###Drive Mounting

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

###Importing

```
import pandas as pd
```

```
adidas = pd.read_csv('/content/drive/MyDrive/STUDY2/DATA ANALYSIS LAB/LABCYCLE/DATASETS/Adidas_USSales_Datasets_8thquestion.csv')
```

###Cleaning

```
adidas['Price per Unit'] = adidas['Price per Unit'].str.replace('$','').astype(float)
adidas['Units Sold'] = adidas['Units Sold'].str.replace(',','').astype(int)
adidas['Total Sales'] = adidas['Total Sales'].str.replace('$','')
adidas['Total Sales'] = adidas['Total Sales'].str.replace(',','').astype(int)
adidas['Operating Profit'] = adidas['Operating Profit'].str.replace('$','')
adidas['Operating Profit'] = adidas['Operating Profit'].str.replace(',','').astype(int)
adidas['Operating Margin'] = adidas['Operating Margin'].str.replace('%','').astype(int)
adidas.head(5)
```

```
<ipython-input-91-49eb9a91bc9a>:1: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  adidas['Price per Unit'] = adidas['Price per Unit'].str.replace('$','').astype(float)
<ipython-input-91-49eb9a91bc9a>:3: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  adidas['Total Sales'] = adidas['Total Sales'].str.replace('$','')
<ipython-input-91-49eb9a91bc9a>:5: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  adidas['Operating Profit'] = adidas['Operating Profit'].str.replace('$','')
```

	Retailer	Retailer ID	Invoice Date	Region	State	City	Product	Price per Unit	Units Sold	Total Sales	Operating Profit	Operating Margin	Sales Method
0	Foot Locker	1185732	1/1/2020	Northeast	New York	New York	Men's Street Footwear	50.0	1200	600000	300000	50	In-store
1	Foot Locker	1185732	1/2/2020	Northeast	New York	New York	Men's Athletic Footwear	50.0	1000	500000	150000	30	In-store
2	Foot Locker	1185732	1/3/2020	Northeast	New York	New York	Women's Street Footwear	40.0	1000	400000	140000	35	In-store
3	Foot Locker	1185732	1/4/2020	Northeast	New York	New York	Women's Athletic Footwear	45.0	850	382500	133875	35	In-store
4	Foot Locker	1185732	1/5/2020	Northeast	New York	New York	Men's Apparel	60.0	900	540000	162000	30	In-store

```
adidas.dtypes
```

```
Retailer             object
Retailer ID           int64
Invoice Date         object
Region               object
State                object
City                 object
Product              object
Price per Unit      float64
Units Sold            int64
Total Sales           int64
Operating Profit      int64
Operating Margin      int64
Sales Method         object
dtype: object
```

###a. List all the products sold in every region.

```
products_region = adidas.groupby(['Region','Product']).nunique().reset_index()
products_region = products_region[['Region','Product']]
products_region.pivot_table(index=['Region','Product'])
```

Region	Product
Midwest	Men's Apparel
	Men's Athletic Footwear
	Men's Street Footwear
	Women's Apparel
	Women's Athletic Footwear
	Women's Street Footwear
Northeast	Men's Apparel
	Men's Athletic Footwear
	Men's Street Footwear

Region	Product
	Women's Apparel
	Women's Athletic Footwear
	Women's Street Footwear
South	Men's Apparel
	Men's Athletic Footwear
	Men's Street Footwear
	Women's Apparel
	Women's Athletic Footwear
	Women's Street Footwear
Southeast	Men's Apparel
	Men's Athletic Footwear
	Men's Street Footwear
	Women's Apparel
	Women's Athletic Footwear
	Women's Street Footwear
West	Men's Apparel
	Men's Athletic Footwear
	Men's Street Footwear
	Women's Apparel
	Women's Athletic Footwear
	Women's Street Footwear

###b. Find the Cities & the retailers who sold womens related products.

]:
```
womans = adidas[adidas['Product'].str.contains('Women',case=False)]
city_retailer_woman = womans.groupby(['City','Retailer','Product']).nunique().reset_index()
city_retailer_woman = city_retailer_woman[['City','Retailer','Product']]
city_retailer_woman.pivot_table(index=['City','Retailer','Product'])
```

]:

City	Retailer	Product
Albany	Kohl's	Women's Apparel
		Women's Athletic Footwear
		Women's Street Footwear
	West Gear	Women's Apparel
		Women's Athletic Footwear
...	...	...
Wilmington	Foot Locker	Women's Athletic Footwear
		Women's Street Footwear
	Kohl's	Women's Apparel
		Women's Athletic Footwear
		Women's Street Footwear

310 rows × 0 columns

###c. Find the total sales of each womens product in in-store method.

]:
```
womans_instore = womans[womans['Sales Method']=='In-store']
womans_totalsales = womans_instore.groupby(['Product'])['Total Sales'].sum()
womans_totalsales
```

]:
```
Product
Women's Apparel              70248750
Women's Athletic Footwear    40520000
Women's Street Footwear      48349250
Name: Total Sales, dtype: int64
```

###d. For each product, find region wise total sales & units sold.

]:
```
region_totalsales = adidas.groupby(['Product','Region'])['Total Sales','Units Sold'].sum().reset_index()
region_totalsales.pivot_table(index=['Product','Region'])
```

]:
```
<ipython-input-95-fc9a6c6992c9>:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use
a list instead.
  region_totalsales = adidas.groupby(['Product','Region'])['Total Sales','Units Sold'].sum().reset_index()
```

]:

		Total Sales	Units Sold
Product	Region		
Men's Apparel	Midwest	18125661	45304
	Northeast	25744412	62031
	South	19703069	60641
	Southeast	24461487	54385
	West	35694003	84322
Men's Athletic Footwear	Midwest	21305539	65120
	Northeast	28874237	81474

		Total Sales	Units Sold
Product	Region		
	South	25710545	90079
	Southeast	27777020	71129
	West	50006339	127724
Men's Street Footwear	Midwest	38322810	109861
	Northeast	51025024	134252
	South	28444561	106545
	Southeast	36019236	91867
	West	55014613	150795
Women's Apparel	Midwest	28206383	69435
	Northeast	37543083	90048
	South	29607187	88740
	Southeast	31491161	68839
	West	52191046	116765
Women's Athletic Footwear	Midwest	13595168	44808
	Northeast	19796138	59464
	South	18420722	63998
	Southeast	20302798	55292
	West	34517070	93674
Women's Street Footwear	Midwest	16244898	56809
	Northeast	23341173	74010
	South	22777097	82257
	Southeast	23119534	65488
	West	42520111	113705

###e. For men's & women's products, find state wise units sold & total sales.

]:
```
adidas['Product Sex'] = adidas['Product'].str.split(' ').str[0].str.strip()
sex_product = adidas.groupby(['Product Sex','State'])['Units Sold','Total Sales'].sum().reset_index()
sex_product.pivot_table(index=['Product Sex','State'])
```

]:
```
<ipython-input-102-b7071b376f9e>:2: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  sex_product = adidas.groupby(['Product Sex','State'])['Units Sold','Total Sales'].sum().reset_index()
```

]:

		Total Sales	Units Sold
Product Sex	State		
Men's	Alabama	8422577	32129
	Alaska	8598187	17630
	Arizona	8195450	25053
	Arkansas	7135887	26714
	California	30642904	84877
...	...	...	...
Women's	Virginia	10198009	24468
	Washington	12529495	21915
	West Virginia	4588640	13167
	Wisconsin	3127872	9965
	Wyoming	8424072	23558

100 rows × 2 columns

###f. Find states where women's products sold were more than men's products.

]:
```
men = sex_product[sex_product['Product Sex']=="Men's"]
women = sex_product[sex_product['Product Sex']=="Women's"]

men_state = men.groupby(['State'])['Total Sales'].sum()
women_state = women.groupby(['State'])['Total Sales'].sum()

states_with_women = women_state[women_state>men_state].index.tolist()
states_with_women
```

]:
```
['Alabama', 'Idaho', 'Tennessee', 'Texas']
```

###g. Find region wise units sold for each product

]:
```
region_product = adidas.groupby(['Product','Region'])['Units Sold'].sum().reset_index()
region_product.pivot_table(index=['Product','Region'])
```

]:

		Units Sold
Product	Region	
Men's Apparel	Midwest	45304
	Northeast	62031
	South	60641

		Units Sold
Product	Region	
	Southeast	54385
	West	84322
Men's Athletic Footwear	Midwest	65120
	Northeast	81474
	South	90079
	Southeast	71129
	West	127724
Men's Street Footwear	Midwest	109861
	Northeast	134252
	South	106545
	Southeast	91867
	West	150795
Women's Apparel	Midwest	69435
	Northeast	90048
	South	88740
	Southeast	68839
	West	116765
Women's Athletic Footwear	Midwest	44808
	Northeast	59464
	South	63998
	Southeast	55292
	West	93674
Women's Street Footwear	Midwest	56809
	Northeast	74010
	South	82257
	Southeast	65488
	West	113705

###h. Find region wise profit for every retailer.

]:
```
retailer_profit = adidas.groupby(['Retailer','Region'])['Operating Profit'].sum().reset_index()
retailer_profit.pivot_table(index=['Retailer','Region'])
```

]:

		Operating Profit
Retailer	Region	
Amazon	Midwest	6833803
	Northeast	13398873
	South	146947
	Southeast	4295096
	West	4143814
Foot Locker	Midwest	18245550
	Northeast	23914694
	South	3679978
	Southeast	22531788
	West	12350224
Kohl's	Midwest	8552980
	Northeast	5172721
	South	1357043
	West	21728558
Sports Direct	Midwest	10684452
	Northeast	8395072
	South	29929316
	Southeast	20755666
	West	4568516
Walmart	Northeast	4617140
	South	13021023
	Southeast	6446692
	West	1697208
West Gear	Midwest	8494627
	Northeast	12522188
	South	13003737
	Southeast	6526220
	West	45121196

###i. Find the states along with units sold where products sold in more than one city in the state.

]:
```
cities_per_state_product = adidas.groupby(['State', 'Product'])['City'].nunique().reset_index()
states_with_multiple_cities = cities_per_state_product[cities_per_state_product['City'] > 1]

result = pd.merge(states_with_multiple_cities, adidas[['State', 'Product', 'Units Sold']], on=['State', 'Product'])
```

```
result = result.groupby(['State'])['Units Sold'].nunique().reset_index()
result
```

]:

	State	Units Sold
0	California	129
1	Florida	160
2	New York	154
3	Texas	156

###j. Draw plot to show monthly sales in 2020 in every region

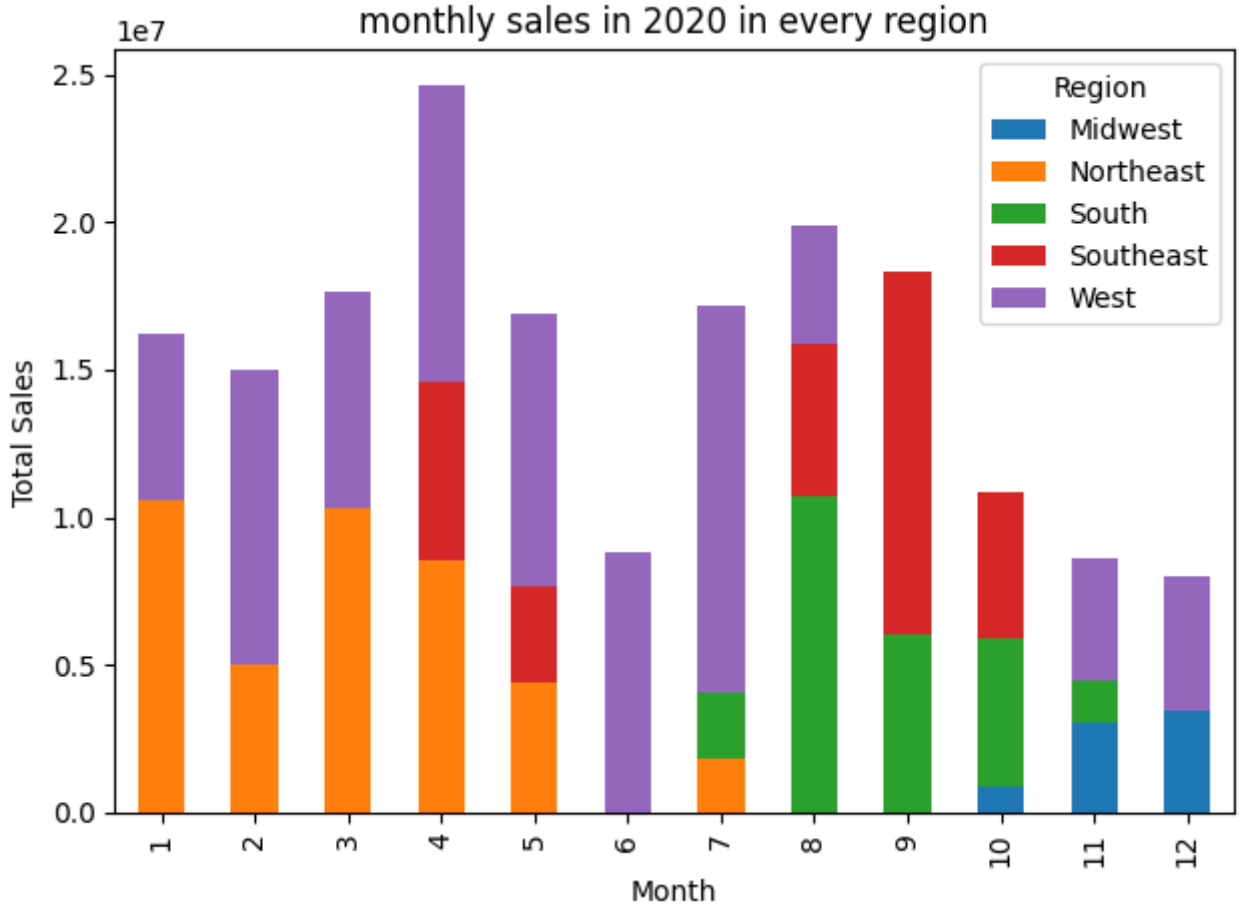
]:

```
import matplotlib.pyplot as plt
adidas['Year'] = adidas['Invoice Date'].str.split('/').str[2].str.strip().astype(int)
adidas['Month'] = adidas['Invoice Date'].str.split('/').str[0].str.strip().astype(int)
adidas_2020 = adidas[adidas['Year']==2020]

adidas_2020_monthlysales = adidas_2020.groupby(['Region','Month'])['Total Sales'].sum().reset_index().sort_values(by='Month')
pivot_table = adidas_2020_monthlysales.pivot_table(index='Month', columns='Region', values='Total Sales', aggfunc='sum')

plt.figure(figsize=(5,4))
pivot_table.plot(kind='bar', stacked=True)
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.title('monthly sales in 2020 in every region')
plt.tight_layout()
plt.show()
```

]:



###k. Draw the plot to show year wise sales in every region.

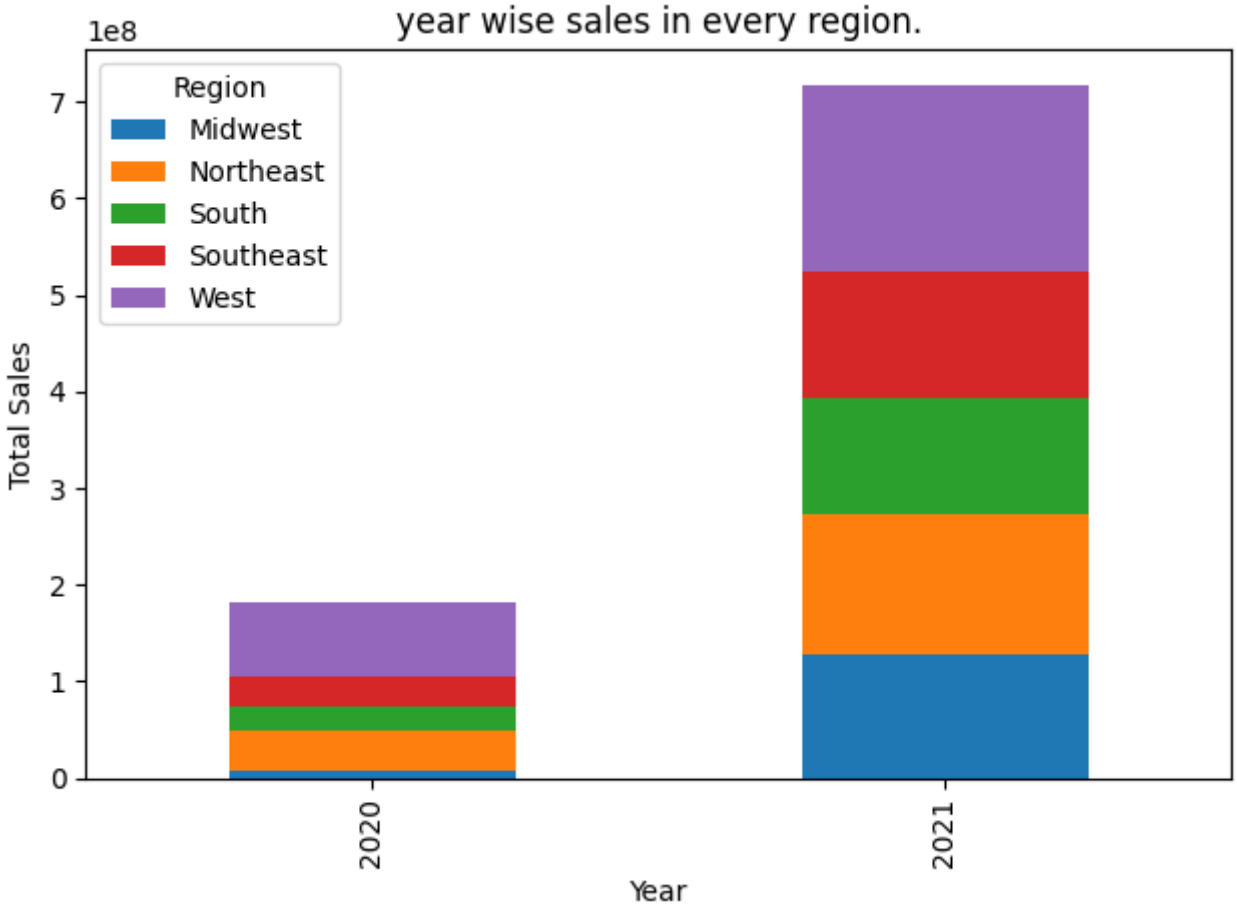
]:

```
adidas_yearlysales = adidas.groupby(['Year','Region'])['Total Sales'].sum().reset_index().sort_values(by='Year')
pivot_table = adidas_yearlysales.pivot_table(index='Year', columns='Region', values='Total Sales', aggfunc='sum')

plt.figure(figsize=(5,4))
pivot_table.plot(kind='bar', stacked=True)
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.title('year wise sales in every region.')
plt.tight_layout()
plt.show()
```

]:

]:



###I. Draw plots to show Region wise sales in every year.

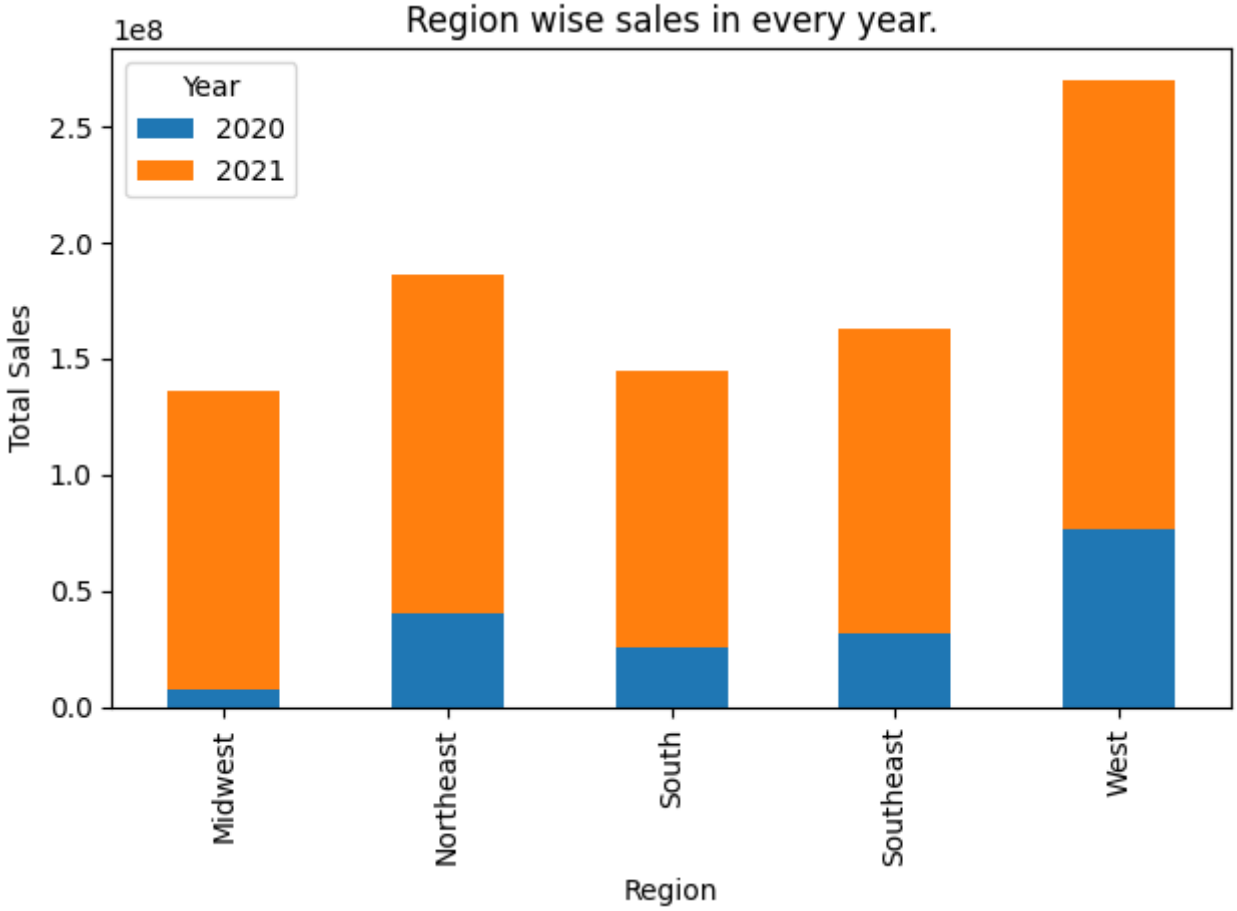
]:

```
adidas_yearlysales = adidas.groupby(['Year','Region'])['Total Sales'].sum().reset_index().sort_values(by='Year')
pivot_table = adidas_yearlysales.pivot_table(index='Region', columns='Year', values='Total Sales', aggfunc='sum')

plt.figure(figsize=(5,4))
pivot_table.plot(kind='bar', stacked=True)
plt.xlabel('Region')
plt.ylabel('Total Sales')
plt.title('Region wise sales in every year.')
plt.tight_layout()
plt.show()
```

]:

]:


Region wise sales in every year.
1e8
Year
2020
2021
Total Sales
0.0
0.5
1.0
1.5
2.0
2.5
Midwest
Northeast
South
Southeast
West
Region