Ben Deatsman

Module 4 Lab

Part 1

In [515…
```python
import pandas as pd

df = pd.read_excel('mbta.xlsx', header=1, index_col=0)
```

In [516…
```python
df
```

Out[516…

	mode	2007-01	2007-02	2007-03	2007-04	2007-05	2007-06	2007-07	2007-08	2007-09	...	2011-01	2011-02	2011-03	2011-04	2011-05	2011-06	20
1	All Modes by Qtr	NaN	NaN	1187.653	NaN	NaN	1245.959	NaN	NaN	1256.571	...	NaN	NaN	1223.452	NaN	NaN	1302.414	
2	Boat	4.000	3.600	40.000	4.300	4.900	5.800	6.521	6.572	5.469	...	3.140	3.284	3.674	4.251	4.431	5.474	
3	Bus	335.819	338.675	339.867	352.162	354.367	350.543	357.519	355.479	372.598	...	334.958	346.234	380.399	380.446	385.289	376.317	36
4	Commuter Rail	142.200	138.500	137.700	139.500	139.000	143.000	142.391	142.364	143.051	...	128.396	125.463	134.374	134.169	136.140	135.581	13
5	Heavy Rail	435.294	448.271	458.583	472.201	474.579	477.032	471.735	461.605	499.566	...	468.418	504.068	516.730	528.631	528.122	529.528	53
6	Light Rail	227.231	240.262	241.444	255.557	248.262	246.108	243.286	234.907	265.748	...	198.450	219.886	227.935	242.280	225.776	221.865	23
7	Pct Chg / Yr	0.020	-0.040	0.114	-0.002	0.049	0.096	-0.037	0.004	-0.007	...	-0.028	0.008	0.050	0.036	0.050	0.054	
8	Private Bus	4.772	4.417	4.574	4.542	4.768	4.722	3.936	3.946	4.329	...	2.213	2.570	2.559	2.762	2.776	2.815	
9	RIDE	4.900	5.000	5.500	5.400	5.400	5.600	5.253	5.308	5.609	...	6.735	7.463	8.387	8.145	8.059	8.377	
10	Trackless Trolley	12.757	12.913	13.057	13.444	13.479	13.323	13.311	13.142	14.393	...	11.104	11.695	12.601	12.599	12.291	12.128	1
11	TOTAL	1166.974	1191.639	1204.725	1247.105	1244.755	1246.129	1243.952	1223.323	1310.764	...	1153.413	1220.663	1286.660	1313.283	1302.884	1292.085	128

11 rows × 59 columns

In [517…
```python
df.isnull().sum()
```

Out[517…
```
mode       0
2007-01    1
2007-02    1
2007-03    0
2007-04    1
2007-05    1
2007-06    0
2007-07    1
2007-08    1
2007-09    0
2007-10    1
2007-11    1
2007-12    0
2008-01    1
2008-02    1
2008-03    0
2008-04    1
2008-05    1
2008-06    0
2008-07    1
2008-08    1
2008-09    0
2008-10    1
2008-11    1
2008-12    0
2009-01    1
2009-02    1
2009-03    0
2009-04    1
2009-05    1
2009-06    0
2009-07    1
2009-08    1
2009-09    0
2009-10    1
2009-11    1
2009-12    0
2010-01    1
2010-02    1
2010-03    0
2010-04    1
2010-05    1
2010-06    0
2010-07    1
2010-08    1
2010-09    0
2010-10    1
2010-11    1
2010-12    0
2011-01    1
2011-02    1
2011-03    0
2011-04    1
2011-05    1
2011-06    0
2011-07    1
2011-08    1
2011-09    0
2011-10    1
dtype: int64
```

In [518…
```python
print('There are', df.isnull().sum().sum(), 'missing values.')
```

```
There are 39 missing values.
```

In [519…
```python
df = df.drop([1, 7, 11])
df
print('The df size is', df.shape)
```

```
The df size is (8, 59)
```

In [520…
```python
df = df.melt(
    id_vars = 'mode',
    var_name = 'date',
    value_name = 'num_riders_1000'
)
df
```

Out[520…

	mode	date	num_riders_1000
0	Boat	2007-01	4.000
1	Bus	2007-01	335.819
2	Commuter Rail	2007-01	142.200
3	Heavy Rail	2007-01	435.294
4	Light Rail	2007-01	227.231
...	...	...	...
459	Heavy Rail	2011-10	554.932
460	Light Rail	2011-10	237.768
461	Private Bus	2011-10	2.967
462	RIDE	2011-10	8.598
463	Trackless Trolley	2011-10	12.297

464 rows × 3 columns

In [521…
```python
print('The new dimensions are', df.shape)
```

```
The new dimensions are (464, 3)
```

In [522…
```python
df[['year', 'month']] = df['date'].str.split('-', expand=True)
df.drop(columns=['date'], inplace=True)
df.head()
```

Out[522…

	mode	num_riders_1000	year	month
0	Boat	4.000	2007	01
1	Bus	335.819	2007	01
2	Commuter Rail	142.200	2007	01
3	Heavy Rail	435.294	2007	01
4	Light Rail	227.231	2007	01

In [523…
```python
df[df['num_riders_1000'] == 40]
```

Out[523…

	mode	num_riders_1000	year	month
16	Boat	40.0	2007	03

In [524…
```python
df['num_riders_1000'].replace(40.000, 4.000, inplace=True)
```

In [525…
```python
df.groupby('mode', as_index=False).agg({'num_riders_1000': 'mean'})
```

Out[525…

	mode	num_riders_1000
0	Boat	4.447276
1	Bus	358.590448
2	Commuter Rail	137.351534
3	Heavy Rail	489.293483
4	Light Rail	232.988810
5	Private Bus	3.352190
6	RIDE	6.603672
7	Trackless Trolley	12.125052

In [526…
```python
df['month'].replace({'01': 'Jan',
                     '02': 'Feb',
                     '03': 'Mar',
                     '04': 'Apr',
                     '05': 'May',
                     '06': 'Jun',
                     '07': 'Jul',
                     '08': 'Aug',
                     '09': 'Sep',
                     '10': 'Oct',
                     '11': 'Nov',
                     '12': 'Dec'}, inplace=True)
```

In [527…
```python
df_sort = df[df['month'] == 'Jan']
df_sort.groupby(['mode', 'month'], as_index=False).agg({'num_riders_1000': 'mean'})
```

Out[527…

	mode	month	num_riders_1000
0	Boat	Jan	3.3072
1	Bus	Jan	342.3718
2	Commuter Rail	Jan	137.0198
3	Heavy Rail	Jan	460.2818
4	Light Rail	Jan	217.3920
5	Private Bus	Jan	3.4746
6	RIDE	Jan	5.9354
7	Trackless Trolley	Jan	12.4658

In [528…
```python
df_boat = df[df['mode'] == 'Boat']
df_boat.groupby(['mode', 'year'], as_index=False).agg({'num_riders_1000': 'sum'}).sort_values('num_riders_1000', ascending=False)
```

Out[528…

	mode	year	num_riders_1000
0	Boat	2007	57.055
3	Boat	2010	52.931
2	Boat	2009	50.552
1	Boat	2008	50.349
4	Boat	2011	47.055

In [529…
```python
df_heavyrail = df[df['mode'] == 'Heavy Rail']
df_heavyrail.groupby(['mode', 'month'], as_index=False).agg({'num_riders_1000': 'mean'}).sort_values('num_riders_1000', ascending=False)
```

Out[529…

	mode	month	num_riders_1000
11	Heavy Rail	Sep	517.70580
10	Heavy Rail	Oct	516.42520
0	Heavy Rail	Apr	501.91240
6	Heavy Rail	Jun	497.65420
5	Heavy Rail	Jul	495.26940
8	Heavy Rail	May	494.73740
9	Heavy Rail	Nov	489.65700
7	Heavy Rail	Mar	483.99600
3	Heavy Rail	Feb	481.39340
1	Heavy Rail	Aug	477.42940
4	Heavy Rail	Jan	460.28180
2	Heavy Rail	Dec	446.59225

Part 2

In [531…
```python
from completejourney_py import get_data

cj_data = get_data()

import pandas as pd
transactions = cj_data['transactions']
demographics = cj_data['demographics']
products = cj_data['products']
```

In [532…
```python
cj_data.keys()
```

Out[532…
```
dict_keys(['campaign_descriptions', 'coupons', 'promotions', 'campaigns', 'demographics', 'transactions', 'coupon_redemptions', 'products'])
```

In [533…
```python
transactions.columns.intersection(demographics.columns)
```

Out[533…
```
Index(['household_id'], dtype='object')
```

In [534…
```python
df1 = transactions.merge(demographics, how='left', on='household_id')
df1
```

Out[534…

	household_id	store_id	basket_id	product_id	quantity	sales_value	retail_disc	coupon_disc	coupon_match_disc	week	transaction_timestamp	age	inco
0	900	330	31198570044	1095275	1	0.50	0.00	0.0	0.0	1	2017-01-01 11:53:26	35-44	35-4
1	900	330	31198570047	9878513	1	0.99	0.10	0.0	0.0	1	2017-01-01 12:10:28	35-44	35-4
2	1228	406	31198655051	1041453	1	1.43	0.15	0.0	0.0	1	2017-01-01 12:26:30	45-54	10 12
3	906	319	31198705046	1020156	1	1.50	0.29	0.0	0.0	1	2017-01-01 12:30:27	55-64	Unc 1
4	906	319	31198705046	1053875	2	2.78	0.80	0.0	0.0	1	2017-01-01 12:30:27	55-64	Unc 1
...	...	...	...	...	...	...	...	...	...	...	...	...	
1469302	679	447	41453103606	14025548	1	0.79	0.20	0.0	0.0	53	2018-01-01 03:50:03	NaN	N
1469303	2070	311	41453083334	909894	1	1.73	0.17	0.0	0.0	53	2018-01-01 04:01:20	45-54	50-7
1469304	2070	311	41453083334	933067	2	5.00	2.98	0.0	0.0	53	2018-01-01 04:01:20	45-54	50-7
1469305	2070	311	41453083334	1029743	1	2.60	0.29	0.0	0.0	53	2018-01-01 04:01:20	45-54	50-7
1469306	2070	311	41453083334	1061220	1	1.19	0.13	0.0	0.0	53	2018-01-01 04:01:20	45-54	50-7

1469307 rows × 18 columns

In [535…
```python
df1.notnull().sum()
```

Out[535…
```
household_id             1469307
store_id                 1469307
basket_id                1469307
product_id               1469307
quantity                 1469307
sales_value              1469307
retail_disc              1469307
coupon_disc              1469307
coupon_match_disc        1469307
week                     1469307
transaction_timestamp    1469307
age                       828850
income                    828850
home_ownership            607943
marital_status            687418
household_size            828850
household_comp            828850
kids_count                828850
dtype: int64
```

In [536…
```python
(
 transactions
 .merge(demographics, how='left', on='household_id')
 .groupby('age', as_index=False)
 .agg({'sales_value': 'sum'})
 .sort_values(by='sales_value', ascending=False)
)
```

Out[536…

	age	sales_value
3	45-54	971822.19
2	35-44	724357.40
1	25-34	453372.46
5	65+	176600.84
4	55-64	173153.70
0	19-24	125673.05

In [595…
```python
pizza_filter = products['product_type'].str.contains('pizza', case=False, na=False)
(products[pizza_filter]
 .merge(cj_data['transactions'] , how='inner', on='product_id')
 .groupby(['product_id', 'product_type', 'product_category'], as_index=False)
 .agg({'sales_value': 'sum'})
 .sort_values(by='sales_value', ascending=False)
)
```

Out[595…

	product_id	product_type	product_category	sales_value
97	944139	PIZZA/TRADITIONAL	FROZEN PIZZA	1344.50
70	906838	PIZZA/PREMIUM	FROZEN PIZZA	1263.78
357	12648296	PIZZA/TRADITIONAL	FROZEN PIZZA	1230.36
113	969568	PIZZA/TRADITIONAL	FROZEN PIZZA	1125.18
85	925626	PIZZA/ECONOMY	FROZEN PIZZA	1017.60
...	...	...	...	...
29	610240	PIZZA/ECONOMY	FROZEN PIZZA	0.99
203	1231548	PIZZA/ECONOMY	FROZEN PIZZA	0.99
232	2063332	PIZZA/ECONOMY	FROZEN PIZZA	0.67
266	6876820	PIZZA/ECONOMY	FROZEN PIZZA	0.59
334	10341942	QSR FD: COLD PIZZA	PREPARED FOOD	0.00

394 rows × 4 columns

In [597…
```python
pizza_cat_filter = products['product_category'].str.contains('pizza', case=False, na=False) & products['product_type'].str.contains('snack|appetizer',
(products[pizza_cat_filter]
 .merge(cj_data['transactions'] , how='inner', on='product_id')
 .groupby(['product_id', 'product_type', 'product_category'], as_index=False)
 .agg({'sales_value': 'sum'})
 .sort_values(by='sales_value', ascending=False)
)
```

Out[597…

	product_id	product_type	product_category	sales_value
37	907631	SNACKS/APPETIZERS	FROZEN PIZZA	1990.50
34	890695	SNACKS/APPETIZERS	FROZEN PIZZA	1686.28
18	845193	SNACKS/APPETIZERS	FROZEN PIZZA	1507.54
22	856252	SNACKS/APPETIZERS	FROZEN PIZZA	1133.90
15	829291	SNACKS/APPETIZERS	FROZEN PIZZA	1020.61
...	...	...	...	...
113	2215639	SNACKS/APPETIZERS	FROZEN PIZZA	1.89
186	15831485	SNACKS/APPETIZERS	FROZEN PIZZA	1.79
99	1335834	SNACKS/APPETIZERS	FROZEN PIZZA	1.25
110	2017119	SNACKS/APPETIZERS	FROZEN PIZZA	0.48
108	1917329	SNACKS/APPETIZERS	FROZEN PIZZA	0.38

188 rows × 4 columns

In [601…
```python
pb_filter = products['product_type'].str.contains('peanut butter', case=False, na=False)
pb_products = products[pb_filter]

print('There are', pb_products['product_id'].nunique(), 'unique peanut butter products')
```

```
There are 144 unique peanut butter products
```

In [588…
```python
pb_merge = pb_products.merge(cj_data['transactions'] , how='inner', on='product_id')
pb_merge['transaction_month'] = pd.to_datetime(pb_merge['transaction_timestamp']).dt.month

(pb_merge.groupby('transaction_month', as_index=False)
                 .agg({'sales_value': 'sum'})
                 .sort_values(by='sales_value', ascending=False))
```

Out[588…

	transaction_month	sales_value
11	12	1019.08
8	9	905.36
7	8	895.10
10	11	882.32
9	10	878.89
5	6	865.15
6	7	863.38
0	1	863.36
4	5	850.37
3	4	790.18
1	2	755.76
2	3	733.60

January has the highest sales value for products containing peanut butter in the product type

In [ ]: