ASSIGNMENT – 1

Name : Bhavan Vishva S

Roll No : 73771921114

Sales Analysis

Import necessary libraries

```
import pandas as pd
import os
```

Let's read a single csv file of the month of April and view the contents.

```
df = pd.read_csv('./dataset/Sales_April_2019.csv')
df.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

Task 1: Merge 12 months of sales data into a single CSV file

The dataset contains 12 different CSV files for 12 months of 2019. Let's now read all the files from the '/dataset' folder.

```
filenames = [file for file in os.listdir('./dataset')]
print(*filenames, sep='\n')
```

```
Sales_April_2019.csv
Sales_August_2019.csv
Sales_December_2019.csv
```

```
Sales_February_2019.csv
Sales_January_2019.csv
Sales_July_2019.csv
Sales_June_2019.csv
Sales_March_2019.csv
Sales_May_2019.csv
Sales_November_2019.csv
Sales_October_2019.csv
Sales_September_2019.csv
```

Now that we have read the filenames from the '/dataset' folder, we can concatenate into a single CSV file.

```
all_data = pd.DataFrame()
for file in filenames:
    df = pd.read_csv('./dataset/'+file)
    all_data = pd.concat([all_data, df])

all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

```
# Saving all_data to a CSV file
all_data.to_csv('all_data.csv', index=False)
```

We can read the merged and updated data directly from 'all_data.csv' so that we don't need to run the code above every time.

```
all_data = pd.read_csv('all_data.csv')
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

```
all_data.shape
```

```
(186850, 6)
```

```
all_data.describe()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
count	186305	186305	186305	186305	186305	186305
unique	178438	20	10	24	142396	140788
top	Order ID	USB-C Charging Cable	1	11.95	Order Date	Purchase Address
freq	355	21903	168552	21903	355	355

Clean up the data!

```
# Find rows with any NaN
any_nan_df = all_data[all_data.isna().any(axis=1)]
any_nan_df.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1	NaN	NaN	NaN	NaN	NaN	NaN
356	NaN	NaN	NaN	NaN	NaN	NaN
735	NaN	NaN	NaN	NaN	NaN	NaN
1433	NaN	NaN	NaN	NaN	NaN	NaN
1553	NaN	NaN	NaN	NaN	NaN	NaN

```
any_nan_df.shape
```

```
(545, 6)
```

```
print('Looks like there are '+str(any_nan_df.shape[0])+' rows with atleast
one NaN!')
```

```
Looks like there are 545 rows with atleast one NaN!
```

```
all_nan_df = all_data[all_data.isna().all(axis=1)]
all_nan_df.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1	NaN	NaN	NaN	NaN	NaN	NaN
356	NaN	NaN	NaN	NaN	NaN	NaN
735	NaN	NaN	NaN	NaN	NaN	NaN
1433	NaN	NaN	NaN	NaN	NaN	NaN
1553	NaN	NaN	NaN	NaN	NaN	NaN

```
all_nan_df.shape
```

```
(545, 6)
```

```
print('And there are '+str(all_nan_df.shape[0])+' rows with all NaN\'s!')
```

```
And there are 545 rows with all NaN's!
```

That means we can drop these rows as all of them have NaN's.

Drop rows with NaN

```
all_data = all_data.dropna(how='all')
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

```
all_data.shape
```

```
(186305, 6)
```

Convert columns to correct type

```
all_data.dtypes
```

```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
dtype: object
```

Data Cleanup Contd: Find 'Or' rows and delete them

```
temp_df = all_data[all_data['Order Date'].str[0:2] == 'Or']
temp_df.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
519	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1149	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1155	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2878	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2893	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

```
temp_df.shape
```

```
(355, 6)
```

```
print('Looks like there are '+str(temp_df.shape[0])+' rows with the header
row duplicated!')
```

```
Looks like there are 355 rows with the header row duplicated!
```

```
del temp_df
```

Let's drop those rows now.

```
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']
all_data.shape
```

```
(185950, 6)
```

```
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each'] = pd.to_numeric(all_data['Price Each'])
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Augment data with additional columns

Task 2: Create new 'Month' column from 'Order Date' column

```
# Slicing first two characters from Order Date for month
all_data['Month'] = all_data['Order Date'].str[0:2]

#Convert Month from str to int32
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

Okay, so we have successfully converted the 'Month' column from str to int.

Task 3: Add a Sales column

```
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

Task 4: Add a City column

```
# Let's use the .apply()
def get_city(address):
    return address.split(',')[1]

def get_state(address):
    return (address.split(',')[2]).split(' ')[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x:
get_city(x)+' ('+get_state(x)+')')

all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99	Los Angeles (CA)