Syrian Arab Republic
Ministry of Higher Education
Syrian Virtual University
MWS_F23

الجمهورية العربية السورية
وزارة التعليم العالي
الجامعة الافتراضية السورية

Data Analysis

Dr. Bassel Al khatib

إعداد الطلاب :

محمد عمر الشيخ علي :	**mohammed_omar_288315**
إيمان عبود :	**eman_309016**
نيرمين المزنه :	**nrmin_292622**
محمد عبده :	**Mohammed_290522**

1. Load the dataset into a Pandas Data Frame and perform any necessary data cleaning, such as handling missing values and converting data types.

الخطوة الأولى:

نقوم باستيراد مكتبة Pandas

نقوم بقراءة ملف CSV

نستعرض أول خمسة أعمدة من Data frame

```
In [1]: #import library
        import pandas as pd
        # load the dataset into Data Frame
        df=pd.read_csv("D:\sales_data_sample.csv", encoding = "cp1252")
        # show the first five rows from Data frame
        df.head()
```

Out[1]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	ADDRESSLINE1	ADDF
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	2	2003	...	897 Long Airport Avenue	
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	5	2003	...	59 rue de l'Abbaye	
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	...	27 rue du Colonel Pierre Avia	
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	8	2003	...	78934 Hillside Dr.	
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	10	2003	...	7734 Strong St.	

5 rows × 25 columns

الخطوة الثانية:

طباعة شكل البيانات لمعرفة عدد الأسطر والأعمدة لإطار البيانات

```
In [2]: # Print data format
        df.shape
```

Out[2]: (2823, 25)

ثم نعرض معلومات عن إطار البيانات التي لدينا ومن خلالها نستطيع أن نعرف عدد الأعمدة الكي وعدد الاسطر ونوع بيانات لكل عمود ونستطيع معرفة وجود القيم الغير فارغة في كل عامود وبالتالي معرفة عدد الخلايا الفارغة داخل كل عمود

```
In [3]: #show the basic information about the Data Frame
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

نقوم بتحويل حالة أحرف الاعمدة الى أحرف صغيرة من أجل سهولة التعامل معها لاحقاً

```
In [4]: # Make column Letters Lowercase
        df.rename(columns= lambda x:x.lower() , inplace = True)
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ordernumber       2823 non-null   int64
 1   quantityordered   2823 non-null   int64
 2   priceeach         2823 non-null   float64
 3   orderlinenumber   2823 non-null   int64
 4   sales             2823 non-null   float64
 5   orderdate         2823 non-null   object
 6   status            2823 non-null   object
 7   qtr_id            2823 non-null   int64
 8   month_id          2823 non-null   int64
 9   year_id           2823 non-null   int64
 10  productline       2823 non-null   object
 11  msrp              2823 non-null   int64
 12  productcode       2823 non-null   object
 13  customername      2823 non-null   object
 14  phone             2823 non-null   object
 15  addressline1      2823 non-null   object
 16  addressline2      302 non-null    object
 17  city              2823 non-null   object
 18  state             1337 non-null   object
 19  postalcode        2747 non-null   object
 20  country           2823 non-null   object
 21  territory         1749 non-null   object
 22  contactlastname   2823 non-null   object
 23  contactfirstname  2823 non-null   object
 24  dealsize          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

نقوم بإعادة تسمية بعض الاعمدة من أجل التعامل مع أسماء أكثر وضوحاً سهولة التعامل معها وجعلها تناسب مع أسماء الاعمدة المذكورة في نص الوظيفة ثم قمنا بعمل إطار بيانات جديد ووضعنا بداخله فقط الأعمدة التي قد نحتاجها (اضفنا العمود status ضمن الاعمدة لأنه لاحظنا انه يمثل حالة الطلب ويمكن أن يحتوي معلومات مفيدة) من أجل تنفيذ العمليات عليها واستخراج المعلومات منها وتعرض معلومات عن إطار البيانات الجديد ونلاحظ:

- أنه لا يوجد لدينا بيانات فارغة ضمن الاعمدة التي سنعمل عليها
- وأن جميع أنواع بيانات الأعمدة مناسبة تماما للأعمدة
- أصبح لدينا 8 أعمدة و 2823 سطر التي سنعمل عليها من أجل استخراج المعلومات وتحليل البيانات

```
In [5]: # rename some of columns by put it in Variable
        newname={"ordernumber" :'order id',"productline":'product' , 'quantityordered' :'quantity ordered',
                            'priceeach':'price each','country':'Purchase Address'}
        df.rename(columns=newname, inplace=True)

        # Use some Columns and put it in new DataFrame
        df2=df[['order id','product','quantity ordered', 'price each' ,"status","city","month_id","year_id"]]

        df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   order id          2823 non-null   int64
 1   product           2823 non-null   object
 2   quantity ordered  2823 non-null   int64
 3   price each        2823 non-null   float64
 4   status            2823 non-null   object
 5   city              2823 non-null   object
 6   month_id          2823 non-null   int64
 7   year_id           2823 non-null   int64
dtypes: float64(1), int64(4), object(3)
memory usage: 176.6+ KB
```

نقوم بعرض البيانات الإحصائية للأعمدة الرقمية لمعرفة أذا كانت هناك قيم متطرفة أو غير منطقية في إطار البيانات لدينا:

```
In [6]: # Statistics information about numeric columns
        df2.describe()
```

Out[6]:

	order id	quantity ordered	price each	month_id	year_id
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.00000
mean	10258.725115	35.092809	83.658544	7.092455	2003.81509
std	92.085478	9.741443	20.174277	3.656633	0.69967
min	10100.000000	6.000000	26.880000	1.000000	2003.00000
25%	10180.000000	27.000000	68.860000	4.000000	2003.00000
50%	10262.000000	35.000000	95.700000	8.000000	2004.00000
75%	10333.500000	43.000000	100.000000	11.000000	2004.00000
max	10425.000000	97.000000	100.000000	12.000000	2005.00000

من خلال النتيجة نلاحظ:

اسم العمود	
order id	يبدأ ترقيمه من 10100 ويمتد الى 10425
quantity ordered	نلاحظ أن أقل عدد طلب لمنتج من المنتجات هو 6 وأكبر عدد هو 97
price each	سعر أقل منتج هو 26.88 وأن اعلا سعر منتج يبلغ 100
month_id	يدل على الأشهر التي تمت فيها عملية البيع
year_id	يدل على سنوات التي تمت فيها عمليات البيع ومنه نستنتج أن عمليات البيع و البيانات التي لدينا كانت خلال ثلاث السنوات التالية : 2005 , 2004 , 2003

نعرض البيانات الإحصائية للأعمدة النصية (الفئوية):

```
In [7]: df2[["product","status","city"]].describe()
```

Out[7]:

	product	status	city
count	2823	2823	2823
unique	7	6	73
top	Classic Cars	Shipped	Madrid
freq	967	2617	304

اسم العمود	
Product	يمثل اسم المنتج ونلاحظ انه لدينا سبعة أنواع من المنتجات وأكثر المنتج مبيعاً هو Classic Cars و تكررت 967 مرة
status	تمثل حالة الطلب لكل منتج وتحتوي على ستة أنواع مختلفة من القيم و القيمة الأكثر تكراراً هي Shipped و تكررت 2617 مرة أي احتمال ان يكون هناك دلالات أخرى فيها
city	تمثل المدن التي شحنت اليها الطلبات و عددها ٧٣ مدينة و إن أكثر مدينة تمتلك أكثر طلبات Madrid وذكرت ٣٠٤ مرة

نقوم بعرض القيم الستة الموجودة داخل عامود status لمعرفة ان كانت المعلومات بداخلها قد تفيد أو لا :

```
In [8]: # show values of STATUS
        Status_types = df.groupby(["status"])
        for key in Status_types.groups:
            print (key ,":",len(Status_types.groups[key]))

        Cancelled : 60
        Disputed : 14
        In Process : 41
        On Hold : 44
        Resolved : 47
        Shipped : 2617
```

قيم العمود status	
Cancelled	يفترض أنها تمثل الطلبات التي تم الغائها وعددها ٦٠ طلب
Disputed	وهي تمثل المبيعات المتنازع عليها لعدة أسباب منها أخطاء في الحسابات و غيرها وعددها ١٤ طلب
In Process	قيد المعالجة أي لم يتم تحديد أنها تمت معالجتها او لا وعددها ٤١ طلب
On Hold	قيد الانتظار وهي لم يتم تحديد انها تمت العملية الشحن أم لا وعددها ٤٤ طلب
Resolved	مبيعات تم حل مشكلتها وعددها ٤٧ طلب
Shipped	المبيعات التي تم شحنها وعددها ٢٦١٧ طلب

من خلالها نجد أنه يجب أن نستبعد الاسطر التي تحتوي قيم مشبوهة والتي لا يجب اخذها بعين الاعتبار مثل الطلبات التي تم إلغائها والمتنازع عليها وقيد المعالجة وقيد الانتظار للوصول الى نتائج أدق نستبعد هذه البيانات من إطار البيانات لدينا بحذفها:

```
In [9]: # delete rows that's STATUS is not Shipped or Resolved
        df2 = df2.drop(df2[df2["status"] == "Cancelled"].index)
        df2 = df2.drop(df2[df2["status"] == "Disputed"].index)
        df2 = df2.drop(df2[df2["status"] == "In Process"].index)
        df2 = df2.drop(df2[df2["status"] == "On Hold"].index)
        # show the New Rows in df2
        df2.info()

        <class 'pandas.core.frame.DataFrame'>
        Index: 2664 entries, 0 to 2821
        Data columns (total 8 columns):
         #   Column            Non-Null Count  Dtype
        ---  ------            --------------  -----
         0   order id          2664 non-null   int64
         1   product           2664 non-null   object
         2   quantity ordered  2664 non-null   int64
         3   price each        2664 non-null   float64
         4   status            2664 non-null   object
         5   city              2664 non-null   object
         6   month_id          2664 non-null   int64
         7   year_id           2664 non-null   int64
        dtypes: float64(1), int64(4), object(3)
        memory usage: 187.3+ KB
```

أصبح لدينا الآن: ٢٦٦٤ سطر في كل عامود

نقوم بفحص الاسطر لمعرفة إذا كان لدينا أسطر مكررة في إطار البيانات:

```
In [9]: #To show duplicate rows
        sum(df2.duplicated())

Out[9]: 53
```

حيث تعليمة () duplicated تعيد true / false لكل سطر من الاسطر ونقوم بجمعها لمعرفة اذا كان لدينا اسطر نتيجتها true و التي تمثل العدد 1 false تمثل العدد 0 و بما أنه أعاد الرقم 53 بعد تطبيق عمليه الجمع أي لدينا قيم مكررة ٥٣
نقوم بحذف هذه القيم المكررة : وأصبح إطار البيانات لدينا مكون من 2611 سطر

```
[19]: # drop the dublicated values
      df2.drop_duplicates(inplace = True )
      df2.info()

      <class 'pandas.core.frame.DataFrame'>
      Index: 2611 entries, 0 to 2821
      Data columns (total 8 columns):
       #   Column            Non-Null Count  Dtype
      ---  ------            --------------  -----
       0   order id          2611 non-null   int64
       1   product           2611 non-null   object
       2   quantity ordered  2611 non-null   int64
       3   price each        2611 non-null   float64
       4   status            2611 non-null   object
       5   city              2611 non-null   object
       6   month_id          2611 non-null   int64
       7   year_id           2611 non-null   int64
      dtypes: float64(1), int64(4), object(3)
      memory usage: 183.6+ KB
```

نقوم بعرض البيانات الإحصائية لجميع الأعمدة الرقمية و النصية ل إطار البيانات بعد حذف الاسطر منه:

```
In [22]: # Statistics information about numeric columns after deleting some rows
         df2.describe(include = "all")

Out[22]:
```

	order id	product	quantity ordered	price each	status	city	month_id	year_id
count	2611.000000	2611	2611.000000	2611.000000	2611	2611	2611.000000	2611.000000
unique	NaN	7	NaN	NaN	2	73	NaN	NaN
top	NaN	Classic Cars	NaN	NaN	Shipped	Madrid	NaN	NaN
freq	NaN	891	NaN	NaN	2564	266	NaN	NaN
mean	10253.960552	NaN	34.906549	83.312206	NaN	NaN	7.178859	2003.776714
std	89.717498	NaN	9.353246	20.306634	NaN	NaN	3.713673	0.681115
min	10100.000000	NaN	6.000000	27.220000	NaN	NaN	1.000000	2003.000000
25%	10177.000000	NaN	27.000000	68.095000	NaN	NaN	4.000000	2003.000000
50%	10259.000000	NaN	34.000000	94.710000	NaN	NaN	8.000000	2004.000000
75%	10329.000000	NaN	43.000000	100.000000	NaN	NaN	11.000000	2004.000000
max	10419.000000	NaN	97.000000	100.000000	NaN	NaN	12.000000	2005.000000

نلاحظ تغير القيم في البيانات الإحصائية للأعمدة بعد حذف الاسطر من أهمها:

اسم العمود	
price each	سعر أقل منتج هو 27.22 وأن اعلا سعر منتج يبلغ 100
quantity ordered	نلاحظ أن أقل عدد طلب لمنتج من المنتجات هو 6 وأكبر عدد هو 97
Product	المنتج Classic Cars اصبح تكراره 891
status	أصبحت فقط تمثل قيمتين shipped , resolved
city	إن أكثر مدينة تكررت Madrid وذكرت 266 مرة

2. Compute the following metrics for each product:

- Total revenue generated for each product :

نقوم بإنشاء عمود جديد داخل إطار البيانات total revenue الذي يمثل سعر الكمية المطلوبة في سطر هو ناتج ضرب سعر السلعة ب الكمية المطلوبة ثم طبقنا عملية الجمع على العمود وبالتالي يظهر لنا مجموع الإيرادات الناتجة لكل منتج ثم قمنا بحساب المجموع الإجمالي لكل المنتجات معاً

Total revenue generated for each product :

```
#Create a new column in df2 to calculate the price of quantity ordered
df2["total revenue"] = df2['price each'].multiply(df2["quantity ordered"])
#Grouping the values by product and sum the values :
product_revenue_total= df2.groupby(["product"])["total revenue"].sum()
#show the result
print ("Total revenue generated for each product  :\n" ,product_revenue_total)
#The Total revenue generated for all product :
print( "The Total revenue generated for all product: ",round(product_revenue_total.sum(),2))
```

```
Total revenue generated for each product  :
 product
Classic Cars        2706615.14
Motorcycles          932883.81
Planes               790305.12
Ships                599668.69
Trains               194804.26
Trucks and Buses     852917.45
Vintage Cars        1515613.53
Name: total revenue, dtype: float64
The Total revenue generated for all product:  7592808.0
```

- Total units sold for each product:

من أجل حســـاب العدد الكلي للوحدات المباعة لكل منتج من المنتجات من خلال عمل تجميع للبيانات بالنســـبة للعمود الذي يحتوي أنواع المنتجات product و تطبيق دالة ( ) sum الجمع على عامود عدد المنتجات quantity ordered الناتج عن عملية التجميع من خلال الدالة ( ) agg

Total units sold for each product

```
]:  1 #Total units sold for each product
    2 total_units_product= df2.groupby(["product"])["quantity ordered"].agg("sum")
    3 print ("Tota units sold for each product : \n ", total_units_product )
```

```
Tota units sold for each product :
  product
Classic Cars        31140
Motorcycles         11267
Planes               9663
Ships                7196
Trains               2622
Trucks and Buses     9781
Vintage Cars        19472
Name: quantity ordered, dtype: int64
```

- Average price per unit :

من أجل حساب متوسط سعر كل منتج نقوم بعملية التجميع على عامود المنتج product و تطبيق دالة المتوسط الحسابي mean على عامود price each الناتج عن عملية التجميع من خلال دالة ()agg

```
In [38]: #Average price per unit
         Ava_price = df2.groupby(["product"])['price each'].agg("mean")
         print (Ava_price)
```

```
product
Classic Cars        86.871605
Motorcycles         82.642888
Planes              81.683022
Ships               83.702560
Trains              75.005467
Trucks and Buses    87.069018
Vintage Cars        77.994423
Name: price each, dtype: float64
```

- Total number of orders for each product:

قمنا بعملية التجميع عامود المنتج product و طبقنا دالة ( ) count على عمود order id حيث يكون الناتج تجميع البيانات على حسب المنتج و عدد مرات طلب المنتج في الاسطر من خلال عمود order id

Total number of orders for each units :

```
6]: 1 #Total number of orders for each units
    2 total_orders = df2.groupby(["product"])["order id"].count()
    3 print ("Total number of order is " , total_orders)
```

```
Total number of order is  product
Classic Cars        891
Motorcycles         322
Planes              278
Ships               207
Trains               75
Trucks and Buses    275
Vintage Cars        563
Name: order id, dtype: int64
```

- Rank the products based on these metrics and provide recommendations on which products the company should focus on selling more.

نقوم بحساب جميع المقاييس السابقة وتطبيقها من خلال عملية التجميع الاعمدة على عامود product حيث سيقوم بتجميع البيانات على حسب أنواع المنتجات فيه و نقوم بحساب :

١- مجموع المبيعات الكلي لكل منتج total_revenue
٢- مجموع عدد الوحدات المباعة لصالح كل منتج sold_units
٣- حساب المتوسط الحسابي لسعر القطعة average_price
٤- حساب مجموع الطلبات لصالح كل منتج total_orders

ثم قمنا بترتيب النتيجة من خلال دالة ( ) sort_values على عامود total_revenue بحيث تعرض لنا ترتيب المنتجات حسب مجموع مبيعات المنتج من الأكثر مبيعاً الى الأدنى

Rank the products based on these metrics

```
38]: 1 #Rank the products based on Previous metrics :
     2 rank = df2.groupby(["product"]).agg(total_revenue=("total revenue","sum"),sold_units= ("quantity ordered","sum")
     3                                     ,average_price = ('price each',"mean"),total_orders = ("order id","count"))
     4 #sort the result by column total revenue
     5 rank_sort = rank.sort_values(by = ["total_revenue"], ascending = False)
     6 print (rank_sort)
```

product	total_revenue	sold_units	average_price	total_orders
Classic Cars	2706615.14	31140	86.871605	891
Vintage Cars	1515613.53	19472	77.994423	563
Motorcycles	932883.81	11267	82.642888	322
Trucks and Buses	852917.45	9781	87.069018	275
Planes	790305.12	9663	81.683022	278
Ships	599668.69	7196	83.702560	207
Trains	194804.26	2622	75.005467	75

من خلال هذه المعلومات أنه يجب التركيز على المنتجات التالية بالترتيب لتحقيق إيرادات أعلا مع المحافظة على وضـــع المنتج Classic cars كما هو لأن المنتجات التالية القطع المباعة منها وعدد الطلبات عليها أقل بكثير من مجموع القطع المباعة وعدد الطلبات من المنتج Classic cars ومتوســـط الأسعار لا يعد فرقاً كبيراً بين المنتجات

١- Trains
٢- Ships
٣- Planes
٤- Trucks and Buses
٥- Motorcycles
٦- Vintage cars

3.Compute the following metrics for each month:

- Total revenue generated :

ســـابقاً قمنا بإدخال عمود month_id في إطار البيانات الخاص بتحليل البيانات الذي يمثل الأشـــهر ١٢وقمنا بإدخال year_id التي تمثل سنة المبيع وتتكون من ١٢ شـهراً ولاحظنا سـابقاً انه لدينا بيانات تتمثل في ثلاث سنوات بدء من ٢٠٠٣ ل ٢٠٠٥ سوف نستخدم هذين العمودين في الحصول على تقارير شهرية لكل سنة

قمنا بعملية تجميع البيانات على أســـاس الأشـــهر ثم طبقنا دالة sum() من خلال agg على العمود " total revenue" الناتج من عملية تجميع الاعمدة يكون الناتج هو الأشهر ومجموع الإيرادات في كل شهر

1- Total revenue generated for each month :

```
18]:  1 #Total revenue generated for each month :
      2 total_revenue_monthly=df2.groupby(["month_id"])["total revenue"].agg("sum")
      3 print(total_revenue_monthly)
```

```
month_id
1      630682.29
2      655728.15
3      608686.73
4      435877.39
5      513928.10
6      289549.08
7      412473.34
8      533432.91
9      457200.12
10     858781.22
11    1693895.53
12     502573.14
Name: total revenue, dtype: float64
```

- Total units sold for each month :

من أجل حساب المجموع الكلي للوحدات المباعة في كل شهر total_sold_monthly

قمنا بعملية تجميع البيانات على العمود month_id ليتم تجميع البيانات على أساس الشهر و طبقنا عملية الجمع ( ) sum على العمود quantity ordered الناتج عن عملية التجميع ليعرض لنا مجموع الكلي للطلبات المباعة في كل شهر

2- Total units sold fo each month

```
]:  1 # Total units sold for each month :
    2 total_sold_monthly = df2.groupby (["month_id"])[["quantity ordered"]].sum()
    3 print (total_sold_monthly)
```

```
          quantity ordered
month_id
1                     7708
2                     7777
3                     7410
4                     5247
5                     6119
6                     3499
7                     4814
8                     6351
9                     5504
10                   10289
11                   20336
12                    6087
```

- Average price per unit for each month:

نحسب المتوسط الحسابي لسعر المنتجات في كل شهر avg_price_units_monthly من خلال تطبيق عملية التجميع على العمودين product , minth_id و تطبيق المتوسط الحسابي mean من خلال دالة ( ) agg

ثم نقوم بحساب المتوسط الحسابي لكل منتج من خلال تطبيق دالة الفلترة filter ونمرر لها اسم المنتج الذي نريده ليقوم بعرض المتوسط الحسابي لسعر المنتج خلال كل شهر وسوف نعيد هذه الخطوة من أجل كل منتج

من أجل المنتج : Classic Cars :

Average price for Classic Cars monthly

```
0]:  1 #Average price per unit.
     2 avg_price_units_monthly=df2.groupby(["month_id","product"])['price each'].agg("mean")
     3 #Avaerage price for Classic Cars monthly:
     4 avg_price_classic_cars_monthly=avg_price_units_monthly.filter(like ="Classic Cars")
     5 print (avg_price_classic_cars_monthly)
```

```
month_id  product
1         Classic Cars    88.294730
2         Classic Cars    86.014203
3         Classic Cars    85.041452
4         Classic Cars    87.414762
5         Classic Cars    88.816857
6         Classic Cars    86.322105
7         Classic Cars    89.147500
8         Classic Cars    87.660000
9         Classic Cars    87.295862
10        Classic Cars    88.743925
11        Classic Cars    85.187897
12        Classic Cars    84.765000
Name: price each, dtype: float64
```

من أجل المنتج : Vintage Cars: فقط نقوم بتغير الاسم الذي نريد عمل الفلترة عليه ونجعله Vintage cars

Average price for Vintage Cars monthly

```
[21]: 1 #Average price for Vintage Cars monthly:
      2 avg_price_vintage_cars_monthly=avg_price_units_monthly.filter(like ="Vintage Cars")
      3 print (avg_price_vintage_cars_monthly)
```

```
month_id  product
1         Vintage Cars    76.461500
2         Vintage Cars    79.098158
3         Vintage Cars    79.868421
4         Vintage Cars    75.566129
5         Vintage Cars    71.831250
6         Vintage Cars    77.988000
7         Vintage Cars    81.103636
8         Vintage Cars    80.086875
9         Vintage Cars    77.327143
10        Vintage Cars    76.296912
11        Vintage Cars    79.000000
12        Vintage Cars    79.120256
Name: price each, dtype: float64
```

من أجل المنتج : Motorcycles: فقط نقوم بتغير الاسم الذي نريد عمل الفلترة عليه ونجعله Motorcycles

Average price for Motorcycles monthly:

```
]: 1 #Average price for Motorcycles monthly:
   2 avg_price_motorcycles_monthly=avg_price_units_monthly.filter(like ="Motorcycles")
   3 print (avg_price_motorcycles_monthly)
```

```
month_id  product
1         Motorcycles    81.185217
2         Motorcycles    87.313824
3         Motorcycles    77.538333
4         Motorcycles    83.461667
5         Motorcycles    81.360455
6         Motorcycles    81.499286
7         Motorcycles    83.654737
8         Motorcycles    79.733871
9         Motorcycles    80.191429
10        Motorcycles    82.457813
11        Motorcycles    84.514571
12        Motorcycles    80.405333
Name: price each, dtype: float64
```

من أجل المنتج : Trucks and Buses: فقط نقوم بتغير الاسم الذي نريد عمل الفلترة عليه ونجعله Trucks and Buses

Average price for Trucks and Buses monthly:

```
3]:  1 #Average price for Trucks and Buses monthly:
     2 avg_price_trucks_buses_monthly=avg_price_units_monthly.filter(like ="Trucks and Buses")
     3 print (avg_price_trucks_buses_monthly)
```

```
month_id  product
1         Trucks and Buses    85.930000
2         Trucks and Buses    87.250000
3         Trucks and Buses    83.599444
4         Trucks and Buses    90.550000
5         Trucks and Buses    88.968621
6         Trucks and Buses    85.178571
7         Trucks and Buses    87.840000
8         Trucks and Buses    87.042857
9         Trucks and Buses    86.900526
10        Trucks and Buses    87.576667
11        Trucks and Buses    87.695455
12        Trucks and Buses    85.705200
Name: price each, dtype: float64
```

من أجل المنتج : Planes: فقط نقوم بتغير الاسم الذي نريد عمل الفلترة عليه ونجعله Planes

Average price for Planes monthly:

```
4]:  1 #Average price for Planes monthly:
     2 avg_price_planes_monthly =avg_price_units_monthly.filter(like ="Planes")
     3 print (avg_price_planes_monthly)
```

```
month_id  product
1         Planes     76.437647
2         Planes     82.909677
3         Planes     77.664583
4         Planes     79.716957
5         Planes     83.690455
6         Planes     82.542143
7         Planes     82.427500
8         Planes     80.329565
9         Planes     84.229167
10        Planes     85.233571
11        Planes     81.956471
12        Planes     82.392381
Name: price each, dtype: float64
```

من أجل المنتج : Ships: فقط نقوم بتغير الاسم الذي نريد عمل الفلترة عليه ونجعله Ships

Average price for Ships monthly:

```
[5]:
#Average price for Ships monthly:
avg_price_ships_monthly=avg_price_units_monthly.filter(like ="Ships")
print (avg_price_ships_monthly)
```

```
month_id  product
1         Ships      89.193889
2         Ships      84.750000
3         Ships      85.647273
4         Ships      81.405455
5         Ships      95.950000
6         Ships      81.145333
7         Ships      83.121111
8         Ships      82.771111
9         Ships      84.169333
10        Ships      82.008947
11        Ships      83.325625
12        Ships      79.005000
Name: price each, dtype: float64
```

من أجل المنتج : Trains: فقط نقوم بتغير الاسم الذي نريد عمل الفلترة عليه ونجعله Trains

Avaerage price for Trains monthly:

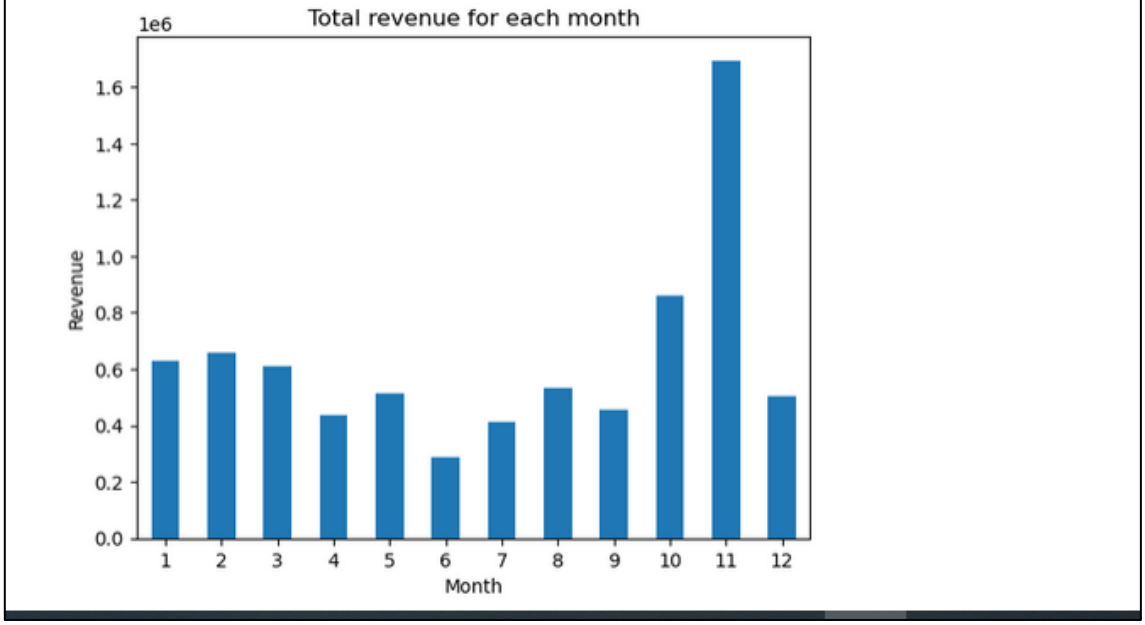
```
[26]:
#Avaerage price for Trains monthly:
avg_price_trains_monthly =avg_price_units_monthly.filter(like ="Trains")
print (avg_price_trains_monthly)
```

```
month_id  product
1         Trains     75.375714
2         Trains     71.165714
3         Trains     76.478333
4         Trains     69.463333
5         Trains     56.780000
6         Trains     79.986667
7         Trains     73.816667
8         Trains     74.766667
9         Trains     69.983333
10        Trains     72.294444
11        Trains     80.677333
12        Trains     85.753333
Name: price each, dtype: float64
```

- Plot these metrics over time and identify any trends or patterns that can help the company improve its sales :

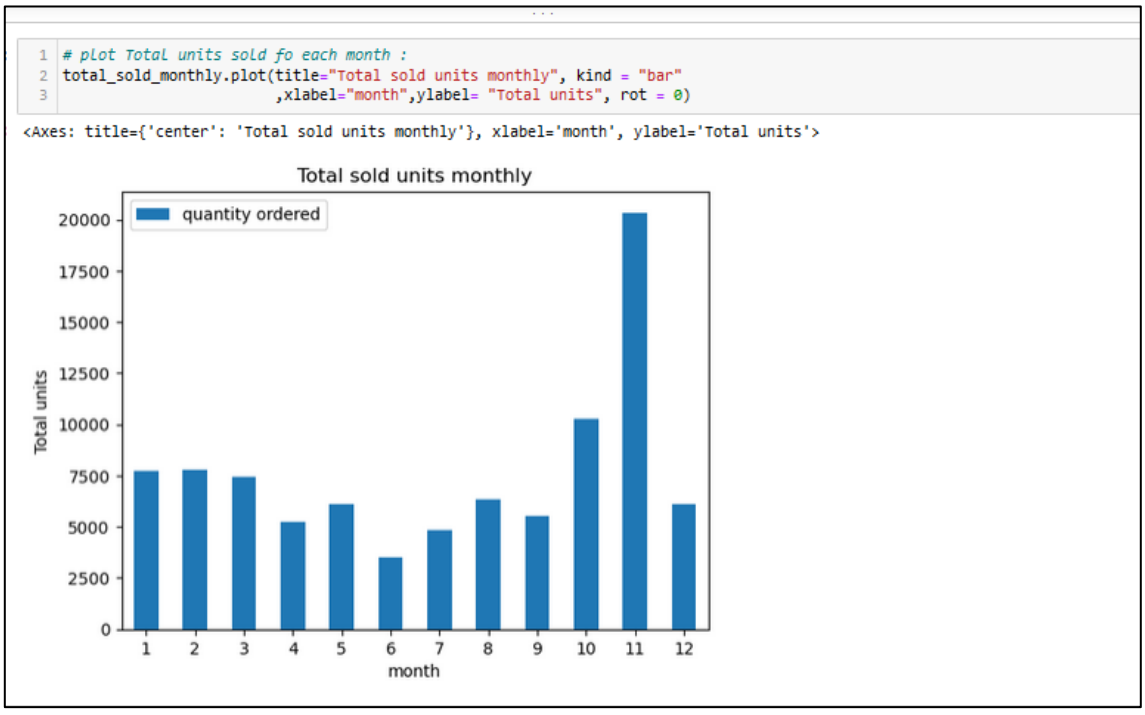
Plot for Total revenue for each month :

```
0]: # plot Total revenue generated for each month
    total_revenue_monthly.plot(title= "Total revenue for each month" ,xlabel="Month",
                               ylabel="Revenue", kind = "bar",rot = 0)
0]: <Axes: title={'center': 'Total revenue for each month'}, xlabel='Month', ylabel='Revenue'>
```



Plot for Total units sold for each month:

```
# plot Total units sold fo each month :
total_sold_monthly.plot(title="Total sold units monthly", kind = "bar"
                        ,xlabel="month",ylabel= "Total units", rot = 0)
<Axes: title={'center': 'Total sold units monthly'}, xlabel='month', ylabel='Total units'>
```

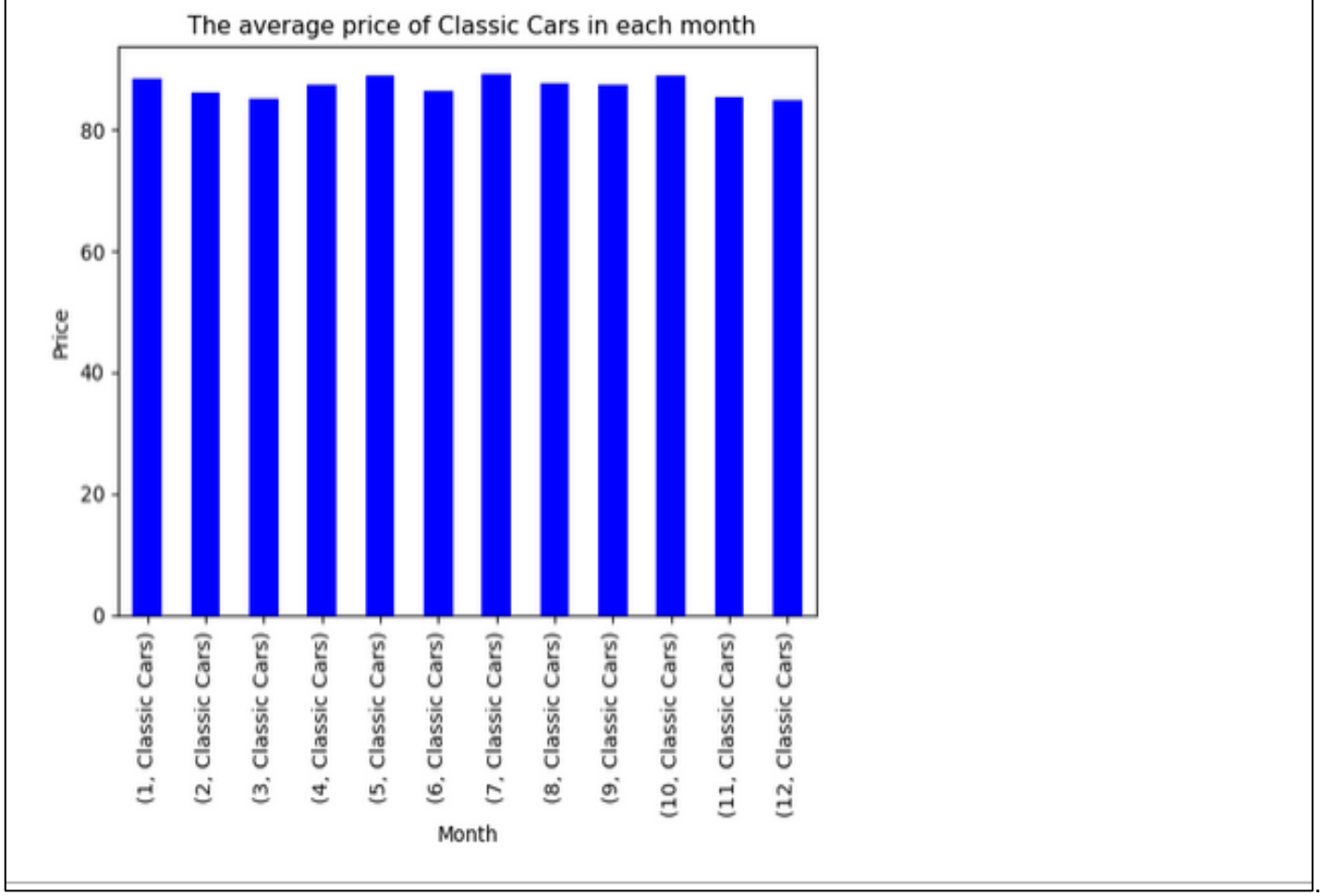


Average price per units for each month :

- Average price Classic Cars for each month

```
]: 1 #Average price Classic Cars for each month
   2 avg_price_classic_cars_monthly.plot(kind= "bar" ,title = " The average price of Classic Cars in each month",
   3                                     xlabel="Month",ylabel="Price",rot=90 ,color="b" )
   4
```

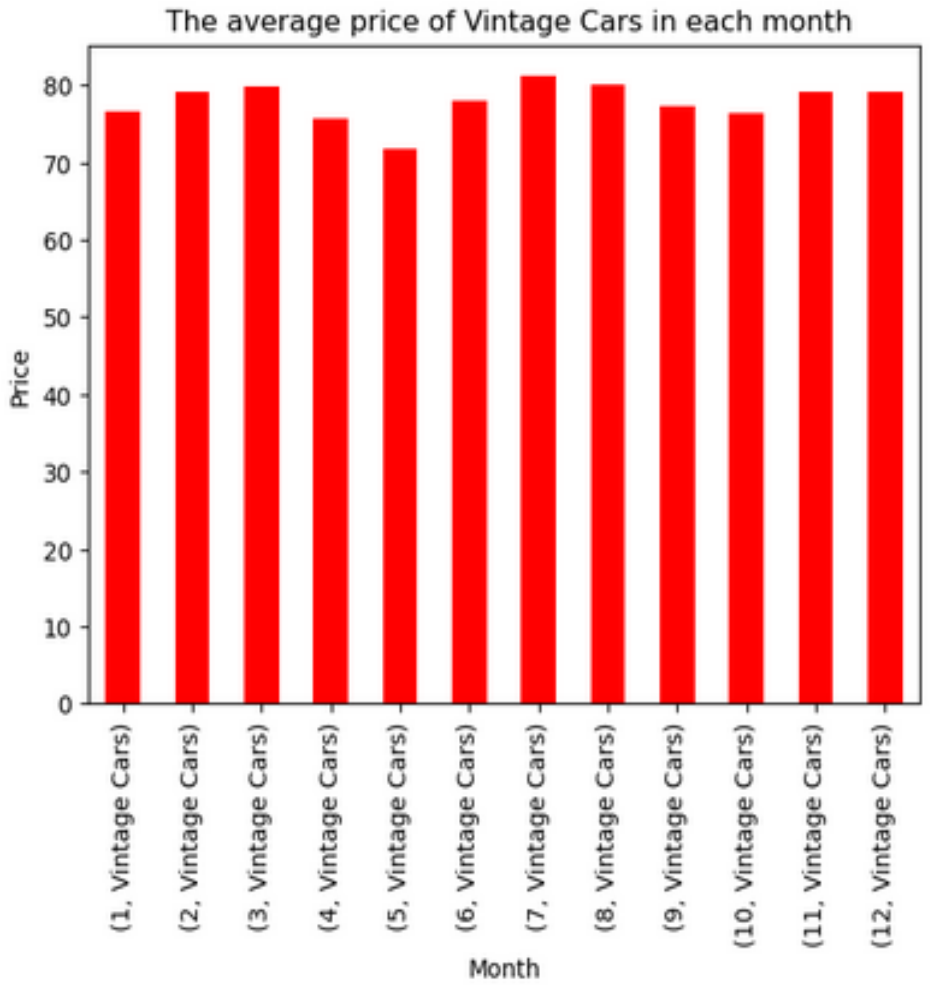
```
]: <Axes: title={'center': ' The average price of Classic Cars in each month'}, xlabel='Month', ylabel='Price'>
```



- Average price Vintage Cars for each month

```
]:  1 #Average price Vintage Cars for each month
    2 avg_price_vintage_cars_monthly.plot(kind= "bar" ,title = " The average price of Vintage Cars in each month",
    3                                     xlabel="Month",ylabel="Price",rot=90 ,color="r" )
```

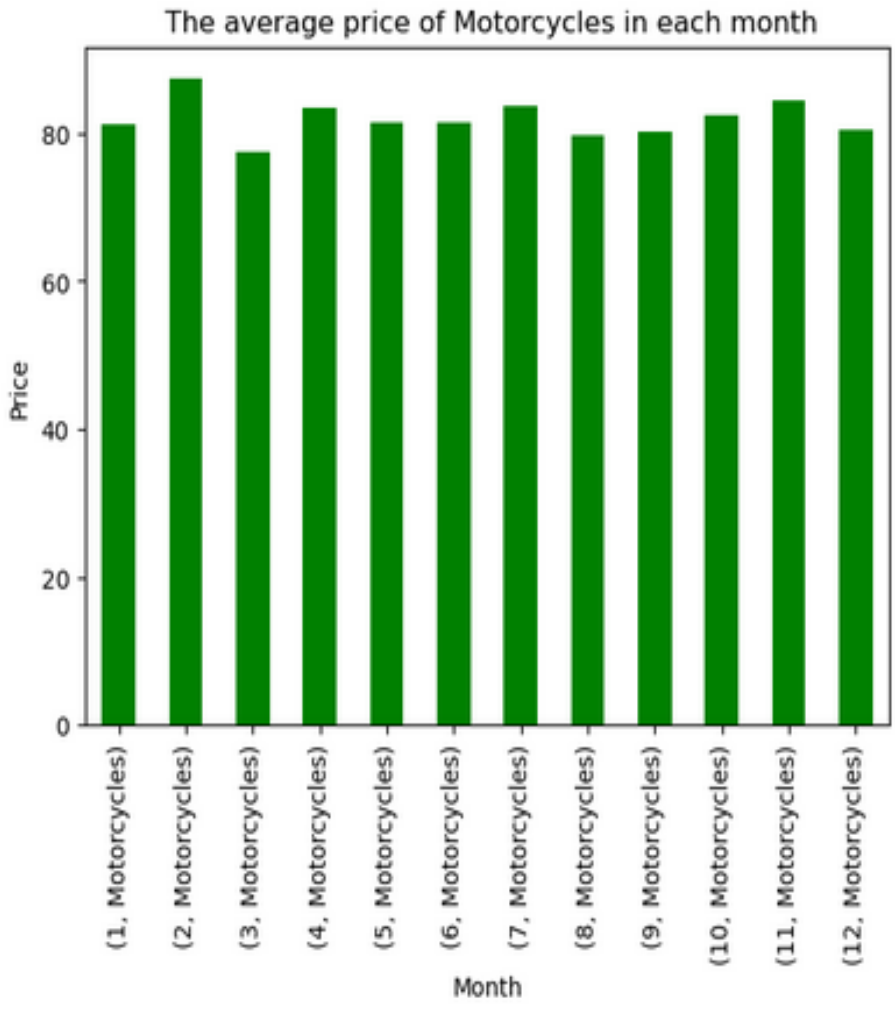
```
]: <Axes: title={'center': ' The average price of Vintage Cars in each month'}, xlabel='Month', ylabel='Price'>
```



- Average price Motorcycles for each month

```
[0]: 1 #Average price Motorcycles for each month
     2 avg_price_motorcycles_monthly.plot(kind= "bar" ,title = " The average price of Motorcycles in each month",
     3                                    xlabel="Month",ylabel="Price",rot=90 ,color="g" )
```

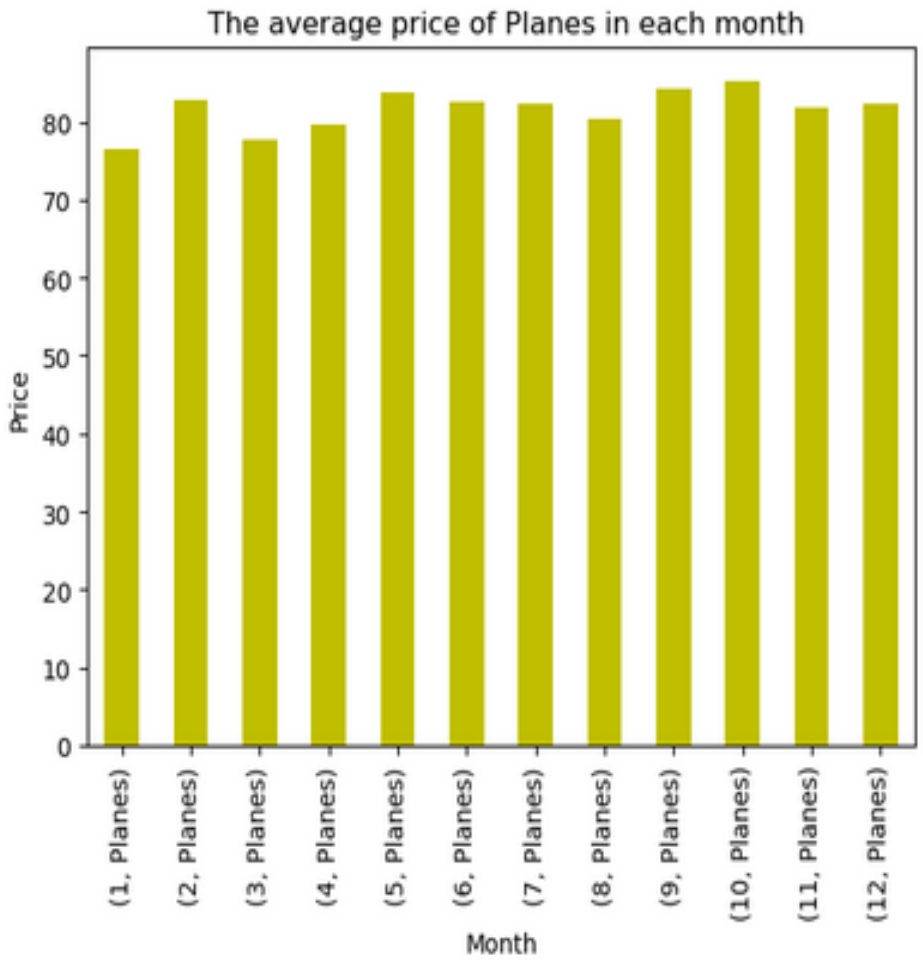
```
[0]: <Axes: title={'center': ' The average price of Motorcycles in each month'}, xlabel='Month', ylabel='Price'>
```



- Average price Planes for each month

```
#Average price Planes for each month
avg_price_planes_monthly.plot(kind= "bar" ,title = " The average price of Planes in each month",
                              xlabel="Month",ylabel="Price",rot=90 ,color="y" )
```

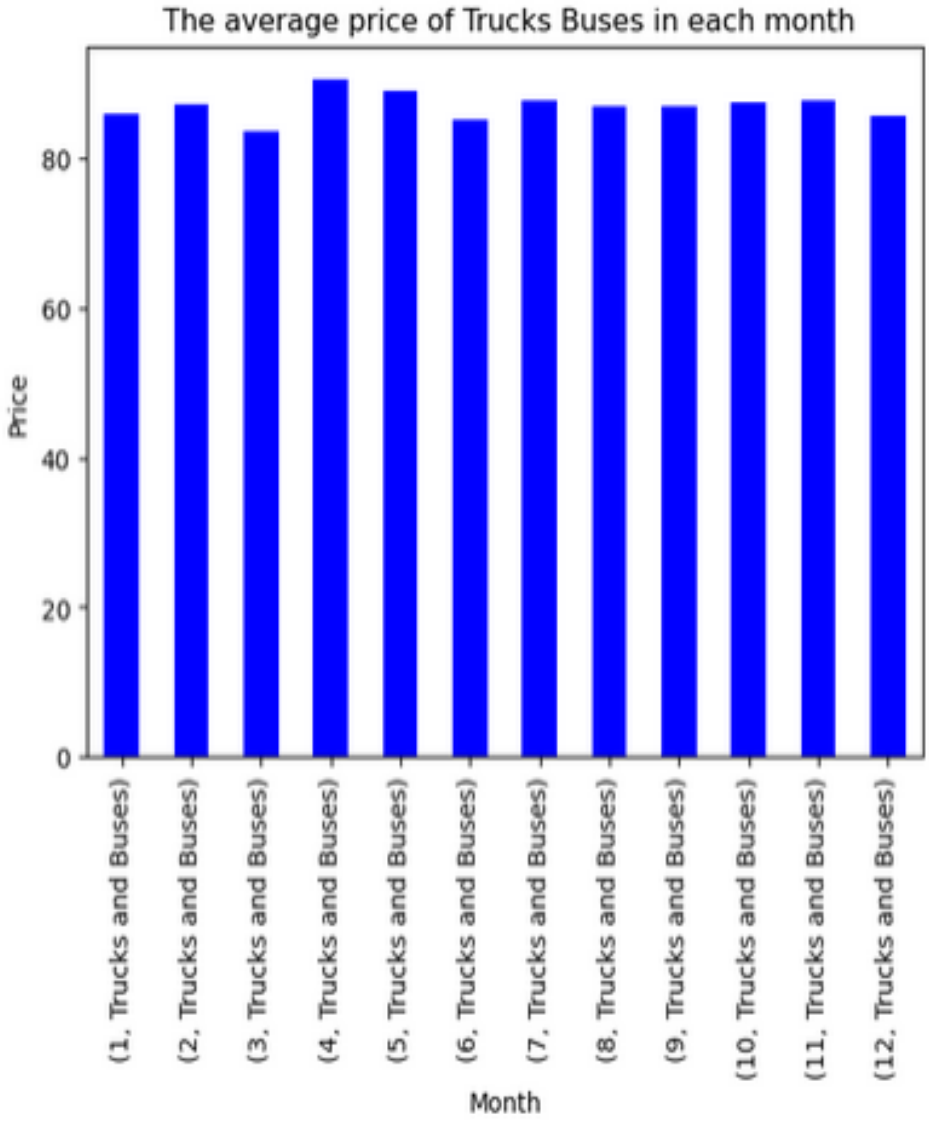
```
<Axes: title={'center': ' The average price of Planes in each month'}, xlabel='Month', ylabel='Price'>
```



- Average price Trucks Buses for each month

```
#Average price Trucks Buses for each month
avg_price_trucks_buses_monthly.plot(kind= "bar" ,title = " The average price of Trucks Buses in each month",
                                    xlabel="Month",ylabel="Price",rot=90 ,color="b" )
```

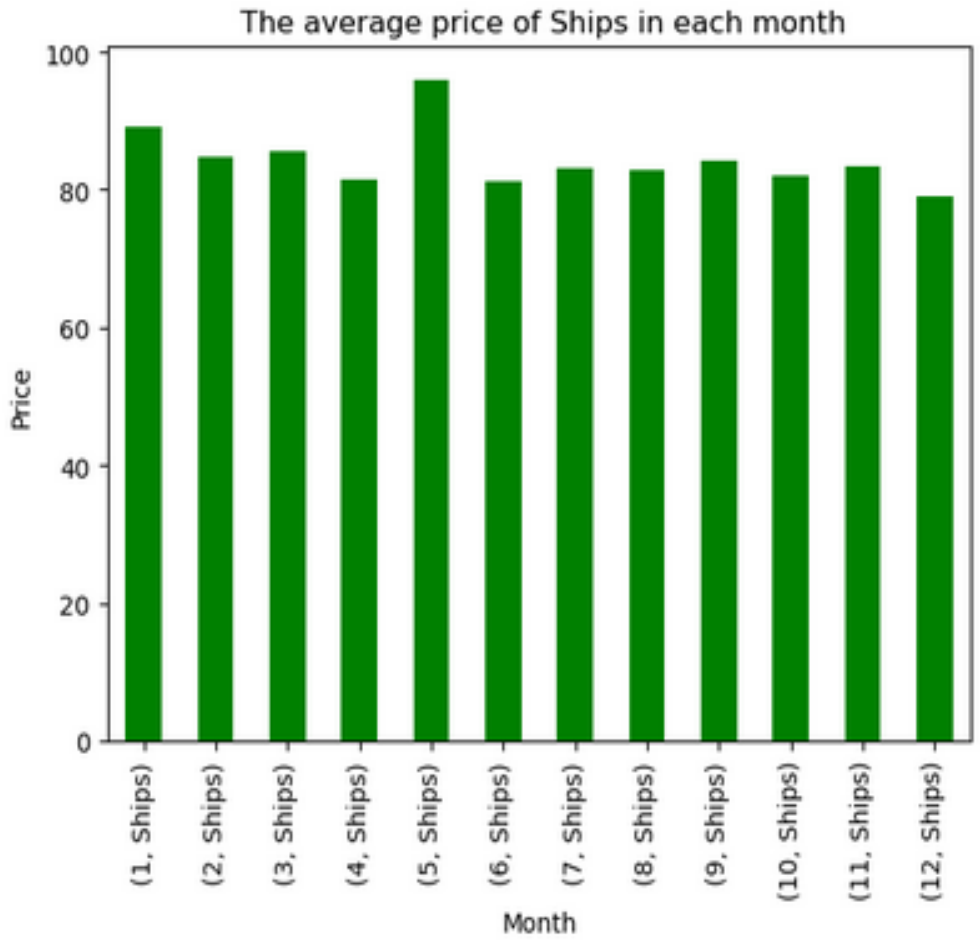
```
<Axes: title={'center': ' The average price of Trucks Buses in each month'}, xlabel='Month', ylabel='Price'>
```



- Average price Ships for each month

```
[ ]: 1 #Average price Ships for each month
     2 avg_price_ships_monthly.plot(kind= "bar" ,title = " The average price of Ships in each month",
     3                              xlabel="Month",ylabel="Price",rot=90 ,color="g" )
```

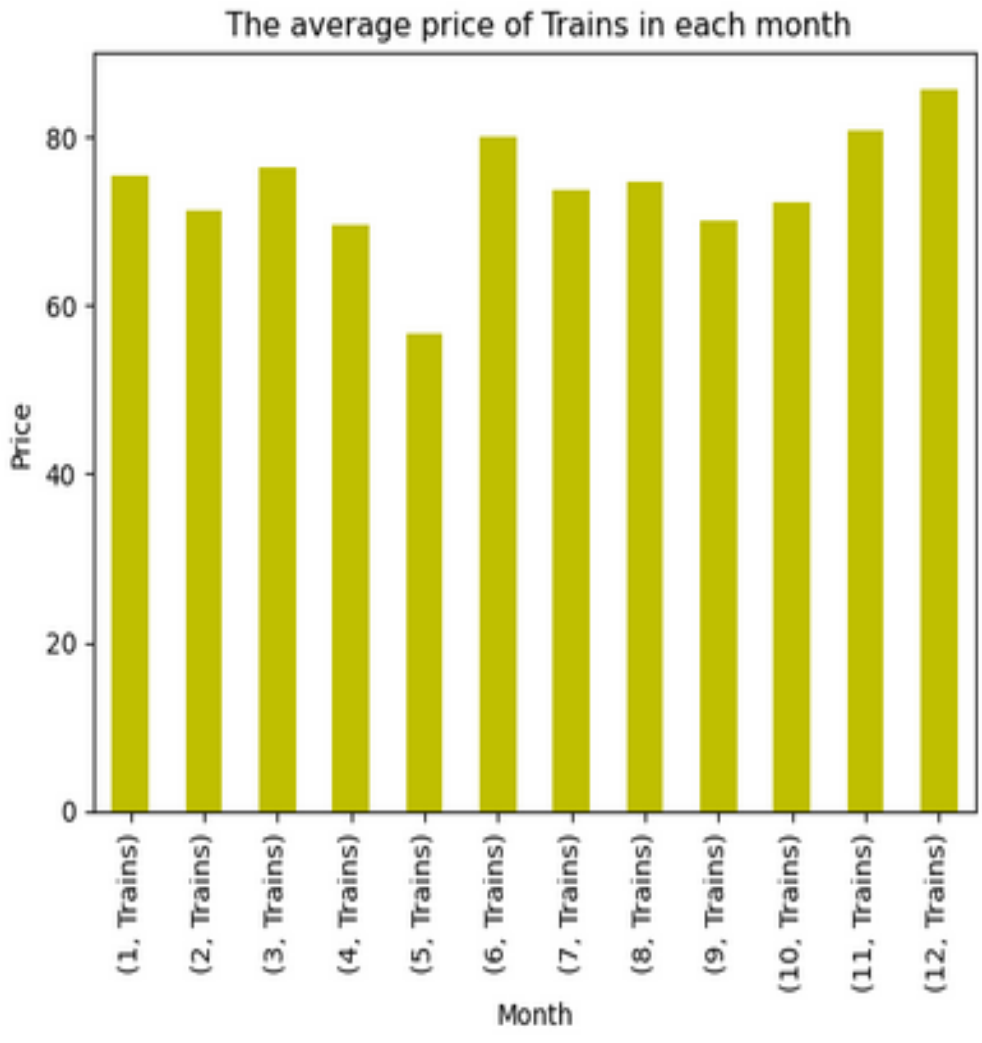
```
[ ]: <Axes: title={'center': ' The average price of Ships in each month'}, xlabel='Month', ylabel='Price'>
```



- Average price Trains for each month

```
#Average price Trains for each month
avg_price_trains_monthly.plot( kind= "bar" ,title = " The average price of Trains in each month",
                               xlabel="Month",ylabel="Price",rot=90 ,color="y" )
```

```
<Axes: title={'center': ' The average price of Trains in each month'}, xlabel='Month', ylabel='Price'>
```



4- Identify the top 5 cities where the company has the highest sales and provide recommendations on how the company can increase its sales in other cities.

4- Identify the top 5 cities where the company has the highest sales

[37]:

```
#Top 5 cities that has the highest sales:
top5_cities= df2.groupby(["city"])[["total revenue"]].sum().sort_values(by=["total revenue"] , ascending = False)
print (top5_cities.head())

```

```
           total revenue
city
Madrid         768571.50
San Rafael     510259.19
NYC            392873.13
Singapore      225985.50
Paris          208859.82
```