Final Project

Ok, so today we'll be working with the transactions, product, and hh_demographic tables in the project_data folder.

- First, read in the transactions data.
- Read in the only columns `household_key`, `BASKET_ID`, `DAY`, `PRODUCT_ID`, `QUANTITY`, and `SALES_VALUE`.
- Convert `DAY`, `QUANTITY`, and `PRODUCT_ID` to the smallest appropriate integer types.

In [1]:
```
import pandas as pd
import numpy as np
```

In [10]:
```
path = "../project_data/project_transactions.csv"
cols = ["household_key", "BASKET_ID", "DAY", "PRODUCT_ID", "QUANTITY", "SALES_VALUE
dtypes = {"DAY" : "Int16", "QUANTITY" : "Int32", "PRODUCT_ID" : "Int32"}
transactions = pd.read_csv(path,
                           usecols = cols,
                           dtype = dtypes
                          )
transactions.head()
```

Out[10]:

	household_key	BASKET_ID	DAY	PRODUCT_ID	QUANTITY	SALES_VALUE
0	1364	26984896261	1	842930	1	2.19
1	1364	26984896261	1	897044	1	2.99
2	1364	26984896261	1	920955	1	3.09
3	1364	26984896261	1	937406	1	2.50
4	1364	26984896261	1	981760	1	0.60

In [13]:
```
transactions.info(memory_usage = "deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2146311 entries, 0 to 2146310
Data columns (total 6 columns):
 #   Column         Dtype
---  ------         -----
 0   household_key  int64
 1   BASKET_ID      int64
 2   DAY            Int16
 3   PRODUCT_ID     Int32
 4   QUANTITY       Int32
 5   SALES_VALUE    float64
dtypes: Int16(1), Int32(2), float64(1), int64(2)
memory usage: 75.7 MB
```

In [12]:
```
transactions.describe().round()
```

Out[12]:

	household_key	BASKET_ID	DAY	PRODUCT_ID	QUANTITY	SALES_VALUE
count	2146311.0	2.146311e+06	2146311.0	2146311.0	2146311.0	2146311.0
mean	1056.0	3.404897e+10	390.0	-861.0	101.0	3.0
std	605.0	4.723748e+09	190.0	3831949.0	1152.0	4.0
min	1.0	2.698490e+10	1.0	25671.0	0.0	0.0
25%	548.0	3.040798e+10	229.0	917231.0	1.0	1.0
50%	1042.0	3.281176e+10	392.0	1027960.0	1.0	2.0
75%	1581.0	4.012804e+10	555.0	1132771.0	1.0	3.0
max	2099.0	4.230536e+10	711.0	18316298.0	89638.0	840.0

In [31]:
```
transactions = (
    transactions
    .assign(date = (pd.to_datetime("2016", format='%Y')
                    + pd.to_timedelta(transactions["DAY"].sub(1).astype(str) + " da
            )
    .drop(["DAY"], axis=1)
)
```

In [32]:
```
transactions
```

Out[32]:

	household_key	BASKET_ID	PRODUCT_ID	QUANTITY	SALES_VALUE	date
0	1364	26984896261	842930	1	2.19	2016-01-01
1	1364	26984896261	897044	1	2.99	2016-01-01
2	1364	26984896261	920955	1	3.09	2016-01-01
3	1364	26984896261	937406	1	2.50	2016-01-01
4	1364	26984896261	981760	1	0.60	2016-01-01
...	...	...	...	...	...	...
2146306	1598	42305362535	92130	1	0.99	2017-12-11
2146307	1598	42305362535	114102	1	8.89	2017-12-11
2146308	1598	42305362535	133449	1	6.99	2017-12-11
2146309	1598	42305362535	6923644	1	4.50	2017-12-11
2146310	1598	42305362535	14055192	1	6.99	2017-12-11

2146311 rows × 6 columns

In [33]:
```
transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2146311 entries, 0 to 2146310
Data columns (total 6 columns):
 #   Column         Dtype         
---  ------         -----         
 0   household_key  int64         
 1   BASKET_ID      int64         
 2   PRODUCT_ID     Int32         
 3   QUANTITY       Int32         
 4   SALES_VALUE    float64       
 5   date           datetime64[ns]
dtypes: Int32(2), datetime64[ns](1), float64(1), int64(2)
memory usage: 86.0 MB
```

TIME BASED ANALYSIS

- Plot the sum of sales by month. Are sales growing over time?
- Next, plot the same series after filtering down to dates April 2016 and October 2017.

- Then, plot the sum of monthly sales in 2016 vs the monthly sales 2017.
- Finally, plot total sales by day of week.

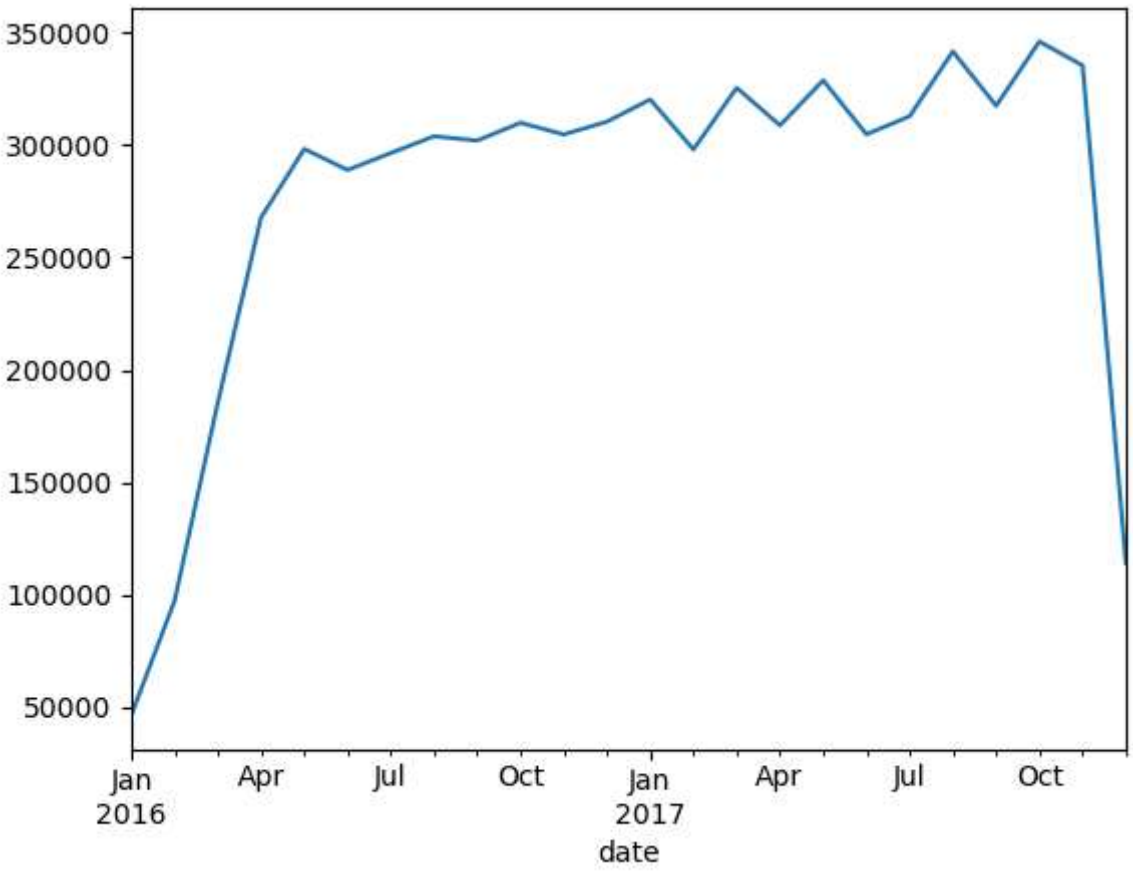
In [34]:
```
transactions.head()
```

Out[34]:

	household_key	BASKET_ID	PRODUCT_ID	QUANTITY	SALES_VALUE	date
0	1364	26984896261	842930	1	2.19	2016-01-01
1	1364	26984896261	897044	1	2.99	2016-01-01
2	1364	26984896261	920955	1	3.09	2016-01-01
3	1364	26984896261	937406	1	2.50	2016-01-01
4	1364	26984896261	981760	1	0.60	2016-01-01

In [40]:
```
(transactions.set_index("date")
.loc[:,"SALES_VALUE"]
.resample("M")
.sum()
.plot())
```

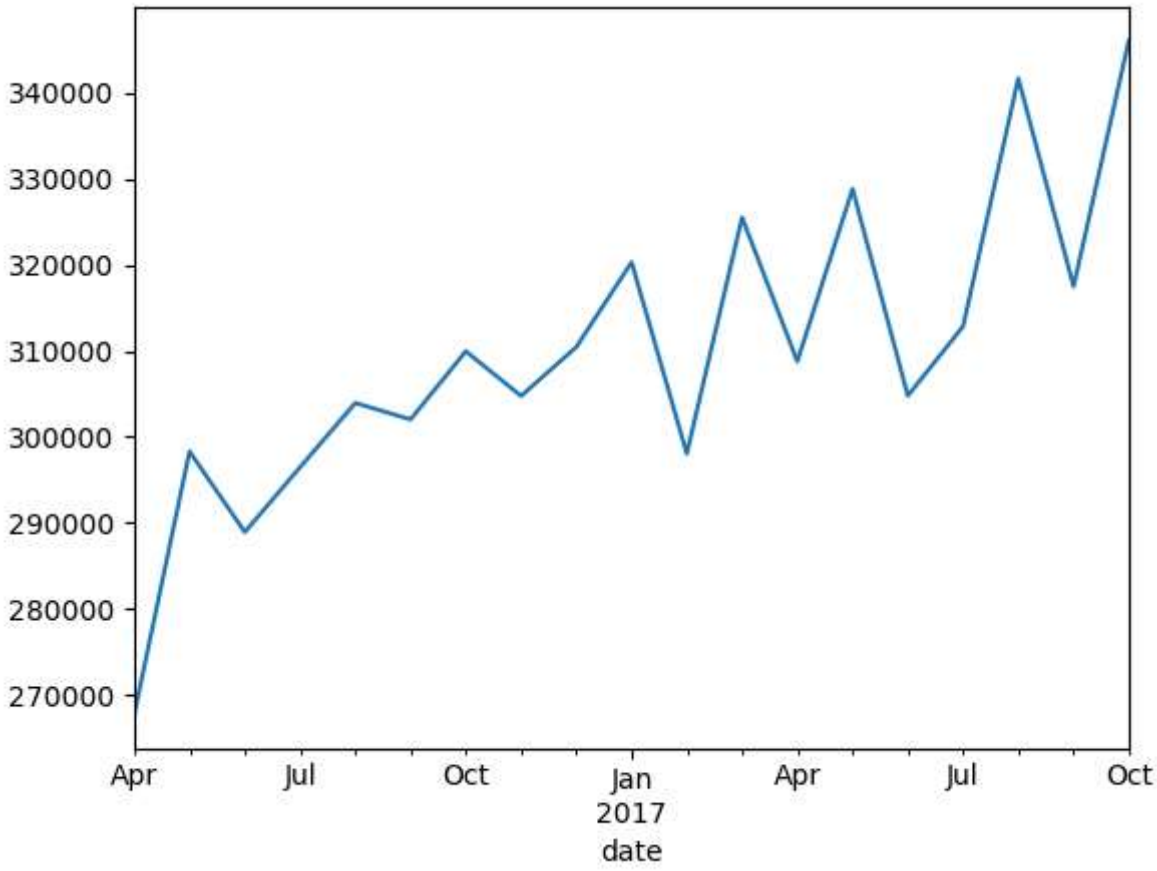
Out[40]: <Axes: xlabel='date'>



In [42]:
```
(transactions.set_index("date")
.loc["2016-04":"2017-10","SALES_VALUE"]
.resample("M")
```

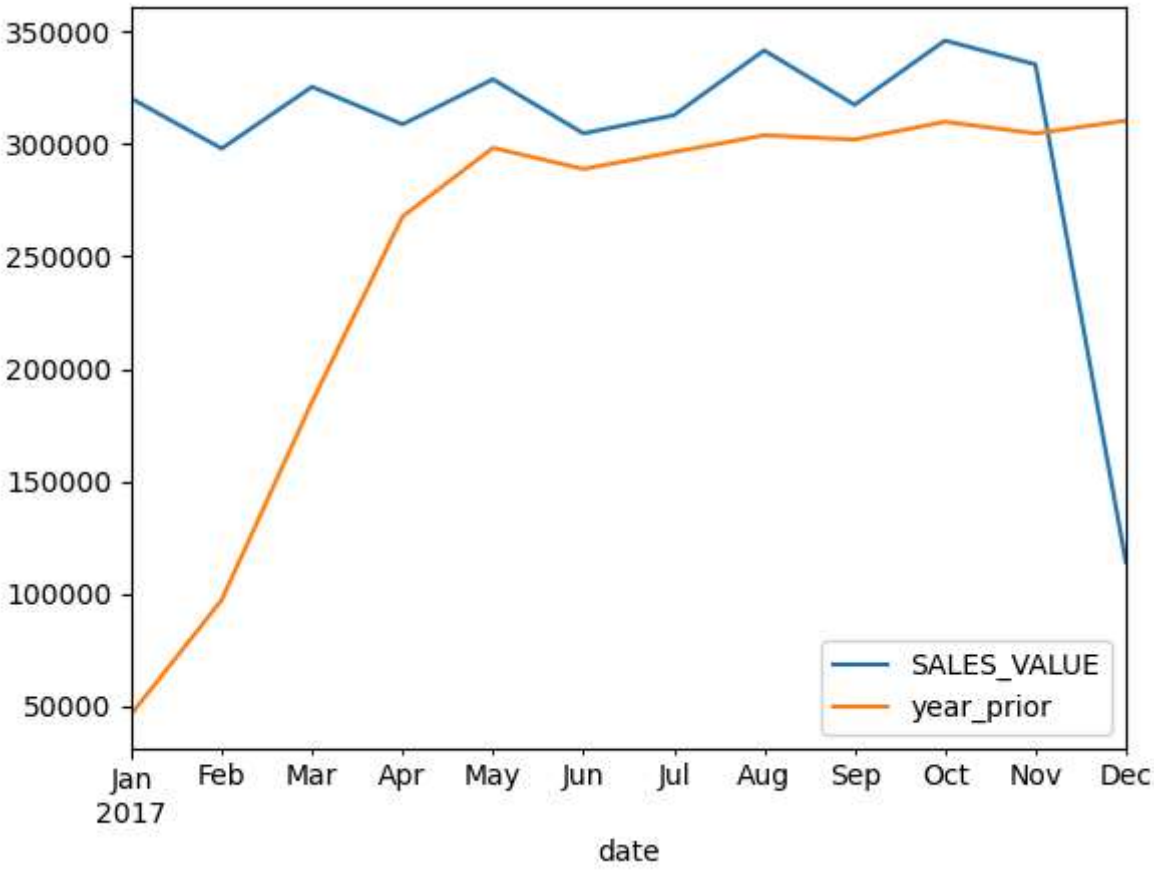
```
.sum()
.plot())
```

Out[42]: <Axes: xlabel='date'>



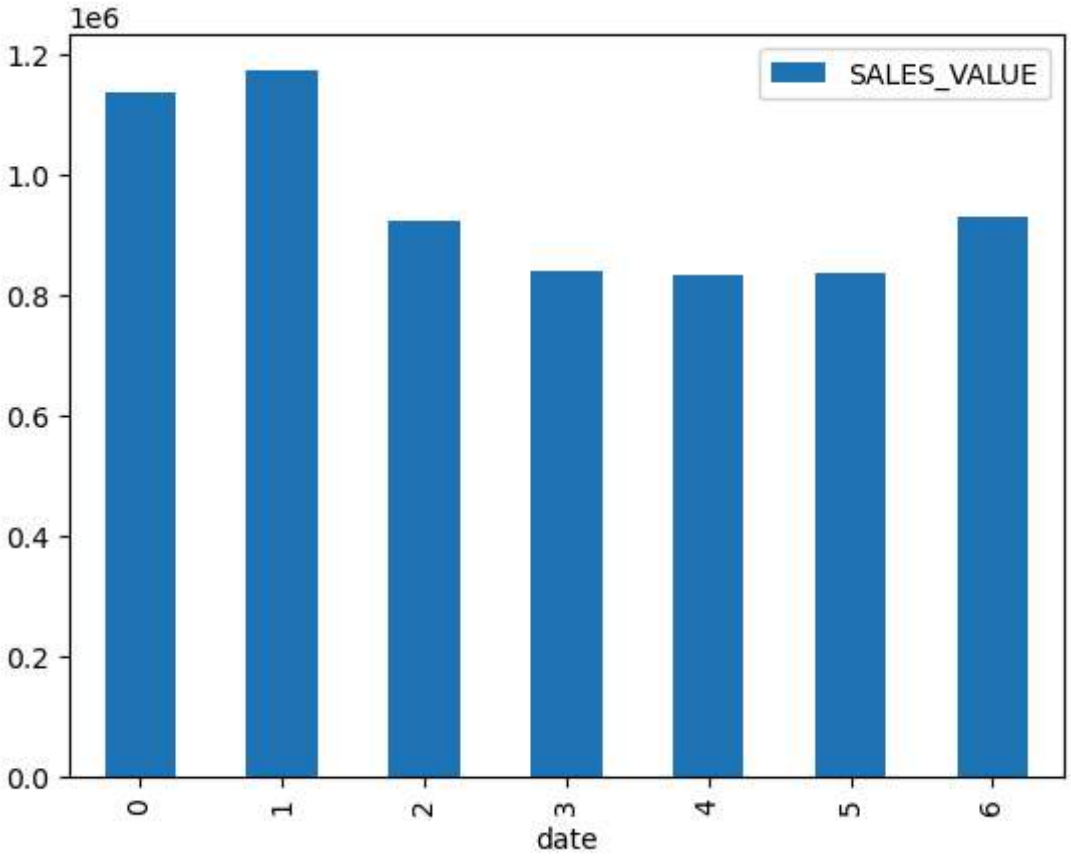
In [50]:
```
(transactions
 .set_index("date")
 .loc[:, ["SALES_VALUE"]]
 .resample("M")
 .sum()
 .assign(year_prior = lambda x: x["SALES_VALUE"].shift(12))
 .loc["2017"]
 .plot()
)
```

Out[50]: <Axes: xlabel='date'>



In [56]:
```
# Group transactions by dayofweek, then calculate sum and plot a bar chart
(transactions
.groupby(transactions["date"].dt.dayofweek)
.agg({"SALES_VALUE" : "sum"})
.plot.bar())
```

Out[56]: <Axes: xlabel='date'>



DEMOGRAPHICS

- Read in the `hh_demographic.csv` file, but only the columns `AGE_DESC`, `INCOME_DESC`, `household_key`, and `HH_COMP_DESC`. Convert the appropriate columns to the category dtype.
- Then group the transactions table by household_id, and calculate the sum of SALES VALUE by household.
- Once you've done that, join the demographics DataFrame to the aggregated transactions table. Since we're interested in analyzing the demographic data we have, make sure not to include rows from transactions that don't match.
- Plot the sum of sales by age_desc and income_desc (in separate charts).
- Then, create a pivot table of the mean household sales by `AGE_DESC` and `HH_COMP_DESC`. Which of our demographics have the highest average sales?

In [64]:
```
path_1 = '../project_data/hh_demographic.csv'
cols_hh = ["AGE_DESC", "INCOME_DESC", "household_key", "HH_COMP_DESC"]
dem_dtype = {"AGE_DESC" : "category", "INCOME_DESC" : "category", "HH_COMP_DESC" :
demographics = pd.read_csv(path_1,
                           usecols = cols_hh,
```

```
                           dtype = dem_dtype)
demographics
```

Out[64]:

	AGE_DESC	INCOME_DESC	HH_COMP_DESC	household_key
0	65+	35-49K	2 Adults No Kids	1
1	45-54	50-74K	2 Adults No Kids	7
2	25-34	25-34K	2 Adults Kids	8
3	25-34	75-99K	2 Adults Kids	13
4	45-54	50-74K	Single Female	16
...	...	...	...	...
796	35-44	50-74K	2 Adults No Kids	2494
797	45-54	75-99K	Unknown	2496
798	45-54	35-49K	Single Male	2497
799	25-34	50-74K	2 Adults No Kids	2498
800	25-34	Under 15K	2 Adults Kids	2499

801 rows × 4 columns

In [65]:
```
demographics.info(memory_usage = "deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype   
---  ------         --------------  -----   
 0   AGE_DESC       801 non-null    category
 1   INCOME_DESC    801 non-null    category
 2   HH_COMP_DESC   801 non-null    category
 3   household_key  801 non-null    int64   
dtypes: category(3), int64(1)
memory usage: 10.9 KB
```

In [67]:
```
transactions.head()
```

Out[67]:

	household_key	BASKET_ID	PRODUCT_ID	QUANTITY	SALES_VALUE	date
0	1364	26984896261	842930	1	2.19	2016-01-01
1	1364	26984896261	897044	1	2.99	2016-01-01
2	1364	26984896261	920955	1	3.09	2016-01-01
3	1364	26984896261	937406	1	2.50	2016-01-01
4	1364	26984896261	981760	1	0.60	2016-01-01

In [73]:
```
# Create total sales by household dataframe
household_sales = (transactions
                   .groupby("household_key")
                   .agg({"SALES_VALUE" : "sum"})
                  )
household_sales.head()
```

Out[73]:

household_key	SALES_VALUE
1	4330.16
2	1954.34
3	2653.21
4	1200.11
5	779.06

In [74]:
```
demographics.head()
```

Out[74]:

	AGE_DESC	INCOME_DESC	HH_COMP_DESC	household_key
0	65+	35-49K	2 Adults No Kids	1
1	45-54	50-74K	2 Adults No Kids	7
2	25-34	25-34K	2 Adults Kids	8
3	25-34	75-99K	2 Adults Kids	13
4	45-54	50-74K	Single Female	16

In [75]:
```
# Join household sales and demographics table on household_key (inner since we're i
household_sales_demo = (household_sales.merge(demographics,
                                              how = "inner",
                                              left_on = "household_key",
                                              right_on = "household_key")
                        )
household_sales_demo
```

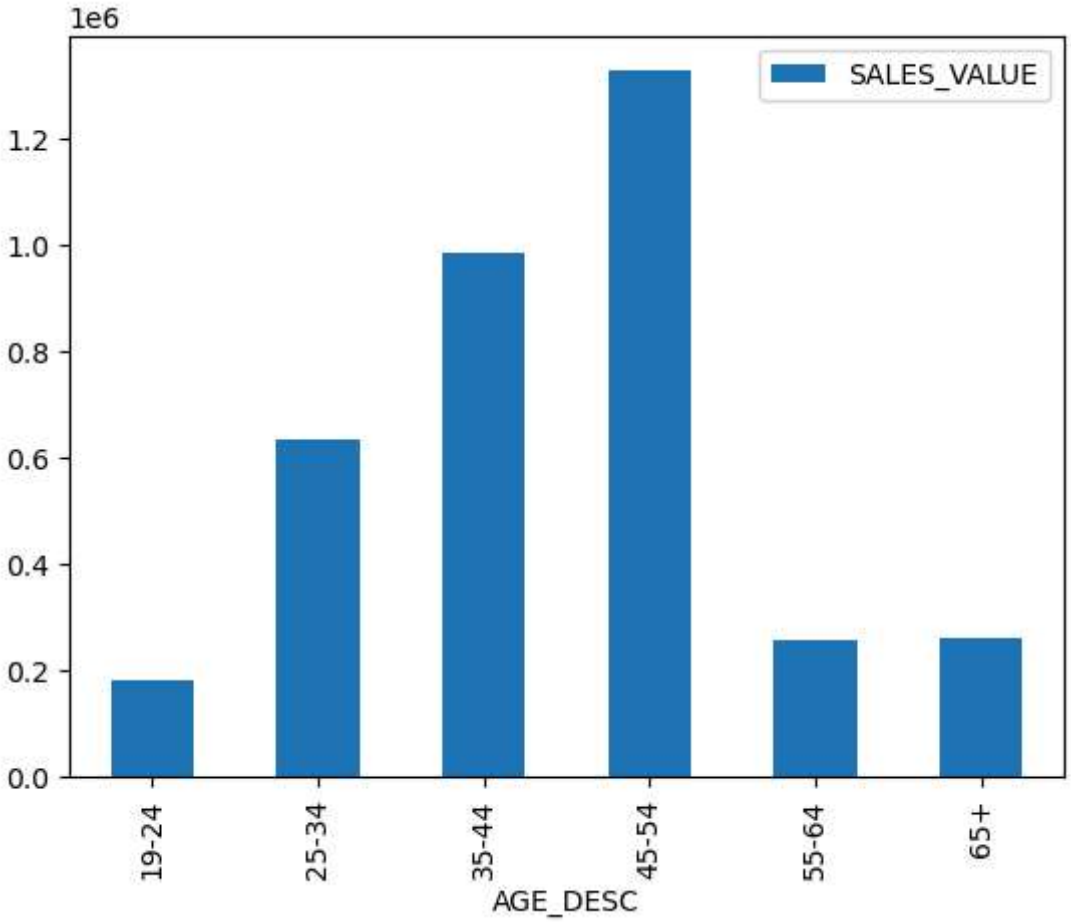
Out[75]:

	household_key	SALES_VALUE	AGE_DESC	INCOME_DESC	HH_COMP_DESC
0	1	4330.16	65+	35-49K	2 Adults No Kids
1	7	3400.05	45-54	50-74K	2 Adults No Kids
2	8	5534.97	25-34	25-34K	2 Adults Kids
3	13	13190.92	25-34	75-99K	2 Adults Kids
4	16	1512.02	45-54	50-74K	Single Female
...	...	...	...	...	...
663	2087	7832.74	45-54	100-124K	Unknown
664	2088	4905.71	25-34	35-49K	Single Female
665	2092	2290.54	35-44	Under 15K	1 Adult Kids
666	2094	3055.52	45-54	50-74K	2 Adults No Kids
667	2097	8823.83	35-44	15-24K	2 Adults Kids

668 rows × 5 columns

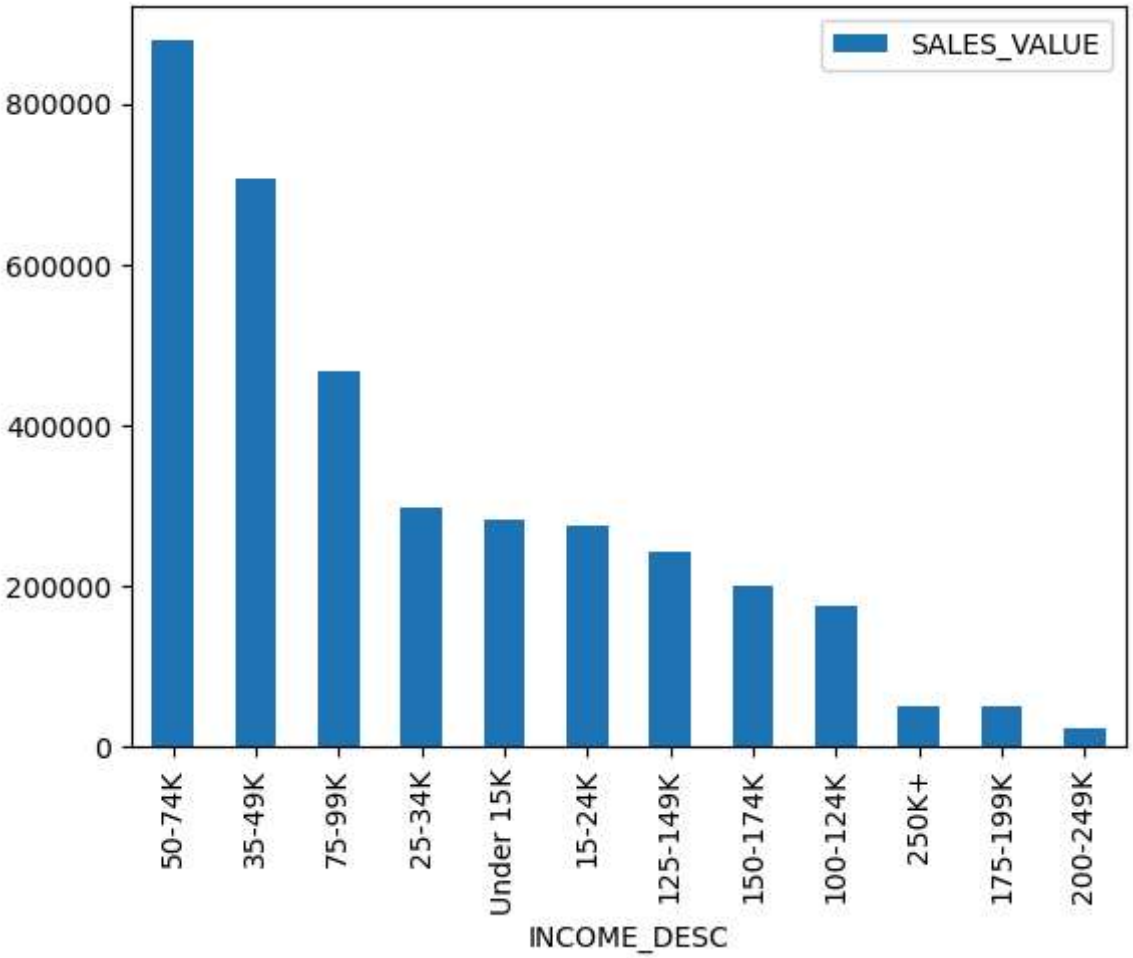
In [79]:
```
# Calculate sum of sales by age group
(household_sales_demo
 .groupby("AGE_DESC")
 .agg({"SALES_VALUE" : "sum"})
 .plot.bar()
)
```

Out[79]:
```
<Axes: xlabel='AGE_DESC'>
```



In [83]:
```
# Calculate sum of sales by income, ordered by magnitude
(household_sales_demo
 .groupby("INCOME_DESC")
 .agg({"SALES_VALUE" : "sum"})
 .sort_values("SALES_VALUE", ascending = False)
 .plot.bar()
)
```

Out[83]: <Axes: xlabel='INCOME_DESC'>



In [98]:
```
# Calculate mean household spend by Age Description and HH Composition
# Format with a heatmap across all cells
(household_sales_demo.pivot_table(index = "AGE_DESC",
                                  columns = "HH_COMP_DESC",
                                  values = "SALES_VALUE",
                                  aggfunc = "mean",
                                  margins = True)
 .style.background_gradient(cmap = "RdYlGn", axis = None)
)
```

Out[98]:

HH_COMP_DESC	1 Adult Kids	2 Adults Kids	2 Adults No Kids	Single Female	Single Male	Unkn
AGE_DESC						
19-24	7268.796667	5428.945000	4020.800000	4576.095556	3216.835000	4911.275
25-34	5512.196875	5753.973514	5638.515833	4807.440588	4909.522381	7356.270
35-44	6297.737778	6691.772264	6260.412444	6015.192069	4844.192000	4227.691
45-54	6632.569167	6610.484490	5839.527027	4549.365405	4636.637083	4843.995
55-64	3064.870000	4695.655000	5752.413684	4816.148462	3922.546250	7973.750
65+	4040.810000	5536.866667	4614.108571	4059.699412	3871.556000	2879.290
All	6032.802143	6280.069103	5599.857756	4895.928361	4544.646750	4936.127

In [ ]:
```
# delete dfs we won't use anymore
del[household_sales_demo, household_sales]
```

PRODUCT DEMOGRAPHICS

- Read in the product csv file.
- Only read in product_id and department from product (consider converting columns).
- Join the product DataFrame to transactions and demographics tables, performing an inner join when joining both tables.
- Finally, pivot the fully joined dataframe by AGE_DESC and DEPARTMENT, calculating the sum of sales. Which category does our youngest demographic perform well in?

In [103…
```
path_2 = '../project_data/product.csv'
prod_cols = ["PRODUCT_ID", "DEPARTMENT"]
prod_dtypes = {"PRODUCT_ID" : "Int32", "DEPARTMENT" : "category"}
products = pd.read_csv(path_2,
                       usecols = prod_cols,
                       dtype = prod_dtypes)
products.head()
```

Out[103…

	PRODUCT_ID	DEPARTMENT
0	25671	GROCERY
1	26081	MISC. TRANS.
2	26093	PASTRY
3	26190	GROCERY
4	26355	GROCERY

In [102…

```
products.info(memory_usage = "deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92353 entries, 0 to 92352
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   PRODUCT_ID  92353 non-null  Int32   
 1   DEPARTMENT  92353 non-null  category
dtypes: Int32(1), category(1)
memory usage: 545.2 KB
```

In [110…

```
# Join all three tables together with an inner join
# Join product on product_id (only shared column)
trans_demo_dept = (transactions
                   .merge(demographics,
                          how = "inner",
                          left_on = "household_key",
                          right_on = "household_key")
                   .merge(products,
                         how = "inner",
                         left_on = "PRODUCT_ID",
                         right_on = "PRODUCT_ID")
                  )
trans_demo_dept
```

Out[110…

	household_key	BASKET_ID	PRODUCT_ID	QUANTITY	SALES_VALUE	date	AG
0	1364	26984896261	842930	1	2.19	2016-01-01	
1	304	27950201718	842930	1	1.67	2016-03-21	
2	575	34186568032	842930	1	1.67	2017-04-29	
3	77	28023861855	842930	3	5.00	2016-03-27	
4	454	42136182098	842930	1	1.67	2017-11-29	
…	…	…	…	…	…	…	
1161570	540	41413346731	913709	1	10.99	2017-10-06	
1161571	540	41587471481	1024615	1	11.99	2017-10-19	
1161572	540	41587471481	9831733	1	0.00	2017-10-19	
1161573	540	41834711355	830676	1	8.99	2017-11-07	
1161574	540	41865495241	924549	1	4.09	2017-11-09	

1161575 rows × 10 columns

In [111…

```
trans_demo_dept.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1161575 entries, 0 to 1161574
Data columns (total 10 columns):
 #   Column         Non-Null Count    Dtype         
---  ------         --------------    -----         
 0   household_key  1161575 non-null  int64         
 1   BASKET_ID      1161575 non-null  int64         
 2   PRODUCT_ID     1161575 non-null  Int32         
 3   QUANTITY       1161575 non-null  Int32         
 4   SALES_VALUE    1161575 non-null  float64       
 5   date           1161575 non-null  datetime64[ns]
 6   AGE_DESC       1161575 non-null  category      
 7   INCOME_DESC    1161575 non-null  category      
 8   HH_COMP_DESC   1161575 non-null  category      
 9   DEPARTMENT     1161575 non-null  category      
dtypes: Int32(2), category(4), datetime64[ns](1), float64(1), int64(2)
memory usage: 59.8 MB
```

In [121…

```
# Where does our youngest demographic rank near the top in sales?
(trans_demo_dept.pivot_table(index = "DEPARTMENT",
                             columns = "AGE_DESC",
                             values = "SALES_VALUE",
                             aggfunc = "sum")
 .style.background_gradient(cmap = "RdYlGn", axis = 1)
)
```

Out[121...

AGE_DESC	19-24	25-34	35-44	45-54	55-64
DEPARTMENT					
	0.000000	0.000000	0.000000	0.000000	0.000000
AUTOMOTIVE	11.640000	21.250000	72.580000	55.920000	0.000000
CHARITABLE CONT	0.000000	0.000000	0.000000	0.000000	0.000000
CHEF SHOPPE	81.300000	134.160000	348.530000	418.240000	80.860000
CNTRL/STORE SUP	2.000000	0.000000	1.000000	9.950000	2.000000
COSMETICS	698.630000	2273.030000	4362.020000	5187.570000	986.260000
COUP/STR & MFG	7.490000	48.420000	121.200000	154.550000	40.680000
DAIRY DELI	3.800000	3.850000	7.390000	16.750000	3.140000
DELI	4043.300000	18181.940000	34577.290000	44334.220000	9850.540000
DELI/SNACK BAR	0.000000	0.000000	6.980000	1.560000	0.000000
DRUG GM	25297.430000	85298.050000	126480.340000	177007.130000	29220.930000
ELECT &PLUMBING	0.000000	0.000000	0.000000	0.000000	0.000000
FLORAL	776.990000	2355.570000	5246.600000	6835.690000	1112.690000
FROZEN GROCERY	1.640000	53.050000	108.960000	84.500000	54.220000
GARDEN CENTER	41.980000	380.110000	701.830000	1487.900000	248.070000
GM MERCH EXP	0.000000	0.000000	17.760000	30.370000	12.050000
GRO BAKERY	0.000000	0.000000	0.000000	2.180000	0.000000
GROCERY	99008.270000	327926.160000	490616.030000	667162.980000	127082.010000
HBC	0.000000	0.000000	0.000000	0.000000	0.000000
HOUSEWARES	0.000000	0.000000	0.000000	0.000000	0.000000
KIOSK-GAS	8465.180000	50817.910000	92614.660000	96858.440000	16329.770000
MEAT	11957.340000	37162.660000	61003.950000	87407.670000	20001.910000
MEAT-PCKGD	10453.130000	30029.690000	46499.110000	59855.460000	11891.430000
MEAT-WHSE	0.000000	0.000000	1.000000	4.000000	1.000000

AGE_DESC	19-24	25-34	35-44	45-54	55-64
DEPARTMENT					
MISC SALES TRAN	2031.730000	8200.660000	9976.190000	23617.850000	7762.980000
MISC. TRANS.	73.520000	757.370000	1334.700000	859.080000	688.730000
NUTRITION	1146.400000	11067.450000	15941.860000	16366.510000	2504.010000
PASTRY	2386.730000	8161.420000	13706.810000	19534.790000	3601.860000
PHARMACY SUPPLY	0.000000	5.970000	3.980000	1.990000	0.000000
PHOTO	4.980000	5.190000	6.980000	2.490000	0.000000
PORK	0.000000	0.000000	0.000000	0.000000	0.000000
POSTAL CENTER	0.000000	0.000000	0.000000	5.960000	1.000000
PROD-WHS SALES	0.000000	0.000000	2.520000	5.000000	0.000000
PRODUCE	10170.590000	41706.460000	67779.890000	96442.730000	21326.150000
RESTAURANT	1.390000	389.040000	234.110000	540.310000	33.900000
RX	0.000000	0.000000	26.880000	26.780000	0.000000
SALAD BAR	1330.150000	2050.060000	3631.810000	4770.150000	925.770000
SEAFOOD	461.180000	2080.940000	3101.910000	5551.320000	1363.950000
SEAFOOD-PCKGD	1500.270000	4189.130000	6346.770000	10079.840000	2975.970000
SPIRITS	2983.750000	2474.460000	1491.370000	3218.340000	263.180000
TOYS	0.000000	0.000000	0.000000	1.490000	0.000000
TRAVEL & LEISUR	50.220000	173.560000	283.190000	431.480000	81.150000
VIDEO	0.000000	7.990000	13.990000	0.000000	0.000000
VIDEO RENTAL	0.000000	0.000000	0.000000	0.000000	0.000000

EXPORT

Finally, export your pivot table to an excel file. Make sure to provide a sheet name.

In [127…

```
# Call to_excel on pivot table above - note the formatting gets passed to excel too
(trans_demo_dept.pivot_table(index = "DEPARTMENT",
                             columns = "AGE_DESC",
                             values = "SALES_VALUE",
                             aggfunc = "sum")
 .style.background_gradient(cmap = "RdYlGn", axis = 1)
 .to_excel("demographic_category_sales.xlsx", sheet_name = "sales_pivot")
)
```