{1, 4, 5, 6, 7, 8, 10, 11, 14, 15, 16}
```

Mari cek jumlah transaksi yang terjadi pada tiap merchant dan negara juga kota, dan pada kepemilikan kartu mana saja yang banyak terjadi fraud transaksi

In [12]:
```python
total_fraud_transactions = data['flag_transaksi_fraud'].sum()
print("Total Keseluruhan Transaksi Fraud yang Terjadi:", total_fraud_transactions)
```

```
Total Keseluruhan Transaksi Fraud yang Terjadi: 910
```

In [13]:
```python
total_fraud_by_city = data[data['flag_transaksi_fraud'] == 1].groupby('nama_kota')['flag_transaksi_fraud'].count()

print("Total Transaksi Fraud per Kota:")
print(total_fraud_by_city)
```

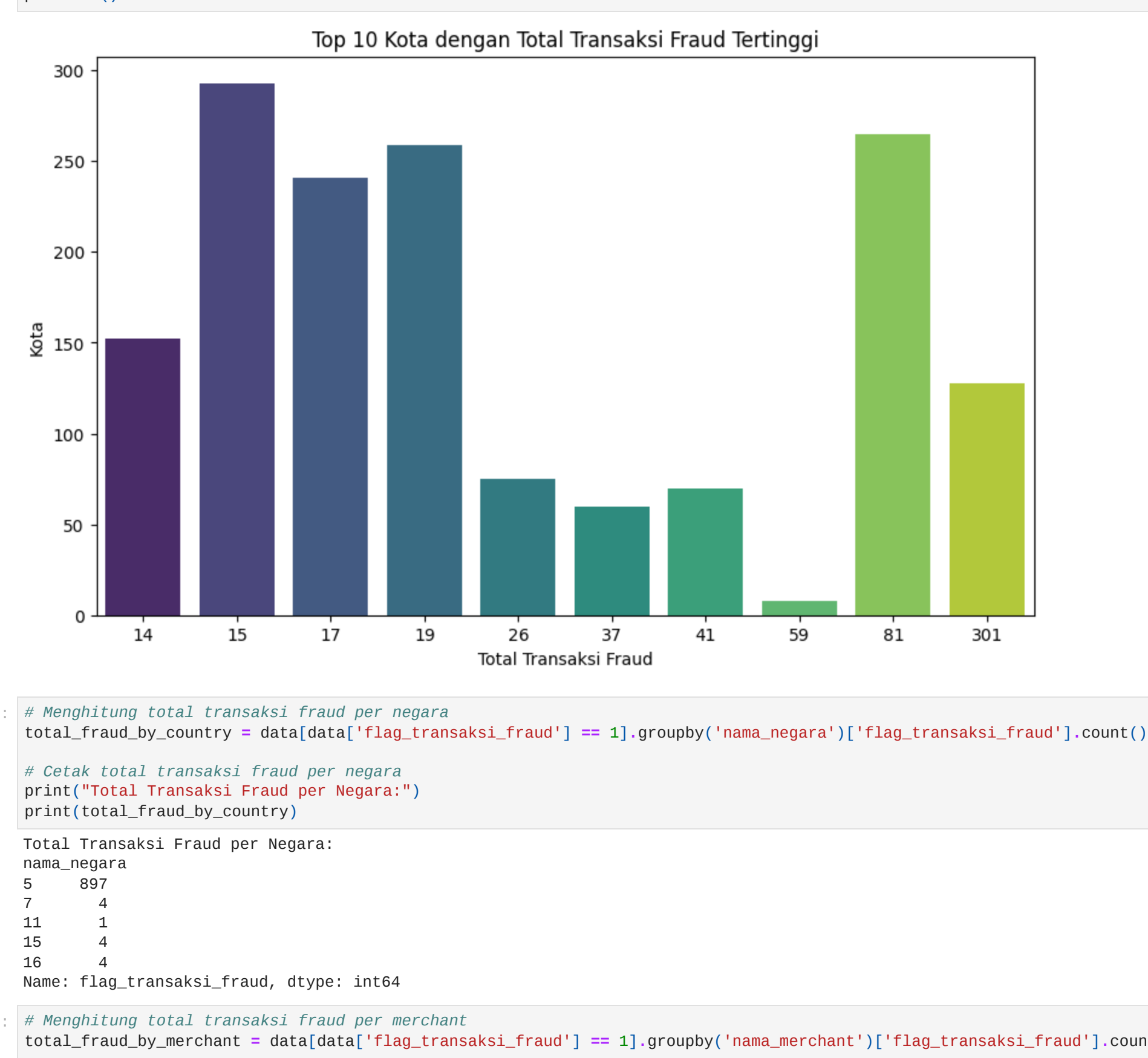
```
Total Transaksi Fraud per Kota:
nama_kota
1      8
3      1
4      1
5      4
7      6
      ..
281    1
283    1
284    2
288    1
293    15
Name: flag_transaksi_fraud, Length: 127, dtype: int64
```

In [14]:
```python
top_10_fraud_cities = total_fraud_by_city.nlargest(10)
print("Top 10 Kota dengan Total Transaksi Fraud Tertinggi:")
print(top_10_fraud_cities)
```

```
Top 10 Kota dengan Total Transaksi Fraud Tertinggi:
nama_kota
128    301
265    81
8      59
78     41
60     37
75     26
259    19
241    17
293    15
152    14
Name: flag_transaksi_fraud, dtype: int64
```

In [15]:
```python
warnings.filterwarnings('ignore')
top_10_fraud_cities = total_fraud_by_city.nlargest(10)

# Buat plot bar
plt.figure(figsize=(10, 6))
sns.barplot(x=top_10_fraud_cities.values, y=top_10_fraud_cities.index, palette="viridis")
plt.title("Top 10 Kota dengan Total Transaksi Fraud Tertinggi")
plt.xlabel("Total Transaksi Fraud")
plt.ylabel("Kota")
plt.show()
```



In [16]:
```python
# Menghitung total transaksi fraud per negara
total_fraud_by_country = data[data['flag_transaksi_fraud'] == 1].groupby('nama_negara')['flag_transaksi_fraud'].count()

# Cetak total transaksi fraud per negara
print("Total Transaksi Fraud per Negara:")
print(total_fraud_by_country)
```

```
Total Transaksi Fraud per Negara:
nama_negara
5     897
7       4
11      1
15      4
16      4
Name: flag_transaksi_fraud, dtype: int64
```

In [17]:
```python
# Menghitung total transaksi fraud per merchant
total_fraud_by_merchant = data[data['flag_transaksi_fraud'] == 1].groupby('nama_merchant')['flag_transaksi_fraud'].count()

# Cetak total transaksi fraud per merchant
print("Total Transaksi Fraud per Merchant:")
print(total_fraud_by_merchant)
```

```
Total Transaksi Fraud per Merchant:
nama_merchant
7       1
17      1
27      1
39      1
92      1
102     1
118     1
122     1
128     1
173     1
176     1
191     1
239     1
258     1
269     1
272     1
354     1
422     1
434     1
559     1
656     1
661     1
692     1
720     1
842     1
849     1
903     1
927     1
969     1
974     1
985     1
1000    1
1011    1
1142    1
1153    1
1191    1
1192    1
1205    1
1400    1
1465    1
1485    2
1512    1
1574    1
1585    1
1588    1
1645    1
1686    1
1732    1
1735    1
1758    1
1798    856
1834    1
1835    1
1836    1
1850    1
1859    1
Name: flag_transaksi_fraud, dtype: int64
```

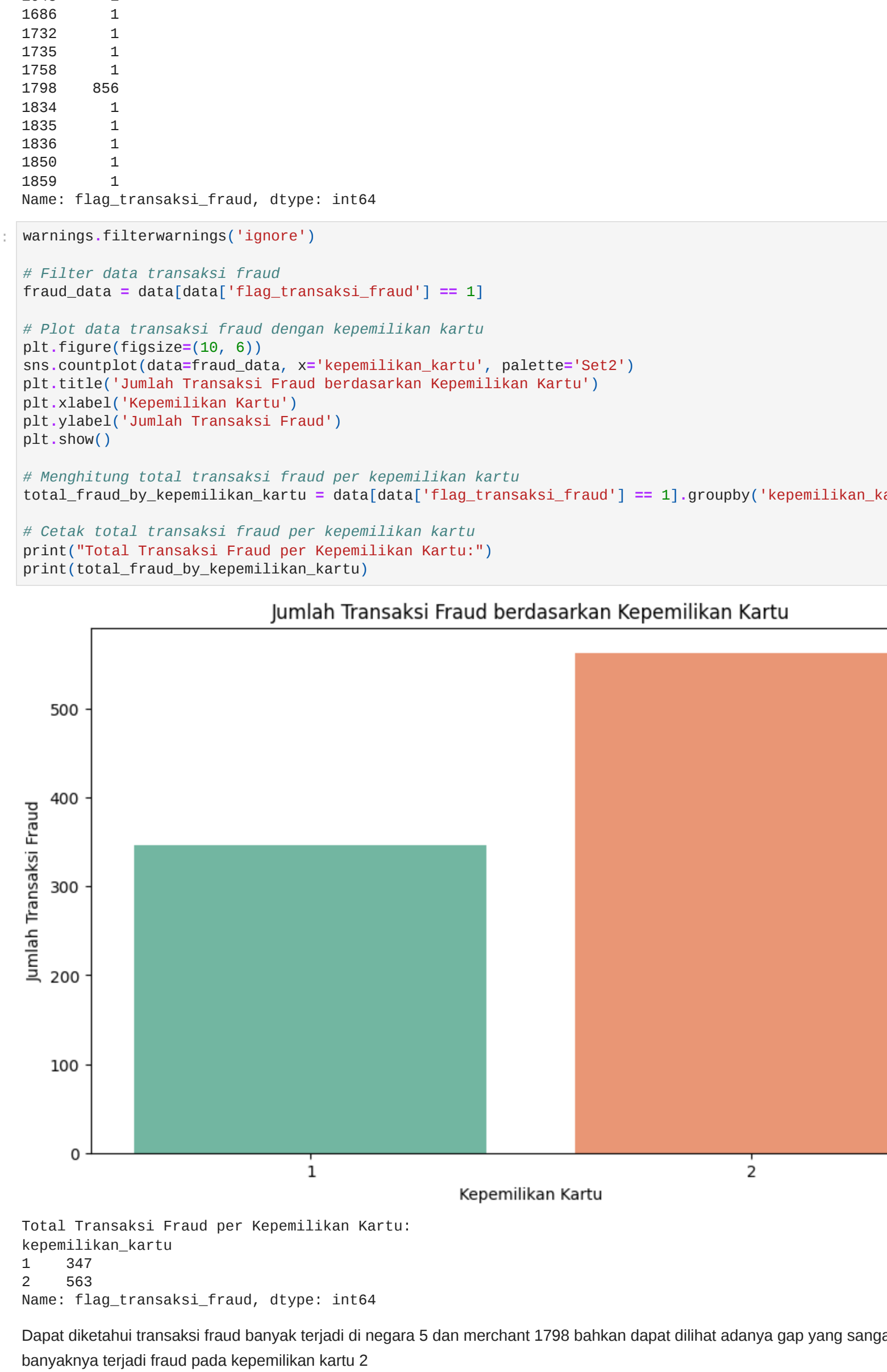
In [18]:
```python
warnings.filterwarnings('ignore')

# Filter data transaksi fraud
fraud_data = data[data['flag_transaksi_fraud'] == 1]

# Plot data transaksi fraud dengan kepemilikan kartu
plt.figure(figsize=(10, 6))
sns.countplot(data=fraud_data, x='kepemilikan_kartu', palette='Set2')
plt.title('Jumlah Transaksi Fraud berdasarkan Kepemilikan Kartu')
plt.xlabel('Kepemilikan Kartu')
plt.ylabel('Jumlah Transaksi Fraud')
plt.show()

# Menghitung total transaksi fraud per kepemilikan kartu
total_fraud_by_kepemilikan_kartu = data[data['flag_transaksi_fraud'] == 1].groupby('kepemilikan_kartu')['flag_transaksi_fraud'].count()

# Cetak total transaksi fraud per kepemilikan kartu
print("Total Transaksi Fraud per Kepemilikan Kartu:")
print(total_fraud_by_kepemilikan_kartu)
```




```
Total Transaksi Fraud per Kepemilikan Kartu:
kepemilikan_kartu
1    347
2    563
Name: flag_transaksi_fraud, dtype: int64
```

Dapat diketahui transaksi fraud banyak terjadi di negara 5 dan merchant 1798 bahkan dapat dilihat adanya gap yang sangat jauh dengan negara dan merchant lainnya, lalu banyaknya terjadi fraud pada kepemilikan kartu 2

In [19]:
```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import f1_score
```

In [20]:
```python
# Menggantikan nilai yang hilang dengan mean (atau median)
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(data.drop(columns=['flag_transaksi_fraud']))
y = imputer.fit_transform(data[['flag_transaksi_fraud']])

# Bagi dataset menjadi data pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standarisasi fitur (opsional, tergantung pada model yang digunakan)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [21]:
```python
# Membuat model Regresi Logistik
model = LogisticRegression()

# Melatih model
model.fit(X_train, y_train)

# Memprediksi data pengujian
y_pred = model.predict(X_test)

# Mengukur akurasi
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi Model:", accuracy)

from sklearn.metrics import classification_report

report = classification_report(y_test, y_pred)
print(report)

# Lakukan cross-validation dengan 5 fold untuk model LogisticRegression
cv_scores_ada = cross_val_score(model, X_train, y_train, cv=5)

# Tampilkan hasil cross-validation
print(f"CV Scores (LogisticRegression): {cv_scores_ada}")
print(f"Mean CV Score (LogisticRegression): {cv_scores_ada.mean()}")

# Hitung F-Measure
f_measure = f1_score(y_test, y_pred)

print(f'F-Measure: {f_measure}')
```

```
Akurasi Model: 0.9321904761904762
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.96      2444
         1.0       0.53      0.15      0.23       181

    accuracy                           0.93      2625
   macro avg       0.73      0.57      0.60      2625
weighted avg       0.91      0.93      0.91      2625

CV Scores (LogisticRegression): [0.93       0.92952381 0.93047619 0.93095238 0.93047619]
Mean CV Score (LogisticRegression): 0.9302857142857143
F-Measure: 0.23275862068965514
```

Precision:

Untuk kelas 0 (non-fraud), precision cukup tinggi (0.94), yang berarti sebagian besar transaksi yang diprediksi sebagai non-fraud adalah benar-benar non-fraud. Namun, untuk kelas 1 (fraud), precision rendah (0.53), yang berarti sebagian besar transaksi yang diprediksi sebagai fraud sebenarnya bukan fraud.

Recall:

Recall untuk kelas 0 (non-fraud) sangat tinggi (0.99), yang berarti sebagian besar transaksi non-fraud berhasil terdeteksi. Namun, untuk kelas 1 (fraud), recall rendah (0.15), yang berarti hanya sedikit transaksi fraud yang terdeteksi dari yang sebenarnya.

F1-score:

F1-score adalah rata-rata harmonik dari precision dan recall. Untuk kelas 0, F1-score tinggi (0.96), sedangkan untuk kelas 1, F1-score rendah (0.23).

Accuracy:

Akurasi keseluruhan adalah 0.93, yang mengindikasikan sebagian besar prediksi yang benar. Namun, karena dataset tidak seimbang, akurasi bisa menyesatkan. Dalam kasus ini, akurasi tinggi disebabkan oleh prediksi yang benar untuk kelas non-fraud (kelas mayoritas), sementara sebagian besar transaksi fraud tidak terdeteksi.

In [22]:
```python
# Inisialisasi model AdaBoost Classifier
ada_model = AdaBoostClassifier(n_estimators=50, random_state=42)

# Melatih model pada data pelatihan
ada_model.fit(X_train, y_train)

# Membuat prediksi
y_pred_ada = ada_model.predict(X_test)

# Mengukur akurasi model AdaBoost
accuracy_ada = accuracy_score(y_test, y_pred_ada)
print(f'Akurasi Model AdaBoost: {accuracy_ada}')

# Menampilkan laporan klasifikasi
print(classification_report(y_test, y_pred_ada))

# Lakukan cross-validation dengan 5 fold untuk model AdaBoost
cv_scores_ada = cross_val_score(ada_model, X_train, y_train, cv=5)

# Tampilkan hasil cross-validation
print(f"CV Scores (AdaBoost): {cv_scores_ada}")
print(f"Mean CV Score (AdaBoost): {cv_scores_ada.mean()}")

# Hitung F-Measure
f_measure = f1_score(y_test, y_pred)

print(f'F-Measure: {f_measure}')
```

```
Akurasi Model AdaBoost: 0.9321904761904762
              precision    recall  f1-score   support

         0.0       0.95      0.98      0.96      2444
         1.0       0.52      0.23      0.32       181

    accuracy                           0.93      2625
   macro avg       0.73      0.61      0.64      2625
weighted avg       0.92      0.93      0.92      2625

CV Scores (AdaBoost): [0.93190476 0.93714286 0.93904762 0.93619048 0.93666667]
Mean CV Score (AdaBoost): 0.9361904761904762
F-Measure: 0.23275862068965514
```

In [23]:
```python
# Inisialisasi dan melatih model RandomForest
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Membuat prediksi
y_pred = model.predict(X_test)

warnings.filterwarnings('ignore')

# Mengukur akurasi model
accuracy = accuracy_score(y_test, y_pred)
print(f'Akurasi: {accuracy}')

# Menampilkan laporan klasifikasi
print(classification_report(y_test, y_pred))

# Inisialisasi model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Lakukan cross-validation dengan 5 fold
cv_scores = cross_val_score(model, X_train, y_train, cv=5)

# Tampilkan hasil cross-validation
print(f"CV Scores: {cv_scores}")
print(f"Mean CV Score: {cv_scores.mean()}")

# Hitung F-Measure
f_measure = f1_score(y_test, y_pred)

print(f'F-Measure: {f_measure}')
```

```
Akurasi: 0.9428571428571428
              precision    recall  f1-score   support

         0.0       0.95      0.99      0.97      2444
         1.0       0.73      0.27      0.40       181

    accuracy                           0.94      2625
   macro avg       0.84      0.63      0.68      2625
weighted avg       0.93      0.94      0.93      2625

CV Scores: [0.93428571 0.94380952 0.94142857 0.93714286 0.94047619]
Mean CV Score: 0.9394285714285715
F-Measure: 0.3951612903225807
```

Model Random Forest telah memberikan hasil yang cukup baik dalam hal akurasi dan metrik klasifikasi seperti presisi, recall, dan f1-score. Selain itu, hasil dari cross-validation (CV Scores) juga menunjukkan bahwa model ini stabil dan tidak overfitting.

Namun, perlu diperhatikan bahwa recall untuk kelas 1 (fraud) masih relatif rendah, yang berarti masih ada banyak kasus fraud yang tidak terdeteksi oleh model. Hal ini mungkin perlu diperbaiki atau disempurnakan melalui penyesuaian hyperparameter model, pemilihan fitur yang lebih tepat, atau penggunaan teknik lain seperti oversampling atau undersampling untuk mengatasi ketidakseimbangan kelas.

Selanjutnya, Anda dapat terus mengoptimalkan model Anda dan melakukan eksperimen lebih lanjut untuk meningkatkan performa deteksi fraud. Juga, penting untuk memastikan bahwa model Anda diterapkan dengan benar dalam lingkungan produksi untuk mendeteksi fraud secara real-time.