Extracting data from the sales raw date and concatenating them into one csv file

```python
#Importing all the required libraries
import pandas as pd
import os
import glob

# Reading the monthly csv files from the folder

files = os.path.join("D:/jspm/SIP project/amazon sales raw data" ,
"*csv")
list_of_files = glob.glob(files)
print(list_of_files)
```

```
['D:/jspm/SIP project/amazon sales raw data\\Sales_April_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_August_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_December_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_February_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_January_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_July_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_June_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_March_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_May_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_November_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_October_2022.csv',
'D:/jspm/SIP project/amazon sales raw data\\Sales_September_2022.csv']
```

```python
#concatenating the data
concatenated_data= pd.concat(map(pd.read_csv,list_of_files) ,
ignore_index=True)

# Writing the concatenating data to a new csv file
Output_file= "D:/jspm/SIP project/Yearly_sales/YEARLY_SALES.csv"


concatenated_data.to_csv(Output_file,index= False)
print("Files concatenated and saved to:",Output_file)
```

```
Files concatenated and saved to: D:/jspm/SIP
project/Yearly_sales/YEARLY_SALES.csv
```

```python
df = pd.read_csv("D:/jspm/SIP project/Yearly_sales/YEARLY_SALES.csv")
df.head()
print(df.shape)
```

```
(186850, 6)
```

```
df

       Order ID                     Product Quantity Ordered Price
Each  \
0        176558        USB-C Charging Cable                2
11.95
1           NaN                         NaN              NaN
NaN
2        176559  Bose SoundSport Headphones                1
99.99
3        176560                Google Phone                1
600
4        176560            Wired Headphones                1
11.99
...         ...                         ...              ...        ..
.
186845   259353      AAA Batteries (4-pack)                3
2.99
186846   259354                      iPhone                1
700
186847   259355                      iPhone                1
700
186848   259356      34in Ultrawide Monitor                1
379.99
186849   259357        USB-C Charging Cable                1
11.95

                 Order Date                         Purchase Address
0       2022-04-19 08:46:00             917 1st St, Dallas, TX 75001
1                       NaN                                      NaN
2       2022-04-07 22:30:00        682 Chestnut St, Boston, MA 02215
3       2022-04-12 14:38:00     669 Spruce St, Los Angeles, CA 90001
4       2022-04-12 14:38:00     669 Spruce St, Los Angeles, CA 90001
...                     ...                                      ...
186845  2022-09-17 20:56:00   840 Highland St, Los Angeles, CA 90001
186846  2022-09-01 16:00:00  216 Dogwood St, San Francisco, CA 94016
186847  2022-09-23 07:39:00     220 12th St, San Francisco, CA 94016
186848  2022-09-19 17:30:00   511 Forest St, San Francisco, CA 94016
186849  2022-09-30 00:18:00   250 Meadow St, San Francisco, CA 94016

[186850 rows x 6 columns]

def DESC(dataframe):
    print(f"Shape of data : {dataframe.shape}\n{'-'*50}")
    print(f"{dataframe.info()}\n{'-'*50}")
    print(f"Count of null values in columns :\
n{dataframe.isna().sum()} \n{'-'*50}")

DESC(df)
```

```
Shape of data : (186850, 6)
--------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186850 entries, 0 to 186849
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   Order ID          186305 non-null  object
 1   Product           186305 non-null  object
 2   Quantity Ordered  186305 non-null  object
 3   Price Each        186305 non-null  object
 4   Order Date        185950 non-null  object
 5   Purchase Address  186305 non-null  object
dtypes: object(6)
memory usage: 8.6+ MB
None
--------------------------------------------------
Count of null values in columns : 
Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
Order Date          900
Purchase Address    545
dtype: int64
--------------------------------------------------

# In the dataframe there are some rows which are completely null so 
dropping them
Final_data = df.dropna()

DESC(Final_data)

Shape of data : (185950, 6)
--------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   Order ID          185950 non-null  object
 1   Product           185950 non-null  object
 2   Quantity Ordered  185950 non-null  object
 3   Price Each        185950 non-null  object
 4   Order Date        185950 non-null  object
 5   Purchase Address  185950 non-null  object
dtypes: object(6)
memory usage: 9.9+ MB
None
--------------------------------------------------
```

```
Count of null values in columns :
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
dtype: int64
--------------------------------------------------
```

```
# For Analysis we need City and State columns
Final_data[[ 'City', 'State']] = Final_data['Purchase
Address'].str.split(',', expand=True).loc[:,1:]
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\3596567923.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data[[ 'City', 'State']] = Final_data['Purchase
Address'].str.split(',', expand=True).loc[:,1:]
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\3596567923.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data[[ 'City', 'State']] = Final_data['Purchase
Address'].str.split(',', expand=True).loc[:,1:]
```

```
Final_data.head()
```

```
  Order ID                     Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99
5   176561            Wired Headphones                1      11.99

            Order Date                      Purchase Address
City  \
0  2022-04-19 08:46:00          917 1st St, Dallas, TX 75001
Dallas
2  2022-04-07 22:30:00     682 Chestnut St, Boston, MA 02215
Boston
```

```
3  2022-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001   Los
Angeles
4  2022-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001   Los
Angeles
5  2022-04-30 09:27:00     333 8th St, Los Angeles, CA 90001   Los
Angeles

       State
0   TX 75001
2   MA 02215
3   CA 90001
4   CA 90001
5   CA 90001
```

```
# As we can see State Column contains Postal Code which is not needed
for Analysis so Extracting only State name
Final_data['State'] = Final_data['State'].str.split("
",expand=True).loc[:,1]

Final_data.head()
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\1391076204.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data['State'] = Final_data['State'].str.split("
",expand=True).loc[:,1]

  Order ID                     Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99
5   176561            Wired Headphones                1      11.99

            Order Date                       Purchase Address
City  \
0  2022-04-19 08:46:00         917 1st St, Dallas, TX 75001
Dallas
2  2022-04-07 22:30:00     682 Chestnut St, Boston, MA 02215
Boston
3  2022-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001   Los
Angeles
4  2022-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001   Los
Angeles
5  2022-04-30 09:27:00     333 8th St, Los Angeles, CA 90001   Los
```

```
Angeles

  State
0    TX
2    MA
3    CA
4    CA
5    CA

Final_data["State"].unique()

array(['TX', 'MA', 'CA', 'WA', 'GA', 'NY', 'OR', 'ME'], dtype=object)

# Matching names with their Abbrevation and replacing Abbrevation in
State Column with full name of state

state_name =
{"TX":"Texas","MA":"Massachusetts","CA":"California","WA":"Washington"
,"GA":"Georgia","NY":"New York","OR":"Oregon","ME":"Maine"}

Final_data["State"] = Final_data["State"].map(state_name)

Final_data.head()

C:\Users\admin\AppData\Local\Temp\ipykernel_6072\1283795104.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data["State"] = Final_data["State"].map(state_name)

  Order ID                     Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99
5   176561            Wired Headphones                1      11.99

             Order Date                      Purchase Address
City  \
0  2022-04-19 08:46:00          917 1st St, Dallas, TX 75001
Dallas
2  2022-04-07 22:30:00     682 Chestnut St, Boston, MA 02215
Boston
3  2022-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001   Los
Angeles
4  2022-04-12 14:38:00  669 Spruce St, Los Angeles, CA 90001   Los
Angeles
```

```
5  2022-04-30 09:27:00     333 8th St, Los Angeles, CA 90001   Los
Angeles

           State
0          Texas
2  Massachusetts
3     California
4     California
5     California

DESC(Final_data)

Shape of data : (185950, 8)
--------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          185950 non-null  object
 1   Product           185950 non-null  object
 2   Quantity Ordered  185950 non-null  object
 3   Price Each        185950 non-null  object
 4   Order Date        185950 non-null  object
 5   Purchase Address  185950 non-null  object
 6   City              185950 non-null  object
 7   State             185950 non-null  object
dtypes: object(8)
memory usage: 12.8+ MB
None
--------------------------------------------------
Count of null values in columns :
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
Purchase Address    0
City                0
State               0
dtype: int64
--------------------------------------------------

# Datatype of OrderID ,Quantity Ordered and Price Each Column is
Object which is wrong so changing it to correct dtype

Final_data["Order ID"]=pd.to_numeric(Final_data["Order ID"])
Final_data["Quantity Ordered "]=pd.to_numeric(Final_data["Quantity
Ordered"])
Final_data["Price Each"]=pd.to_numeric(Final_data["Price Each"])
```

```
Final_data["Order Date"]=pd.to_datetime(Final_data["Order
Date"],infer_datetime_format=True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\2523658474.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data["Order ID"]=pd.to_numeric(Final_data["Order ID"])
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\2523658474.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data["Quantity Ordered "]=pd.to_numeric(Final_data["Quantity
Ordered"])
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\2523658474.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data["Price Each"]=pd.to_numeric(Final_data["Price Each"])
C:\Users\admin\AppData\Local\Temp\ipykernel_6072\2523658474.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final_data["Order Date"]=pd.to_datetime(Final_data["Order
Date"],infer_datetime_format=True)
```

```
# Dropping Purchase Address from Dataframe as we have extracted City
and State Columns

Final_data= Final_data.drop("Purchase Address", axis=1)

DESC(Final_data)
```

```
Shape of data : (185950, 8)
--------------------------------------------------
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Order ID           185950 non-null  int64
 1   Product            185950 non-null  object
 2   Quantity Ordered   185950 non-null  object
 3   Price Each         185950 non-null  float64
 4   Order Date         185950 non-null  datetime64[ns]
 5   City               185950 non-null  object
 6   State              185950 non-null  object
 7   Quantity Ordered   185950 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 12.8+ MB
None
--------------------------------------------------
Count of null values in columns :
Order ID            0
Product             0
Quantity Ordered    0
Price Each          0
Order Date          0
City                0
State               0
Quantity Ordered    0
dtype: int64
--------------------------------------------------
```

Now we will derive Category Column from Product Column

```
Final_data["Product"].unique()

array(['USB-C Charging Cable', 'Bose SoundSport Headphones',
       'Google Phone', 'Wired Headphones', 'Macbook Pro Laptop',
       'Lightning Charging Cable', '27in 4K Gaming Monitor',
       'AA Batteries (4-pack)', 'Apple Airpods Headphones',
       'AAA Batteries (4-pack)', 'iPhone', 'Flatscreen TV',
       '27in FHD Monitor', '20in Monitor', 'LG Dryer', 'ThinkPad
Laptop',
       'Vareebadd Phone', 'LG Washing Machine', '34in Ultrawide
Monitor'],
      dtype=object)

def product_category(dataframe, column_name, product_name_ends_with,
category_name):

index_list=dataframe[dataframe[column_name].str.endswith(product_name_
ends_with)==True].index
```

```
    dataframe.loc[index_list,"Category"]= category_name
    print(f"'Category' column created Successfully for Category
'{category_name}'")


product_category(Final_data, "Product", "Charging Cable", "Charging
Cable")
```

```
'Category' column created Successfully for Category 'Charging Cable'
```

```
product_category(Final_data, "Product", "Headphones", "Headphones")
```

```
'Category' column created Successfully for Category 'Headphones'
```

```
product_category(Final_data, "Product", "Phone", "Phone")
```

```
'Category' column created Successfully for Category 'Phone'
```

```
product_category(Final_data, "Product", "Laptop", "Laptop")
```

```
'Category' column created Successfully for Category 'Laptop'
```

```
product_category(Final_data, "Product", "Monitor", "Monitor")
```

```
'Category' column created Successfully for Category 'Monitor'
```

```
product_category(Final_data, "Product", "Batteries (4-pack)",
"Batteries")
```

```
'Category' column created Successfully for Category 'Batteries'
```

```
product_category(Final_data, "Product", "TV", "TV")
```

```
'Category' column created Successfully for Category 'TV'
```

```
product_category(Final_data, "Product", "Dryer", "Dryer")
```

```
'Category' column created Successfully for Category 'Dryer'
```

```
product_category(Final_data, "Product", "Washing Machine", "Washing
Machine")
```

```
'Category' column created Successfully for Category 'Washing Machine'
```

```
Final_data.head()
```

```
   Order ID                     Product Quantity Ordered  Price
Each  \
0    176558        USB-C Charging Cable                2      11.95

2    176559  Bose SoundSport Headphones                1      99.99

3    176560                Google Phone                1     600.00
```

```
4    176560           Wired Headphones                1       11.99

5    176561           Wired Headphones                1       11.99


           Order Date          City          State  Quantity Ordered
\
0 2022-04-19 08:46:00        Dallas          Texas                 2

2 2022-04-07 22:30:00        Boston  Massachusetts                 1

3 2022-04-12 14:38:00   Los Angeles     California                 1

4 2022-04-12 14:38:00   Los Angeles     California                 1

5 2022-04-30 09:27:00   Los Angeles     California                 1


         Category
0  Charging Cable
2      Headphones
3           Phone
4      Headphones
5      Headphones
```

```
Final_data= Final_data.drop(df.columns[2],axis=1)
```

```
DESC(Final_data)
```

```
Shape of data : (185950, 8)
--------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Order ID          185950 non-null  int64
 1   Product           185950 non-null  object
 2   Price Each        185950 non-null  float64
 3   Order Date        185950 non-null  datetime64[ns]
 4   City              185950 non-null  object
 5   State             185950 non-null  object
 6   Quantity Ordered  185950 non-null  int64
 7   Category          185950 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 16.8+ MB
None
--------------------------------------------------
Count of null values in columns :
Order ID            0
Product             0
```

```
Price Each          0
Order Date          0
City                0
State               0
Quantity Ordered    0
Category            0
dtype: int64
--------------------------------------------------
```

Now there are no null values, column datatype is correct and also derived neccessary columns. So we will create a new csv file with this dataframe

```
Final_data.to_csv("SALES_2022", index= False)

Final_data.to
```