Imports

In [74]:
```
import pandas as pd
import numpy as np
```

Exploration

In [71]:
```
df = pd.read_csv("C:/Users/robah/OneDrive/Desktop/Konecta/Session1/dirty_caf
```

In [10]:
```
df
#jupyter shortcut; it outputs data in an organized table format and rows wil
```

Out[10]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Tr
0	TXN_1961373	Coffee	2	2.0	4.0	Credit Card	Takeaway	
1	TXN_4977031	Cake	4	3.0	12.0	Cash	In-store	
2	TXN_4271903	Cookie	4	1.0	ERROR	Credit Card	In-store	
3	TXN_7034554	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN	
4	TXN_3160411	Coffee	2	2.0	4.0	Digital Wallet	In-store	
...	...	...	...	...	...	...	...	
9995	TXN_7672686	Coffee	2	2.0	4.0	NaN	UNKNOWN	
9996	TXN_9659401	NaN	3	NaN	3.0	Digital Wallet	NaN	
9997	TXN_5255387	Coffee	4	2.0	8.0	Digital Wallet	NaN	
9998	TXN_7695629	Cookie	3	NaN	3.0	Digital Wallet	NaN	
9999	TXN_6170729	Sandwich	3	4.0	12.0	Cash	In-store	

10000 rows × 8 columns

In [11]:
```
print(df)
#will output data as a string with no table-like format and rows will be tru
```

```
     Transaction ID      Item Quantity Price Per Unit Total Spent  \
0       TXN_1961373    Coffee        2            2.0         4.0   
1       TXN_4977031      Cake        4            3.0        12.0   
2       TXN_4271903    Cookie        4            1.0       ERROR   
3       TXN_7034554     Salad        2            5.0        10.0   
4       TXN_3160411    Coffee        2            2.0         4.0   
...             ...       ...      ...            ...         ...   
9995    TXN_7672686    Coffee        2            2.0         4.0   
9996    TXN_9659401       NaN        3            NaN         3.0   
9997    TXN_5255387    Coffee        4            2.0         8.0   
9998    TXN_7695629    Cookie        3            NaN         3.0   
9999    TXN_6170729  Sandwich        3            4.0        12.0   

      Payment Method  Location Transaction Date  
0        Credit Card  Takeaway       2023-09-08  
1               Cash  In-store       2023-05-16  
2        Credit Card  In-store       2023-07-19  
3            UNKNOWN   UNKNOWN       2023-04-27  
4     Digital Wallet  In-store       2023-06-11  
...              ...       ...              ...  
9995             NaN   UNKNOWN       2023-08-30  
9996  Digital Wallet       NaN       2023-06-02  
9997  Digital Wallet       NaN       2023-03-02  
9998  Digital Wallet       NaN       2023-12-02  
9999            Cash  In-store       2023-11-07  

[10000 rows x 8 columns]
```

In [271…

```
#print(df.to_string())
#table-like format and it output all rows
```

In [ ]:

```
#Notes about dataset
#All columns have either NAN,UNKOWN or ERROR except for the transaction ID
```

In [16]:

```
#first 5 rows if no parameter given else prints frist x rows where x is the
df.head(30)
```

Out[16]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Tra
0	TXN_1961373	Coffee	2	2.0	4.0	Credit Card	Takeaway	20
1	TXN_4977031	Cake	4	3.0	12.0	Cash	In-store	20
2	TXN_4271903	Cookie	4	1.0	ERROR	Credit Card	In-store	20
3	TXN_7034554	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN	20
4	TXN_3160411	Coffee	2	2.0	4.0	Digital Wallet	In-store	20
5	TXN_2602893	Smoothie	5	4.0	20.0	Credit Card	NaN	20
6	TXN_4433211	UNKNOWN	3	3.0	9.0	ERROR	Takeaway	20
7	TXN_6699534	Sandwich	4	4.0	16.0	Cash	UNKNOWN	20
8	TXN_4717867	NaN	5	3.0	15.0	NaN	Takeaway	20
9	TXN_2064365	Sandwich	5	4.0	20.0	NaN	In-store	20
10	TXN_2548360	Salad	5	5.0	25.0	Cash	Takeaway	20
11	TXN_3051279	Sandwich	2	4.0	8.0	Credit Card	Takeaway	
12	TXN_7619095	Sandwich	2	4.0	8.0	Cash	In-store	20
13	TXN_9437049	Cookie	5	1.0	5.0	NaN	Takeaway	20
14	TXN_8915701	ERROR	2	1.5	3.0	NaN	In-store	20
15	TXN_2847255	Salad	3	5.0	15.0	Credit Card	In-store	20
16	TXN_3765707	Sandwich	1	4.0	4.0	NaN	NaN	20
17	TXN_6769710	Juice	2	3.0	6.0	Cash	In-store	20
18	TXN_8876618	Cake	5	3.0	15.0	Cash	ERROR	20
19	TXN_3709394	Juice	4	3.0	12.0	Cash	Takeaway	20
20	TXN_3522028	Smoothie	ERROR	4.0	20.0	Cash	In-store	20
21	TXN_3567645	Smoothie	4	4.0	16.0	Credit Card	Takeaway	20
22	TXN_5132361	Sandwich	3	4.0	12.0	Digital Wallet	Takeaway	20
23	TXN_2616390	Sandwich	2	4.0	8.0	NaN	NaN	20
24	TXN_9400181	Sandwich	5	4.0	20.0	Cash	In-store	20
25	TXN_7958992	Smoothie	3	4.0	NaN	UNKNOWN	UNKNOWN	20
26	TXN_5183041	Cookie	5	1.0	5.0	Credit Card	In-store	20

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Tra
27	TXN_5695074	Juice	4	3.0	12.0	Credit Card	Takeaway	2(
28	TXN_8467949	Smoothie	5	4.0	20.0	Credit Card	NaN	2(
29	TXN_7640952	Cake	4	3.0	12.0	Digital Wallet	Takeaway	

In [17]:
```
#last 5 rows by default or last x rows where x is the parameter of tail fucr
df.tail(30)
```

Out[17]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
9970	TXN_5762440	Sandwich	5	4.0	20.0	Cash	In-store
9971	TXN_6120851	Salad	5	5.0	25.0	Cash	Takeaway
9972	TXN_3124078	Cake	4	3.0	12.0	UNKNOWN	In-store
9973	TXN_7936002	Salad	2	5.0	10.0	Digital Wallet	NaN
9974	TXN_8076061	Tea	4	1.5	6.0	Cash	In-store
9975	TXN_9668108	Cake	1	3.0	3.0	Cash	In-store
9976	TXN_3528020	Cookie	1	1.0	1.0	NaN	Takeaway
9977	TXN_5548914	Juice	2	3.0	ERROR	Digital Wallet	In-store
9978	TXN_4302199	Tea	3	1.5	4.5	NaN	NaN
9979	TXN_9933628	Smoothie	5	4.0	20.0	Cash	In-store
9980	TXN_6796890	Tea	4	1.5	6.0	UNKNOWN	NaN
9981	TXN_4583012	ERROR	5	4.0	20.0	Digital Wallet	NaN
9982	TXN_8567525	Cookie	2	1.0	2.0	NaN	Takeaway
9983	TXN_9226047	Smoothie	3	4.0	12.0	Cash	NaN
9984	TXN_3142496	Smoothie	UNKNOWN	4.0	4.0	Cash	Takeaway
9985	TXN_3297457	Cake	2	3.0	6.0	NaN	UNKNOWN
9986	TXN_2858441	Sandwich	2	4.0	8.0	Credit Card	In-store
9987	TXN_1784478	Juice	5	3.0	15.0	Cash	NaN
9988	TXN_9594133	Cake	5	3.0	NaN	ERROR	NaN
9989	TXN_1741685	Juice	5	3.0	15.0	Cash	NaN
9990	TXN_1538510	Coffee	5	2.0	10.0	Digital Wallet	NaN
9991	TXN_3897619	Sandwich	3	4.0	12.0	Cash	Takeaway
9992	TXN_2739140	Smoothie	4	4.0	16.0	UNKNOWN	In-store
9993	TXN_4766549	Smoothie	2	4.0	NaN	Cash	NaN
9994	TXN_7851634	UNKNOWN	4	4.0	16.0	NaN	NaN
9995	TXN_7672686	Coffee	2	2.0	4.0	NaN	UNKNOWN
9996	TXN_9659401	NaN	3	NaN	3.0	Digital Wallet	NaN
9997	TXN_5255387	Coffee	4	2.0	8.0	Digital Wallet	NaN

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
9998	TXN_7695629	Cookie	3	NaN	3.0	Digital Wallet	NaN
9999	TXN_6170729	Sandwich	3	4.0	12.0	Cash	In-store

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Transaction ID    10000 non-null  object
 1   Item              9667 non-null   object
 2   Quantity          9862 non-null   object
 3   Price Per Unit    9821 non-null   object
 4   Total Spent       9827 non-null   object
 5   Payment Method    7421 non-null   object
 6   Location          6735 non-null   object
 7   Transaction Date  9841 non-null   object
dtypes: object(8)
memory usage: 625.1+ KB
```

In [37]:
```
df.describe()
```

Out[37]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transa
count	10000	9667	9862	9821	9827	7421	6735	
unique	10000	10	7	8	19	5	4	
top	TXN_1961373	Juice	5	3.0	6.0	Digital Wallet	Takeaway	UNK
freq	1	1171	2013	2429	979	2291	3022	

In [22]:
```
#select n random rows from the df
df.sample(10)
```

Out[22]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Tra
8989	TXN_7353685	Sandwich	2	4.0	NaN	NaN	In-store	(
2284	TXN_8379880	Juice	UNKNOWN	3.0	9.0	NaN	Takeaway	2
6044	TXN_1900906	Coffee	5	2.0	ERROR	Cash	NaN	2
5461	TXN_5697778	Cake	3	3.0	9.0	Cash	NaN	2
6228	TXN_4602865	Cookie	5	1.0	5.0	Digital Wallet	Takeaway	2
3190	TXN_6149748	Salad	3	5.0	15.0	Credit Card	NaN	2
8508	TXN_8787149	Juice	1	3.0	3.0	NaN	ERROR	2
7255	TXN_6598343	Juice	4	3.0	12.0	NaN	NaN	2
2218	TXN_4959129	NaN	2	3.0	6.0	Credit Card	In-store	2
3400	TXN_6131195	Salad	1	5.0	5.0	NaN	NaN	2

In [ ]:
```
#output error most probably because the dtype is object not numerical; the c
df.nlargest(10, 'Quantity')
df.nsmallest(10, 'Price')
```

In [26]:
```
#outputs an errir because the column has strings as NAN and can not be compa
df[df['Price Per Unit'] > 50]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 df[df[                  ] > 50]

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\ops\common.py:7
6, in _unpack_zerodim_and_defer.<locals>.new_method(self, other)
     72             return NotImplemented
     74 other = item_from_zerodim(other)
---> 76 return method(self, other)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\arraylike.py:56,
in OpsMixin.__gt__(self, other)
     54 @unpack_zerodim_and_defer("__gt__")
     55 def __gt__(self, other):
---> 56     return self._cmp_method(other, operator.gt)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\series.py:6130,
in Series._cmp_method(self, other, op)
   6127 lvalues = self._values
   6128 rvalues = extract_array(other, extract_numpy=True, extract_range=Tru
e)
-> 6130 res_values = ops.comparison_op(lvalues, rvalues, op)
   6132 return self._construct_result(res_values, name=res_name)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\ops\array_ops.p
y:344, in comparison_op(left, right, op)
    341     return invalid_comparison(lvalues, rvalues, op)
    343 elif lvalues.dtype == object or isinstance(rvalues, str):
--> 344     res_values = comp_method_OBJECT_ARRAY(op, lvalues, rvalues)
    346 else:
    347     res_values = _na_arithmetic_op(lvalues, rvalues, op, is_cmp=Tru
e)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\ops\array_ops.p
y:129, in comp_method_OBJECT_ARRAY(op, x, y)
    127     result = libops.vec_compare(x.ravel(), y.ravel(), op)
    128 else:
--> 129     result = libops.scalar_compare(x.ravel(), y, op)
    130 return result.reshape(x.shape)

File pandas/_libs/ops.pyx:107, in pandas._libs.ops.scalar_compare()

TypeError: '>' not supported between instances of 'str' and 'int'
```

```
In [27]: df[df['Item'] == 'Juice']
```

Out[27]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transac I
17	TXN_6769710	Juice	2	3.0	6.0	Cash	In-store	2023-0
19	TXN_3709394	Juice	4	3.0	12.0	Cash	Takeaway	2023-0
27	TXN_5695074	Juice	4	3.0	12.0	Credit Card	Takeaway	2023-0
43	TXN_9620080	Juice	4	3.0	12.0	NaN	Takeaway	2023-1
46	TXN_8078640	Juice	4	3.0	12.0	Digital Wallet	In-store	2023-1
...	...	...	...	...	...	...	...	
9960	TXN_3546629	Juice	5	3.0	15.0	NaN	In-store	2023-0
9967	TXN_8563793	Juice	4	3.0	12.0	NaN	In-store	2023-0
9977	TXN_5548914	Juice	2	3.0	ERROR	Digital Wallet	In-store	2023-1
9987	TXN_1784478	Juice	5	3.0	15.0	Cash	NaN	2023-0
9989	TXN_1741685	Juice	5	3.0	15.0	Cash	NaN	2023-0

1171 rows × 8 columns

In [32]:
```python
#outputs an error without the brackets
df[(df['Item'] == 'Juice') & (df['Payment Method'] == 'Cash')]
```

Out[32]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transac D
17	TXN_6769710	Juice	2	3.0	6.0	Cash	In-store	2023-0
19	TXN_3709394	Juice	4	3.0	12.0	Cash	Takeaway	2023-0
48	TXN_8201146	Juice	5	3.0	15.0	Cash	NaN	2023-0
66	TXN_8501819	Juice	NaN	3.0	6.0	Cash	NaN	2023-0
79	TXN_3829165	Juice	4	3.0	12.0	Cash	In-store	2023-0
...	...	...	...	...	...	...	...	
9899	TXN_6188262	Juice	1	3.0	3.0	Cash	Takeaway	2023-1
9929	TXN_6049240	Juice	1	3.0	3.0	Cash	NaN	2023-0
9941	TXN_4224427	Juice	4	3.0	12.0	Cash	Takeaway	2023-0
9987	TXN_1784478	Juice	5	3.0	15.0	Cash	NaN	2023-0
9989	TXN_1741685	Juice	5	3.0	15.0	Cash	NaN	2023-0

266 rows × 8 columns

```
In [36]: df[['Location']]
```

Out[36]:

	Location
0	Takeaway
1	In-store
2	In-store
3	UNKNOWN
4	In-store
...	...
9995	UNKNOWN
9996	NaN
9997	NaN
9998	NaN
9999	In-store

10000 rows × 1 columns

```
In [34]: df[['Item', 'Quantity']]
```

Out[34]:

	Item	Quantity
0	Coffee	2
1	Cake	4
2	Cookie	4
3	Salad	2
4	Coffee	2
...	...	...
9995	Coffee	2
9996	NaN	3
9997	Coffee	4
9998	Cookie	3
9999	Sandwich	3

10000 rows × 2 columns

```
In [35]: #select all columns between Item and location columns
         df.loc[:, 'Item':'Location']
```

Out[35]:

	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
0	Coffee	2	2.0	4.0	Credit Card	Takeaway
1	Cake	4	3.0	12.0	Cash	In-store
2	Cookie	4	1.0	ERROR	Credit Card	In-store
3	Salad	2	5.0	10.0	UNKNOWN	UNKNOWN
4	Coffee	2	2.0	4.0	Digital Wallet	In-store
...	...	...	...	...	...	...
9995	Coffee	2	2.0	4.0	NaN	UNKNOWN
9996	NaN	3	NaN	3.0	Digital Wallet	NaN
9997	Coffee	4	2.0	8.0	Digital Wallet	NaN
9998	Cookie	3	NaN	3.0	Digital Wallet	NaN
9999	Sandwich	3	4.0	12.0	Cash	In-store

10000 rows × 6 columns

In [38]:
```
df.Item.unique()
```

Out[38]:
```
array(['Coffee', 'Cake', 'Cookie', 'Salad', 'Smoothie', 'UNKNOWN',
       'Sandwich', nan, 'ERROR', 'Juice', 'Tea'], dtype=object)
```

In [44]:
```
df['Price Per Unit'].unique()
```

Out[44]:
```
array(['2.0', '3.0', '1.0', '5.0', '4.0', '1.5', nan, 'ERROR', 'UNKNOWN'],
      dtype=object)
```

In [45]:
```
df.Item.value_counts()
```

Out[45]:
```
Item
Juice       1171
Coffee      1165
Salad       1148
Cake        1139
Sandwich    1131
Smoothie    1096
Cookie      1092
Tea         1089
UNKNOWN      344
ERROR        292
Name: count, dtype: int64
```

In [46]:
```
#error becuase of string values in price column
df['Price Per Unit'].sum()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[46], line 1
----> 1 df[                ].sum()

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\series.py:6539,
in Series.sum(self, axis, skipna, numeric_only, min_count, **kwargs)
   6530 @doc(make_doc("sum", ndim=1))
   6531 def sum(
   6532     self,
   (...)   6537     **kwargs,
   6538 ):
-> 6539     return NDFrame.sum(self, axis, skipna, numeric_only, min_count,
**kwargs)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\generic.py:1252
5, in NDFrame.sum(self, axis, skipna, numeric_only, min_count, **kwargs)
  12517 def sum(
  12518     self,
  12519     axis: Axis | None = 0,
   (...)  12523     **kwargs,
  12524 ):
> 12525     return self._min_count_stat_function(
  12526                  , nanops.nansum, axis, skipna, numeric_only, min_count,
**kwargs
  12527     )

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\generic.py:1250
8, in NDFrame._min_count_stat_function(self, name, func, axis, skipna, numer
ic_only, min_count, **kwargs)
  12505 elif axis is lib.no_default:
  12506     axis = 0
> 12508 return self._reduce(
  12509     func,
  12510     name=name,
  12511     axis=axis,
  12512     skipna=skipna,
  12513     numeric_only=numeric_only,
  12514     min_count=min_count,
  12515 )

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\series.py:6468,
in Series._reduce(self, op, name, axis, skipna, numeric_only, filter_type, *
*kwds)
   6463     # GH#47500 - change to TypeError to match other methods
   6464     raise TypeError(
   6465         f"Series.{name} does not allow {kwd_name}={numeric_only} "
   6466         "with non-numeric dtypes."
   6467     )
-> 6468 return op(delegate, skipna=skipna, **kwds)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:85, in
disallow.__call__.<locals>._f(*args, **kwargs)
     81     raise TypeError(
     82         f"reduction operation '{f_name}' not allowed for this dtype"
     83     )
     84 try:
---> 85     return f(*args, **kwargs)
     86 except ValueError as e:
     87     # we want to transform an object array
     88     # ValueError message to the more typical TypeError
     89     # e.g. this is normally a disallowed function on
     90     # object arrays that contain strings
     91     if is_object_dtype(args[0]):

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:404, i
n _datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwarg
s)
    401 if datetimelike and mask is None:
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:
    407     result = _wrap_results(result, orig_values.dtype, fill_value=iNa
T)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:477, i
n maybe_operate_rowwise.<locals>.newfunc(values, axis, **kwargs)
    474         results = [func(x, **kwargs) for x in arrs]
    475     return np.array(results)
--> 477 return func(values, axis=axis, **kwargs)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:646, i
n nansum(values, axis, skipna, min_count, mask)
    643 elif dtype.kind == "m":
    644     dtype_sum = np.dtype(np.float64)
--> 646 the_sum = values.sum(axis, dtype=dtype_sum)
    647 the_sum = _maybe_null_out(the_sum, axis, mask, values.shape, min_cou
nt=min_count)
    649 return the_sum

File ~\miniconda\envs\konecta\Lib\site-packages\numpy\core\_methods.py:49, i
n _sum(a, axis, dtype, out, keepdims, initial, where)
     47 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     48          initial=_NoValue, where=True):
---> 49     return umr_sum(a, axis, dtype, out, keepdims, initial, where)

TypeError: can only concatenate str (not "int") to str
```

In [49]:
```python
df['Price Per Unit'].min()
df['Price Per Unit'].max()
df['Price Per Unit'].mean()
df['Price Per Unit'].median()
df['Price Per Unit'].var()
df['Price Per Unit'].std()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 1
----> 1 df[                ].min()

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\series.py:6518,
in Series.min(self, axis, skipna, numeric_only, **kwargs)
   6510 @doc(make_doc("min", ndim=1))
   6511 def min(
   6512     self,
   (...)   6516     **kwargs,
   6517 ):
-> 6518     return NDFrame.min(self, axis, skipna, numeric_only, **kwargs)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\generic.py:1240
7, in NDFrame.min(self, axis, skipna, numeric_only, **kwargs)
  12400 def min(
  12401     self,
  12402     axis: Axis | None = 0,
   (...)  12405     **kwargs,
  12406 ):
> 12407     return self._stat_function(
  12408                  ,
  12409         nanops.nanmin,
  12410         axis,
  12411         skipna,
  12412         numeric_only,
  12413         **kwargs,
  12414     )

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\generic.py:1239
6, in NDFrame._stat_function(self, name, func, axis, skipna, numeric_only, *
*kwargs)
  12392 nv.validate_func(name, (), kwargs)
  12394 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
> 12396 return self._reduce(
  12397     func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_
only
  12398 )

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\series.py:6468,
in Series._reduce(self, op, name, axis, skipna, numeric_only, filter_type, *
*kwds)
   6463     # GH#47500 - change to TypeError to match other methods
   6464     raise TypeError(
   6465         f"Series.{name} does not allow {kwd_name}={numeric_only} "
   6466         "with non-numeric dtypes."
   6467     )
-> 6468 return op(delegate, skipna=skipna, **kwds)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:147, i
n bottleneck_switch.__call__.<locals>.f(values, axis, skipna, **kwds)
    145         result = alt(values, axis=axis, skipna=skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result
```

```
File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:404, i
n _datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwarg
s)
    401 if datetimelike and mask is None:
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:
    407     result = _wrap_results(result, orig_values.dtype, fill_value=iNa
T)

File ~\miniconda\envs\konecta\Lib\site-packages\pandas\core\nanops.py:1098,
in _nanminmax.<locals>.reduction(values, axis, skipna, mask)
   1093     return _na_for_min_count(values, axis)
   1095 values, mask = _get_values(
   1096     values, skipna, fill_value_typ=fill_value_typ, mask=mask
   1097 )
-> 1098 result = getattr(values, meth)(axis)
   1099 result = _maybe_null_out(result, axis, mask, values.shape)
   1100 return result

File ~\miniconda\envs\konecta\Lib\site-packages\numpy\core\_methods.py:45, i
n _amin(a, axis, out, keepdims, initial, where)
     43 def _amin(a, axis=None, out=None, keepdims=False,
     44           initial=_NoValue, where=True):
---> 45     return umr_minimum(a, axis, None, out, keepdims, initial, where)

TypeError: '<=' not supported between instances of 'str' and 'float'
```

In [50]:
```
#return true is data is missing (nan) error,unknown are not null
df.isnull()
```

Out[50]:

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transact Da
0	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	True	False	Fa
9996	False	True	False	True	False	False	True	Fa
9997	False	False	False	False	False	False	True	Fa
9998	False	False	False	True	False	False	True	Fa
9999	False	False	False	False	False	False	False	Fa

10000 rows × 8 columns

```
In [67]: df.isnull().sum()
```

```
Out[67]: Transaction ID         0
         Item                 333
         Quantity             138
         Price Per Unit       179
         Total Spent          173
         Payment Method      2579
         Location            3265
         Transaction Date     159
         dtype: int64
```

```
In [57]: df.columns[df.isin(['UNKNOWN', 'ERROR']).any()]
```

```
Out[57]: Index(['Item', 'Quantity', 'Price Per Unit', 'Total Spent', 'Payment Metho
         d',
                'Location', 'Transaction Date'],
               dtype='object')
```

Data Cleaning

```
In [125…: #must assign inplace = true to see the effect
          df.replace(to_replace=['UNKNOWN', 'ERROR'], value=np.nan, inplace=True)
```

```
In [206…: df['Quantity'] = pd.to_numeric(df['Quantity'], errors='coerce')
          df['Price Per Unit'] = pd.to_numeric(df['Price Per Unit'], errors='coerce')
          df['Total Spent'] = pd.to_numeric(df['Total Spent'], errors='coerce')
```

```
In [84]: df.isnull().sum()
```

```
Out[84]: Transaction ID         0
         Item                 333
         Quantity             479
         Price Per Unit       533
         Total Spent          173
         Payment Method      2579
         Location            3265
         Transaction Date     159
         dtype: int64
```

Logical Relationships

```
In [ ]: # Total spent= Quantity * Price per unit
        # Price per unit = Total spent/ Quantity
        # Some items have their prices written once but other times NAN so they can
```

```
In [127…: df.loc[(
              df['Quantity'].notna() &
              df['Price Per Unit'].notna() &
              pd.isna(df['Total Spent'])
          ), 'Total Spent' ] = df['Quantity'] * df['Price Per Unit']
```

In [128…
```
df.isnull().sum()
```

Out[128…
```
Transaction ID         0
Item                 969
Quantity              38
Price Per Unit        54
Total Spent           23
Payment Method      3178
Location            3961
Transaction Date     460
dtype: int64
```

In [129…
```
df.loc[(
    df['Total Spent'].notna() &
    df['Price Per Unit'].notna() &
    pd.isna(df['Quantity'])
), 'Quantity' ] = df['Total Spent'] / df['Price Per Unit']
```

In [130…
```
df.isnull().sum()
```

Out[130…
```
Transaction ID         0
Item                 969
Quantity              23
Price Per Unit        54
Total Spent           23
Payment Method      3178
Location            3961
Transaction Date     460
dtype: int64
```

In [131…
```
df[['Item','Price Per Unit']]
```

Out[131…

	Item	Price Per Unit
0	Coffee	2.0
1	Cake	3.0
2	Cookie	1.0
3	Salad	5.0
4	Coffee	2.0
...	...	...
9995	Coffee	2.0
9996	NaN	NaN
9997	Coffee	2.0
9998	Cookie	1.0
9999	Sandwich	4.0

10000 rows × 2 columns

```
In [132… df.loc[(
              df['Item'] == 'Coffee'), 'Price Per Unit' ] = 2.0
```

```
In [133… df.loc[(
              df['Item'] == 'Cake'), 'Price Per Unit' ] = 3.0
```

```
In [134… df.loc[(
              df['Item'] == 'Cookie'), 'Price Per Unit' ] = 1.0
```

```
In [135… df.loc[(
              df['Item'] == 'Sandwich'), 'Price Per Unit' ] = 4.0
```

```
In [136… df.loc[(
              df['Item'] == 'Salad'), 'Price Per Unit' ] = 5.0
```

```
In [137… df.loc[(
              df['Item'] == 'Tea'), 'Price Per Unit' ] = 1.5
```

```
In [138… df.loc[(
              df['Item'] == 'Juice'), 'Price Per Unit' ] = 3.0
```

```
In [139… df.loc[(
              df['Item'] == 'Smoothie'), 'Price Per Unit' ] = 4.0
```

```
In [140… df.isnull().sum()
```

```
Out[140… Transaction ID         0
         Item                 969
         Quantity              23
         Price Per Unit        54
         Total Spent           23
         Payment Method      3178
         Location            3961
         Transaction Date     460
         dtype: int64
```

```
In [144… df.loc[(
              df['Total Spent'].notna() &
              df['Quantity'].notna() &
              pd.isna(df['Price Per Unit'])
          ), 'Price Per Unit' ] = df['Total Spent'] / df['Quantity']
```

```
In [145… df.isnull().sum()
```

Out[145…
```
Transaction ID         0
Item                 969
Quantity              23
Price Per Unit         6
Total Spent           23
Payment Method      3178
Location            3961
Transaction Date     460
dtype: int64
```

In [147…
```
df.loc[(
    df['Price Per Unit'] == 2.0), 'Item' ] = 'Coffee'
```

In [149…
```
df.loc[(
    df['Price Per Unit'] == 1.0), 'Item' ] = 'Cookie'
```

In [151…
```
df.loc[(
    df['Price Per Unit'] == 1.5), 'Item' ] = 'Tea'
```

In [153…
```
df.loc[(
    df['Price Per Unit'] == 5.0), 'Item' ] = 'Salad'
```

In [154…
```
df.isnull().sum()
```

Out[154…
```
Transaction ID         0
Item                 480
Quantity              23
Price Per Unit         6
Total Spent           23
Payment Method      3178
Location            3961
Transaction Date     460
dtype: int64
```

In [272…
```
#print(df.to_string())
# NaN in Price per unit bec NaN in item
#can drop rows where item and total price are Nan
#there are rows having total price with item, quantity and price per unit Na
#there are rows with quantity and total price Nan
#Quantity and Nan are missing together
```

Normalize Date Column

In [157…
```
df['Transaction Date'] = pd.to_datetime(df['Transaction Date'], format='mixe
```

In [160…
```
#the first line is wrong because it assigns a sorted dataframe to a single c
#df['Transaction Date'] = df.sort_values('Transaction Date', ascending=True)
df = df.sort_values('Transaction Date', ascending=True)
```

In [164…
```
df['Transaction Date'] = df['Transaction Date'].ffill()
```

In [166…
```
df.isnull().sum()
```

```
Out[166…  Transaction ID        0
          Item                480
          Quantity             23
          Price Per Unit        6
          Total Spent          23
          Payment Method     3178
          Location           3961
          Transaction Date      0
          dtype: int64
```

```
In [275…  #print(df.to_string())
```

Standerdize Categorical Columns

```
In [177…  df['Item'] = df['Item'].str.capitalize()
          df['Payment Method'] = df['Payment Method'].str.capitalize()
          df['Location'] = df['Location'].str.capitalize()
```

```
In [179…  df['Item'] = df['Item'].str.strip()
          df['Location'] = df['Location'].str.strip()
          df['Payment Method'] = df['Payment Method'].str.strip()
```

```
In [180…  df['Item'] = df['Item'].str.replace(r'[^\w\s]', '', regex=True)
          df['Payment Method'] = df['Payment Method'].str.replace(r'[^\w\s]', '', rege
          df['Location'] = df['Location'].str.replace(r'[^\w\s]', '', regex=True)
```

```
In [191…  df[['Item', 'Location', 'Payment Method']] = df[['Item', 'Location', 'Paymen
          df['Payment Method'] = df['Payment Method'].str.replace('Digital wallet', 'E
```

```
In [194…  df['Location'].unique()
```

```
Out[194…  array(['Takeaway', 'Instore', 'Unspecified'], dtype=object)
```

```
In [193…  df['Payment Method'].unique()
```

```
Out[193…  array(['E-wallet', 'Unspecified', 'Cash', 'Credit card'], dtype=object)
```

```
In [201…  df.isnull().sum()
```

```
Out[201…  Transaction ID      0
          Item                0
          Quantity           23
          Price Per Unit      6
          Total Spent        23
          Payment Method      0
          Location            0
          Transaction Date    0
          dtype: int64
```

```
In [273…  #print(df.to_string())
```

Removing Duplicates

In [202…

```
df_cleaned = df.drop_duplicates()
df
```

Out[202…

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
8015	TXN_4801947	Juice	1.0	3.0	3.0	E-wallet	Takeaway
9063	TXN_9161256	Smoothie	2.0	4.0	8.0	E-wallet	Instore
7309	TXN_6093955	Tea	5.0	1.5	7.5	Unspecified	Takeaway
1425	TXN_8842223	Sandwich	5.0	4.0	20.0	E-wallet	Instore
1777	TXN_7367474	Juice	5.0	3.0	15.0	E-wallet	Takeaway
...	...	...	...	...	...	...	...
9933	TXN_9460419	Cake	1.0	3.0	3.0	Unspecified	Takeaway
9937	TXN_8253472	Cake	1.0	3.0	3.0	Unspecified	Unspecified
9949	TXN_3130865	Juice	3.0	3.0	9.0	Unspecified	Instore
9983	TXN_9226047	Smoothie	3.0	4.0	12.0	Cash	Unspecified
9988	TXN_9594133	Cake	5.0	3.0	15.0	Unspecified	Unspecified

10000 rows × 8 columns

In [270…

```
df['Transaction ID'].nunique()
```

Out[270…

```
9977
```

In [210…

```
df[df['Quantity'] <= 0.0]
#no negative or 0 in quantity column
```

Out[210…

Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date

In [211…

```
df[df['Price Per Unit'] <= 0.0]
```

Out[211…

Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date

In [213…

```
df[df['Total Spent'] <= 0.0]
```

Out[213…

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date

In [215…

```
df[df['Quantity'].isnull()]
```

Out[215…

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
7297	TXN_9944500	Smoothie	NaN	4.0	NaN	Cash	Instore
9869	TXN_1975184	Coffee	NaN	2.0	NaN	E-wallet	Unspecified
9590	TXN_9924732	Sandwich	NaN	4.0	NaN	Credit card	Instore
641	TXN_2962976	Juice	NaN	3.0	NaN	Unspecified	Unspecified
3224	TXN_6297232	Coffee	NaN	2.0	NaN	Unspecified	Unspecified
8574	TXN_2546684	Juice	NaN	3.0	NaN	E-wallet	Takeaway
278	TXN_3229409	Juice	NaN	3.0	NaN	Cash	Takeaway
8021	TXN_2428781	Salad	NaN	5.0	NaN	Unspecified	Instore
738	TXN_8696094	Sandwich	NaN	4.0	NaN	Unspecified	Takeaway
236	TXN_8562645	Salad	NaN	5.0	NaN	Unspecified	Instore
8443	TXN_2023651	Sandwich	NaN	4.0	NaN	Cash	Instore
3779	TXN_7376255	Unspecified	NaN	NaN	25.0	Unspecified	Instore
8465	TXN_9669616	Coffee	NaN	2.0	NaN	Unspecified	Unspecified
5841	TXN_5884081	Cookie	NaN	1.0	NaN	E-wallet	Instore
3401	TXN_3251829	Tea	NaN	1.5	NaN	E-wallet	Instore
8732	TXN_4550558	Cookie	NaN	1.0	NaN	Credit card	Instore
9819	TXN_1208561	Unspecified	NaN	NaN	20.0	Credit card	Unspecified
8479	TXN_1547245	Sandwich	NaN	4.0	NaN	Unspecified	Takeaway
4257	TXN_6470865	Coffee	NaN	2.0	NaN	E-wallet	Takeaway
3203	TXN_4565754	Smoothie	NaN	4.0	NaN	E-wallet	Takeaway
2796	TXN_9188692	Cake	NaN	3.0	NaN	Credit card	Unspecified
7597	TXN_1082717	Unspecified	NaN	NaN	9.0	E-wallet	Instore
7029	TXN_4628338	Coffee	NaN	2.0	NaN	Cash	Unspecified

In [216…

```
df[df['Total Spent'].isnull()]
```

Out[216…

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location
7297	TXN_9944500	Smoothie	NaN	4.0	NaN	Cash	Instore
9869	TXN_1975184	Coffee	NaN	2.0	NaN	E-wallet	Unspecified
9590	TXN_9924732	Sandwich	NaN	4.0	NaN	Credit card	Instore
1761	TXN_3611851	Unspecified	4.0	NaN	NaN	Credit card	Unspecified
641	TXN_2962976	Juice	NaN	3.0	NaN	Unspecified	Unspecified
3224	TXN_6297232	Coffee	NaN	2.0	NaN	Unspecified	Unspecified
8574	TXN_2546684	Juice	NaN	3.0	NaN	E-wallet	Takeaway
278	TXN_3229409	Juice	NaN	3.0	NaN	Cash	Takeaway
8021	TXN_2428781	Salad	NaN	5.0	NaN	Unspecified	Instore
738	TXN_8696094	Sandwich	NaN	4.0	NaN	Unspecified	Takeaway
236	TXN_8562645	Salad	NaN	5.0	NaN	Unspecified	Instore
8443	TXN_2023651	Sandwich	NaN	4.0	NaN	Cash	Instore
8465	TXN_9669616	Coffee	NaN	2.0	NaN	Unspecified	Unspecified
5841	TXN_5884081	Cookie	NaN	1.0	NaN	E-wallet	Instore
3401	TXN_3251829	Tea	NaN	1.5	NaN	E-wallet	Instore
8732	TXN_4550558	Cookie	NaN	1.0	NaN	Credit card	Instore
8479	TXN_1547245	Sandwich	NaN	4.0	NaN	Unspecified	Takeaway
4257	TXN_6470865	Coffee	NaN	2.0	NaN	E-wallet	Takeaway
3203	TXN_4565754	Smoothie	NaN	4.0	NaN	E-wallet	Takeaway
2796	TXN_9188692	Cake	NaN	3.0	NaN	Credit card	Unspecified
2289	TXN_7524977	Unspecified	4.0	NaN	NaN	Unspecified	Unspecified
4152	TXN_9646000	Unspecified	2.0	NaN	NaN	Unspecified	Instore
7029	TXN_4628338	Coffee	NaN	2.0	NaN	Cash	Unspecified

In [217…

```
#drop rows where Total Spent isnull
df = df.dropna(subset=['Total Spent'])
```

In [235…

```
avg_quantity = df['Quantity'].mean()
avg_quantity
df.loc[:, 'Quantity'] = df['Quantity'].fillna(avg_quantity)
df.loc[:, 'Price Per Unit'] = df['Price Per Unit'].fillna(df['Total Spent']/
```

In [237…

```
df.isnull().sum()
```

Out[237…
```
Transaction ID      0
Item                0
Quantity            0
Price Per Unit      0
Total Spent         0
Payment Method      0
Location            0
Transaction Date    0
dtype: int64
```

Derived Columns

In [239…
```
df.loc[:,'Month'] = df['Transaction Date'].dt.month_name()
df.loc[:,'Weekday'] = df['Transaction Date'].dt.day_name()
df.loc[:,'Hour'] = df['Transaction Date'].dt.hour
```

In [274…
```
#print(df.to_string())
```

In [245…
```
avg_price_per_item = df.groupby(by='Item')['Price Per Unit'].mean()
avg_price_per_item
df.loc[:,'Avg Price Per Item'] = df['Item'].map(avg_price_per_item)
```

In [250…
```
avg_qu_per_item = df.groupby(by='Item')['Quantity'].mean()
avg_qu_per_item
df.loc[:,'Avg Quantity Per Item'] = df['Item'].map(avg_qu_per_item)
```

In [261…
```
avg_price_per_location = df.groupby('Location')['Price Per Unit'].mean()
df.loc[:,'Avg Price Per Location'] = df['Location'].map(avg_price_per_locati
```

Plotting

In [258…
```
conda install matplotlib
```

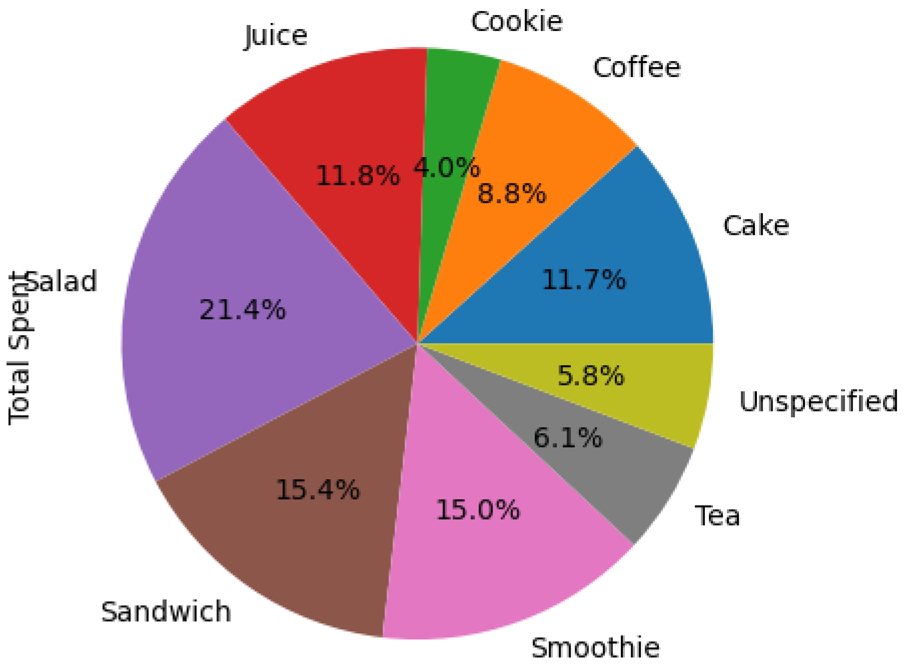
```
Jupyter detected...
3 channel Terms of Service accepted
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

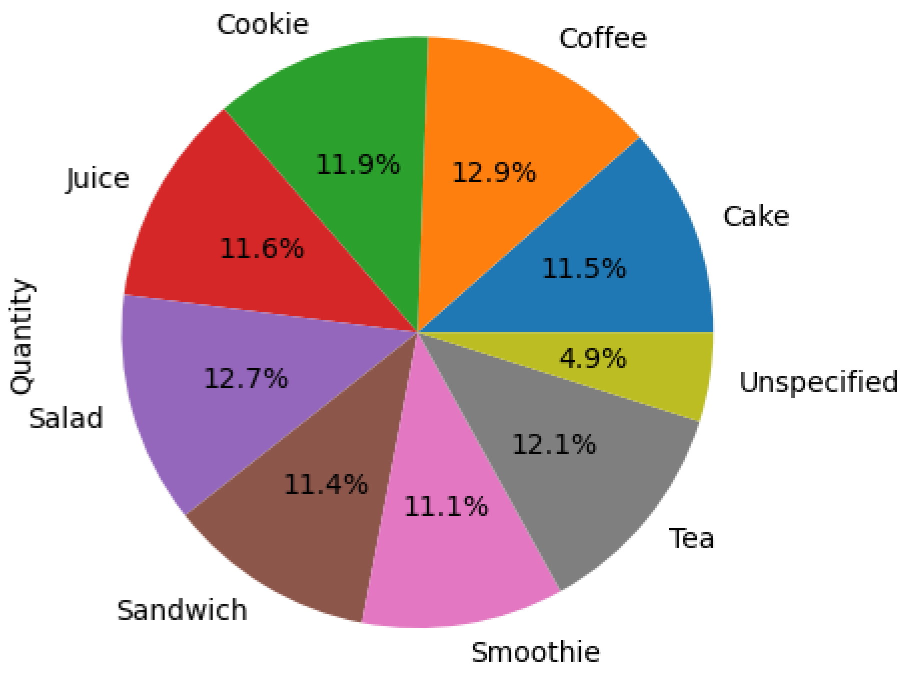
In [265…
```
item_revenue = df.groupby('Item')['Total Spent'].sum()
item_revenue.plot.pie(autopct='%1.1f%%', y='')
```

Out[265…
```
<Axes: ylabel='Total Spent'>
```



```
In [266…  item_quantity = df.groupby('Item')['Quantity'].sum()
          item_quantity.plot.pie(autopct='%1.1f%%', y='')
```

```
Out[266…  <Axes: ylabel='Quantity'>
```



```
In [268…  total_revenue = df['Total Spent'].sum()
          total_revenue
```

Out[268…

```
89096.0
```

In [267…

```
item_quantity = df.groupby('Item')['Quantity'].sum()
item_quantity
```

Out[267…

```
Item
Cake           3468
Coffee         3904
Cookie         3598
Juice          3505
Salad          3819
Sandwich       3429
Smoothie       3336
Tea            3650
Unspecified    1470
Name: Quantity, dtype: int32
```

In [269…

```
item_revenue
```

Out[269…

```
Item
Cake           10404.0
Coffee          7808.0
Cookie          3598.0
Juice          10515.0
Salad          19095.0
Sandwich       13716.0
Smoothie       13344.0
Tea             5475.0
Unspecified     5141.0
Name: Total Spent, dtype: float64
```

In [ ]: