1. What was the total revenue generated by the company over the course of the year?
2. Which product had the highest revenue? How much revenue did it generate?
3. What was the average price of a product sold by the company?
4. What was the total quantity of products sold by the company?
5. Which category had the highest revenue? How much revenue did it generate?
6. What was the average revenue per sale?
7. What was the total revenue generated in each quarter of the year? (i.e. Q1, Q2, Q3, Q4)

at the begining of all we shall be importing all libraies we will need.

```
import numpy as np
import pandas as pd
import plotly.express as px
```

- date: The date of the sale (in YYYY-MM-DD format)
- product: The name of the product sold
- category: The category of the product (e.g. "electronics", "clothing", etc.)
- price: The price of the product (in USD)
- quantity: The quantity of the product sold
- revenue: The total revenue generated by the sale (i.e. price * quantity)

```
sales_data = pd.read_csv("./Data/sales_data.csv")
```

now as we have imported the data we will now discover the data

```
sales_data.head()
```

```
         date     product     category   price  quantity  revenue
0  2022-01-01  Smartphone  Electronics   600.0      10.0   6000.0
1  2022-01-01      Laptop  Electronics  1200.0       5.0   6000.0
2  2022-01-02     T-Shirt     Clothing    20.0      50.0   1000.0
3  2022-01-03  Headphones  Electronics   100.0      20.0   2000.0
4  2022-01-04     T-Shirt     Clothing    20.0      25.0    500.0
```

lets find out what is our data data types

```
sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 369 entries, 0 to 368
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   date      369 non-null    object 
 1   product   369 non-null    object 
 2   category  369 non-null    object 
 3   price     367 non-null    float64
 4   quantity  368 non-null    float64
 5   revenue   368 non-null    float64
dtypes: float64(3), object(3)
memory usage: 17.4+ KB
```

lets now look for missing values

```
sales_data.isnull().sum()
```

```
date        0
product     0
category    0
price       2
quantity    1
revenue     1
dtype: int64
```

we can see that we have to problems the first is the date column data type and the seconde is that we have missing values

```
sales_data["date"] = pd.to_datetime(sales_data["date"], format='%Y-%m-%d')

sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 369 entries, 0 to 368
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   date      369 non-null    datetime64[ns]
 1   product   369 non-null    object        
 2   category  369 non-null    object        
 3   price     369 non-null    float64       
 4   quantity  369 non-null    float64       
 5   revenue   369 non-null    float64       
dtypes: datetime64[ns](1), float64(3), object(2)
memory usage: 17.4+ KB
```

now the fist problem solved lets go the second one

lets fist see the index of the missing data

```
sales_data[sales_data['price'].isnull()].index.tolist()
```

```
[193, 320]
```

```
sales_data[sales_data['quantity'].isnull()].index.tolist()
```

```
[122]
```

```
sales_data[sales_data['revenue'].isnull()].index.tolist()
```

```
[96]
```

as we can opserve missing values are in different positions and we now that revenue=quantity * price we now can derive every missing value from the other two

```
sales_data['price'].iloc[193] =sales_data['revenue'].iloc[193] /
sales_data['quantity'].iloc[193]
sales_data['price'].iloc[320] =sales_data['revenue'].iloc[320] /
sales_data['quantity'].iloc[320]
sales_data['quantity'].iloc[122] = sales_data['revenue'].iloc[122] /
sales_data['price'].iloc[122]
sales_data['revenue'].iloc[96] = sales_data['quantity'].iloc[96] *
sales_data['price'].iloc[96]
```

```
/home/turing/anaconda3/lib/python3.9/site-packages/pandas/core/
indexing.py:1732: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

we will just ignore this warning an continue our work and see if it worked

```
sales_data.isnull().sum()
```

```
date        0
product     0
category    0
price       0
quantity    0
revenue     0
dtype: int64
```

it worked!

```
total_revenue = sales_data["revenue"].sum()
total_revenue
```

```
760330.0
```

by knowing the agerage revenue we can compare it to the previous years to get more insights

```
highest_revenue_index = sales_data["revenue"].argmax()
highest_revenue_product =
sales_data["product"].iloc[highest_revenue_index]
highest_revenue = sales_data["revenue"].iloc[highest_revenue_index]

highest_revenue_product, highest_revenue
```

```
('Smartphone', 7200.0)
```

```
highest_total_revenue_product =
sales_data.groupby('product').agg(Sum=('revenue', np.sum))

highest_total_revenue_product.loc[highest_total_revenue_product.idxmax
()]
```

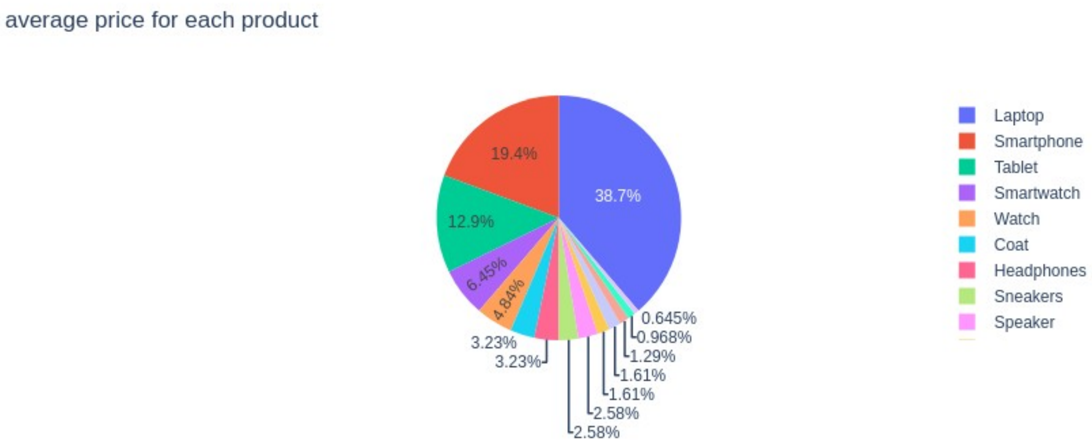
```
                 Sum
product
Smartphone  434400.0
```

hence we know for sure that smart phones gets us more profit more than the other product which means that we might have to increase samrt phone sales

```
average_prices_per_product =
sales_data.groupby('product').agg(average_price=('price',
np.mean)).sort_values(by='average_price',
ascending=False).reset_index()
average_prices_per_product
```

```
       product  average_price
0       Laptop         1200.0
1   Smartphone          600.0
2       Tablet          400.0
3   Smartwatch          200.0
4        Watch          150.0
5         Coat          100.0
6   Headphones          100.0
7     Sneakers           80.0
8      Speaker           80.0
9     Backpack           50.0
10       Jeans           50.0
11      Hoodie           40.0
12      Wallet           30.0
13     T-Shirt           20.0
```

```
px.pie(data_frame=average_prices_per_product, names='product',
values='average_price', title="average price for each product",
labels={
    'product':'product',
    'average_price':'average_price in usd'
})
```



hence we see that price of laptops is the highest however smart phone gets us more profit so we might consider decrease the amount of laptops we sale

```
total_quantity_of_products =
sales_data.groupby('product').agg(quantity=('quantity', np.sum))
['quantity'].sum()
total_quantity_of_products

5371.0
```

looking to the total_quantity_of_products we can see the company sales for the whole year

```
###the highest revenue index had already been calculated
highest_revenue_category =
sales_data["category"].iloc[highest_revenue_index]
highest_revenue_category, highest_revenue

('Electronics', 7200.0)

highest_total_revenue_category =
sales_data.groupby('category').agg(total_revenue=('revenue', np.sum))
highest_total_revenue_category.loc[highest_total_revenue_category.idxm
ax()]

             avg_revenue
category
Electronics     516080.0
```

we know surely know that electronics gets us the highest revenue which also pushs us towards saling more electronics

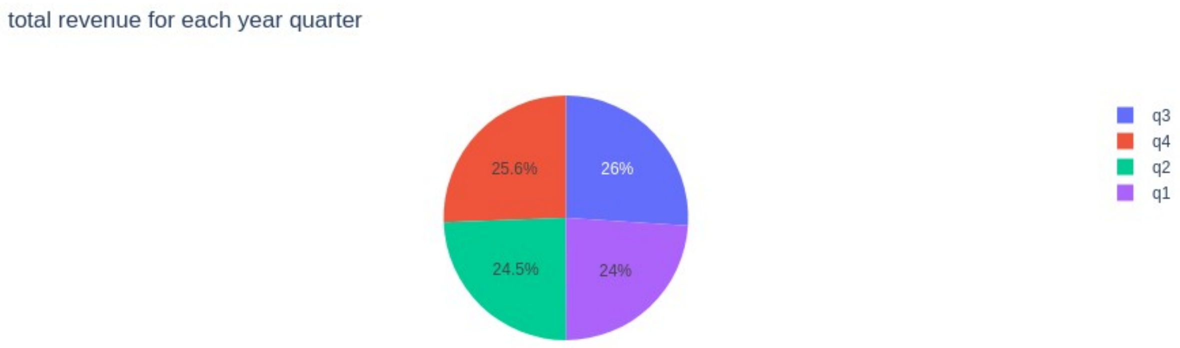
which is basicaly the average revenue over the user

```
average_revenue_per_sale = sales_data["revenue"].mean()
average_revenue_per_sale
```

```
2060.5149051490516
```

by knowing the average revenue we can compare it the previous years to have more insights

```
####first Dividing year into quarters
sales_data["quarters"] = 
sales_data['date'].dt.to_period('Q').dt.strftime('q%q')

sales_data.head(3)

        date     product     category   price  quantity  revenue  
quarters
0 2022-01-01  Smartphone  Electronics   600.0      10.0   6000.0  
q1
1 2022-01-01      Laptop  Electronics  1200.0       5.0   6000.0  
q1
2 2022-01-02     T-Shirt     Clothing    20.0      50.0   1000.0  
q1

revenue_at_quarter = 
sales_data.groupby('quarters').agg(total_revenue=('revenue', 
np.sum)).reset_index()
revenue_at_quarter

  quarters  total_revenue
0       q1       182100.0
1       q2       185970.0
2       q3       197680.0
3       q4       194580.0

px.pie(data_frame=revenue_at_quarter, names='quarters', 
values='total_revenue', labels={
    'quarters':'quarters',
    'total_revenue':'total_revenue in usd'
}, title="total revenue for each year quarter")
```



by looking to this chart we can opserve that the total revenue of each quarter of the year is nearly the same, which means that our sales don't affect by much with seasons, which is basicaly amazing

jackmahouz@github.com