Task 3 - Modeling

This notebook will get you started by helping you to load the data, but then it'll be up to you to complete the task! If you need help, refer to the `modeling_walkthrough.ipynb` notebook.

Section 1 - Setup

First, we need to mount this notebook to our Google Drive folder, in order to access the CSV data file. If you haven't already, watch this video https://www.youtube.com/watch?v=woHxvbBLarQ to help you mount your Google Drive folder.

In [1]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

We want to use dataframes once again to store and manipulate the data.

In [2]:
```
#!pip install pandas
```

In [3]:
```
import pandas as pd
```

Section 2 - Data loading

Similar to before, let's load our data from Google Drive for the 3 datasets provided. Be sure to upload the datasets into Google Drive, so that you can access them here.

In [4]:
```
#path = "/content/drive/MyDrive/Forage - Cognizant AI Program/Task 3/Resources/"

sales_df = pd.read_csv("sales.csv", parse_dates=["timestamp"])
sales_df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')
sales_df.head()
```

Out[4]:

	transaction_id	timestamp	product_id	category	customer_type	unit_price	quantity	total	payment_type
0	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet
1	931ad550-09e8-4da6-beaa-8c9d17be9c60	2022-03-06 10:33:59	ad81b46c-bf38-41cf-9b54-5fe7f5eba93e	fruit	standard	3.99	1	3.99	e-wallet
2	ae133534-6f61-4cd6-b6b8-d1c1d8d90aea	2022-03-04 17:20:21	7c55cbd4-f306-4c04-a030-628cbe7867c1	fruit	premium	0.19	2	0.38	e-wallet

	transaction_id	timestamp	product_id	category	customer_type	unit_price	quantity	total	payment_type
3	157cebd9-aaf0-475d-8a11-7c8e0f5b76e4	2022-03-02 17:23:58	80da8348-1707-403f-8be7-9e6deeccc883	fruit	gold	0.19	4	0.76	e-wallet
4	a81a6cd3-5e0c-44a2-826c-aea43e46c514	2022-03-05 14:32:43	7f5e86e6-f06f-45f6-bf44-27b095c9ad1d	fruit	basic	4.49	2	8.98	debit card

In [5]:

```python
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7829 entries, 0 to 7828
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   transaction_id  7829 non-null   object
 1   timestamp       7829 non-null   datetime64[ns]
 2   product_id      7829 non-null   object
 3   category        7829 non-null   object
 4   customer_type   7829 non-null   object
 5   unit_price      7829 non-null   float64
 6   quantity        7829 non-null   int64
 7   total           7829 non-null   float64
 8   payment_type    7829 non-null   object
dtypes: datetime64[ns](1), float64(2), int64(1), object(5)
memory usage: 550.6+ KB
```

In [6]:

```python
sales_df.describe(include="all", datetime_is_numeric=False)
```

```
C:\Users\Dennis\AppData\Local\Temp/ipykernel_9420/42003929.py:1: FutureWarning: Treating d
atetime data as categorical rather than numeric in `.describe` is deprecated and will be r
emoved in a future version of pandas. Specify `datetime_is_numeric=True` to silence this w
arning and adopt the future behavior now.
  sales_df.describe(include="all", datetime_is_numeric=False)
```

Out[6]:

	transaction_id	timestamp	product_id	category	customer_type	unit_price	quantity	total
count	7829	7829	7829	7829	7829	7829.000000	7829.000000	7829.000000
unique	7829	7738	300	22	5	NaN	NaN	NaN
top	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 19:32:20	ecac012c-1dec-41d4-9ebd-56fb7166f6d9	fruit	non-member	NaN	NaN	NaN
freq	1	2	114	998	1601	NaN	NaN	NaN
first	NaN	2022-03-01 09:00:13	NaN	NaN	NaN	NaN	NaN	NaN
last	NaN	2022-03-07 19:59:54	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	7.819480	2.501597	19.709905
std	NaN	NaN	NaN	NaN	NaN	5.388088	1.122722	17.446680
min	NaN	NaN	NaN	NaN	NaN	0.190000	1.000000	0.190000
25%	NaN	NaN	NaN	NaN	NaN	3.990000	1.000000	6.570000

	transaction_id	timestamp	product_id	category	customer_type	unit_price	quantity	total
50%	NaN	NaN	NaN	NaN	NaN	7.190000	3.000000	14.970000
75%	NaN	NaN	NaN	NaN	NaN	11.190000	4.000000	28.470000
max	NaN	NaN	NaN	NaN	NaN	23.990000	4.000000	95.960000

In [7]:

```
#sales_df[sales_df["transaction_id"] == "4220e505-c247-478d-9831-6b9f87a4488a"]
```

In [8]:

```
stock_df = pd.read_csv("sensor_stock_levels.csv")
stock_df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')
stock_df.head()
```

Out[8]:

	id	timestamp	product_id	estimated_stock_pct
0	4220e505-c247-478d-9831-6b9f87a4488a	2022-03-07 12:13:02	f658605e-75f3-4fed-a655-c0903f344427	0.75
1	f2612b26-fc82-49ea-8940-0751fdd4d9ef	2022-03-07 16:39:46	de06083a-f5c0-451d-b2f4-9ab88b52609d	0.48
2	989a287f-67e6-4478-aa49-c3a35dac0e2e	2022-03-01 18:17:43	ce8f3a04-d1a4-43b1-a7c2-fa1b8e7674c8	0.58
3	af8e5683-d247-46ac-9909-1a77bdebefb2	2022-03-02 14:29:09	c21e3ba9-92a3-4745-92c2-6faef73223f7	0.79
4	08a32247-3f44-4002-85fb-c198434dd4bb	2022-03-02 13:46:18	7f478817-aa5b-44e9-9059-8045228c9eb0	0.22

In [9]:

```
stock_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   15000 non-null  object
 1   timestamp            15000 non-null  object
 2   product_id           15000 non-null  object
 3   estimated_stock_pct  15000 non-null  float64
dtypes: float64(1), object(3)
memory usage: 468.9+ KB
```

In [10]:

```
stock_df.describe(include='all')
```

Out[10]:

	id	timestamp	product_id	estimated_stock_pct
count	15000	15000	15000	15000.000000
unique	15000	14611	300	NaN
top	4220e505-c247-478d-9831-6b9f87a4488a	2022-03-05 17:07:33	89845097-f0ec-4702-bb65-85c67cf94490	NaN
freq	1	3	89	NaN
mean	NaN	NaN	NaN	0.502735
std	NaN	NaN	NaN	0.286842

	id	timestamp	product_id	estimated_stock_pct
min	NaN	NaN	NaN	0.010000
25%	NaN	NaN	NaN	0.260000
50%	NaN	NaN	NaN	0.500000
75%	NaN	NaN	NaN	0.750000
max	NaN	NaN	NaN	1.000000

In [11]:
```
temp_df = pd.read_csv("sensor_storage_temperature.csv")
temp_df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')
temp_df.head()
```

Out[11]:

	id	timestamp	temperature
0	d1ca1ef8-0eac-42fc-af80-97106efc7b13	2022-03-07 15:55:20	2.96
1	4b8a66c4-0f3a-4f16-826f-8cf9397e9d18	2022-03-01 09:18:22	1.88
2	3d47a0c7-1e72-4512-812f-b6b5d8428cf3	2022-03-04 15:12:26	1.78
3	9500357b-ce15-424a-837a-7677b386f471	2022-03-02 12:30:42	2.18
4	c4b61fec-99c2-4c6d-8e5d-4edd8c9632fa	2022-03-05 09:09:33	1.38

Now it's up to you, refer back to the steps in your strategic plan to complete this task. Good luck!

In [12]:
```
df2 = pd.merge(left=sales_df, right=stock_df, how="left", on="product_id")
```

In [13]:
```
df2
```

Out[13]:

	transaction_id	timestamp_x	product_id	category	customer_type	unit_price	quantity	total	payment_t
0	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-w
1	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-w
2	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-w
3	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-w
4	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-w
...	...	...	...	...	...	...	...	...	

	transaction_id	timestamp_x	product_id	category	customer_type	unit_price	quantity	total	payment_t
435017	afd70b4f-ee21-402d-8d8f-0d9e13c2bea6	2022-03-06 13:50:36	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit
435018	afd70b4f-ee21-402d-8d8f-0d9e13c2bea6	2022-03-06 13:50:36	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit
435019	afd70b4f-ee21-402d-8d8f-0d9e13c2bea6	2022-03-06 13:50:36	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit
435020	afd70b4f-ee21-402d-8d8f-0d9e13c2bea6	2022-03-06 13:50:36	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit
435021	afd70b4f-ee21-402d-8d8f-0d9e13c2bea6	2022-03-06 13:50:36	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit

435022 rows × 12 columns

In [14]:
```
df2.columns
```

Out[14]:
```
Index(['transaction_id', 'timestamp_x', 'product_id', 'category',
       'customer_type', 'unit_price', 'quantity', 'total', 'payment_type',
       'id', 'timestamp_y', 'estimated_stock_pct'],
      dtype='object')
```

In [15]:
```
df2.drop(['transaction_id', 'timestamp_x', 'id', 'timestamp_y'], inplace=True, axis=1)
```

In [16]:
```
df2
```

Out[16]:

	product_id	category	customer_type	unit_price	quantity	total	payment_type	estimated_stock_pct
0	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.25
1	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.23
2	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.80
3	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.79

	product_id	category	customer_type	unit_price	quantity	total	payment_type	estimated_stock_pct
4	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.86
...	...	...	...	...	...	...	...	...
435017	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.32
435018	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.80
435019	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.13
435020	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.04
435021	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.32

435022 rows × 8 columns

In [17]:
```
df2.duplicated().sum()
```

Out[17]:
```
175566
```

In [18]:
```
df2.drop_duplicates(inplace=True)
```

In [19]:
```
df2.reset_index(inplace=True, drop=True)
```

In [20]:
```
df2
```

Out[20]:

	product_id	category	customer_type	unit_price	quantity	total	payment_type	estimated_stock_pct
0	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.25
1	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.23

	product_id	category	customer_type	unit_price	quantity	total	payment_type	estimated_stock_pct
2	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.80
3	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.79
4	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet	0.86
...	...	...	...	...	...	...	...	...
259451	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.18
259452	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.51
259453	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.61
259454	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.32
259455	d6ccd088-11be-4c25-aa1f-ea87c01a04db	cleaning products	non-member	14.99	4	59.96	debit card	0.13

259456 rows × 8 columns

In [21]:

```
df2.groupby("category")["estimated_stock_pct"].mean()
```

Out[21]:

```
category
baby products            0.495525
baked goods              0.508719
baking                   0.500581
beverages                0.482044
canned foods             0.501246
cheese                   0.513667
cleaning products        0.506404
condiments and sauces    0.504110
dairy                    0.504433
frozen                   0.508793
fruit                    0.506041
kitchen                  0.497361
meat                     0.488858
medicine                 0.508038
packaged foods           0.500852
personal care            0.486956
```

```
pets                         0.493691
refrigerated items           0.513369
seafood                      0.502254
snacks                       0.499847
spices and herbs             0.496206
vegetables                   0.497701
Name: estimated_stock_pct, dtype: float64
```

In [22]:
```
df2.groupby("product_id")["estimated_stock_pct"].mean()
```

Out[22]:
```
product_id
00e120bb-89d6-4df5-bc48-a051148e3d03    0.518864
01f3cdd9-8e9e-4dff-9b5c-69698a0388d0    0.507500
01ff0803-ae73-4234-971d-5713c97b7f4b    0.494048
02b1a5a2-cd74-4e64-80f0-4667372bc394    0.418929
0363eb21-8c74-47e1-a216-c37e565e5ceb    0.530937
                                          ...
fa9fa800-cd49-4702-b94f-53a53cd4e610    0.473600
fbeb39cc-8cd0-4143-bdfb-77658a02dec9    0.545870
fcc9e0ca-ad36-4925-8306-4369afd6cd41    0.561842
fd66ac0b-3498-4613-8ec0-764686b0d864    0.532857
fd77b5cb-498c-40ca-95d1-0f87f13dd0d8    0.543939
Name: estimated_stock_pct, Length: 300, dtype: float64
```

In [23]:
```
df2.groupby("customer_type")["estimated_stock_pct"].mean()
```

Out[23]:
```
customer_type
basic         0.500622
gold          0.500588
non-member    0.502068
premium       0.501717
standard      0.502111
Name: estimated_stock_pct, dtype: float64
```

In [24]:
```
df2.groupby("payment_type")["estimated_stock_pct"].mean()
```

Out[24]:
```
payment_type
cash           0.501178
credit card    0.502013
debit card     0.501821
e-wallet       0.500750
Name: estimated_stock_pct, dtype: float64
```

In [ ]:
```

```