Lab 4 Data Wrangling

Skills

- Data Format
  - Long vs. Wide
- Merging (merge)
- Reshaping (melt)
- Aggregating (groupby, agg)
- Calculating new variables

In [131…
```
# import modules you need
import pandas as pd
import numpy as np
```

In [132…
```
# Write code to read in the data sets. Name the datasets as follows, respectively: orc
# orders_447.csv
# products.csv
# order_products_447.csv
# OrganicAisle.csv
# ConventionalAisle.csv
orders = pd.read_csv('orders_447.csv')
products = pd.read_csv('products.csv')
order_products = pd.read_csv('order_products_447.csv')
organic = pd.read_csv('OrganicAisle.csv')
conventional = pd.read_csv('ConventionalAisle.csv')
```

Part A: Aggregation

Units of Analysis

In [133…
```
# get the number of rows/observations and print the first few lines of the orders data
orders.shape
orders.head()
```

Out[133]:

	order_id	num_items	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_p
0	2281065	24	108077	prior	8	6	16	
1	3407099	3	195027	prior	8	2	14	
2	2106016	7	94462	prior	2	0	18	
3	1222212	1	141637	prior	66	2	13	
4	2571664	15	119478	prior	28	1	8	

In [134…
```
# get the number of rows/observations and print the first few lines of the products da
```

```
products.shape
products.head()
```

Out[134]:

	product_id	product_name	aisle_id	department_id
0	1	Chocolate Sandwich Cookies	61	19
1	2	All-Seasons Salt	104	13
2	3	Robust Golden Unsweetened Oolong Tea	94	7
3	4	Smart Ones Classic Favorites Mini Rigatoni Wit...	38	1
4	5	Green Chile Anytime Sauce	5	13

In [135…

```
# get the number of rows/observations and print the first few lines of the order_produ
order_products.shape
order_products.head()
```

Out[135]:

	order_id	product_id	add_to_cart_order	reordered	user_id	eval_set	order_number	order_dow	c
0	10853	33120	2	1	73826	prior	25	0	
1	112647	33120	8	1	64174	prior	5	1	
2	207058	33120	1	1	172922	prior	12	3	
3	521908	33120	6	1	131050	prior	28	0	
4	522668	33120	1	1	35583	prior	39	5	

Aggregation

In [136…

```
# aggregate the order_products data set to get the count of the number of items ordere
# reset the index and rename the columns after you've aggregated the data, so the colu
dept_items = order_products[['department_id', 'add_to_cart_order']].groupby(['departme
dept_items = dept_items.reset_index()
dept_items.columns = ['Department', '# of Items Per Department']
dept_items
```

Out[136]:

	Department	# of Items Per Department
0	1	3355
1	2	45
2	3	1806
3	4	14941
4	5	260
5	6	427
6	7	4310
7	8	159
8	9	1393
9	10	64
10	11	702
11	12	1077
12	13	2963
13	14	1091
14	15	1643
15	16	8503
16	17	1094
17	18	716
18	19	4470
19	20	1652
20	21	107

In [137…

```
# aggregate the order_products data set to get the count of the number of items ordere
# reset the index and rename the columns after you've aggregated the data, so the colu
aisle_items = order_products[['aisle_id', 'add_to_cart_order']].groupby(['aisle_id']).
aisle_items = aisle_items.reset_index()
aisle_items.columns = ['Aisle', '# Of Items Per Aisle']
aisle_items
```

Out[137]:

	Aisle	# Of Items Per Aisle
0	1	104
1	2	155
2	3	663
3	4	345
4	5	90
...	...	...
129	130	227
130	131	402
131	132	5
132	133	25
133	134	11

134 rows × 2 columns

In [138…

```
# aggregate the order_products data set to get the count of the number of items ordere
# reset the index and rename the columns after you've aggregated the data, so the colu
dow_items = order_products[['order_dow', 'add_to_cart_order']].groupby(['order_dow']).
dow_items = dow_items.reset_index()
dow_items.columns = ['Day Of The Week', '# Of Items Per Day']
dow_items
# To interpret the data, remember that 0 = Monday, 1 = Tuesday, 2 = Wednesday, etc.
```

Out[138]:

	Day Of The Week	# Of Items Per Day
0	0	9845
1	1	8830
2	2	6907
3	3	5735
4	4	5755
5	5	6540
6	6	7166

In [139…

```
# aggregate the orders data set to get the mean number of items (num_items) ordered pe
# reset the index and rename the columns after you've aggregated the data, so the colu
mean_items = orders[['num_items', 'order_number']].groupby(['num_items']).agg('mean')
mean_items = mean_items.reset_index()
mean_items.columns = ['Order Number', 'Mean # Of Items']
mean_items
```

Out[139]:

	Order Number	Mean # Of Items
0	1	16.516279
1	2	16.545139
2	3	17.145062
3	4	18.994490
4	5	18.308383
5	6	17.626667
6	7	17.136778
7	8	16.025157
8	9	16.039427
9	10	16.558704
10	11	16.163636
11	12	17.982456
12	13	18.481283
13	14	17.593548
14	15	17.364238
15	16	16.727273
16	17	19.672566
17	18	15.464646
18	19	20.058824
19	20	16.962963
20	21	16.571429
21	22	17.183099
22	23	14.754386
23	24	15.418605
24	25	17.756757
25	26	16.125000
26	27	18.565217
27	28	17.666667
28	29	13.315789
29	30	24.181818
30	31	24.000000
31	32	10.300000
32	33	24.750000

	Order Number	Mean # Of Items
33	34	16.142857
34	35	6.666667
35	36	23.111111
36	37	15.000000
37	38	29.000000
38	39	18.500000
39	40	6.500000
40	41	28.500000
41	42	13.166667
42	43	21.250000
43	46	25.000000
44	50	8.000000
45	51	30.000000
46	52	9.000000
47	61	18.000000

In [140…

```
# aggregate the orders data set to get the median number of items (num_items) ordered
# reset the index and rename the columns after you've aggregated the data, so the colu
median_items = orders[['num_items', 'order_number']].groupby(['num_items']).agg('media
median_items = median_items.reset_index()
median_items.columns = ['Order Number', 'Median # Of Items']
median_items
```

Out[140]:

	Order Number	Median # Of Items
0	1	9.0
1	2	11.0
2	3	11.5
3	4	12.0
4	5	13.0
5	6	11.0
6	7	11.0
7	8	10.0
8	9	10.0
9	10	10.0
10	11	10.0
11	12	10.0
12	13	11.0
13	14	11.0
14	15	11.0
15	16	11.0
16	17	14.0
17	18	10.0
18	19	12.0
19	20	10.0
20	21	13.0
21	22	13.0
22	23	10.0
23	24	9.0
24	25	13.0
25	26	10.5
26	27	8.0
27	28	13.0
28	29	10.0
29	30	13.0
30	31	13.5
31	32	9.5
32	33	19.5

	Order Number	Median # Of Items
33	34	10.0
34	35	8.0
35	36	22.0
36	37	9.0
37	38	12.0
38	39	18.5
39	40	6.0
40	41	36.5
41	42	13.5
42	43	21.5
43	46	25.0
44	50	8.0
45	51	30.0
46	52	9.0
47	61	18.0

Questions to think about:

- Which department had the most items ordered? The fewest?
- Which aisle had the most items ordered? The fewest?
- Which day of the week had the most items ordered? The fewest?
- On average do people order more, fewer, or the same number of items after they have tried the service (order numbers 2, 3, etc.) compared to the first time they try the service (order numbers 1)?

Part B: Subsetting/Merging

In [141…
```
# Run this cell to create the example data sets needed for the next section
fruit_weights = pd.DataFrame({'fruit':['apples','bananas','apricots'],'weight_lbs':[0.
fruit_prices = pd.DataFrame({'fruit':['oranges','bananas','apples'],'price_lbs':[1.39,
```

Subsetting

In [142…
```
# In fruit_weights, select rows where the fruit weight in pounds (weight_lbs) is less
fw = fruit_weights.loc[fruit_weights['weight_lbs'] < .4]
fw
```

Out[142]:

	fruit	weight_lbs
1	bananas	0.33
2	apricots	0.10

In [143…
```
# In fruit_prices, select rows where the fruit price per pound (price_lbs) is higher t
fp = fruit_prices.loc[fruit_prices['price_lbs'] > 1.00]
fp
```

Out[143]:

	fruit	price_lbs
0	oranges	1.39
2	apples	2.99

In [144…
```
# In fruit_prices, select only rows where fruit = 'apples'
ft = fruit_prices.loc[fruit_prices['fruit'] == 'apples']
ft
```

Out[144]:

	fruit	price_lbs
2	apples	2.99

Merging

In [145…
```
# Merge fruit_weights and fruit_prices, resulting in a dataframe called merge1. Your f
# Before doing this, ask yourself "What type of join results in all records being pull
# Get the number of rows and print the first few rows
merge1 = pd.merge(fruit_weights, fruit_prices, how = 'outer', on = ['fruit'])
merge1.shape
merge1.head()
```

Out[145]:

	fruit	weight_lbs	price_lbs
0	apples	0.50	2.99
1	bananas	0.33	0.58
2	apricots	0.10	NaN
3	oranges	NaN	1.39

In [50]:
```
# Merge fruit_weights and fruit_prices, resulting in a dataframe called merge2.
# Your final data set should *only* have fruits for which there is a record in both th
# Get the number of rows and print the first few rows
merge2 = pd.merge(fruit_weights, fruit_prices, how = 'inner', on = ['fruit'])
merge2.shape
merge2.head()
```

Out[50]:

	fruit	weight_lbs	price_lbs
0	apples	0.50	2.99
1	bananas	0.33	0.58

In [127…
```
# Merge the organic and conventional data sets, resulting in a dataframe called merge3
# Get the number of rows and print the first few rows
merge3 = pd.merge(organic, conventional, how = 'inner', on = ['aisle_id', 'aisle'])
merge3.shape
merge3.head()
```

Out[127]:

	aisle_id	num_items_x	aisle	num_items_y
0	1	895	prepared soups salads	2679
1	2	36	specialty cheeses	3801
2	3	2289	energy granola bars	15134
3	4	1320	instant foods	8539
4	5	191	marinades meat preparation	2623

In [125…
```
# Merge the organic and conventional data sets, resulting in a dataframe called merge4
# get the number of rows and print the first few rows
merge4 =  pd.merge(organic, conventional, how = 'outer', on = ['aisle_id', 'aisle'])
merge4.shape
merge4.head()
```

Out[125]:

	aisle_id	num_items_x	aisle	num_items_y
0	1	895.0	prepared soups salads	2679
1	2	36.0	specialty cheeses	3801
2	3	2289.0	energy granola bars	15134
3	4	1320.0	instant foods	8539
4	5	191.0	marinades meat preparation	2623

In [146…
```
# Merge the organic and conventional data sets, resulting in a dataframe called merge5
# Get the number of rows and print the first few rows
merge5 = pd.merge(organic, conventional, how = 'left', on = ['aisle_id', 'aisle'])
merge5.shape
merge5.head()
```

Out[146]:

	aisle_id	num_items_x	aisle	num_items_y
0	1	895	prepared soups salads	2679
1	2	36	specialty cheeses	3801
2	3	2289	energy granola bars	15134
3	4	1320	instant foods	8539
4	5	191	marinades meat preparation	2623

In [147…
```
# Merge the organic and conventional data sets, resulting in a dataframe called merge6
# Get the number of rows and print the first few rows
merge6 =  pd.merge(organic, conventional, how = 'right', on = ['aisle_id', 'aisle'])
merge6.shape
merge6.head()
```

Out[147]:

	aisle_id	num_items_x	aisle	num_items_y
0	1	895.0	prepared soups salads	2679
1	2	36.0	specialty cheeses	3801
2	3	2289.0	energy granola bars	15134
3	4	1320.0	instant foods	8539
4	5	191.0	marinades meat preparation	2623

Questions to think about:

- Why shouldn't you merge on "num_items"?
- How do the number of rows differ among the different merges?
- Why do the merges result in different data sets?
- Which merge is the appropriate choice? Why?

Part C: Reshaping/Calculating

Reshaping

In [148…

```
# Run this cell to create the example data sets needed for the next section
fruit_orders = pd.DataFrame({'fruit':['apples','bananas','apricots'],'order1':[4,0,5],
fruit_inventory = pd.DataFrame({'fruit':['apples','apples','oranges','oranges'],'store
fruit_orders
```

Out[148]:

	fruit	order1	order2	order3
0	apples	4	1	6
1	bananas	0	3	5
2	apricots	5	0	4

In [149…

```
# convert fruit_orders to long format
fruit_ordlong = pd.melt(fruit_orders, id_vars='fruit',
                                      value_vars=['order1', 'order2', 'order3'],
                                      var_name='Order Number', value_name='Fruit Type')
fruit_ordlong
```

Out[149]:

	fruit	Order Number	Fruit Type
0	apples	order1	4
1	bananas	order1	0
2	apricots	order1	5
3	apples	order2	1
4	bananas	order2	3
5	apricots	order2	0
6	apples	order3	6
7	bananas	order3	5
8	apricots	order3	4

In [150…

```
# convert fruit_inventory to wide format
fruit_invwide = fruit_inventory.pivot(index='fruit', columns='store', values='inventor
fruit_invwide
```

Out[150]:

store	giant	safeway
fruit		
apples	75	100
oranges	59	62

In [151…

```
# convert the merged conventional/organic data set with how="inner", merge3, to long f
merge3_long = pd.melt(merge3, id_vars = 'aisle_id',
                      value_vars=['num_items_x', 'aisle', 'num_items_y'],
                      var_name='aisle', value_name='num_items')
merge3_long
```

Out[151]:

	aisle_id	aisle	num_items
0	1	num_items_x	895
1	2	num_items_x	36
2	3	num_items_x	2289
3	4	num_items_x	1320
4	5	num_items_x	191
...	...	...	...
364	129	num_items_y	8228
365	130	num_items_y	4848
366	131	num_items_y	8498
367	132	num_items_y	244
368	133	num_items_y	895

369 rows × 3 columns

Calculating

In [152…

```
# Create a new column in merge1, the fruit prices dataset, that equals the total price
# HINT: price_per_item = price per pound (price_lbs) * lbs (weight_lbs)
# print the first few rows
merge1['price_per_item'] = (merge1['price_lbs'] * merge1['weight_lbs'])
merge1.head()
```

Out[152]:

	fruit	weight_lbs	price_lbs	price_per_item
0	apples	0.50	2.99	1.4950
1	bananas	0.33	0.58	0.1914
2	apricots	0.10	NaN	NaN
3	oranges	NaN	1.39	NaN

In [153…

```
# Create a new column in the fruit_orders dataset that equals the total number of item
# HINT: sum number of items ordered in order1, order2, order3
# print the first few rows
fruit_orders['sum of ordered items'] = fruit_orders['order1'] + fruit_orders['order2']
fruit_orders.head()
```

Out[153]:

	fruit	order1	order2	order3	sum of ordered items
0	apples	4	1	6	11
1	bananas	0	3	5	8
2	apricots	5	0	4	9

In [154…

```
# Create a new column in merge1 that equals the weight per item in grams instead of lb
```

```
# use apply with a lambda function
# 1 pound = 454 grams
# HINT: weight grams = weight lbs * (454)
# print the first few rows
merge1['grams per item'] = merge1['weight_lbs'].apply(lambda x:(x*454), 1)
merge1
```

Out[154]:

	fruit	weight_lbs	price_lbs	price_per_item	grams per item
0	apples	0.50	2.99	1.4950	227.00
1	bananas	0.33	0.58	0.1914	149.82
2	apricots	0.10	NaN	NaN	45.40
3	oranges	NaN	1.39	NaN	NaN

In [ ]: