Title:

Turning Retail Data into Business Growth: How Insights Drove Smarter Sales Decisions

Introduction:

In today's competitive retail market, many stores struggle to understand why certain products perform better than others. My retail sales analysis project aimed to bridge that gap — by transforming raw sales data into actionable business insights that could guide smarter decisions.

Problem Statement:

The store was facing challenges with inconsistent sales performance across different product categories. Despite running promotions, results were unpredictable, and inventory imbalances led to overstock in some areas and shortages in others. Management needed clarity on what was truly driving sales trends and how to optimize their strategy.

Dataset Description:

The dataset contained several months of transactional data, including:

Product details: category, price, and stock quantity

Sales data: units sold, transaction date, and total revenue

Promotional information: discount rates and campaign periods

Customer demographics: region and purchase frequency

This data provided a comprehensive view of both customer behavior and product performance.

Steps of the Project:

Data Cleaning:

Removed duplicates and missing values

Standardized column names and formatted date/time fields

Exploratory Data Analysis (EDA):

Examined sales trends over time

Identified top-performing and underperforming product categories

Analyzed promotional impact and seasonal buying patterns

Visualization:

Created sales dashboards showing monthly revenue trends and category performance

Visualized relationships between discounts and sales volume

Insight Generation:

Found that specific products had predictable peak periods (e.g., end of the month)

Revealed that not all discounts led to increased sales — timing and product selection mattered most

Recommendations:

Adjust inventory planning to align with demand peaks

Redesign promotions based on data-backed insights rather than assumptions

In [1]:

```
  Cell In[1], line 1
    python -m pip show numpy pandas
              ^
SyntaxError: invalid syntax
```

Importation of DataSet

In [2]:

```python
import pandas as pd
```

In [3]:

```python
rsales = pd.read_csv(r"C:\Users\shadrach\Downloads\retail_sales_dataset 1.csv")
```

Data Inspection

In [4]:

```python
rsales.head()
```

Out[4]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	24/11/2023	CUST001	Male	34	Beauty	3	50	150
1	2	27/02/2023	CUST002	Female	26	Clothing	2	500	1000
2	3	13/01/2023	CUST003	Male	50	Electronics	1	30	30
3	4	21/05/2023	CUST004	Male	37	Clothing	1	500	500
4	5	06/05/2023	CUST005	Male	30	Beauty	2	50	100

In [5]:

```python
rsales.shape
```

Out[5]:

```
(1000, 9)
```

In [6]:

```python
rsales.columns
```

Out[6]:

```
Index(['Transaction ID', 'Date', 'Customer ID', 'Gender', 'Age',
       'Product Category', 'Quantity', 'Price per Unit', 'Total Amount'],
      dtype='object')
```

In [7]:

```
# rsales.info()
```

In [8]:

```
rsales.describe()
```

Out[8]:

	Transaction ID	Age	Quantity	Price per Unit	Total Amount
count	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000
mean	500.500000	41.39200	2.514000	179.890000	456.000000
std	288.819436	13.68143	1.132734	189.681356	559.997632
min	1.000000	18.00000	1.000000	25.000000	25.000000
25%	250.750000	29.00000	1.000000	30.000000	60.000000
50%	500.500000	42.00000	3.000000	50.000000	135.000000
75%	750.250000	53.00000	4.000000	300.000000	900.000000
max	1000.000000	64.00000	4.000000	500.000000	2000.000000

Data Cleaning

In [9]:

```
rsales.isnull().sum()
```

Out[9]:

```
Transaction ID      0
Date                0
Customer ID         0
Gender              0
Age                 0
Product Category    0
Quantity            0
Price per Unit      0
Total Amount        0
dtype: int64
```

In [10]:

```
rsales.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
###since there is no duplicate ornull values in our data set, let chcek for unique enter
```

In [12]:

```
rsales.nunique()
```

Out[12]:

```
Transaction ID      1000
Date                 345
Customer ID         1000
Gender                 2
Age                   47
Product Category       3
Quantity               4
Price per Unit         5
Total Amount          18
dtype: int64
```

In [13]:

```
###let chcek each if any has a mistake or a false unique
```

In [14]:

```
rsales['Date'].unique()
```

Out[14]:

```
array(['24/11/2023', '27/02/2023', '13/01/2023', '21/05/2023',
       '06/05/2023', '25/04/2023', '13/03/2023', '22/02/2023',
'13/12/2023', '07/10/2023', '14/02/2023', '30/10/2023',
'05/08/2023', '17/01/2023', '16/01/2023', '17/02/2023',
'22/04/2023', '30/04/2023', '16/09/2023', '05/11/2023',
'14/01/2023', '15/10/2023', '12/04/2023', '29/11/2023',
'26/12/2023', '03/08/2023', '23/04/2023', '18/08/2023',
'29/10/2023', '23/05/2023', '04/01/2023', '23/03/2023',
'24/12/2023', '24/06/2023', '21/03/2023', '21/04/2023',
'22/06/2023', '14/07/2023', '19/02/2023', '03/07/2023',
'26/06/2023', '06/11/2023', '16/05/2023', '23/01/2023',
'24/08/2023', '02/10/2023', '05/03/2023', '13/07/2023',
'10/02/2023', '10/10/2023', '31/05/2023', '18/11/2023',
'13/11/2023', '05/07/2023', '23/10/2023', '09/04/2023',
'27/12/2023', '05/02/2023', '24/01/2023', '05/12/2023',
'27/04/2023', '29/05/2023', '21/02/2023', '21/08/2023',
'22/11/2023', '06/07/2023', '25/03/2023', '09/07/2023',
'01/07/2023', '18/04/2023', '10/12/2023', '17/05/2023',
'16/12/2023', '28/11/2023', '06/02/2023', '08/11/2023',
'29/03/2023', '01/10/2023', '25/08/2023', '19/05/2023',
'19/12/2023', '13/10/2023', '17/12/2023', '16/06/2023',
'29/01/2023', '28/04/2023', '11/06/2023', '25/07/2023',
'18/05/2023', '03/02/2023', '19/04/2023', '18/10/2023',
'02/12/2023', '13/09/2023', '22/07/2023', '26/11/2023',
'23/08/2023', '15/03/2023', '07/05/2023', '03/10/2023',
'15/05/2023', '27/10/2023', '08/08/2023', '26/10/2023',
'24/07/2023', '12/03/2023', '18/09/2023', '10/09/2023',
'16/02/2023', '25/01/2023', '26/02/2023', '20/03/2023',
'15/12/2023', '02/11/2023', '02/02/2023', '17/07/2023',
'15/07/2023', '28/08/2023', '28/09/2023', '09/05/2023',
'11/10/2023', '06/01/2023', '28/02/2023', '25/11/2023',
'11/08/2023', '22/03/2023', '02/01/2023', '14/09/2023',
'02/04/2023', '17/09/2023', '24/02/2023', '17/11/2023',
'02/06/2023', '11/07/2023', '24/03/2023', '04/10/2023',
'29/09/2023', '01/01/2023', '03/11/2023', '15/06/2023',
'08/09/2023', '10/01/2023', '07/06/2023', '03/05/2023',
'30/01/2023', '04/05/2023', '13/02/2023', '06/09/2023',
'30/09/2023', '06/03/2023', '07/03/2023', '04/12/2023',
'01/09/2023', '09/10/2023', '26/03/2023', '07/11/2023',
'20/12/2023', '13/04/2023', '01/01/2024', '09/06/2023',
'13/08/2023', '22/09/2023', '20/08/2023', '03/03/2023',
'26/04/2023', '23/06/2023', '11/01/2023', '29/12/2023',
'20/11/2023', '31/01/2023', '04/02/2023', '19/06/2023',
'21/09/2023', '02/05/2023', '09/12/2023', '20/04/2023',
'09/03/2023', '20/10/2023', '31/08/2023', '05/05/2023',
'28/07/2023', '08/04/2023', '18/02/2023', '09/08/2023',
'30/07/2023', '28/01/2023', '11/12/2023', '01/12/2023',
'27/11/2023', '20/02/2023', '01/02/2023', '26/07/2023',
'25/02/2023', '08/05/2023', '04/04/2023', '08/02/2023',
'15/08/2023', '26/01/2023', '30/11/2023', '08/01/2023',
'27/03/2023', '04/09/2023', '19/07/2023', '27/05/2023',
'23/12/2023', '12/10/2023', '07/09/2023', '01/06/2023',
'24/10/2023', '05/10/2023', '10/06/2023', '02/09/2023',
'15/09/2023', '11/02/2023', '06/04/2023', '01/11/2023',
'12/12/2023', '01/05/2023', '19/10/2023', '21/01/2023',
'14/11/2023', '03/12/2023', '17/10/2023', '25/09/2023',
'14/05/2023', '15/04/2023', '03/06/2023', '07/02/2023',
'05/01/2023', '15/11/2023', '16/10/2023', '28/06/2023',
'26/05/2023', '06/10/2023', '04/06/2023', '10/11/2023',
'08/12/2023', '06/12/2023', '23/02/2023', '10/03/2023',
'01/03/2023', '20/05/2023', '25/05/2023', '25/06/2023',
'18/12/2023', '21/11/2023', '27/01/2023', '22/05/2023',
'20/06/2023', '08/03/2023', '23/11/2023', '28/12/2023',
'07/08/2023', '18/03/2023', '19/01/2023', '17/03/2023',
'22/01/2023', '14/10/2023', '01/04/2023', '31/07/2023',
'20/01/2023', '29/08/2023', '24/04/2023', '29/06/2023',
'06/06/2023', '18/06/2023', '14/12/2023', '15/01/2023',
'25/10/2023', '12/08/2023', '19/09/2023', '11/05/2023',
'24/09/2023', '11/04/2023', '07/12/2023', '16/11/2023',
'08/06/2023', '29/07/2023', '17/06/2023', '04/08/2023',
'31/03/2023', '12/11/2023', '27/07/2023', '08/10/2023',
'05/06/2023', '14/06/2023', '27/08/2023', '28/03/2023',
'21/06/2023', '09/11/2023', '22/08/2023', '01/08/2023',
'19/11/2023', '22/10/2023', '10/04/2023', '16/07/2023',
'11/09/2023', '03/01/2023', '06/08/2023', '23/09/2023',
'26/08/2023', '04/03/2023', '12/06/2023', '17/08/2023',
'19/08/2023', '13/05/2023', '14/08/2023', '09/02/2023',
'13/06/2023', '19/03/2023', '29/04/2023', '22/12/2023',
'16/04/2023', '30/06/2023', '21/07/2023', '31/10/2023',
'12/02/2023', '11/03/2023', '10/05/2023', '25/12/2023',
'12/05/2023', '08/07/2023', '12/07/2023', '23/07/2023',
'04/11/2023', '27/06/2023', '07/01/2023', '10/08/2023',
'05/09/2023', '24/05/2023', '31/12/2023', '09/09/2023',
'09/01/2023', '21/12/2023', '05/04/2023', '26/09/2023',
'04/07/2023', '30/05/2023', '21/10/2023', '03/09/2023',
'02/03/2023', '02/08/2023', '17/04/2023', '30/03/2023',
'28/05/2023'], dtype=object)
```

```
In [15]: rsales['Transaction ID'].unique()
```

```
Out[15]: array([   1,    2,    3,    4,    5,    6,    7,    8,    9,   10,   11,
         12,   13,   14,   15,   16,   17,   18,   19,   20,   21,   22,
         23,   24,   25,   26,   27,   28,   29,   30,   31,   32,   33,
         34,   35,   36,   37,   38,   39,   40,   41,   42,   43,   44,
         45,   46,   47,   48,   49,   50,   51,   52,   53,   54,   55,
         56,   57,   58,   59,   60,   61,   62,   63,   64,   65,   66,
         67,   68,   69,   70,   71,   72,   73,   74,   75,   76,   77,
         78,   79,   80,   81,   82,   83,   84,   85,   86,   87,   88,
         89,   90,   91,   92,   93,   94,   95,   96,   97,   98,   99,
        100,  101,  102,  103,  104,  105,  106,  107,  108,  109,  110,
        111,  112,  113,  114,  115,  116,  117,  118,  119,  120,  121,
        122,  123,  124,  125,  126,  127,  128,  129,  130,  131,  132,
        133,  134,  135,  136,  137,  138,  139,  140,  141,  142,  143,
        144,  145,  146,  147,  148,  149,  150,  151,  152,  153,  154,
        155,  156,  157,  158,  159,  160,  161,  162,  163,  164,  165,
        166,  167,  168,  169,  170,  171,  172,  173,  174,  175,  176,
        177,  178,  179,  180,  181,  182,  183,  184,  185,  186,  187,
        188,  189,  190,  191,  192,  193,  194,  195,  196,  197,  198,
        199,  200,  201,  202,  203,  204,  205,  206,  207,  208,  209,
        210,  211,  212,  213,  214,  215,  216,  217,  218,  219,  220,
        221,  222,  223,  224,  225,  226,  227,  228,  229,  230,  231,
        232,  233,  234,  235,  236,  237,  238,  239,  240,  241,  242,
        243,  244,  245,  246,  247,  248,  249,  250,  251,  252,  253,
        254,  255,  256,  257,  258,  259,  260,  261,  262,  263,  264,
        265,  266,  267,  268,  269,  270,  271,  272,  273,  274,  275,
        276,  277,  278,  279,  280,  281,  282,  283,  284,  285,  286,
        287,  288,  289,  290,  291,  292,  293,  294,  295,  296,  297,
        298,  299,  300,  301,  302,  303,  304,  305,  306,  307,  308,
        309,  310,  311,  312,  313,  314,  315,  316,  317,  318,  319,
        320,  321,  322,  323,  324,  325,  326,  327,  328,  329,  330,
        331,  332,  333,  334,  335,  336,  337,  338,  339,  340,  341,
        342,  343,  344,  345,  346,  347,  348,  349,  350,  351,  352,
        353,  354,  355,  356,  357,  358,  359,  360,  361,  362,  363,
        364,  365,  366,  367,  368,  369,  370,  371,  372,  373,  374,
        375,  376,  377,  378,  379,  380,  381,  382,  383,  384,  385,
        386,  387,  388,  389,  390,  391,  392,  393,  394,  395,  396,
        397,  398,  399,  400,  401,  402,  403,  404,  405,  406,  407,
        408,  409,  410,  411,  412,  413,  414,  415,  416,  417,  418,
        419,  420,  421,  422,  423,  424,  425,  426,  427,  428,  429,
        430,  431,  432,  433,  434,  435,  436,  437,  438,  439,  440,
        441,  442,  443,  444,  445,  446,  447,  448,  449,  450,  451,
        452,  453,  454,  455,  456,  457,  458,  459,  460,  461,  462,
        463,  464,  465,  466,  467,  468,  469,  470,  471,  472,  473,
        474,  475,  476,  477,  478,  479,  480,  481,  482,  483,  484,
        485,  486,  487,  488,  489,  490,  491,  492,  493,  494,  495,
        496,  497,  498,  499,  500,  501,  502,  503,  504,  505,  506,
        507,  508,  509,  510,  511,  512,  513,  514,  515,  516,  517,
        518,  519,  520,  521,  522,  523,  524,  525,  526,  527,  528,
        529,  530,  531,  532,  533,  534,  535,  536,  537,  538,  539,
        540,  541,  542,  543,  544,  545,  546,  547,  548,  549,  550,
        551,  552,  553,  554,  555,  556,  557,  558,  559,  560,  561,
        562,  563,  564,  565,  566,  567,  568,  569,  570,  571,  572,
        573,  574,  575,  576,  577,  578,  579,  580,  581,  582,  583,
        584,  585,  586,  587,  588,  589,  590,  591,  592,  593,  594,
        595,  596,  597,  598,  599,  600,  601,  602,  603,  604,  605,
        606,  607,  608,  609,  610,  611,  612,  613,  614,  615,  616,
        617,  618,  619,  620,  621,  622,  623,  624,  625,  626,  627,
        628,  629,  630,  631,  632,  633,  634,  635,  636,  637,  638,
        639,  640,  641,  642,  643,  644,  645,  646,  647,  648,  649,
        650,  651,  652,  653,  654,  655,  656,  657,  658,  659,  660,
        661,  662,  663,  664,  665,  666,  667,  668,  669,  670,  671,
        672,  673,  674,  675,  676,  677,  678,  679,  680,  681,  682,
        683,  684,  685,  686,  687,  688,  689,  690,  691,  692,  693,
        694,  695,  696,  697,  698,  699,  700,  701,  702,  703,  704,
        705,  706,  707,  708,  709,  710,  711,  712,  713,  714,  715,
        716,  717,  718,  719,  720,  721,  722,  723,  724,  725,  726,
        727,  728,  729,  730,  731,  732,  733,  734,  735,  736,  737,
        738,  739,  740,  741,  742,  743,  744,  745,  746,  747,  748,
        749,  750,  751,  752,  753,  754,  755,  756,  757,  758,  759,
        760,  761,  762,  763,  764,  765,  766,  767,  768,  769,  770,
        771,  772,  773,  774,  775,  776,  777,  778,  779,  780,  781,
        782,  783,  784,  785,  786,  787,  788,  789,  790,  791,  792,
        793,  794,  795,  796,  797,  798,  799,  800,  801,  802,  803,
        804,  805,  806,  807,  808,  809,  810,  811,  812,  813,  814,
        815,  816,  817,  818,  819,  820,  821,  822,  823,  824,  825,
        826,  827,  828,  829,  830,  831,  832,  833,  834,  835,  836,
        837,  838,  839,  840,  841,  842,  843,  844,  845,  846,  847,
        848,  849,  850,  851,  852,  853,  854,  855,  856,  857,  858,
        859,  860,  861,  862,  863,  864,  865,  866,  867,  868,  869,
        870,  871,  872,  873,  874,  875,  876,  877,  878,  879,  880,
        881,  882,  883,  884,  885,  886,  887,  888,  889,  890,  891,
        892,  893,  894,  895,  896,  897,  898,  899,  900,  901,  902,
        903,  904,  905,  906,  907,  908,  909,  910,  911,  912,  913,
        914,  915,  916,  917,  918,  919,  920,  921,  922,  923,  924,
        925,  926,  927,  928,  929,  930,  931,  932,  933,  934,  935,
        936,  937,  938,  939,  940,  941,  942,  943,  944,  945,  946,
        947,  948,  949,  950,  951,  952,  953,  954,  955,  956,  957,
        958,  959,  960,  961,  962,  963,  964,  965,  966,  967,  968,
        969,  970,  971,  972,  973,  974,  975,  976,  977,  978,  979,
        980,  981,  982,  983,  984,  985,  986,  987,  988,  989,  990,
        991,  992,  993,  994,  995,  996,  997,  998,  999, 1000],
      dtype=int64)
```

In [16]:

```
rsales['Customer ID'].unique()
```

Out[16]:

```
array(['CUST001', 'CUST002', 'CUST003', 'CUST004', 'CUST005', 'CUST006',
       'CUST007', 'CUST008', 'CUST009', 'CUST010', 'CUST011', 'CUST012',
       'CUST013', 'CUST014', 'CUST015', 'CUST016', 'CUST017', 'CUST018',
       'CUST019', 'CUST020', 'CUST021', 'CUST022', 'CUST023', 'CUST024',
       'CUST025', 'CUST026', 'CUST027', 'CUST028', 'CUST029', 'CUST030',
       'CUST031', 'CUST032', 'CUST033', 'CUST034', 'CUST035', 'CUST036',
       'CUST037', 'CUST038', 'CUST039', 'CUST040', 'CUST041', 'CUST042',
       'CUST043', 'CUST044', 'CUST045', 'CUST046', 'CUST047', 'CUST048',
       'CUST049', 'CUST050', 'CUST051', 'CUST052', 'CUST053', 'CUST054',
       'CUST055', 'CUST056', 'CUST057', 'CUST058', 'CUST059', 'CUST060',
       'CUST061', 'CUST062', 'CUST063', 'CUST064', 'CUST065', 'CUST066',
       'CUST067', 'CUST068', 'CUST069', 'CUST070', 'CUST071', 'CUST072',
       'CUST073', 'CUST074', 'CUST075', 'CUST076', 'CUST077', 'CUST078',
       'CUST079', 'CUST080', 'CUST081', 'CUST082', 'CUST083', 'CUST084',
       'CUST085', 'CUST086', 'CUST087', 'CUST088', 'CUST089', 'CUST090',
       'CUST091', 'CUST092', 'CUST093', 'CUST094', 'CUST095', 'CUST096',
       'CUST097', 'CUST098', 'CUST099', 'CUST100', 'CUST101', 'CUST102',
       'CUST103', 'CUST104', 'CUST105', 'CUST106', 'CUST107', 'CUST108',
       'CUST109', 'CUST110', 'CUST111', 'CUST112', 'CUST113', 'CUST114',
       'CUST115', 'CUST116', 'CUST117', 'CUST118', 'CUST119', 'CUST120',
       'CUST121', 'CUST122', 'CUST123', 'CUST124', 'CUST125', 'CUST126',
       'CUST127', 'CUST128', 'CUST129', 'CUST130', 'CUST131', 'CUST132',
       'CUST133', 'CUST134', 'CUST135', 'CUST136', 'CUST137', 'CUST138',
       'CUST139', 'CUST140', 'CUST141', 'CUST142', 'CUST143', 'CUST144',
       'CUST145', 'CUST146', 'CUST147', 'CUST148', 'CUST149', 'CUST150',
       'CUST151', 'CUST152', 'CUST153', 'CUST154', 'CUST155', 'CUST156',
       'CUST157', 'CUST158', 'CUST159', 'CUST160', 'CUST161', 'CUST162',
       'CUST163', 'CUST164', 'CUST165', 'CUST166', 'CUST167', 'CUST168',
       'CUST169', 'CUST170', 'CUST171', 'CUST172', 'CUST173', 'CUST174',
       'CUST175', 'CUST176', 'CUST177', 'CUST178', 'CUST179', 'CUST180',
       'CUST181', 'CUST182', 'CUST183', 'CUST184', 'CUST185', 'CUST186',
       'CUST187', 'CUST188', 'CUST189', 'CUST190', 'CUST191', 'CUST192',
       'CUST193', 'CUST194', 'CUST195', 'CUST196', 'CUST197', 'CUST198',
       'CUST199', 'CUST200', 'CUST201', 'CUST202', 'CUST203', 'CUST204',
       'CUST205', 'CUST206', 'CUST207', 'CUST208', 'CUST209', 'CUST210',
       'CUST211', 'CUST212', 'CUST213', 'CUST214', 'CUST215', 'CUST216',
       'CUST217', 'CUST218', 'CUST219', 'CUST220', 'CUST221', 'CUST222',
       'CUST223', 'CUST224', 'CUST225', 'CUST226', 'CUST227', 'CUST228',
       'CUST229', 'CUST230', 'CUST231', 'CUST232', 'CUST233', 'CUST234',
       'CUST235', 'CUST236', 'CUST237', 'CUST238', 'CUST239', 'CUST240',
       'CUST241', 'CUST242', 'CUST243', 'CUST244', 'CUST245', 'CUST246',
       'CUST247', 'CUST248', 'CUST249', 'CUST250', 'CUST251', 'CUST252',
       'CUST253', 'CUST254', 'CUST255', 'CUST256', 'CUST257', 'CUST258',
       'CUST259', 'CUST260', 'CUST261', 'CUST262', 'CUST263', 'CUST264',
       'CUST265', 'CUST266', 'CUST267', 'CUST268', 'CUST269', 'CUST270',
       'CUST271', 'CUST272', 'CUST273', 'CUST274', 'CUST275', 'CUST276',
       'CUST277', 'CUST278', 'CUST279', 'CUST280', 'CUST281', 'CUST282',
       'CUST283', 'CUST284', 'CUST285', 'CUST286', 'CUST287', 'CUST288',
       'CUST289', 'CUST290', 'CUST291', 'CUST292', 'CUST293', 'CUST294',
       'CUST295', 'CUST296', 'CUST297', 'CUST298', 'CUST299', 'CUST300',
       'CUST301', 'CUST302', 'CUST303', 'CUST304', 'CUST305', 'CUST306',
       'CUST307', 'CUST308', 'CUST309', 'CUST310', 'CUST311', 'CUST312',
       'CUST313', 'CUST314', 'CUST315', 'CUST316', 'CUST317', 'CUST318',
       'CUST319', 'CUST320', 'CUST321', 'CUST322', 'CUST323', 'CUST324',
       'CUST325', 'CUST326', 'CUST327', 'CUST328', 'CUST329', 'CUST330',
       'CUST331', 'CUST332', 'CUST333', 'CUST334', 'CUST335', 'CUST336',
       'CUST337', 'CUST338', 'CUST339', 'CUST340', 'CUST341', 'CUST342',
       'CUST343', 'CUST344', 'CUST345', 'CUST346', 'CUST347', 'CUST348',
       'CUST349', 'CUST350', 'CUST351', 'CUST352', 'CUST353', 'CUST354',
       'CUST355', 'CUST356', 'CUST357', 'CUST358', 'CUST359', 'CUST360',
       'CUST361', 'CUST362', 'CUST363', 'CUST364', 'CUST365', 'CUST366',
       'CUST367', 'CUST368', 'CUST369', 'CUST370', 'CUST371', 'CUST372',
       'CUST373', 'CUST374', 'CUST375', 'CUST376', 'CUST377', 'CUST378',
       'CUST379', 'CUST380', 'CUST381', 'CUST382', 'CUST383', 'CUST384',
       'CUST385', 'CUST386', 'CUST387', 'CUST388', 'CUST389', 'CUST390',
       'CUST391', 'CUST392', 'CUST393', 'CUST394', 'CUST395', 'CUST396',
       'CUST397', 'CUST398', 'CUST399', 'CUST400', 'CUST401', 'CUST402',
       'CUST403', 'CUST404', 'CUST405', 'CUST406', 'CUST407', 'CUST408',
       'CUST409', 'CUST410', 'CUST411', 'CUST412', 'CUST413', 'CUST414',
       'CUST415', 'CUST416', 'CUST417', 'CUST418', 'CUST419', 'CUST420',
       'CUST421', 'CUST422', 'CUST423', 'CUST424', 'CUST425', 'CUST426',
       'CUST427', 'CUST428', 'CUST429', 'CUST430', 'CUST431', 'CUST432',
       'CUST433', 'CUST434', 'CUST435', 'CUST436', 'CUST437', 'CUST438',
       'CUST439', 'CUST440', 'CUST441', 'CUST442', 'CUST443', 'CUST444',
       'CUST445', 'CUST446', 'CUST447', 'CUST448', 'CUST449', 'CUST450',
       'CUST451', 'CUST452', 'CUST453', 'CUST454', 'CUST455', 'CUST456',
       'CUST457', 'CUST458', 'CUST459', 'CUST460', 'CUST461', 'CUST462',
       'CUST463', 'CUST464', 'CUST465', 'CUST466', 'CUST467', 'CUST468',
       'CUST469', 'CUST470', 'CUST471', 'CUST472', 'CUST473', 'CUST474',
       'CUST475', 'CUST476', 'CUST477', 'CUST478', 'CUST479', 'CUST480',
       'CUST481', 'CUST482', 'CUST483', 'CUST484', 'CUST485', 'CUST486',
       'CUST487', 'CUST488', 'CUST489', 'CUST490', 'CUST491', 'CUST492',
       'CUST493', 'CUST494', 'CUST495', 'CUST496', 'CUST497', 'CUST498',
       'CUST499', 'CUST500', 'CUST501', 'CUST502', 'CUST503', 'CUST504',
       'CUST505', 'CUST506', 'CUST507', 'CUST508', 'CUST509', 'CUST510',
       'CUST511', 'CUST512', 'CUST513', 'CUST514', 'CUST515', 'CUST516',
       'CUST517', 'CUST518', 'CUST519', 'CUST520', 'CUST521', 'CUST522',
       'CUST523', 'CUST524', 'CUST525', 'CUST526', 'CUST527', 'CUST528',
       'CUST529', 'CUST530', 'CUST531', 'CUST532', 'CUST533', 'CUST534',
       'CUST535', 'CUST536', 'CUST537', 'CUST538', 'CUST539', 'CUST540',
       'CUST541', 'CUST542', 'CUST543', 'CUST544', 'CUST545', 'CUST546',
       'CUST547', 'CUST548', 'CUST549', 'CUST550', 'CUST551', 'CUST552',
       'CUST553', 'CUST554', 'CUST555', 'CUST556', 'CUST557', 'CUST558',
       'CUST559', 'CUST560', 'CUST561', 'CUST562', 'CUST563', 'CUST564',
       'CUST565', 'CUST566', 'CUST567', 'CUST568', 'CUST569', 'CUST570',
       'CUST571', 'CUST572', 'CUST573', 'CUST574', 'CUST575', 'CUST576',
       'CUST577', 'CUST578', 'CUST579', 'CUST580', 'CUST581', 'CUST582',
       'CUST583', 'CUST584', 'CUST585', 'CUST586', 'CUST587', 'CUST588',
       'CUST589', 'CUST590', 'CUST591', 'CUST592', 'CUST593', 'CUST594',
       'CUST595', 'CUST596', 'CUST597', 'CUST598', 'CUST599', 'CUST600',
       'CUST601', 'CUST602', 'CUST603', 'CUST604', 'CUST605', 'CUST606',
       'CUST607', 'CUST608', 'CUST609', 'CUST610', 'CUST611', 'CUST612',
       'CUST613', 'CUST614', 'CUST615', 'CUST616', 'CUST617', 'CUST618',
       'CUST619', 'CUST620', 'CUST621', 'CUST622', 'CUST623', 'CUST624',
       'CUST625', 'CUST626', 'CUST627', 'CUST628', 'CUST629', 'CUST630',
       'CUST631', 'CUST632', 'CUST633', 'CUST634', 'CUST635', 'CUST636',
       'CUST637', 'CUST638', 'CUST639', 'CUST640', 'CUST641', 'CUST642',
       'CUST643', 'CUST644', 'CUST645', 'CUST646', 'CUST647', 'CUST648',
       'CUST649', 'CUST650', 'CUST651', 'CUST652', 'CUST653', 'CUST654',
       'CUST655', 'CUST656', 'CUST657', 'CUST658', 'CUST659', 'CUST660',
       'CUST661', 'CUST662', 'CUST663', 'CUST664', 'CUST665', 'CUST666',
       'CUST667', 'CUST668', 'CUST669', 'CUST670', 'CUST671', 'CUST672',
       'CUST673', 'CUST674', 'CUST675', 'CUST676', 'CUST677', 'CUST678',
       'CUST679', 'CUST680', 'CUST681', 'CUST682', 'CUST683', 'CUST684',
       'CUST685', 'CUST686', 'CUST687', 'CUST688', 'CUST689', 'CUST690',
       'CUST691', 'CUST692', 'CUST693', 'CUST694', 'CUST695', 'CUST696',
       'CUST697', 'CUST698', 'CUST699', 'CUST700', 'CUST701', 'CUST702',
       'CUST703', 'CUST704', 'CUST705', 'CUST706', 'CUST707', 'CUST708',
       'CUST709', 'CUST710', 'CUST711', 'CUST712', 'CUST713', 'CUST714',
       'CUST715', 'CUST716', 'CUST717', 'CUST718', 'CUST719', 'CUST720',
       'CUST721', 'CUST722', 'CUST723', 'CUST724', 'CUST725', 'CUST726',
       'CUST727', 'CUST728', 'CUST729', 'CUST730', 'CUST731', 'CUST732',
       'CUST733', 'CUST734', 'CUST735', 'CUST736', 'CUST737', 'CUST738',
       'CUST739', 'CUST740', 'CUST741', 'CUST742', 'CUST743', 'CUST744',
       'CUST745', 'CUST746', 'CUST747', 'CUST748', 'CUST749', 'CUST750',
       'CUST751', 'CUST752', 'CUST753', 'CUST754', 'CUST755', 'CUST756',
       'CUST757', 'CUST758', 'CUST759', 'CUST760', 'CUST761', 'CUST762',
       'CUST763', 'CUST764', 'CUST765', 'CUST766', 'CUST767', 'CUST768',
       'CUST769', 'CUST770', 'CUST771', 'CUST772', 'CUST773', 'CUST774',
       'CUST775', 'CUST776', 'CUST777', 'CUST778', 'CUST779', 'CUST780',
       'CUST781', 'CUST782', 'CUST783', 'CUST784', 'CUST785', 'CUST786',
       'CUST787', 'CUST788', 'CUST789', 'CUST790', 'CUST791', 'CUST792',
       'CUST793', 'CUST794', 'CUST795', 'CUST796', 'CUST797', 'CUST798',
       'CUST799', 'CUST800', 'CUST801', 'CUST802', 'CUST803', 'CUST804',
       'CUST805', 'CUST806', 'CUST807', 'CUST808', 'CUST809', 'CUST810',
       'CUST811', 'CUST812', 'CUST813', 'CUST814', 'CUST815', 'CUST816',
       'CUST817', 'CUST818', 'CUST819', 'CUST820', 'CUST821', 'CUST822',
       'CUST823', 'CUST824', 'CUST825', 'CUST826', 'CUST827', 'CUST828',
       'CUST829', 'CUST830', 'CUST831', 'CUST832', 'CUST833', 'CUST834',
       'CUST835', 'CUST836', 'CUST837', 'CUST838', 'CUST839', 'CUST840',
       'CUST841', 'CUST842', 'CUST843', 'CUST844', 'CUST845', 'CUST846',
       'CUST847', 'CUST848', 'CUST849', 'CUST850', 'CUST851', 'CUST852',
       'CUST853', 'CUST854', 'CUST855', 'CUST856', 'CUST857', 'CUST858',
       'CUST859', 'CUST860', 'CUST861', 'CUST862', 'CUST863', 'CUST864',
       'CUST865', 'CUST866', 'CUST867', 'CUST868', 'CUST869', 'CUST870',
       'CUST871', 'CUST872', 'CUST873', 'CUST874', 'CUST875', 'CUST876',
       'CUST877', 'CUST878', 'CUST879', 'CUST880', 'CUST881', 'CUST882',
              'CUST883', 'CUST884', 'CUST885', 'CUST886', 'CUST887', 'CUST888',
              'CUST889', 'CUST890', 'CUST891', 'CUST892', 'CUST893', 'CUST894',
              'CUST895', 'CUST896', 'CUST897', 'CUST898', 'CUST899', 'CUST900',
              'CUST901', 'CUST902', 'CUST903', 'CUST904', 'CUST905', 'CUST906',
              'CUST907', 'CUST908', 'CUST909', 'CUST910', 'CUST911', 'CUST912',
              'CUST913', 'CUST914', 'CUST915', 'CUST916', 'CUST917', 'CUST918',
              'CUST919', 'CUST920', 'CUST921', 'CUST922', 'CUST923', 'CUST924',
              'CUST925', 'CUST926', 'CUST927', 'CUST928', 'CUST929', 'CUST930',
              'CUST931', 'CUST932', 'CUST933', 'CUST934', 'CUST935', 'CUST936',
              'CUST937', 'CUST938', 'CUST939', 'CUST940', 'CUST941', 'CUST942',
              'CUST943', 'CUST944', 'CUST945', 'CUST946', 'CUST947', 'CUST948',
              'CUST949', 'CUST950', 'CUST951', 'CUST952', 'CUST953', 'CUST954',
              'CUST955', 'CUST956', 'CUST957', 'CUST958', 'CUST959', 'CUST960',
              'CUST961', 'CUST962', 'CUST963', 'CUST964', 'CUST965', 'CUST966',
              'CUST967', 'CUST968', 'CUST969', 'CUST970', 'CUST971', 'CUST972',
              'CUST973', 'CUST974', 'CUST975', 'CUST976', 'CUST977', 'CUST978',
              'CUST979', 'CUST980', 'CUST981', 'CUST982', 'CUST983', 'CUST984',
              'CUST985', 'CUST986', 'CUST987', 'CUST988', 'CUST989', 'CUST990',
              'CUST991', 'CUST992', 'CUST993', 'CUST994', 'CUST995', 'CUST996',
              'CUST997', 'CUST998', 'CUST999', 'CUST1000'], dtype=object)
```

In [17]:

```
rsales['Gender'].unique()
```

Out[17]:

```
array(['Male', 'Female'], dtype=object)
```

In [18]:

```
rsales['Age'].unique()
```

Out[18]:

```
array([34, 26, 50, 37, 30, 45, 46, 63, 52, 23, 35, 22, 64, 42, 19, 27, 47,
       62, 18, 49, 28, 38, 43, 39, 44, 51, 58, 48, 55, 20, 40, 54, 36, 31,
       21, 57, 25, 56, 29, 61, 32, 41, 59, 60, 33, 53, 24], dtype=int64)
```

In [19]:

```
rsales["Product Category"].unique()
```

Out[19]:

```
array(['Beauty', 'Clothing', 'Electronics'], dtype=object)
```

In [20]:

```
rsales['Product Category'].unique()
```

Out[20]:

```
array(['Beauty', 'Clothing', 'Electronics'], dtype=object)
```

In [21]:

```
rsales['Price per Unit'].unique()
```

Out[21]:

```
array([ 50, 500,  30,  25, 300], dtype=int64)
```

In [22]:

```
rsales['Total Amount'].unique()
```

Out[22]:

```
array([ 150, 1000,   30,  500,  100,   50,  600,  200,   75, 1500,  120,
       2000,  900,  300, 1200,   90,   25,   60], dtype=int64)
```

Exploratory Data Analysis

In [23]:

```
###we therefore have expored the data set hence we can call it clean,fit FOR E.D.A
```

In [24]:

```
##a. Univariate Analysis
###Visualize each variable individually.
```

In [25]:

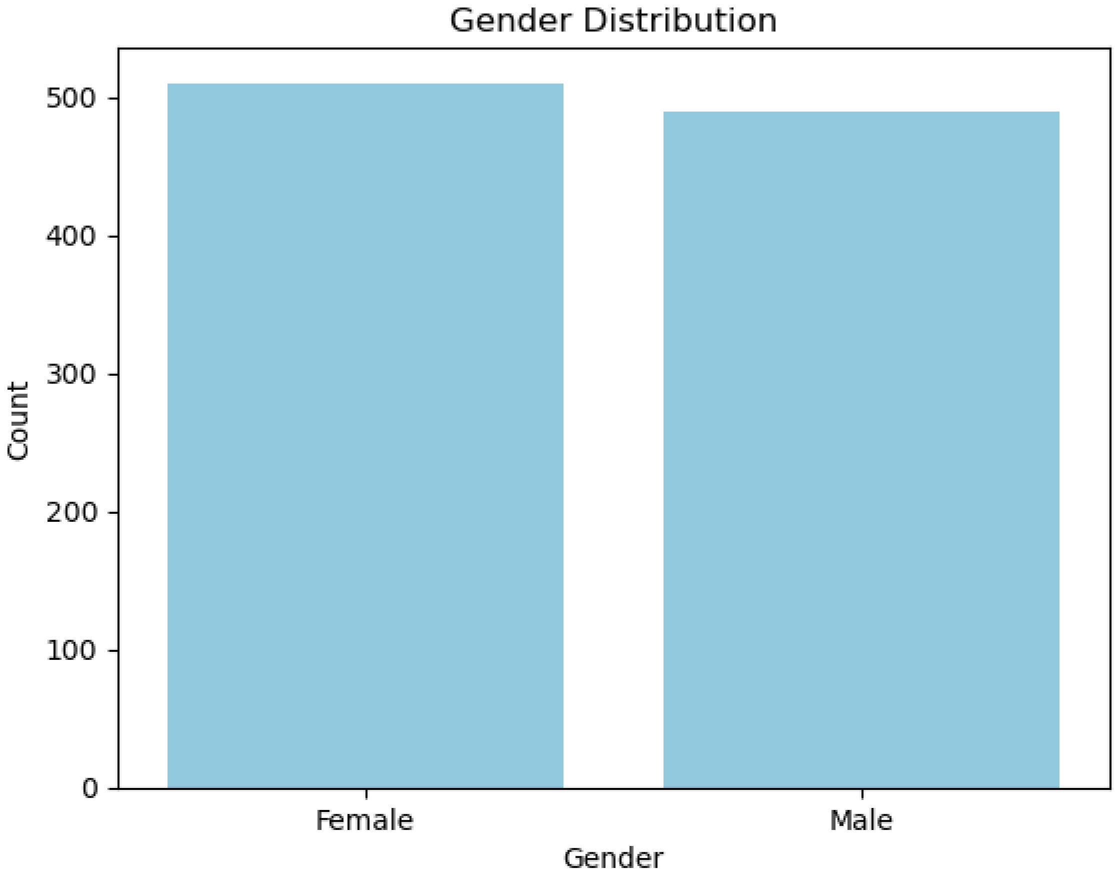
```
##categorical data
```

In [26]:

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

```
sns.barplot(x = rsales['Gender'].value_counts().index, y= rsales['Gender'].value_counts(

plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution')
plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

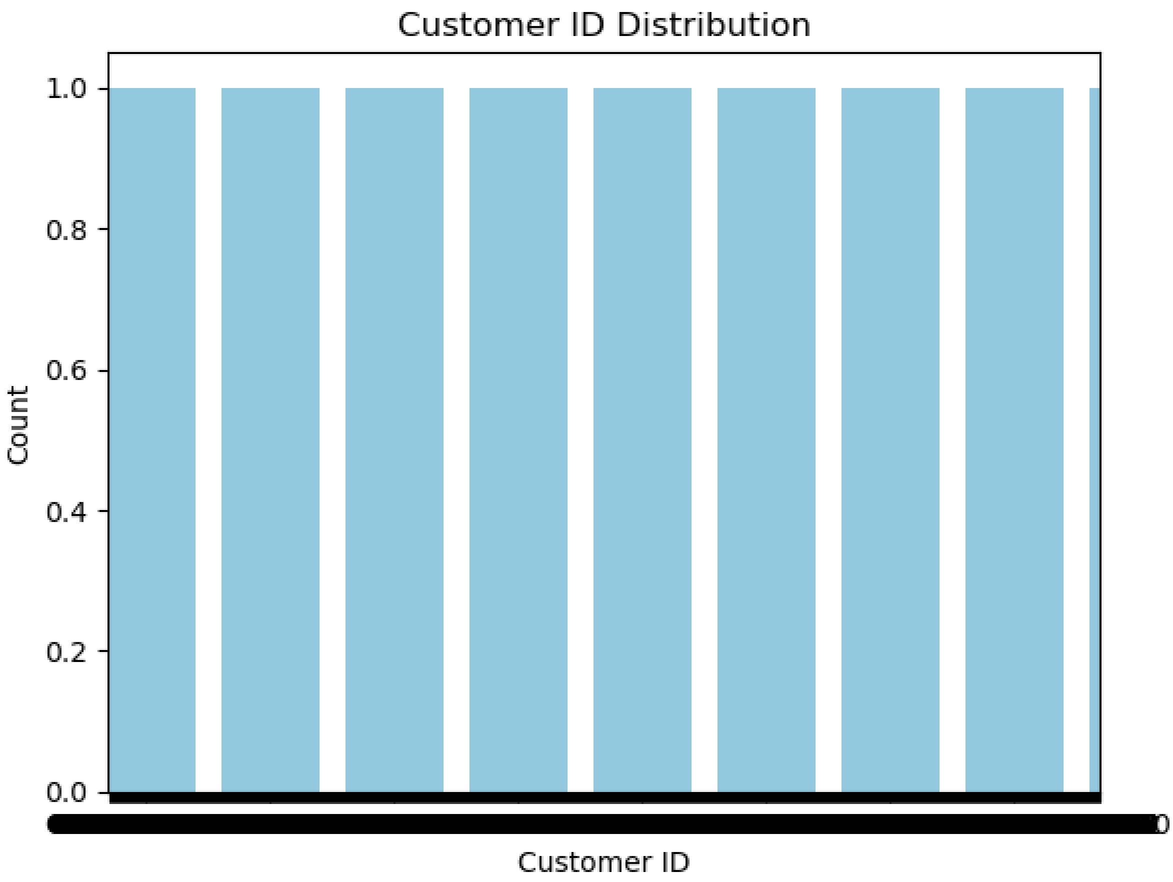


In [27]:

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

sns.barplot(x = rsales['Customer ID'].value_counts().index, y= rsales['Customer ID'].val

plt.xlabel('Customer ID')
plt.ylabel('Count')
plt.title('Customer ID Distribution')
plt.show()
```

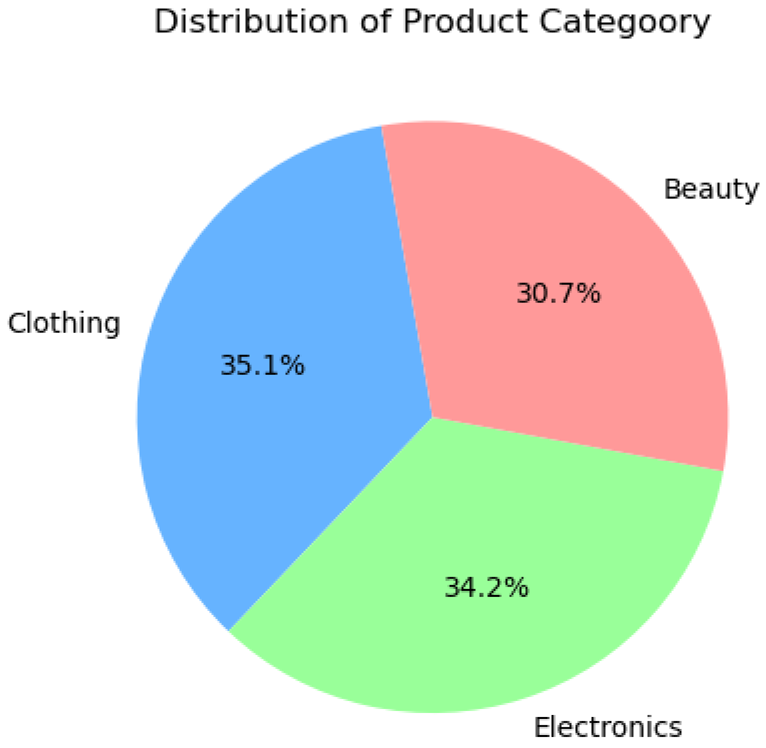
```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
```

```
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



In [28]:

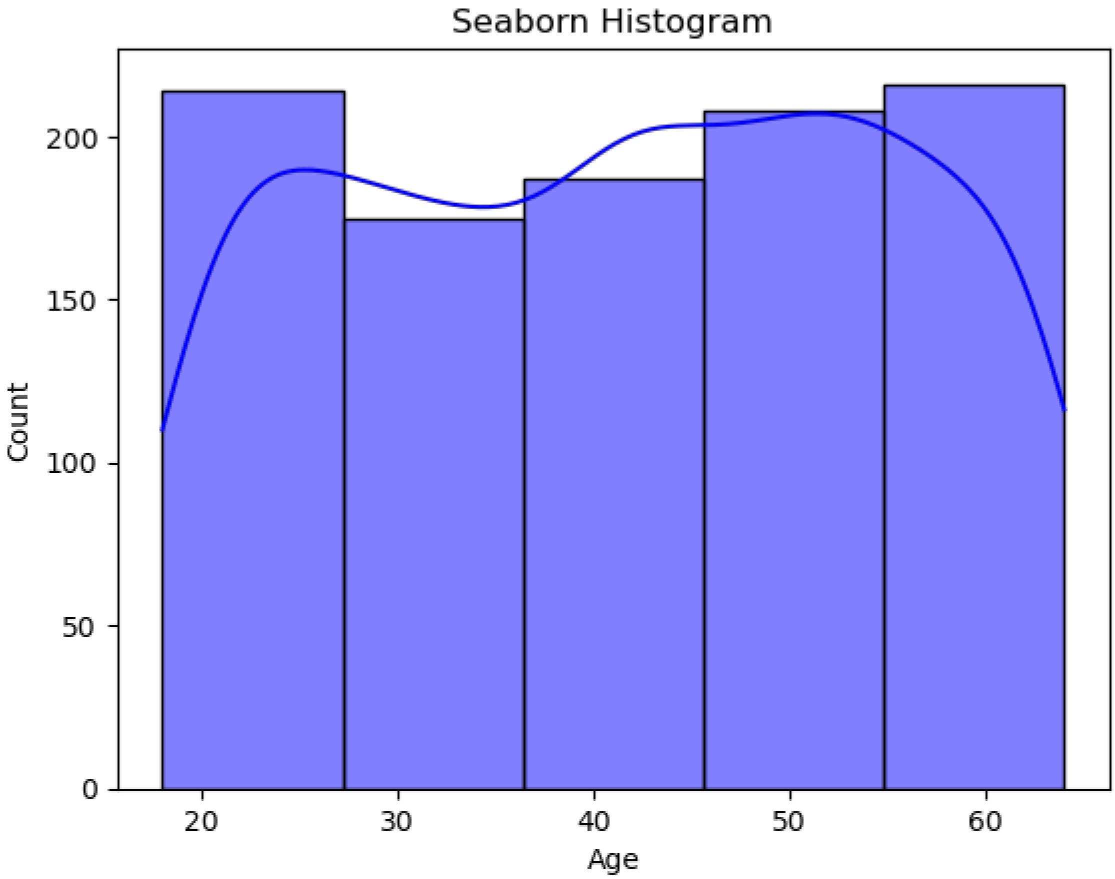
```
plt.pie(
     rsales['Product Category'].value_counts(),
    labels =rsales['Product Category'].value_counts().index,
    colors=['#66b3ff', '#99ff99', '#ff9999'],
    autopct='%1.1f%%',
    startangle=100
)
plt.title("Distribution of Product Categoory")
plt.show()
```



In [29]:
```
##numerical data
```

In [30]:
```
sns.histplot(rsales['Age'], bins=5,kde=True,color='blue')
plt.title('Seaborn Histogram')
plt.show()
```

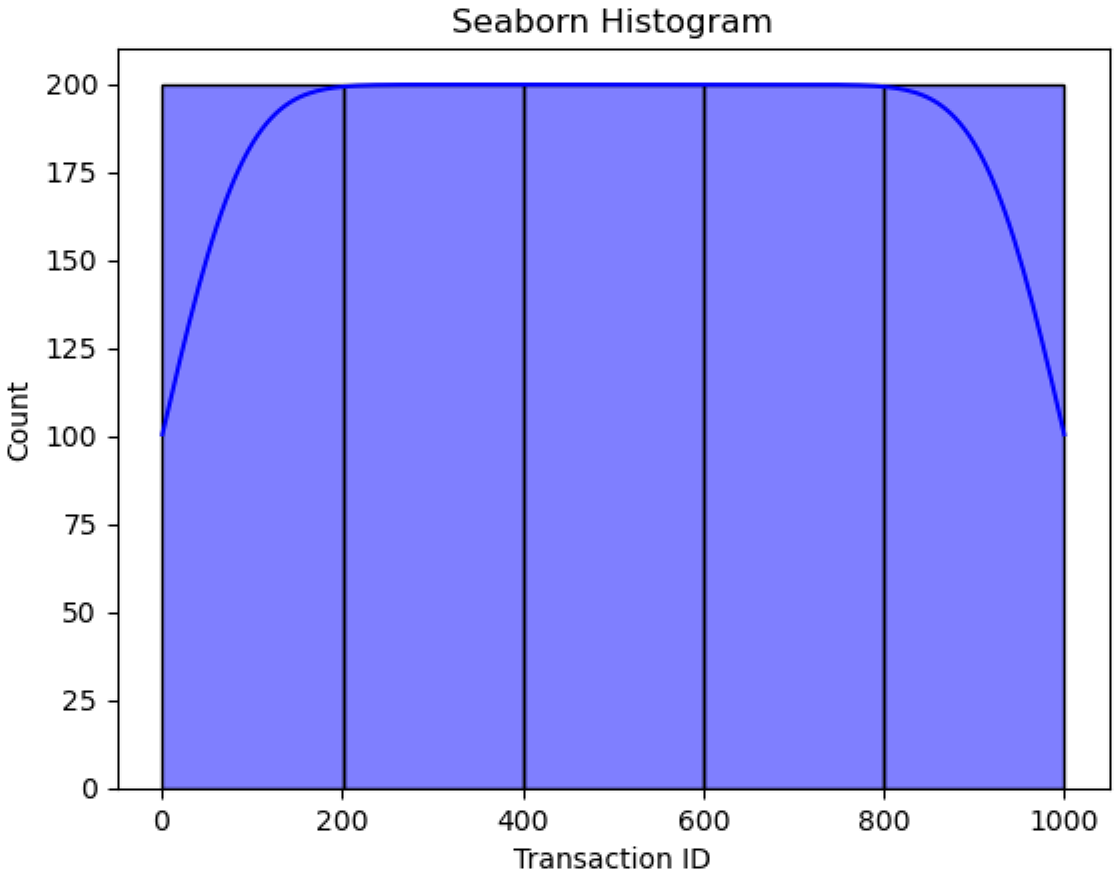
```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [31]:

```
sns.histplot(rsales['Transaction ID'], bins=5,kde=True,color='blue')
plt.title('Seaborn Histogram')
plt.show()
```

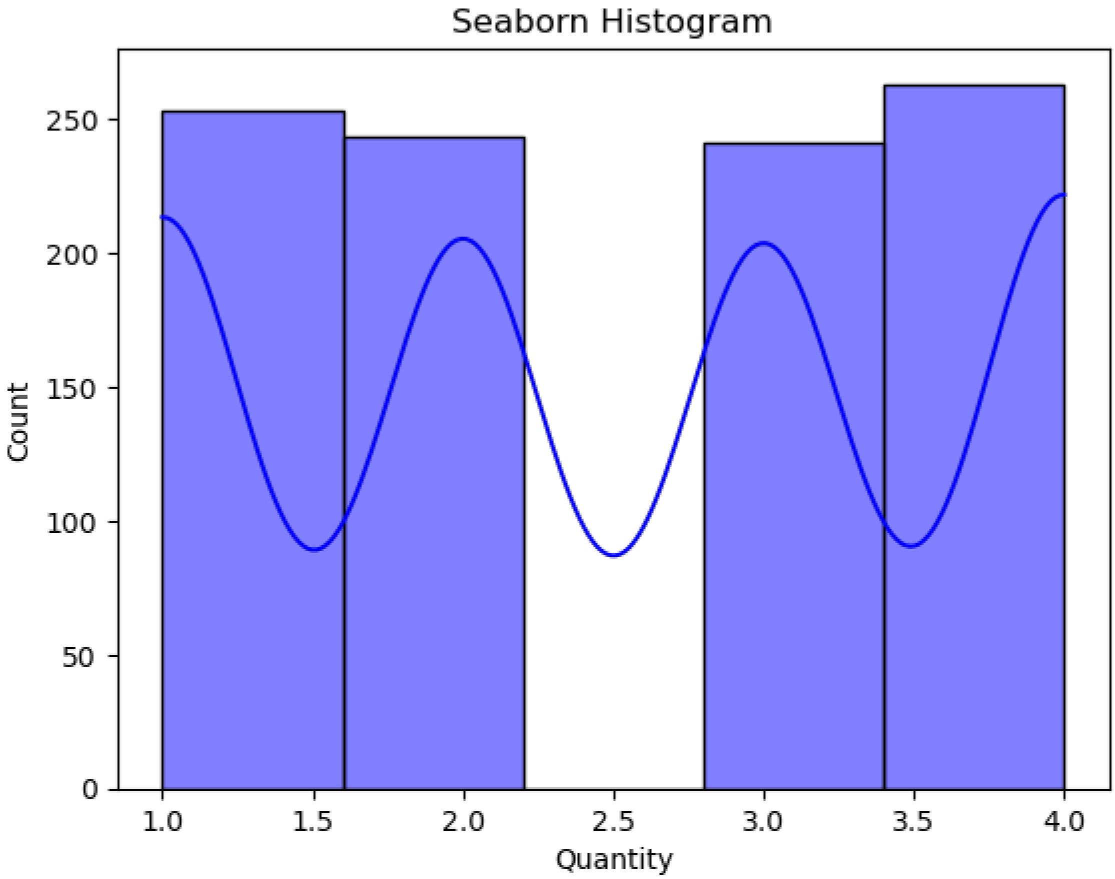
```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [32]:

```
sns.histplot(rsales['Quantity'], bins=5,kde=True,color='blue')
plt.title('Seaborn Histogram')
plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

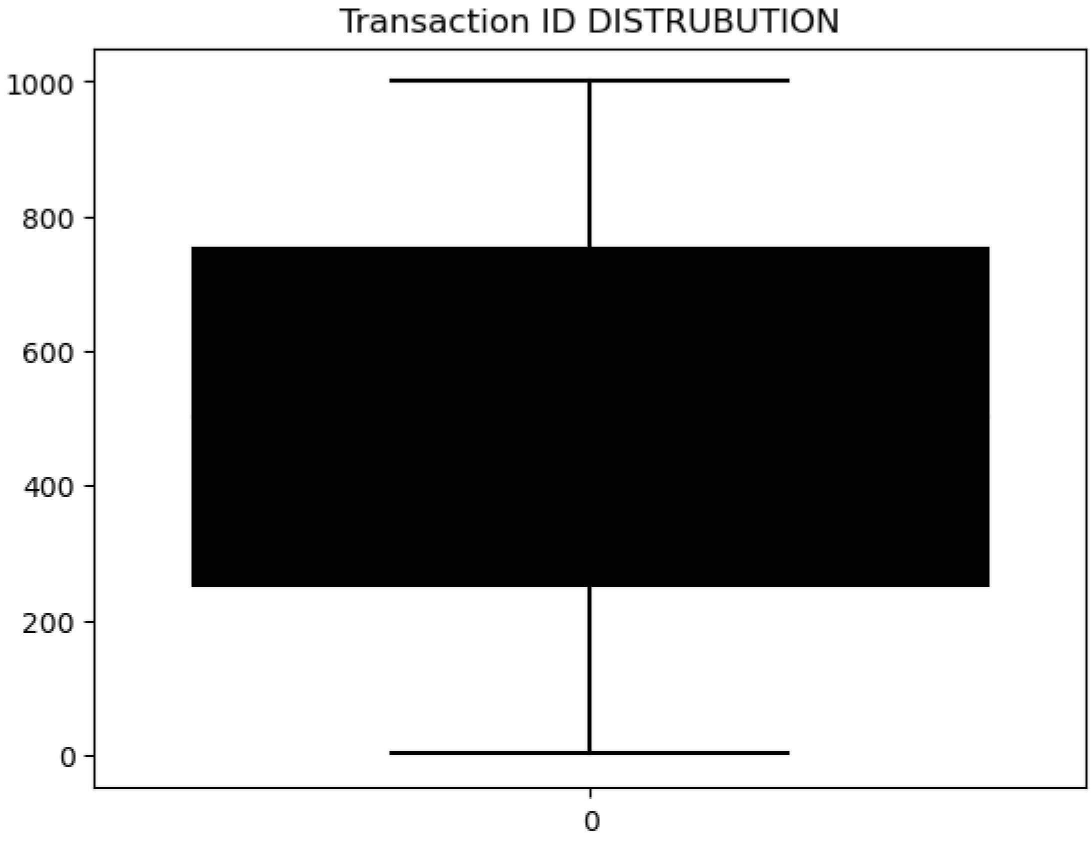


In [33]:

```
#### checking for outliers using the box plot
```

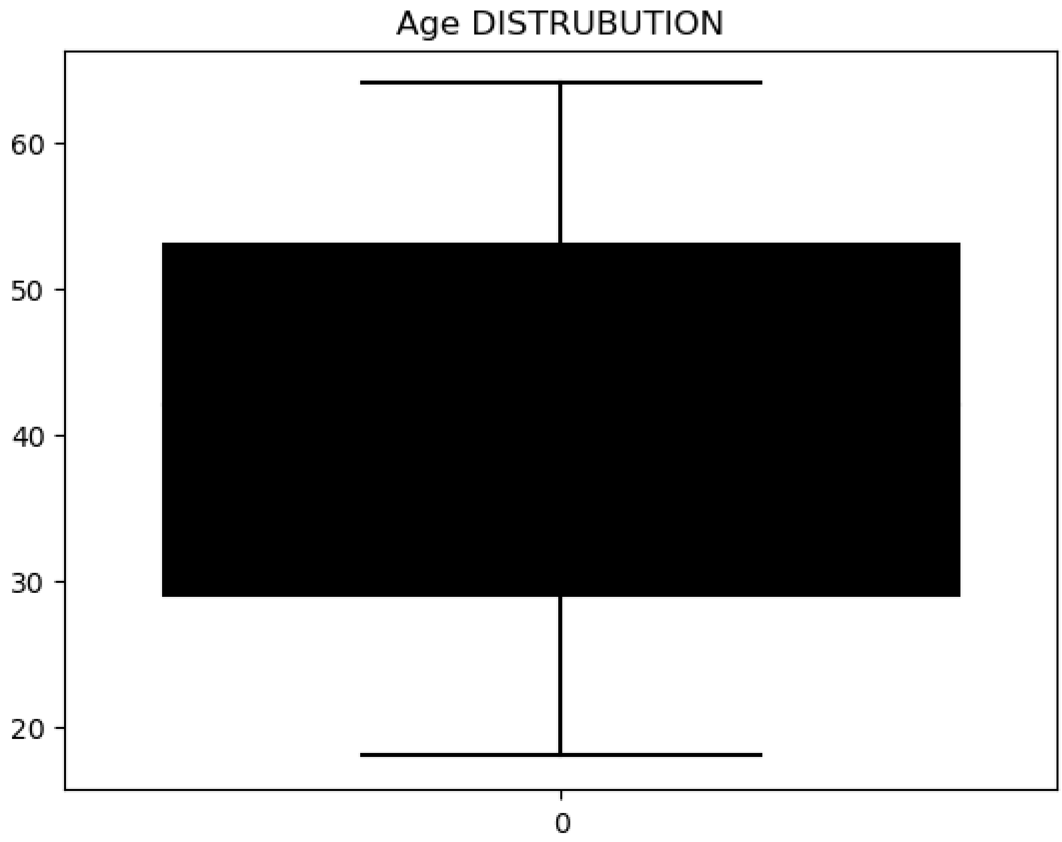
In [34]:

```
sns.boxplot(data=rsales['Transaction ID'], color='black')
plt.title('Transaction ID DISTRUBUTION')
plt.show()
###no outlier detected
```



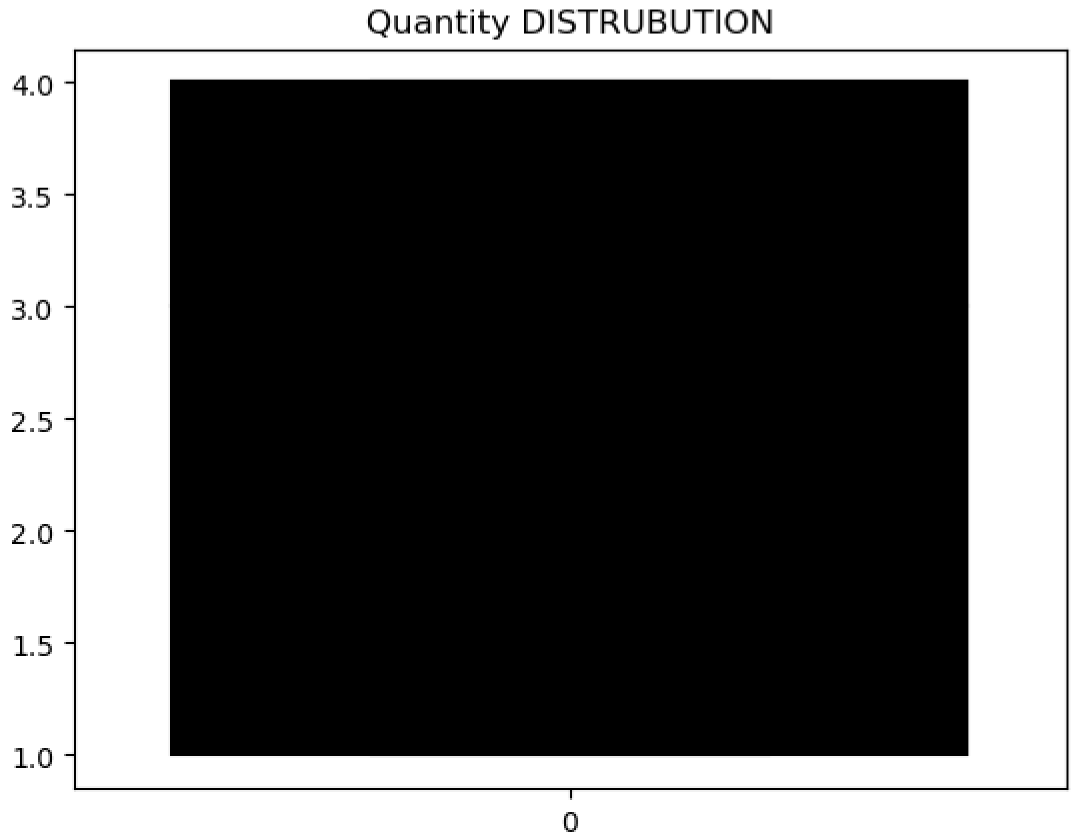
In [35]:

```
sns.boxplot(data=rsales['Age'], color='black')
plt.title('Age DISTRUBUTION')
plt.show()
###no outlier detected
```



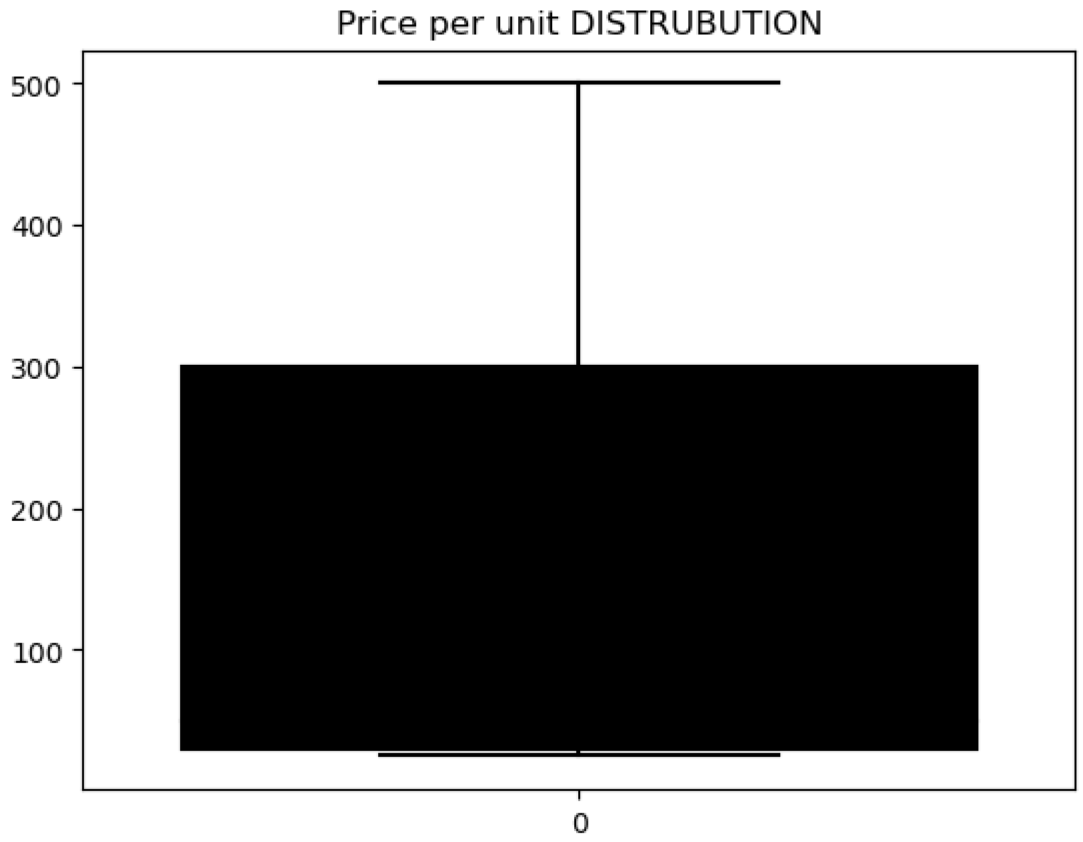
In [36]:

```
sns.boxplot(data=rsales['Quantity'], color='black')
plt.title('Quantity DISTRUBUTION')
plt.show()
###no outlier detected
```



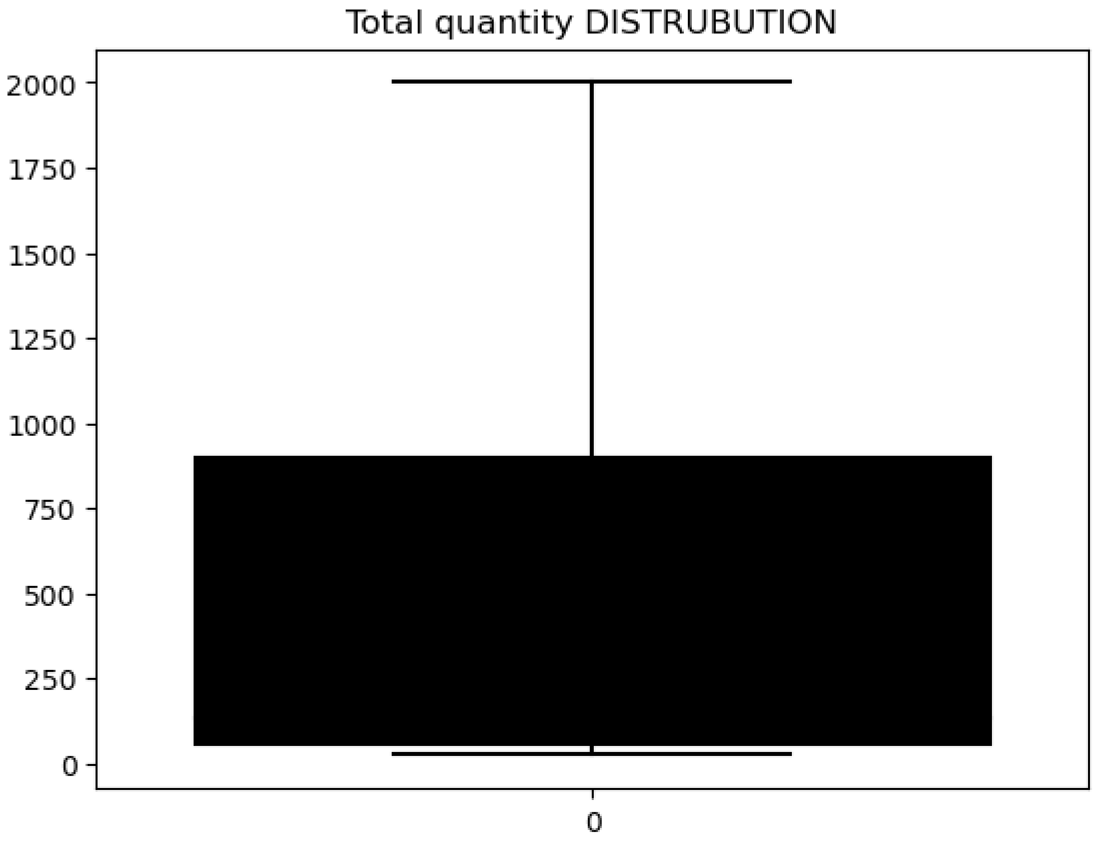
In [37]:

```
    sns.boxplot(data=rsales['Price per Unit'], color='black')
plt.title('Price per unit DISTRUBUTION')
plt.show()
###no outlier detected
```



In [38]:

```
sns.boxplot(data=rsales['Total Amount'], color='black')
plt.title('Total quantity DISTRUBUTION')
plt.show()
###no outlier detected
```



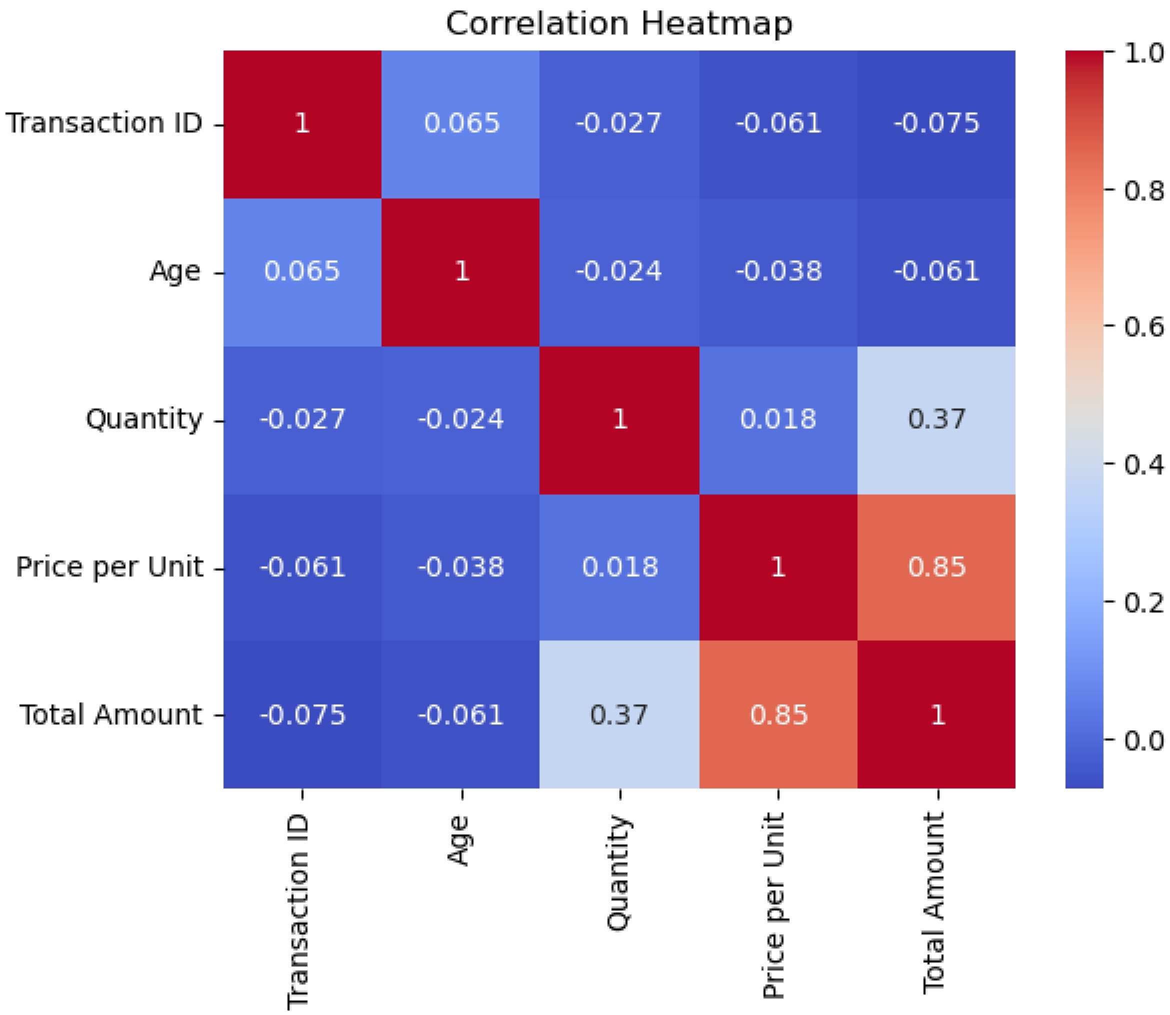
In [ ]:

In [39]:

```
##bivarient analysis
```

In [40]:

```
sns.heatmap(rsales.select_dtypes(include=np.number).corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



In [ ]:

The time date sequenece

In [42]:

```
rsales['Date'] = pd.to_datetime(rsales['Date'], errors='coerce')
print(rsales['Date'].dtypes)
print(rsales['Date'].head())
```

```
datetime64[ns]
0   2023-11-24
1   2023-02-27
2   2023-01-13
3   2023-05-21
4   2023-05-06
Name: Date, dtype: datetime64[ns]
```

```
C:\Users\shadrach\AppData\Local\Temp\ipykernel_14708\2265765993.py:1: UserWarning: Parsi
ng dates in %d/%m/%Y format when dayfirst=False (the default) was specified. Pass `dayfi
rst=True` or specify a format to silence this warning.
  rsales['Date'] = pd.to_datetime(rsales['Date'], errors='coerce')
```

```
### day sequence
```

In [43]:

```
# Convert the 'Date' column to datetime
rsales['Date'] = pd.to_datetime(rsales['Date'], errors='coerce')

# Verify the conversion
print("✅ Date column type:", rsales['Date'].dtypes)
print(rsales['Date'].head())
```

```
✅ Date column type: datetime64[ns]
0   2023-11-24
1   2023-02-27
2   2023-01-13
3   2023-05-21
4   2023-05-06
Name: Date, dtype: datetime64[ns]
```

In [44]:

```
rsales['Year'] = rsales['Date'].dt.year
rsales['Month'] = rsales['Date'].dt.month
rsales['Month_Name'] = rsales['Date'].dt.strftime('%b')
rsales['Week'] = rsales['Date'].dt.isocalendar().week
rsales['Quarter'] = rsales['Date'].dt.quarter
```

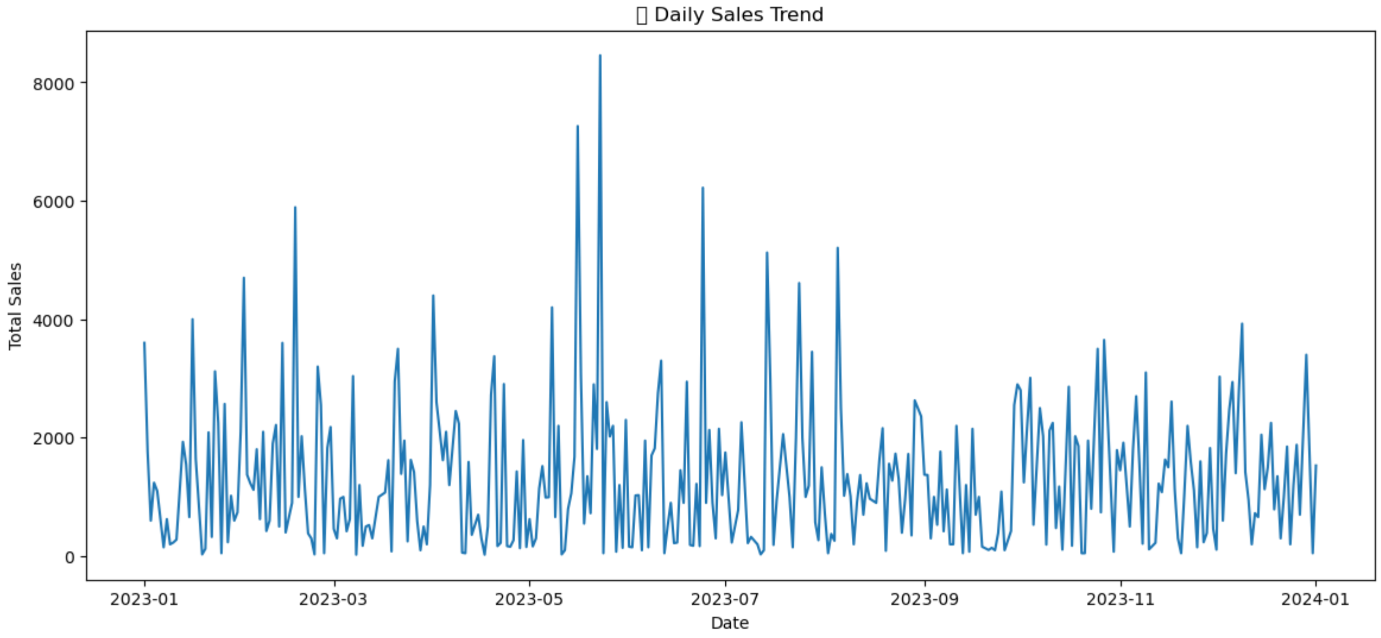
In [45]:

```
# Daily total sales
daily_sales = rsales.groupby('Date')['Total Amount'].sum().reset_index()

# Monthly total sales
monthly_sales = rsales.groupby(['Year', 'Month'])['Total Amount'].sum().reset_index()
monthly_sales['Date'] = pd.to_datetime(monthly_sales[['Year', 'Month']].assign(DAY=1))
```

In [46]:

```
###daily trend

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14,6))
sns.lineplot(x='Date', y='Total Amount', data=daily_sales)
plt.title('📆 Daily Sales Trend')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarnin
g: Glyph 128198 (\N{TEAR-OFF CALENDAR}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



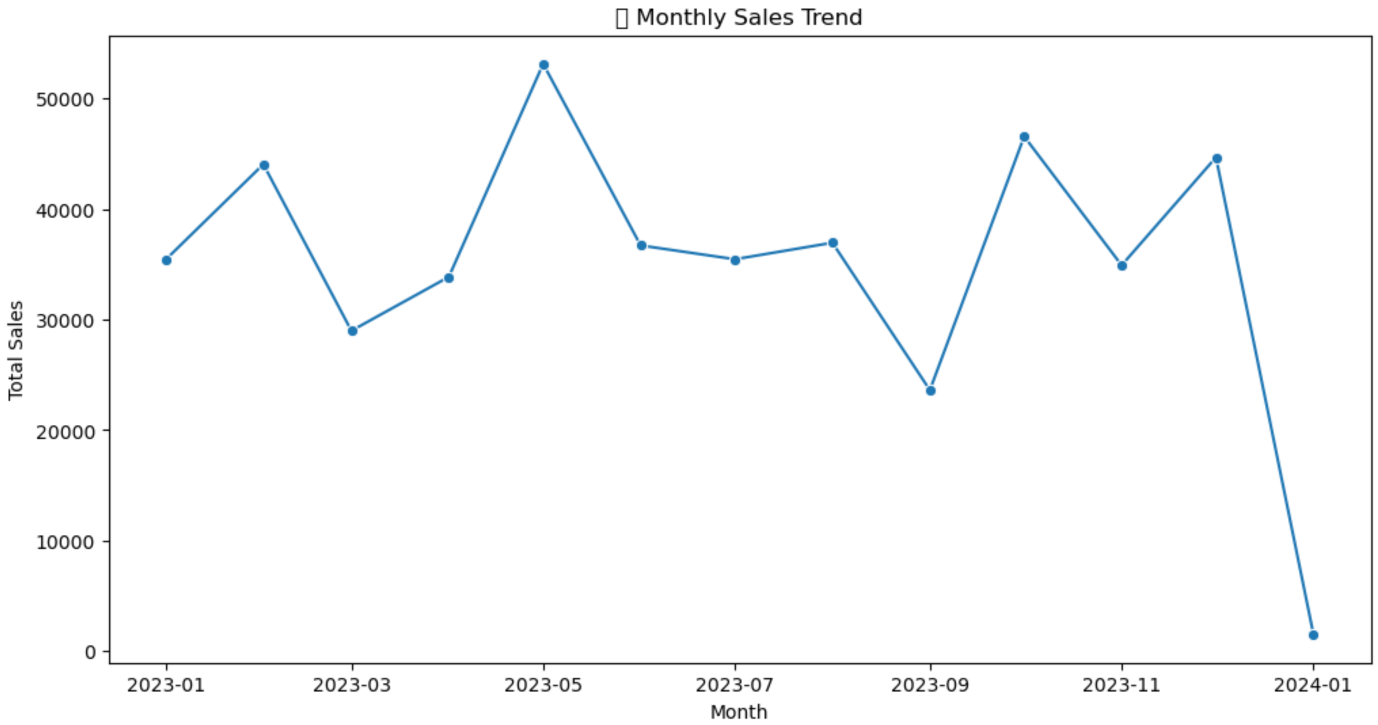
In [47]:

```
#The graph titled "Daily Sales Trend" shows the total daily sales from the beginning of
#Overall Trend: The sales fluctuate significantly on a daily basis throughout the year,
#Peak Sales: The highest sales peaks appear to occur around May and June 2023, with seve
#Sales Range: Most daily sales fall between 0 and 4000, but there are frequent spikes th
#Seasonality: The data shows a high degree of volatility, but there doesn't appear to be
```

In [48]:

```
###monthly trend

plt.figure(figsize=(12,6))
sns.lineplot(x='Date', y='Total Amount', data=monthly_sales, marker='o')
plt.title('📆 Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarnin
g: Glyph 128198 (\N{TEAR-OFF CALENDAR}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [49]:

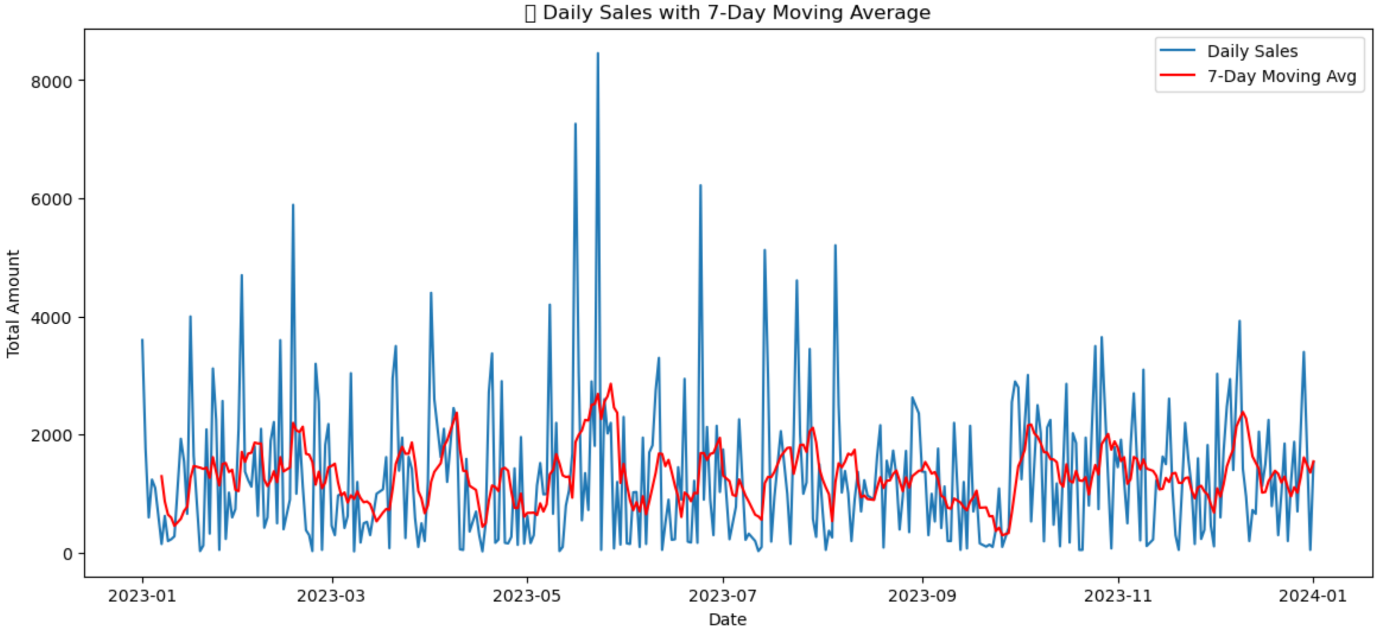
```
#Overall Trend: Sales fluctuated significantly throughout the year, with a general downw
#Peak Sales: The highest sales occurred in May 2023, reaching almost 60,000 units.
#Lowest Sales: The lowest sales were recorded in January 2024, with sales dropping to a
#Significant Fluctuations: There were notable drops in sales in March 2023 and September
```

In [50]:

```
daily_sales['Moving_Avg_7'] = daily_sales['Total Amount'].rolling(window=7).mean()

plt.figure(figsize=(14,6))
sns.lineplot(x='Date', y='Total Amount', data=daily_sales, label='Daily Sales')
sns.lineplot(x='Date', y='Moving_Avg_7', data=daily_sales, label='7-Day Moving Avg', col
plt.title('▯ Daily Sales with 7-Day Moving Average')
plt.legend()
plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
```

```
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarnin
g: Glyph 128201 (\N{CHART WITH DOWNWARDS TREND}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



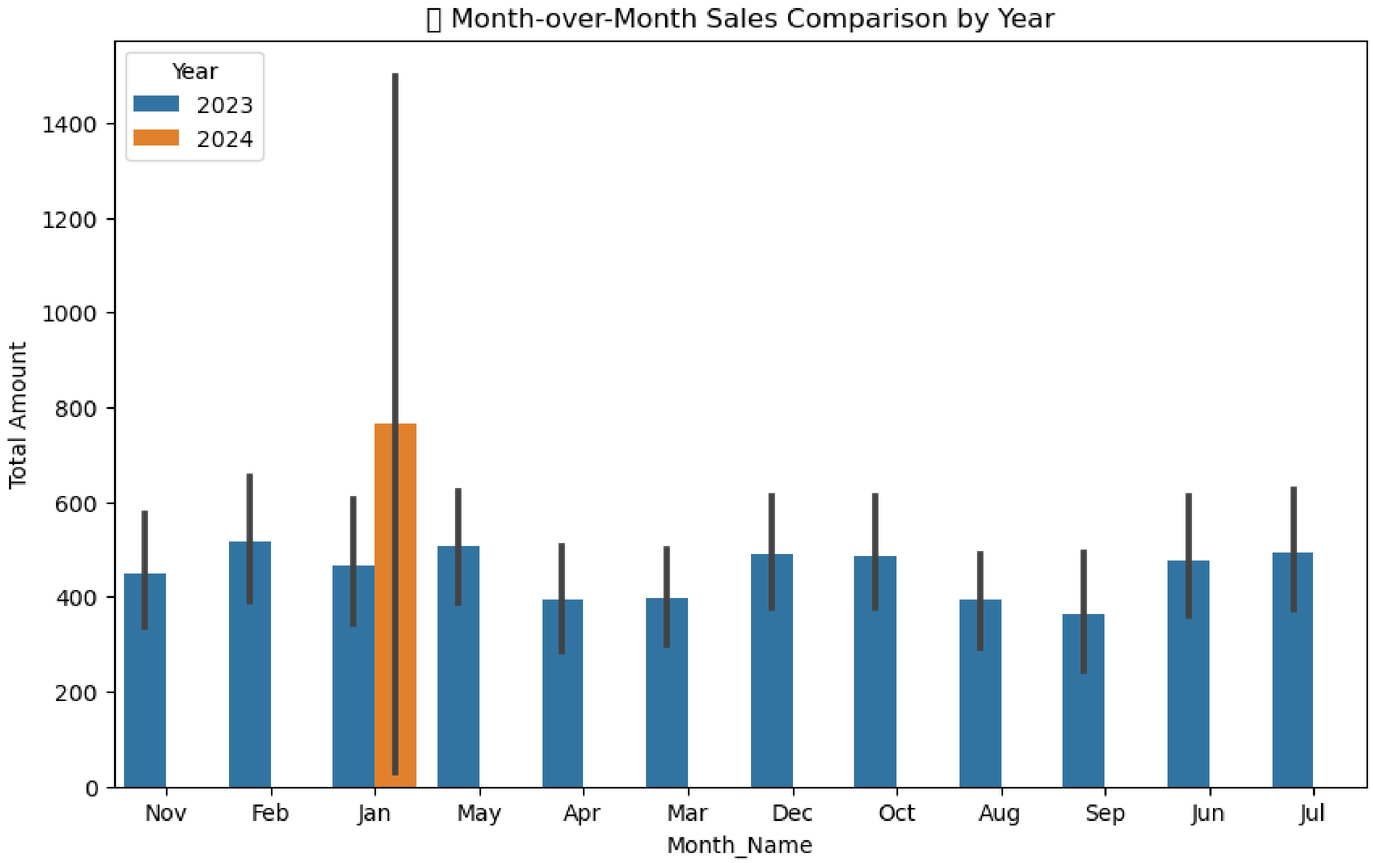
In [51]:

```
#Daily Sales (Blue Line): This line shows significant day-to-day fluctuations, with seve
#7-Day Moving Average (Red Line): This line smooths out the daily fluctuations, providin
```

In [52]:

```
plt.figure(figsize=(10,6))
sns.barplot(x='Month_Name', y='Total Amount', hue='Year', data=rsales)
plt.title('🗓 Month-over-Month Sales Comparison by Year')
plt.show()

###to be discarded reason below
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarnin
g: Glyph 128467 (\N{SPIRAL CALENDAR PAD}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [53]:

```
#The primary reasons for discarding the graph are:
###Inconsistent Data: The graph compares sales data for two years, 2023 and 2024, but on
#Incorrect Order: The months on the x-axis are not in chronological order (Nov, Feb, Jan
#Misleading Title: The title "Month-over-Month Sales Comparison by Year" implies a compr
```

In [ ]:

Customer & Product Analysis

In [55]:

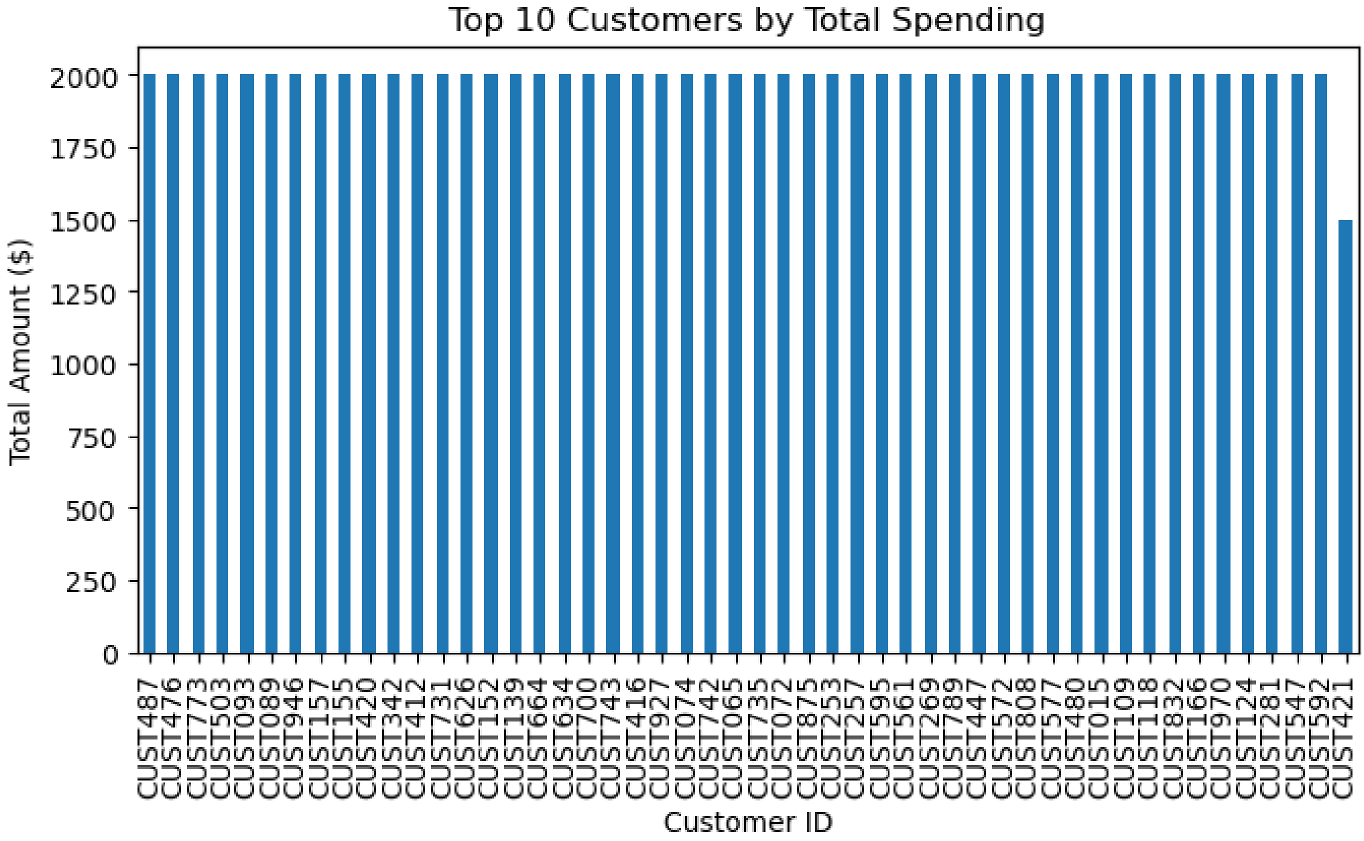
```
# Top 10 customers by total purchase amount
top_customers = rsales.groupby('Customer ID')['Total Amount'].sum().sort_values(ascendin
print(top_customers)
```

```
Customer ID
CUST487    2000
CUST476    2000
CUST773    2000
CUST503    2000
CUST093    2000
CUST089    2000
CUST946    2000
CUST157    2000
CUST155    2000
CUST420    2000
CUST342    2000
CUST412    2000
CUST731    2000
CUST626    2000
CUST152    2000
CUST139    2000
CUST664    2000
CUST634    2000
CUST700    2000
CUST743    2000
```

```
CUST416    2000
CUST927    2000
CUST074    2000
CUST742    2000
CUST065    2000
CUST735    2000
CUST072    2000
CUST875    2000
CUST253    2000
CUST257    2000
CUST595    2000
CUST561    2000
CUST269    2000
CUST789    2000
CUST447    2000
CUST572    2000
CUST808    2000
CUST577    2000
CUST480    2000
CUST015    2000
CUST109    2000
CUST118    2000
CUST832    2000
CUST166    2000
CUST970    2000
CUST124    2000
CUST281    2000
CUST547    2000
CUST592    2000
CUST421    1500
Name: Total Amount, dtype: int64
```

In [56]:

```
# Visualization
top_customers.plot(kind='bar', figsize=(8,4), title='Top 10 Customers by Total Spending'
plt.xlabel('Customer ID')
plt.ylabel('Total Amount ($)')
plt.show()
```

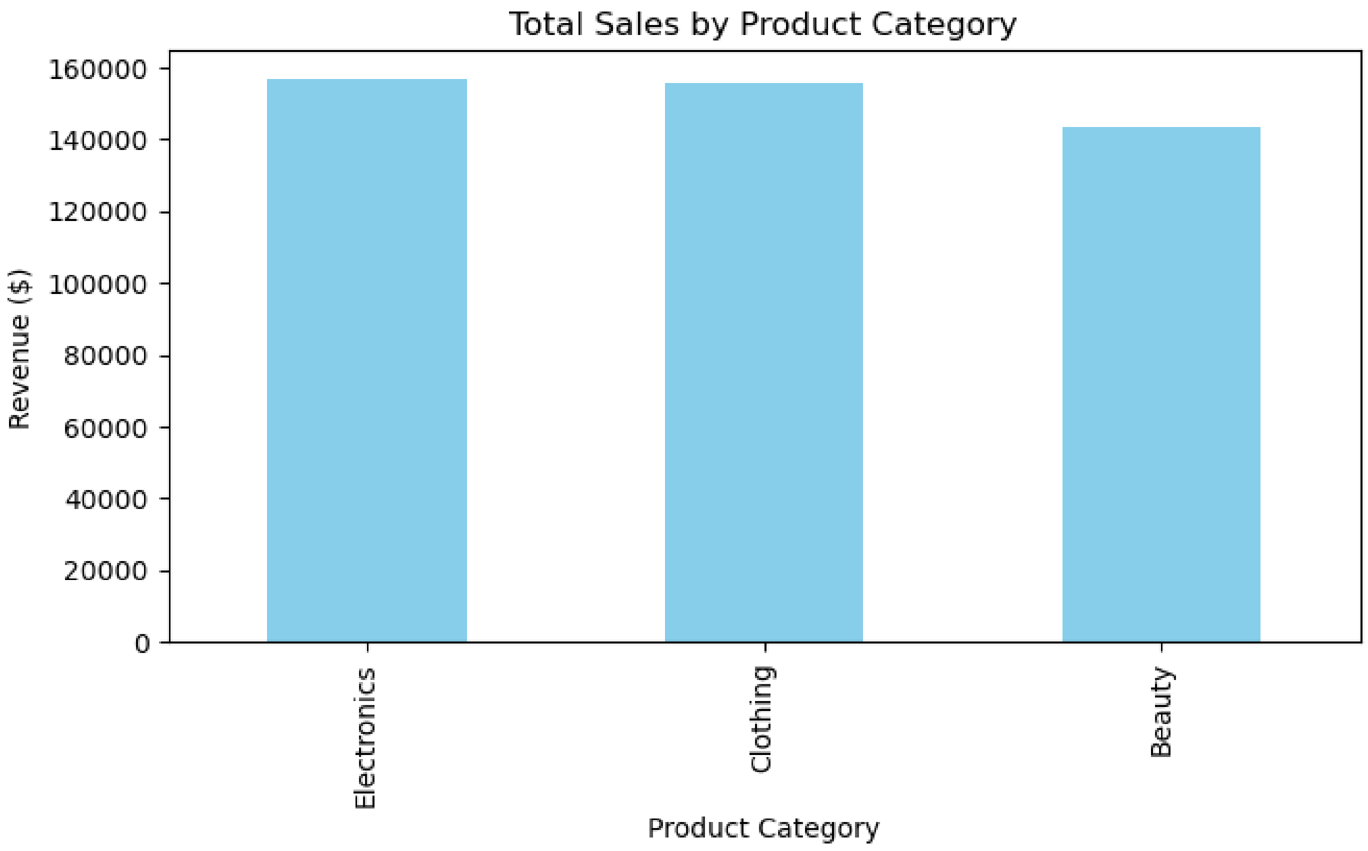


In [57]:
```
#### This information also is not important and does not add anytghing to our analysis


##The dataset might be synthetic or uniformly generated, which fits your dataset descrip

##The “Total Amount” may have been fixed per customer (e.g., a cap or simulated ceiling)
```

In [58]:
```
# Total sales and quantity by product category
category_sales = rsales.groupby('Product Category').agg({
    'Quantity': 'sum',
    'Total Amount': 'sum'
}).sort_values(by='Total Amount', ascending=False)

print(category_sales)
```

```
                  Quantity  Total Amount
Product Category
Electronics            849        156905
Clothing               894        155580
Beauty                 771        143515
```

In [59]:
```
# Visualization
category_sales['Total Amount'].plot(kind='bar', color='skyblue', figsize=(8,4))
plt.title('Total Sales by Product Category')
plt.ylabel('Revenue ($)')
plt.show()
```



In [60]:
```
#There are three main product categories in the dataset: Electronics, Clothing, and Beau

##Electronics recorded the highest total sales, generating revenue of nearly $160,000.

##Clothing followed closely, with total sales slightly below Electronics, also around $1

##Beauty ranked third, with total revenue a little above $140,000, making it the lowest
```

In [61]:
```
###Customer Segmentation (RFM Analysis)
#Reason for carring thia out
```

```
#Customers with low Recency, high Frequency, and high Monetary scores are the most valua
#Segmenting these groups helps design loyalty and reward programs.
```

In [62]:

```
# Latest date for reference
latest_date = rsales['Date'].max()
latest_date
```

Out[62]:

```
Timestamp('2024-01-01 00:00:00')
```

In [63]:

```
# RFM table
rfm = rsales.groupby('Customer ID').agg({
    'Date': lambda x: (latest_date - x.max()).days,   # Recency
    'Transaction ID': 'count',                        # Frequency
    'Total Amount': 'sum'                             # Monetary
}).rename(columns={
    'Date': 'Recency',
    'Transaction ID': 'Frequency',
    'Total Amount': 'Monetary'
})
```

In [64]:

```
rfm.head(50)
```

Out[64]:

	Recency	Frequency	Monetary
Customer ID			
CUST001	38	1	150
CUST002	308	1	1000
CUST003	353	1	30
CUST004	225	1	500
CUST005	240	1	100
CUST006	251	1	30
CUST007	294	1	50
CUST008	313	1	100
CUST009	19	1	600
CUST010	86	1	200
CUST011	321	1	100
CUST012	63	1	75
CUST013	149	1	1500
CUST014	349	1	120
CUST015	350	1	2000
CUST016	318	1	1500
CUST017	254	1	100
CUST018	246	1	50
CUST019	107	1	50
CUST020	57	1	900
CUST021	352	1	500
CUST022	78	1	100

CUST023	264	1	120
CUST024	33	1	300
CUST025	6	1	50
CUST026	86	1	1000
CUST027	151	1	50
CUST028	253	1	500
CUST029	136	1	30
CUST030	64	1	900
CUST031	223	1	1200
CUST032	362	1	90
CUST033	284	1	100
CUST034	8	1	150
CUST035	149	1	900
CUST036	191	1	900
CUST037	223	1	75
CUST038	286	1	200
CUST039	255	1	120
CUST040	193	1	50
CUST041	313	1	50
CUST042	318	1	900
CUST043	171	1	300
CUST044	316	1	25
CUST045	182	1	30
CUST046	189	1	1200
CUST047	56	1	1500
CUST048	230	1	900
CUST049	343	1	1000
CUST050	130	1	75

In [69]:

```
# : create RFM score
rfm['R_Score'] = pd.qcut(rfm['Recency'], 4, labels=[4,3,2,1]).astype(int)
rfm['F_Score'] = pd.qcut(rfm['Frequency'].rank(method='first'), 4, labels=[1,2,3,4]).ast
rfm['M_Score'] = pd.qcut(rfm['Monetary'], 4, labels=[1,2,3,4]).astype(int)

# Combine to numeric score
rfm['RFM_Score_Num'] = rfm['R_Score'] + rfm['F_Score'] + rfm['M_Score']

# Optional: combine as string
rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) + rfm['M_Scor
```

In [70]:

```
 ##Sort by RFM score
top_50_rfm = rfm.sort_values(by='RFM_Score_Num', ascending=False).head(50)
print(top_50_rfm)
```

```
             Recency  Frequency  Monetary  R_Score  M_Score RFM_Score  \
Customer ID
CUST869           68          1      1500        4        4       444
CUST994           14          1      1000        4        4       444
CUST757            7          1      1200        4        4       444
CUST805            3          1      1500        4        4       444
CUST908            3          1      1200        4        4       444
CUST943           77          1      1200        4        4       444
CUST828           23          1      1200        4        4       444
CUST999           27          1       150        4        3       443
CUST662           10          1      1000        4        4       434
CUST937           70          1       500        4        3       443
CUST938           43          1       200        4        3       443
CUST939           14          1       300        4        3       443
CUST634           85          1      2000        4        4       434
CUST795           34          1       300        4        3       443
CUST950           55          1       900        4        3       443
CUST608           30          1      1500        4        4       434
CUST600           71          1      1000        4        4       434
CUST664            4          1      2000        4        4       434
CUST784           58          1       500        4        3       443
CUST677           66          1      1500        4        4       434
CUST789           93          1      2000        3        4       344
CUST595           53          1      2000        4        4       434
CUST783           15          1       300        4        3       443
CUST700           23          1      2000        4        4       434
CUST781            9          1       500        4        3       443
CUST710           62          1      1500        4        4       434
CUST711           77          1      1500        4        4       434
CUST735           89          1      2000        4        4       434
CUST756          127          1      1200        3        4       344
CUST761           55          1       500        4        3       443
CUST914           82          1       500        4        3       443
CUST777           12          1       150        4        3       443
CUST956          135          1      1500        3        4       344
CUST773          162          1      2000        3        4       344
CUST968           45          1       900        4        3       443
CUST827           53          1       900        4        3       443
CUST868           26          1       300        4        3       443
CUST983           61          1       300        4        3       443
CUST533           46          1      1500        4        4       434
CUST507           60          1      1500        4        4       434
CUST880          133          1      1000        3        4       344
CUST976           83          1       600        4        3       443
CUST844           81          1       150        4        3       443
CUST971           27          1       200        4        3       443
CUST832          112          1      2000        3        4       344
CUST875          148          1      2000        3        4       344
CUST865           11          1       300        4        3       443
CUST842            6          1       600        4        3       443
CUST503           68          1      2000        4        4       434
CUST965           53          1       200        4        3       443

             F_Score  RFM_Score_Num
Customer ID
CUST869            4             12
CUST994            4             12
CUST757            4             12
CUST805            4             12
CUST908            4             12
CUST943            4             12
CUST828            4             12
CUST999            4             11
CUST662            3             11
CUST937            4             11
CUST938            4             11
```

```
CUST939          4           11
CUST634          3           11
CUST795          4           11
CUST950          4           11
CUST608          3           11
CUST600          3           11
CUST664          3           11
CUST784          4           11
CUST677          3           11
CUST789          4           11
CUST595          3           11
CUST783          4           11
CUST700          3           11
CUST781          4           11
CUST710          3           11
CUST711          3           11
CUST735          3           11
CUST756          4           11
CUST761          4           11
CUST914          4           11
CUST777          4           11
CUST956          4           11
CUST773          4           11
CUST968          4           11
CUST827          4           11
CUST868          4           11
CUST983          4           11
CUST533          3           11
CUST507          3           11
CUST880          4           11
CUST976          4           11
CUST844          4           11
CUST971          4           11
CUST832          4           11
CUST875          4           11
CUST865          4           11
CUST842          4           11
CUST503          3           11
CUST965          4           11
```

In [71]:

```
##The scoring variation (55 vs 54) reflects slight differences in spending or recency, n
```

In [72]:

```
#Strategic Recommendation
#Businesses should maintain engagement with the 55 group through appreciation and exclus
```

In [73]:

```
# Spending by Gender
gender_sales = rsales.groupby('Gender')['Total Amount'].sum().sort_values(ascending=Fals

gender_sales
```

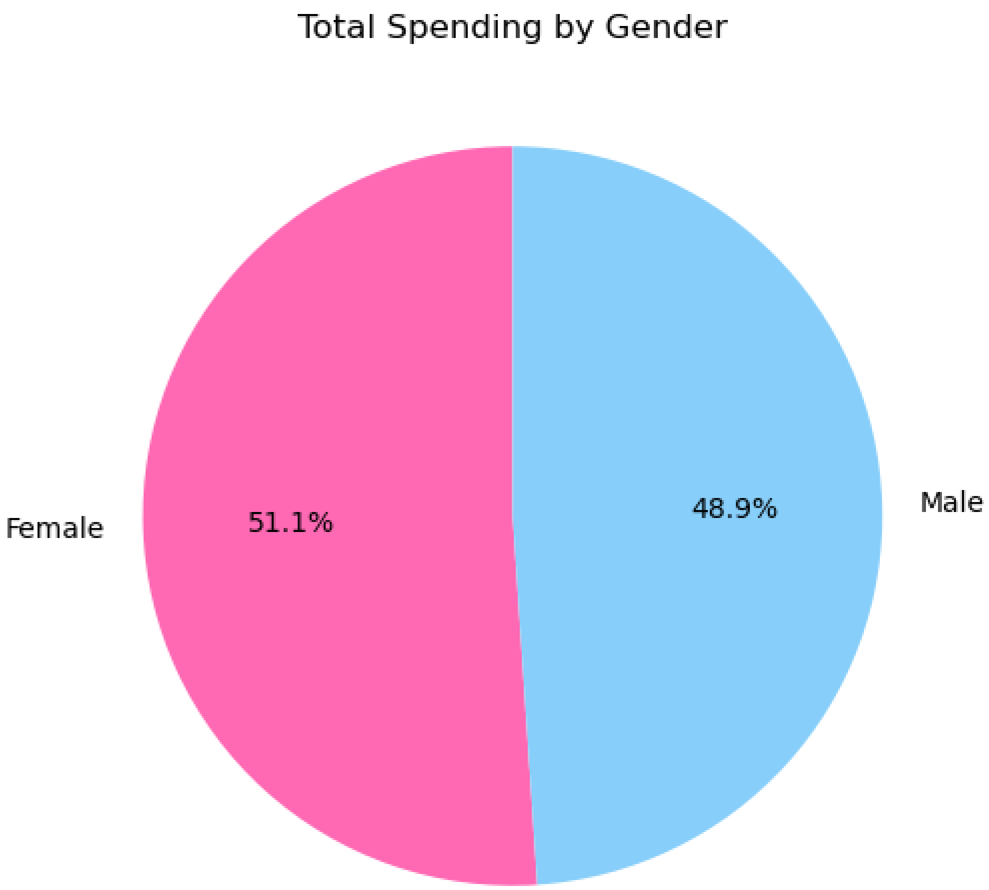
Out[73]:

```
Gender
Female    232840
Male      223160
Name: Total Amount, dtype: int64
```

In [74]:

```
gender_spending = {'Female': 232840, 'Male': 223160}

## Create pie chart
plt.figure(figsize=(6, 6))
plt.pie(gender_spending.values(),
        labels=gender_spending.keys(),
        autopct='%1.1f%%',
        startangle=90,
        colors=['#FF69B4', '#87CEFA'])
```

```
plt.title('Total Spending by Gender')
plt.show()
```



In [75]:
```
## The bar chart  shows total spending distribution by gender.

## Female customers have a slightly higher total spending (₦232,840)
## compared to male customers (₦223,160).

## This suggests that female shoppers contribute marginally more to overall revenue.
## It could indicate stronger purchasing frequency, interest in certain product categori
## (like Beauty or Clothing), or a tendency toward higher-value transactions.

## Male customers, while spending slightly less in total,
## still represent a substantial portion of the customer base,
## showing balanced engagement across genders.

## Overall, spending patterns between males and females are relatively close,
## implying that both genders are important target groups for the retail business.
## Marketing strategies could therefore be gender-inclusive,
## while specific campaigns (e.g., product-based) can still be tailored by gender prefer
```

In [76]:
```
# Spending by Age Group

#creating age group
bins = [17, 25, 35, 45, 55, 100]
labels = ['18-25 (Youth)', '26-35 (Young Adult)', '36-45 (Adult)', '46-55 (Middle-aged)'

rsales['Age Group'] = pd.cut(rsales['Age'], bins=bins, labels=labels)
```

In [77]:
```
rsales['Age Group'] = pd.cut(rsales['Age'], bins=bins, labels=labels)
age_sales = rsales.groupby('Age Group')['Total Amount'].sum().sort_values(ascending=Fals
age_sales
```

```
C:\Users\shadrach\AppData\Local\Temp\ipykernel_14708\3726019061.py:2: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future version
of pandas. Pass observed=False to retain current behavior or observed=True to adopt the
future default and silence this warning.
  age_sales = rsales.groupby('Age Group')['Total Amount'].sum().sort_values(ascending=Fa
lse)
```

Out[77]:

```
Age Group
46-55 (Middle-aged)    100690
26-35 (Young Adult)     98480
36-45 (Adult)           91870
18-25 (Youth)           84550
56+ (Senior)            80410
Name: Total Amount, dtype: int64
```

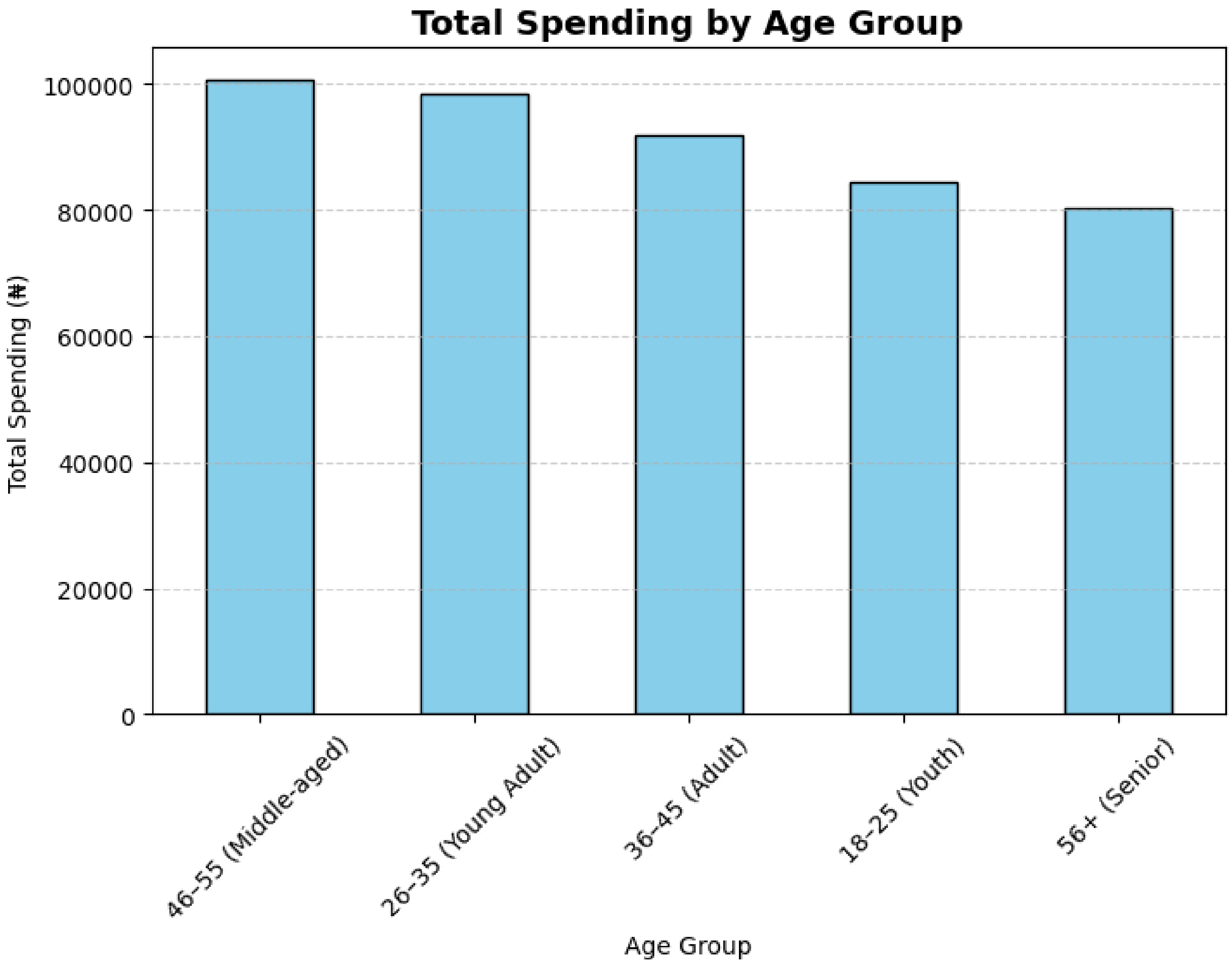
In [ ]:

In [78]:

```python
age_group_sales = {
    '46-55 (Middle-aged)': 100690,
    '26-35 (Young Adult)': 98480,
    '36-45 (Adult)': 91870,
    '18-25 (Youth)': 84550,
    '56+ (Senior)': 80410
}
```

In [79]:

```python
import pandas as pd
age_group_sales = pd.Series(age_group_sales)


plt.figure(figsize=(8, 5))
age_group_sales.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title('Total Spending by Age Group', fontsize=14, fontweight='bold')
plt.xlabel('Age Group')
plt.ylabel('Total Spending (₦)')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.show()
```



In [80]:

```
## The bar chart shows total spending by different age groups.

## Middle-aged customers (46–55 years) are the highest spenders,
## contributing over ₦100,000 in total revenue.
## This indicates that this group has strong purchasing power
## and likely engages in consistent, high-value transactions.

## Young adults (26–35 years) follow closely behind,
## showing that they are also a key consumer segment.
## Their near-equal spending suggests strong engagement,
## possibly influenced by lifestyle, convenience, and digital marketing.

## Adults aged 36–45 also spend considerably,
## representing a financially stable, family-oriented audience
## that contributes significantly to overall revenue.

## Youths (18–25 years) spend less compared to older groups,
## likely due to lower disposable income or financial dependence.
## However, they represent long-term potential for brand loyalty
## if targeted with youth-oriented promotions.

## Seniors (56+ years) record the lowest spending overall,
## which could reflect fixed incomes or lower consumption frequency.

## Overall, the 26–55 age range forms the main revenue backbone,
## suggesting marketing and product strategies should prioritize
## these middle-aged and young adult groups,
## while also developing approaches to engage younger and older customers.
```

In [81]:

```
rsales
```

Out[81]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount	Year	Month	Month_Nar
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150	2023	11	N
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000	2023	2	F
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30	2023	1	J
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500	2023	5	M
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100	2023	5	M
...	...	...	...	...	...	...	...	...	...	...	...	
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50	2023	5	M
996	997	2023-11-17	CUST997	Male	52	Beauty	3	30	90	2023	11	N
997	998	2023-10-29	CUST998	Female	23	Beauty	4	25	100	2023	10	C
998	999	2023-12-05	CUST999	Female	36	Electronics	3	50	150	2023	12	D
999	1000	2023-04-12	CUST1000	Male	47	Electronics	4	30	120	2023	4	A

1000 rows × 15 columns

In [82]:

```
top_products = rsales.groupby('Product Category')['Total Amount'].sum() \
                         .sort_values(ascending=False).head(10)
top_products
```

Out[82]:

```
Product Category
Electronics    156905
Clothing       155580
Beauty         143515
Name: Total Amount, dtype: int64
```

In [ ]:

```

```

In [ ]:

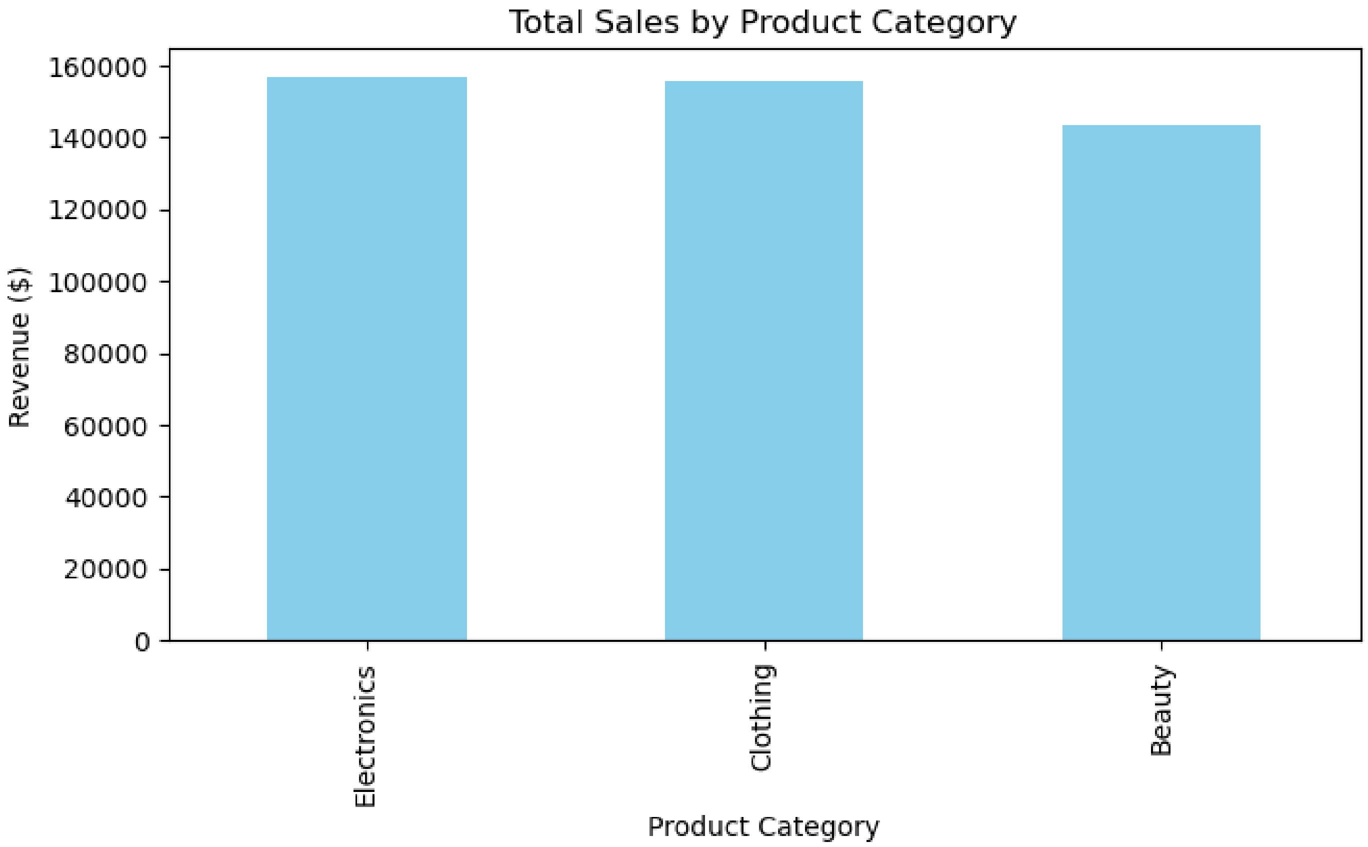
```

```

In [83]:

```
###collectively this visuals
#3SUMMARY
```

In [84]:

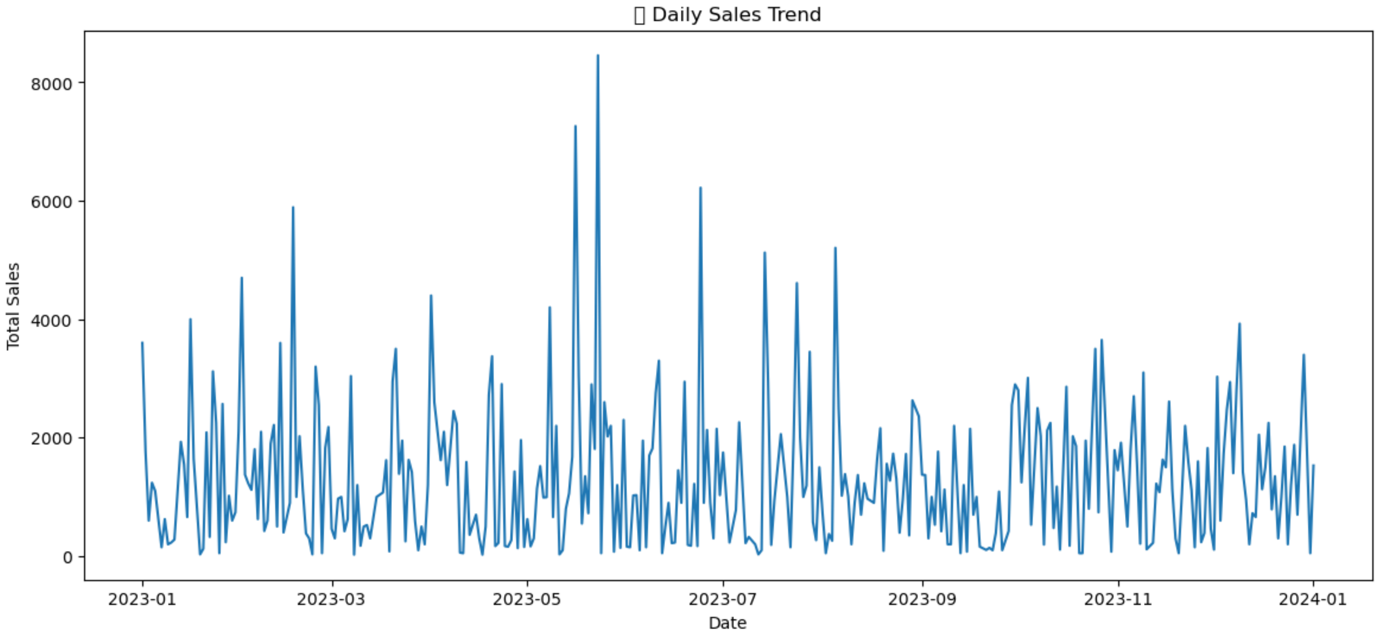
```
category_sales['Total Amount'].plot(kind='bar', color='skyblue', figsize=(8,4))
plt.title('Total Sales by Product Category')
plt.ylabel('Revenue ($)')
plt.show()
```



In [85]:

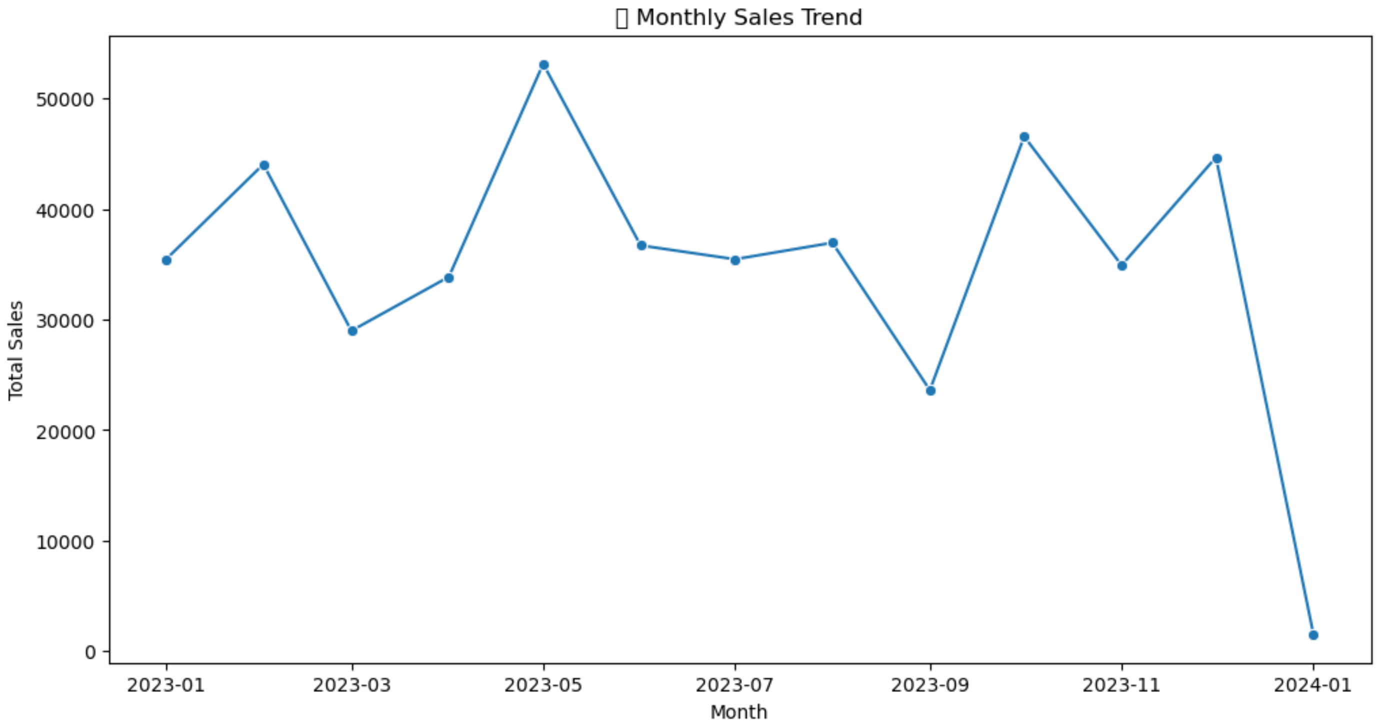
```
plt.figure(figsize=(14,6))
sns.lineplot(x='Date', y='Total Amount', data=daily_sales)
plt.title('📆 Daily Sales Trend')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarnin
g: Glyph 128198 (\N{TEAR-OFF CALENDAR}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



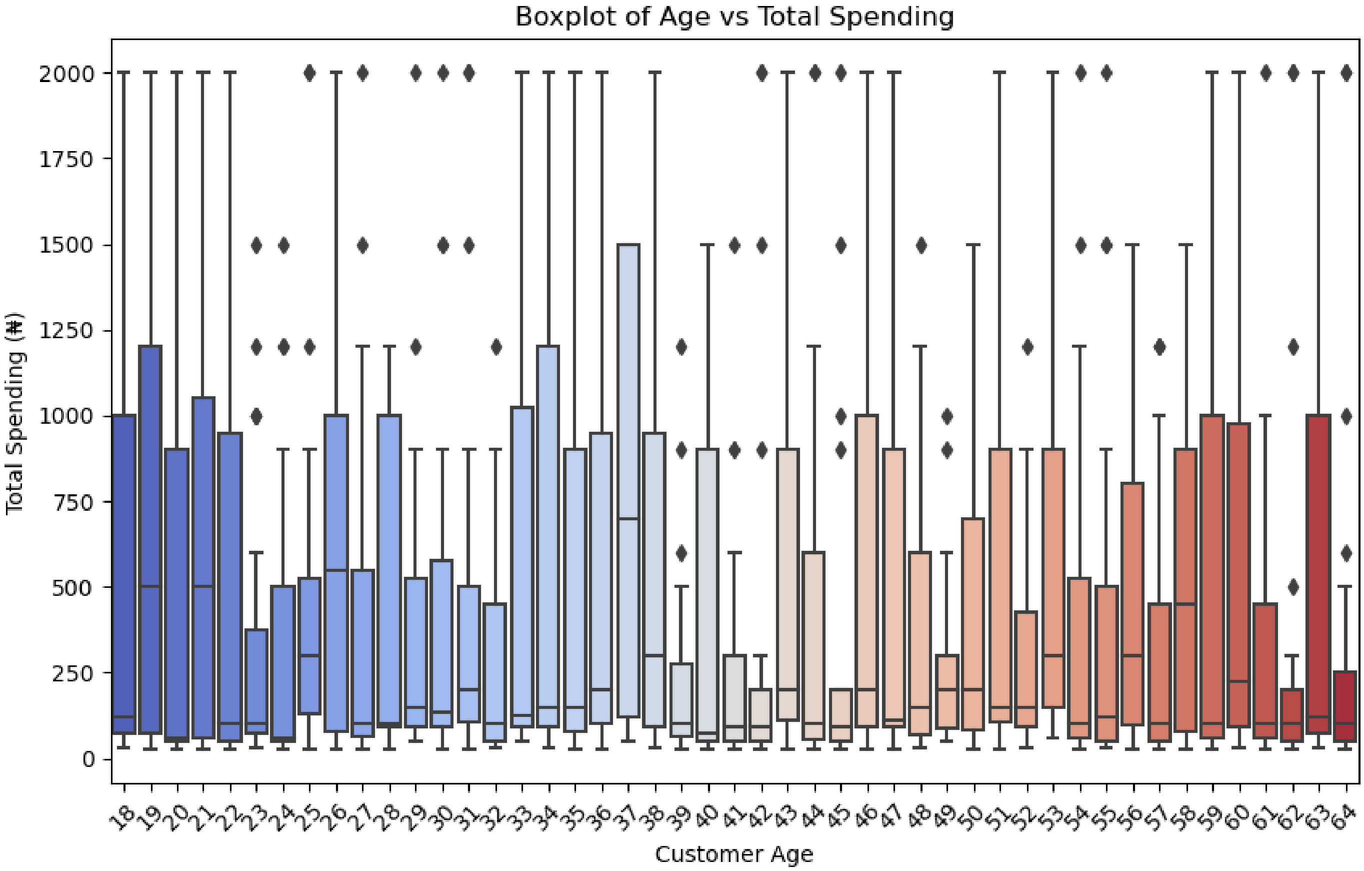
```
In [86]: plt.figure(figsize=(12,6))
         sns.lineplot(x='Date', y='Total Amount', data=monthly_sales, marker='o')
         plt.title('📆 Monthly Sales Trend')
         plt.xlabel('Month')
         plt.ylabel('Total Sales')
         plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf v
alues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\shadrach\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:152: UserWarnin
g: Glyph 128198 (\N{TEAR-OFF CALENDAR}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
In [87]: plt.figure(figsize=(10, 6))
         sns.boxplot(x='Age', y='Total Amount', data=rsales, palette='coolwarm')

         plt.title('Boxplot of Age vs Total Spending')
         plt.xlabel('Customer Age')
         plt.ylabel('Total Spending (₦)')
         plt.xticks(rotation=45)
         plt.show()
```

```
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\shadrach\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: i
s_categorical_dtype is deprecated and will be removed in a future version. Use isinstanc
e(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



Conclusion / Results & Impact:

Through careful data exploration and trend analysis, I uncovered key patterns — including peak sales periods and underperforming promotions. By acting on these findings, the business optimized inventory, refined its marketing efforts, and improved overall sales efficiency.

The impact went beyond numbers — it gave the store clarity, confidence, and control over their operations. This project reaffirmed my belief that data analysis isn't just about figures — it's about uncovering stories that help businesses grow.

```
In [ ]:
```