Data Wrangling di Shopping dataset

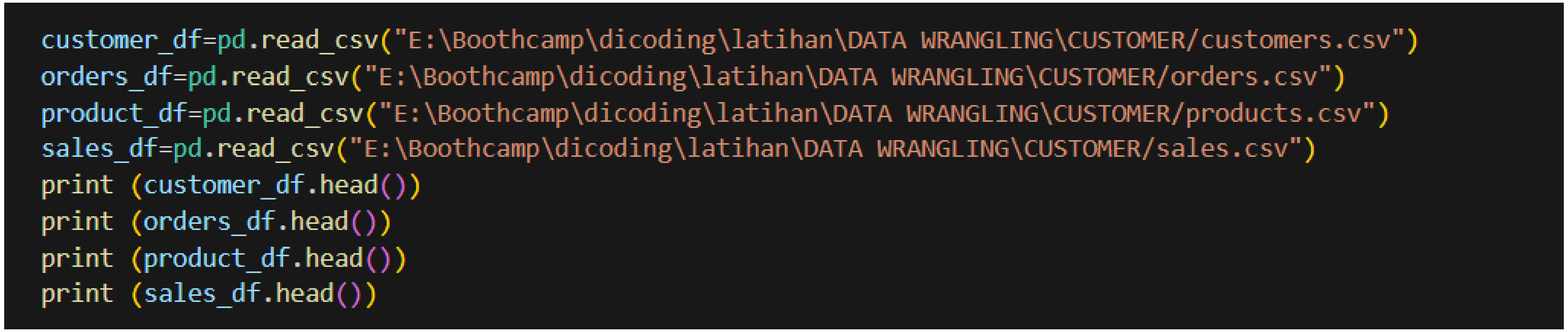
Dataset ini berisi 4 table :

1. Customer_df
2. Order_df
3. Product_df
4. Sales_df

Step for data wrangling

1. **Melihat head dari data**
2. **Melihat jumlah missing value**
3. **Melihat jumlah dupilikat**
4. **Melihat tipe data**
5. **Melihat outliers**

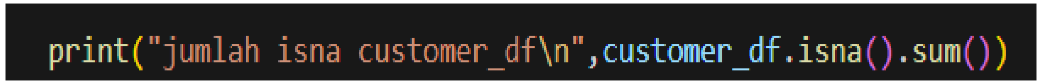
Load Data



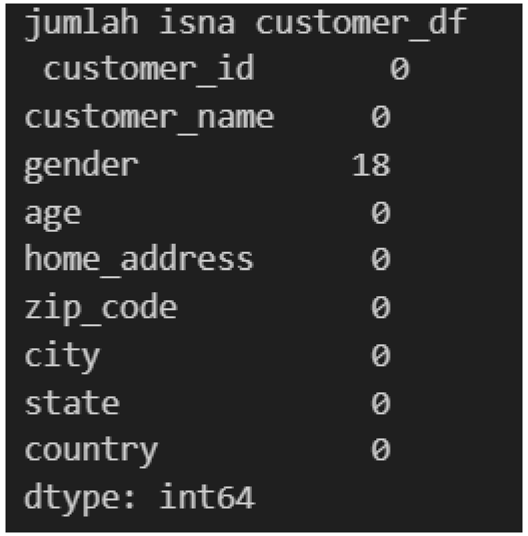
```
customer_df=pd.read_csv("E:\Boothcamp\dicoding\latihan\DATA WRANGLING\CUSTOMER/customers.csv")
orders_df=pd.read_csv("E:\Boothcamp\dicoding\latihan\DATA WRANGLING\CUSTOMER/orders.csv")
product_df=pd.read_csv("E:\Boothcamp\dicoding\latihan\DATA WRANGLING\CUSTOMER/products.csv")
sales_df=pd.read_csv("E:\Boothcamp\dicoding\latihan\DATA WRANGLING\CUSTOMER/sales.csv")
print (customer_df.head())
print (orders_df.head())
print (product_df.head())
print (sales_df.head())
```

Customer df

1. Melihat jumlah missing value dengan isna().sum()

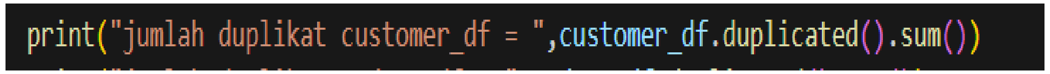


```
print("jumlah isna customer_df\n",customer_df.isna().sum())
```

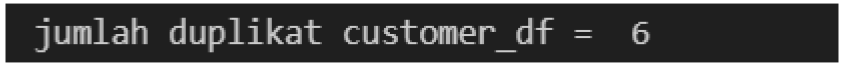


```
jumlah isna customer_df
 customer_id        0
customer_name      0
gender            18
age                0
home_address       0
zip_code           0
city               0
state              0
country            0
dtype: int64
```

2. Melihat data duplicate dengan duplicate().sum()



```
print("jumlah duplikat customer_df = ",customer_df.duplicated().sum())
```



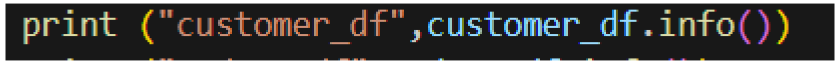
```
jumlah duplikat customer_df =  6
```

Result :

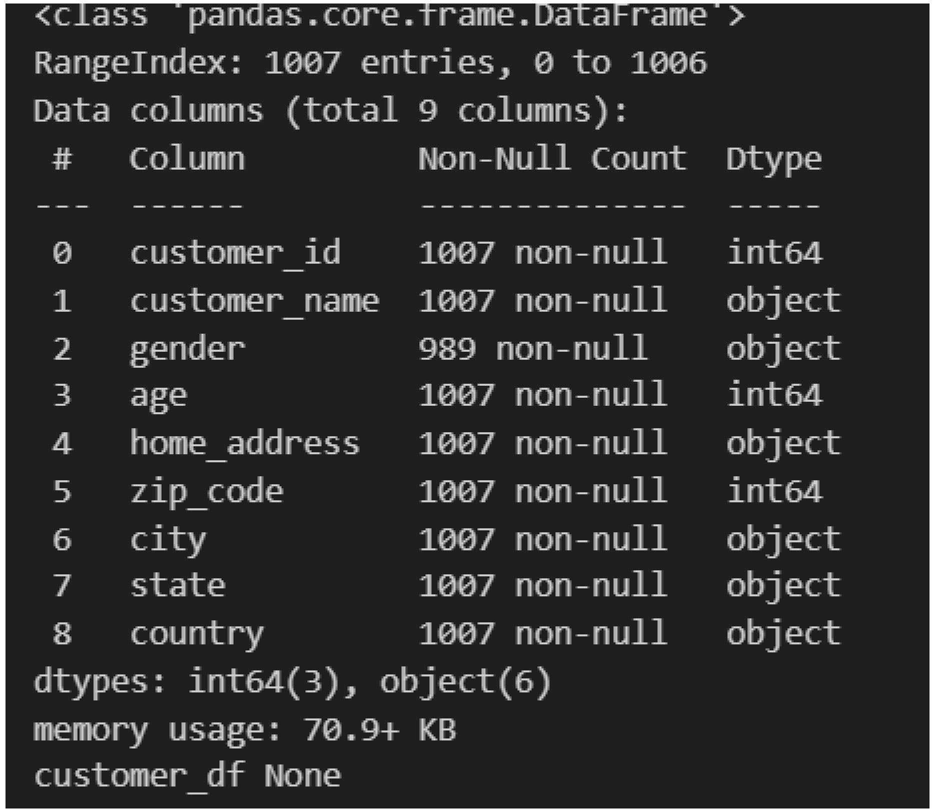
1. Ditemukan 18 baris missing value di kolom gender
2. Ditemukan 6 baris data duplikat

Customer df

3. Melihat tipe data masing2 kolom dengan df.info()

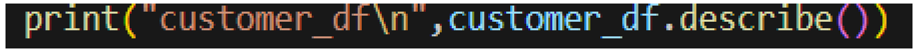


```
print ("customer_df",customer_df.info())
```

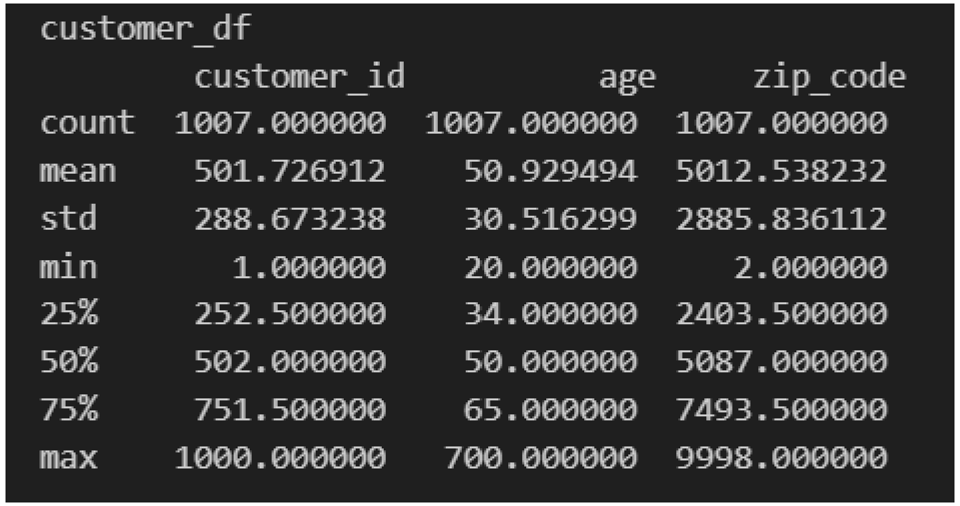


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1007 entries, 0 to 1006
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   customer_id    1007 non-null   int64
 1   customer_name  1007 non-null   object
 2   gender         989 non-null    object
 3   age            1007 non-null   int64
 4   home_address   1007 non-null   object
 5   zip_code       1007 non-null   int64
 6   city           1007 non-null   object
 7   state          1007 non-null   object
 8   country        1007 non-null   object
dtypes: int64(3), object(6)
memory usage: 70.9+ KB
customer_df None
```

4. Melihat outlier dengan df.describe()



```
print("customer_df\n",customer_df.describe())
```

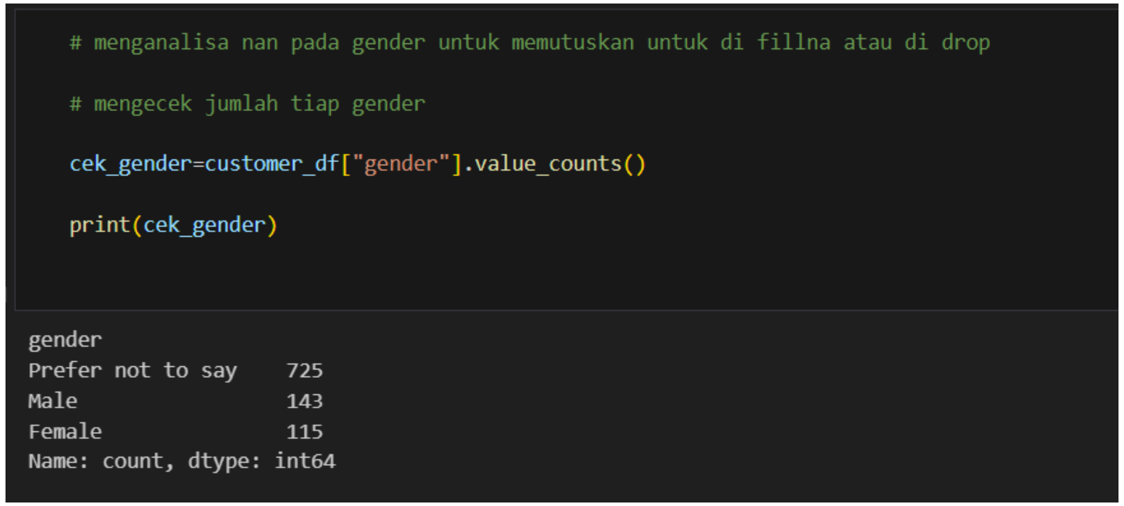


```
 customer_df
         customer_id          age     zip_code
count    1007.000000  1007.000000  1007.000000
mean      501.726912    50.929494  5012.538232
std       288.673238    30.516299  2885.836112
min         1.000000    20.000000     2.000000
25%       252.500000    34.000000  2403.500000
50%       502.000000    50.000000  5087.000000
75%       751.500000    65.000000  7493.500000
max      1000.000000   700.000000  9998.000000
```

Result
3. Tipe data masing2 kolom sudah sesuai
4. Outlier di Max age =700

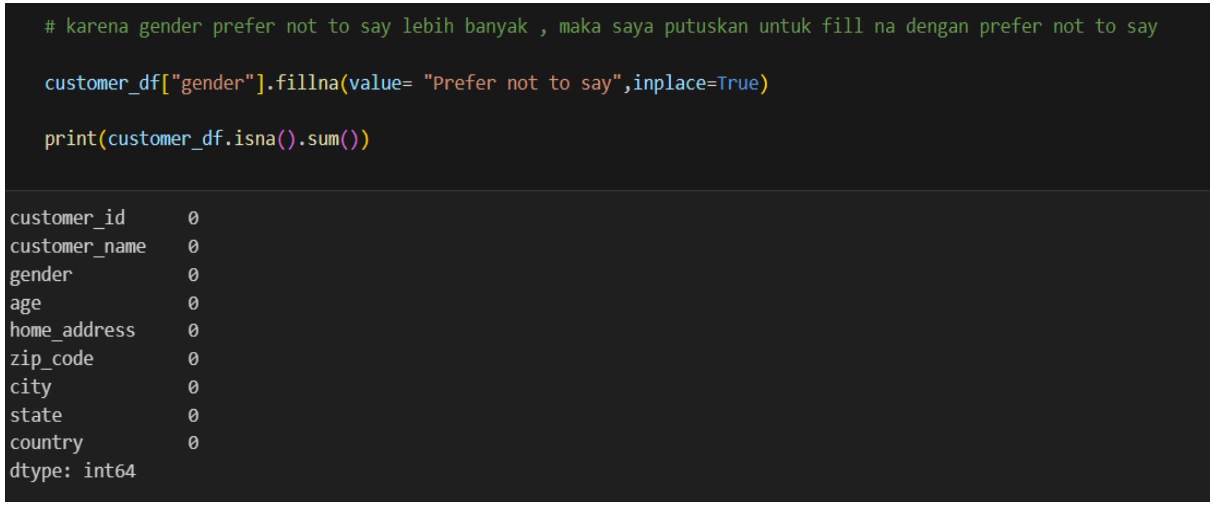
Cleaning Customer_df

1. Fillna missing value pada kolom gender dengan gender terbanyak yaitu "prefer not to say"



```
# menganalisa nan pada gender untuk memutuskan untuk di fillna atau di drop

# mengecek jumlah tiap gender

cek_gender=customer_df["gender"].value_counts()

print(cek_gender)
```

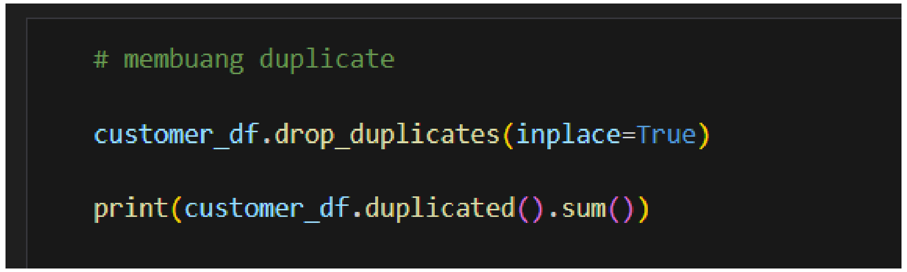
```
gender
Prefer not to say     725
Male                  143
Female                115
Name: count, dtype: int64
```



```
# karena gender prefer not to say lebih banyak , maka saya putuskan untuk fill na dengan prefer not to say

customer_df["gender"].fillna(value= "Prefer not to say",inplace=True)

print(customer_df.isna().sum())
```

```
customer_id      0
customer_name    0
gender           0
age              0
home_address     0
zip_code         0
city             0
state            0
country          0
dtype: int64
```

2. Membuang duplicate dengan drop_duplicate()

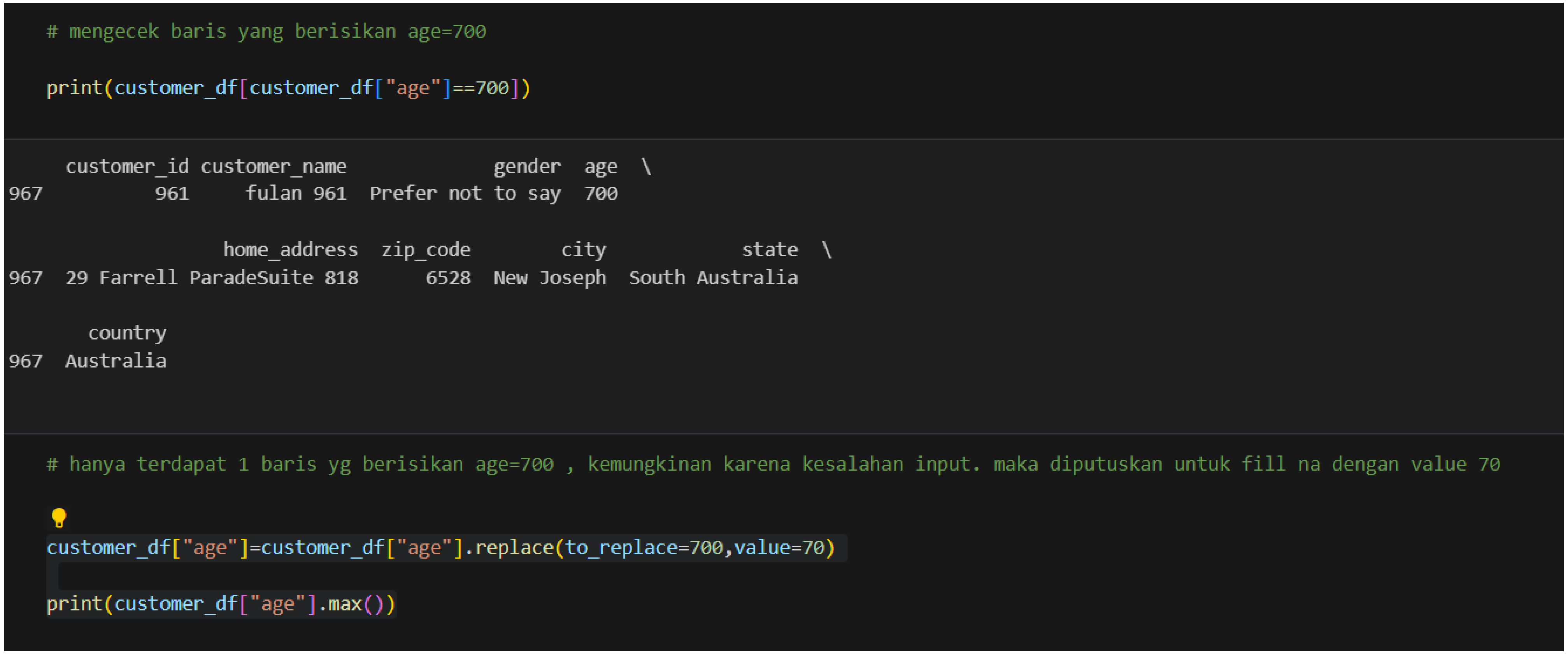


```
# membuang duplicate

customer_df.drop_duplicates(inplace=True)

print(customer_df.duplicated().sum())
```

Cleaning Customer_df

3. Setelah di teliti, terjadi kesalahan penginputan umur yang seharusnya 70 menjadi 700.

Replace age=700 menjadi 70



```
# mengecek baris yang berisikan age=700

print(customer_df[customer_df["age"]==700])
```

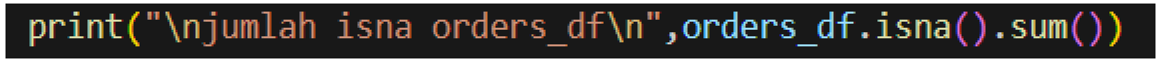
```
    customer_id customer_name             gender  age  \
967         961     fulan 961  Prefer not to say  700

                 home_address  zip_code       city            state  \
967  29 Farrell ParadeSuite 818      6528  New Joseph  South Australia

      country
967  Australia
```

```
# hanya terdapat 1 baris yg berisikan age=700 , kemungkinan karena kesalahan input. maka diputuskan untuk fill na dengan value 70

customer_df["age"]=customer_df["age"].replace(to_replace=700,value=70)

print(customer_df["age"].max())
```

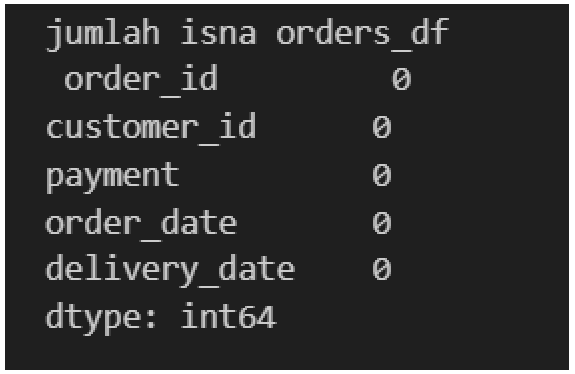
Order_df

Order_df

1. Melihat jumlah missing value dengan isna().sum()

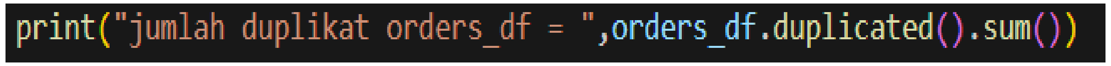


```
print("\njumlah isna orders_df\n",orders_df.isna().sum())
```

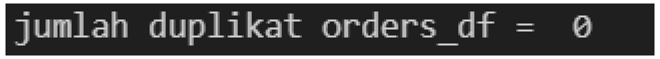


```
jumlah isna orders_df
 order_id         0
customer_id      0
payment          0
order_date       0
delivery_date    0
dtype: int64
```

2. Melihat data duplicate dengan duplicate().sum()



```
print("jumlah duplikat orders_df = ",orders_df.duplicated().sum())
```



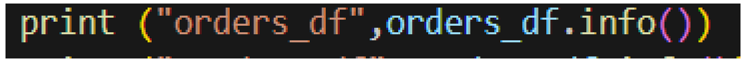
```
jumlah duplikat orders_df =  0
```

Result :

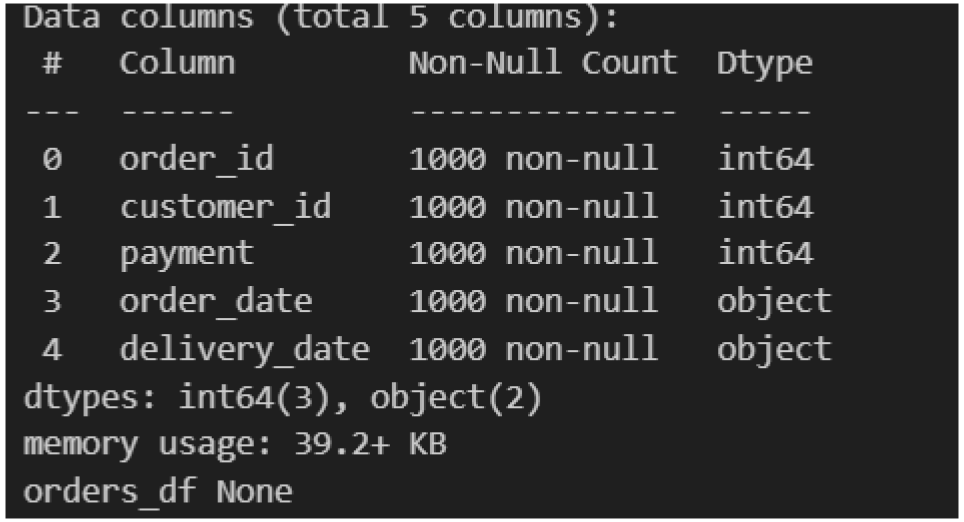
1. Tidak ada missing value (isna)
2. Tidak ada duplikat

Order_df

3. Melihat tipe data masing2 kolom dengan df.info()

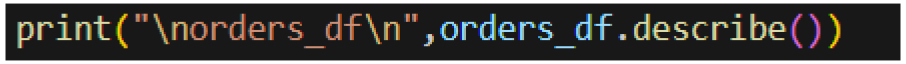


```
print ("orders_df",orders_df.info())
```

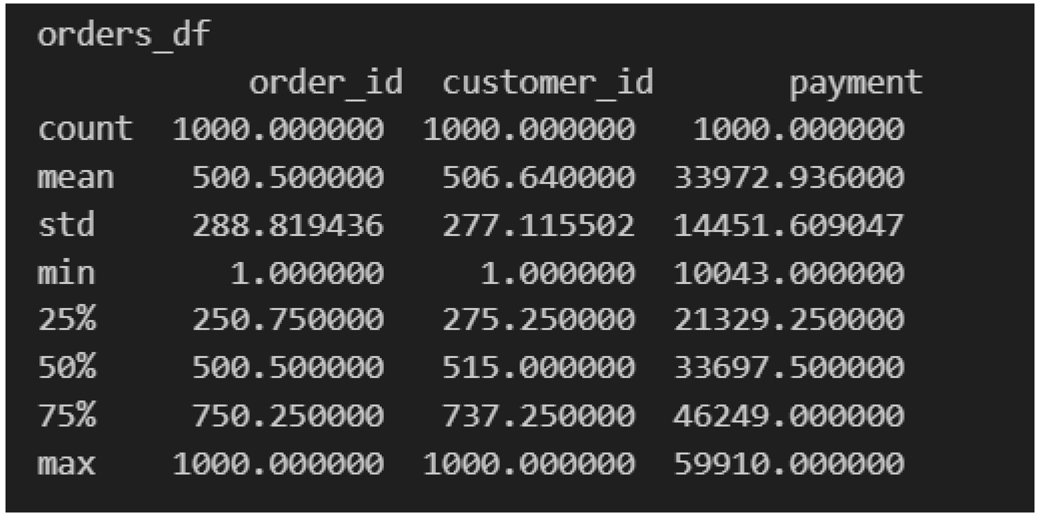


```
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   order_id       1000 non-null   int64
 1   customer_id    1000 non-null   int64
 2   payment        1000 non-null   int64
 3   order_date     1000 non-null   object
 4   delivery_date  1000 non-null   object
dtypes: int64(3), object(2)
memory usage: 39.2+ KB
orders_df None
```

4. Melihat outlier dengan df.describe()



```
print("\norders_df\n",orders_df.describe())
```

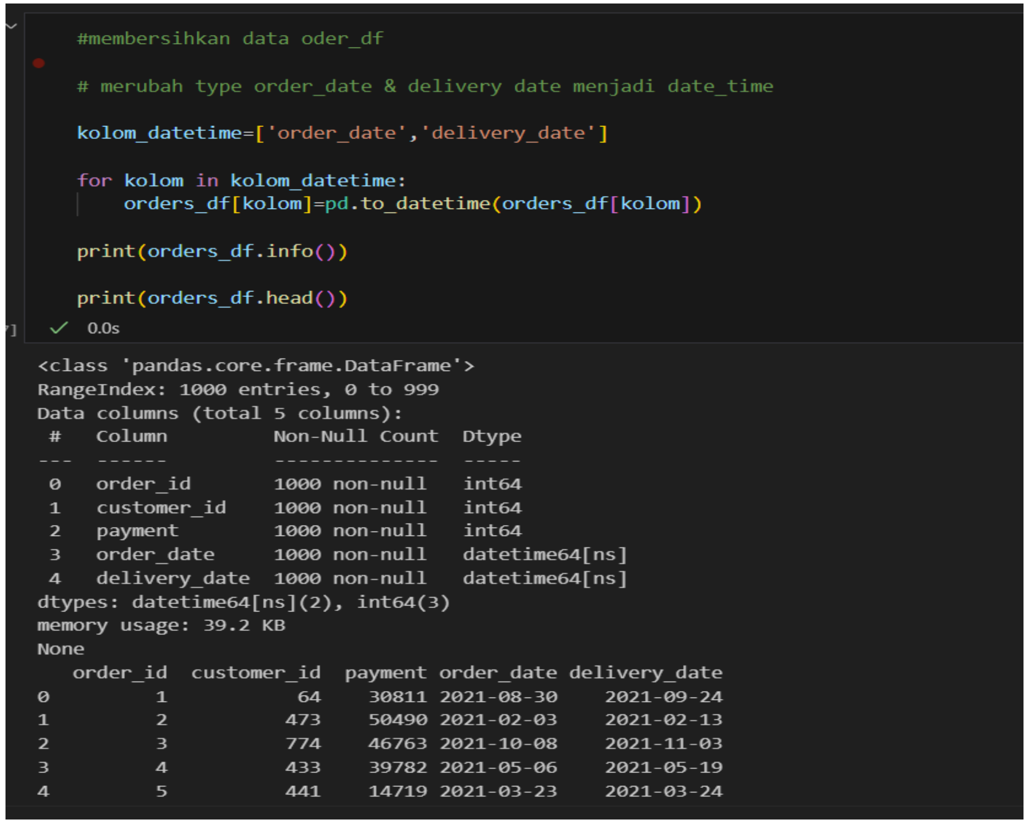


```
 orders_df
          order_id  customer_id       payment
count  1000.000000  1000.000000   1000.000000
mean    500.500000   506.640000  33972.936000
std     288.819436   277.115502  14451.609047
min       1.000000     1.000000  10043.000000
25%     250.750000   275.250000  21329.250000
50%     500.500000   515.000000  33697.500000
75%     750.250000   737.250000  46249.000000
max    1000.000000  1000.000000  59910.000000
```

Result
3. Tipe data order_date dan delivery _date seharusnya datetime
4. Tidak ditemukan outliers

Cleaning- orders_df

1. Merubah kolom order_date & delivery_date dengan pd.to_datetime()



```
#membersihkan data oder_df

# merubah type order_date & delivery date menjadi date_time

kolom_datetime=['order_date','delivery_date']

for kolom in kolom_datetime:
    orders_df[kolom]=pd.to_datetime(orders_df[kolom])

print(orders_df.info())

print(orders_df.head())
```

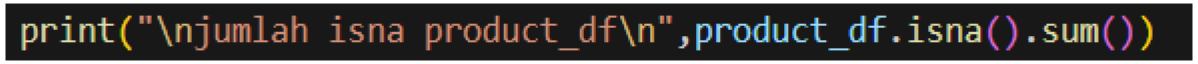
0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   order_id       1000 non-null   int64
 1   customer_id    1000 non-null   int64
 2   payment        1000 non-null   int64
 3   order_date     1000 non-null   datetime64[ns]
 4   delivery_date  1000 non-null   datetime64[ns]
dtypes: datetime64[ns](2), int64(3)
memory usage: 39.2 KB
None
   order_id  customer_id  payment order_date delivery_date
0         1           64    30811 2021-08-30    2021-09-24
1         2          473    50490 2021-02-03    2021-02-13
2         3          774    46763 2021-10-08    2021-11-03
3         4          433    39782 2021-05-06    2021-05-19
4         5          441    14719 2021-03-23    2021-03-24
```

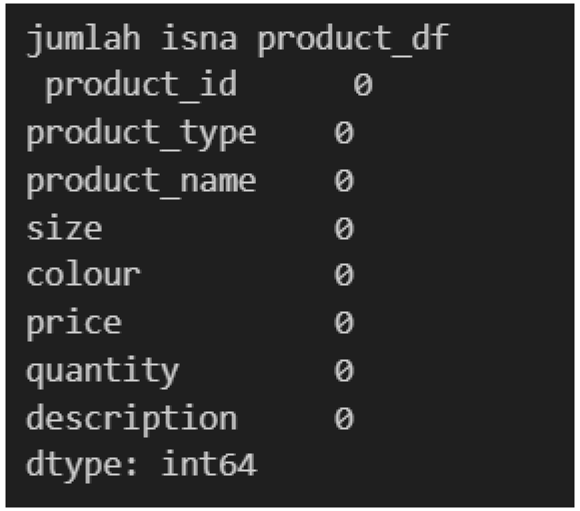
Product_df

Product_df

1. Melihat jumlah missing value dengan isna().sum()

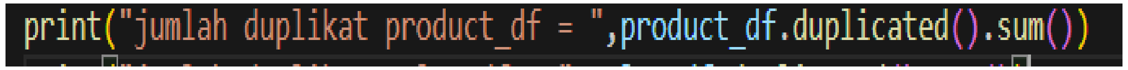


```
print("\njumlah isna product_df\n",product_df.isna().sum())
```

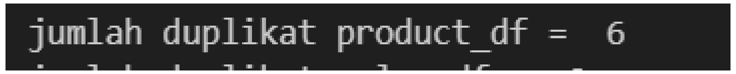


```
jumlah isna product_df
 product_id      0
product_type    0
product_name    0
size            0
colour          0
price           0
quantity        0
description     0
dtype: int64
```

2. Melihat data duplicate dengan duplicate().sum()



```
print("jumlah duplikat product_df = ",product_df.duplicated().sum())
```



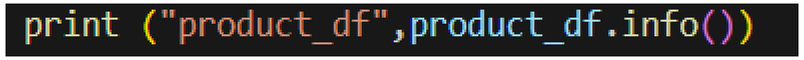
```
jumlah duplikat product_df =  6
```

Result :

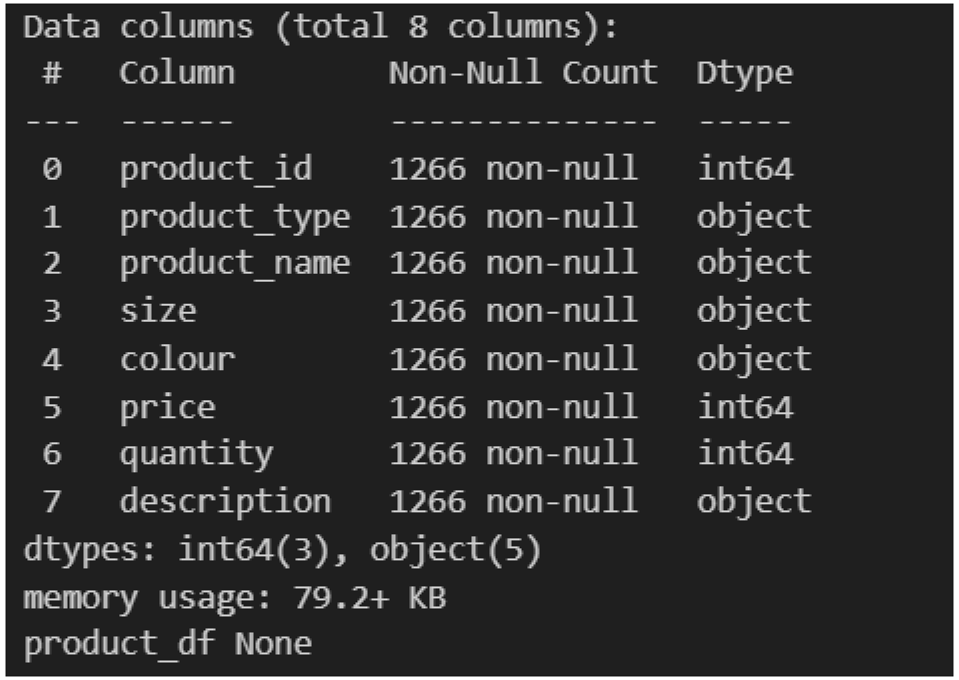
1. Tidak ditemukan missing value
2. Ditemukan 6 baris data duplikat

Product_df

3. Melihat tipe data masing2 kolom dengan df.info()

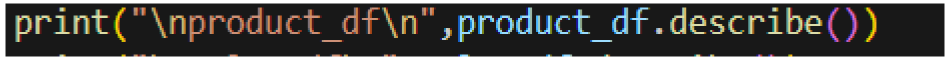


```
print ("product_df",product_df.info())
```

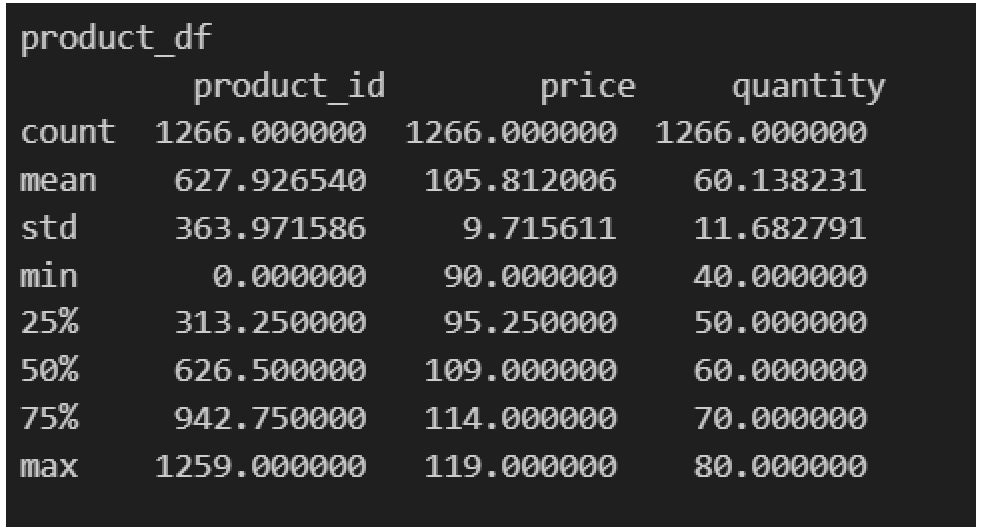


```
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   product_id    1266 non-null   int64
 1   product_type  1266 non-null   object
 2   product_name  1266 non-null   object
 3   size          1266 non-null   object
 4   colour        1266 non-null   object
 5   price         1266 non-null   int64
 6   quantity      1266 non-null   int64
 7   description   1266 non-null   object
dtypes: int64(3), object(5)
memory usage: 79.2+ KB
product_df None
```

4. Melihat outlier dengan df.describe()



```
print("\nproduct_df\n",product_df.describe())
```

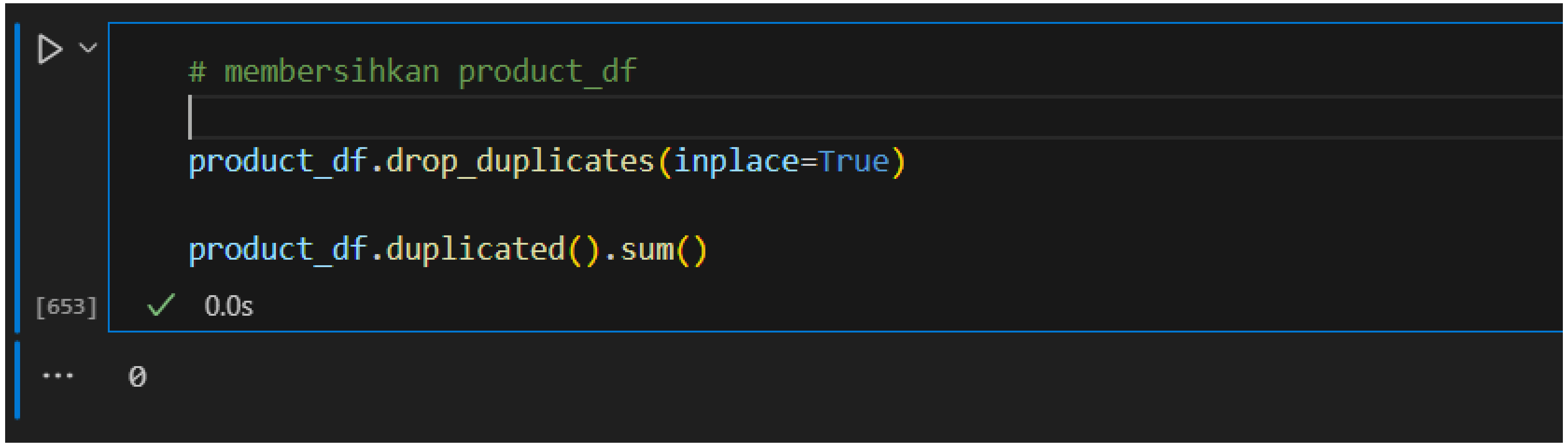


```
product_df
        product_id        price     quantity
count  1266.000000  1266.000000  1266.000000
mean    627.926540   105.812006    60.138231
std     363.971586     9.715611    11.682791
min       0.000000    90.000000    40.000000
25%     313.250000    95.250000    50.000000
50%     626.500000   109.000000    60.000000
75%     942.750000   114.000000    70.000000
max    1259.000000   119.000000    80.000000
```

Result
3. Tipe data masing2 kolom sudah sesuai
4. Tidak ditemukan outliers

Cleaning – product_df

1. Menghapus baris duplicate dengan drop_duplicate()



```
# membersihkan product_df

product_df.drop_duplicates(inplace=True)

product_df.duplicated().sum()
```

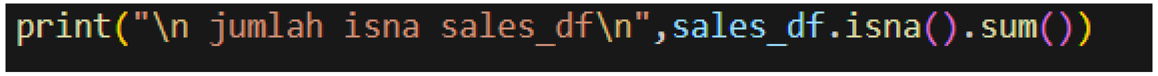
[653] ✓ 0.0s

```
0
```

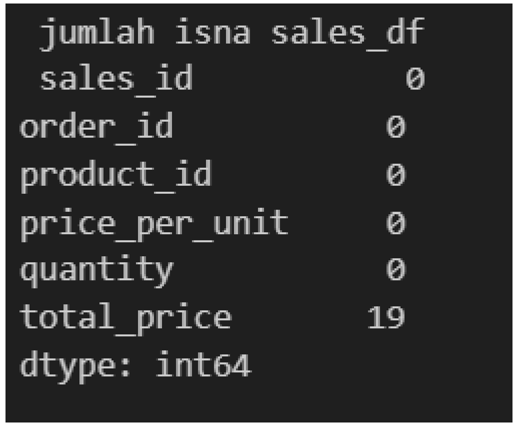
Sales_df

Sales_df

1. Melihat jumlah missing value dengan isna().sum()

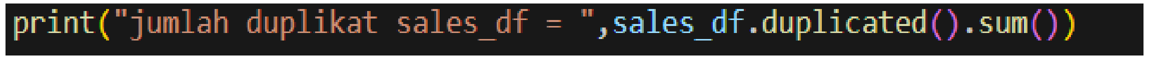


```
print("\n jumlah isna sales_df\n",sales_df.isna().sum())
```

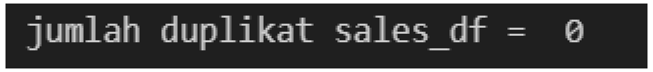


```
 jumlah isna sales_df
 sales_id           0
order_id           0
product_id         0
price_per_unit     0
quantity           0
total_price       19
dtype: int64
```

2. Melihat data duplicate dengan duplicate().sum()



```
print("jumlah duplikat sales_df = ",sales_df.duplicated().sum())
```



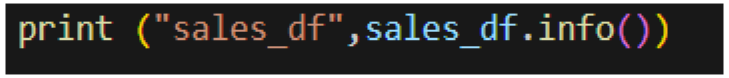
```
jumlah duplikat sales_df =  0
```

Result :

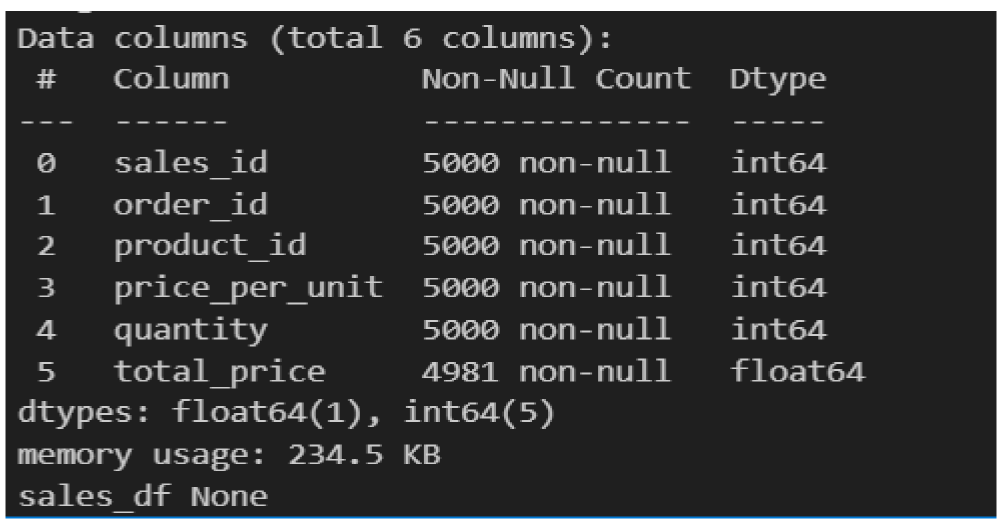
1. Ditemukan 19 baris missing value di kolom total_price
2. Tidak ditemukan data duplikat

Sales_df

3. Melihat tipe data masing2 kolom dengan df.info()



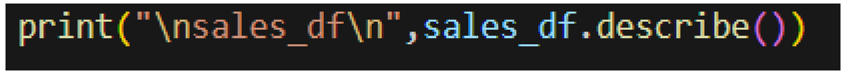
```
print ("sales_df",sales_df.info())
```



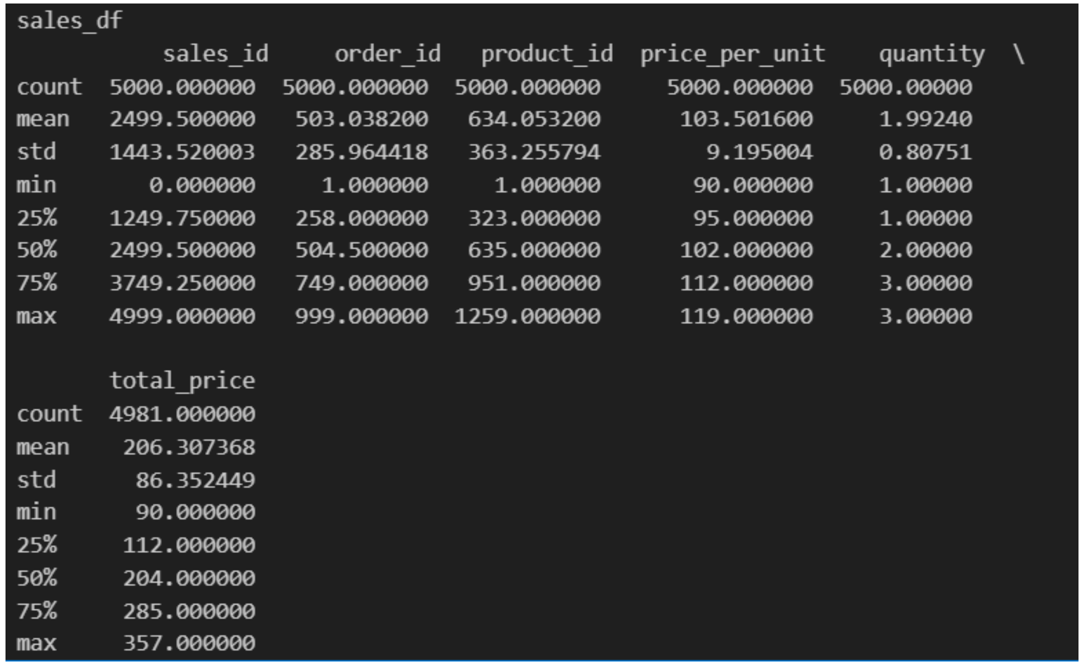
```
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sales_id        5000 non-null   int64
 1   order_id        5000 non-null   int64
 2   product_id      5000 non-null   int64
 3   price_per_unit  5000 non-null   int64
 4   quantity        5000 non-null   int64
 5   total_price     4981 non-null   float64
dtypes: float64(1), int64(5)
memory usage: 234.5 KB
sales_df None
```

Result
3. Tipe data masing2 kolom sudah sesuai
4. Tidak ditemukan outliers

4. Melihat outlier dengan df.describe()



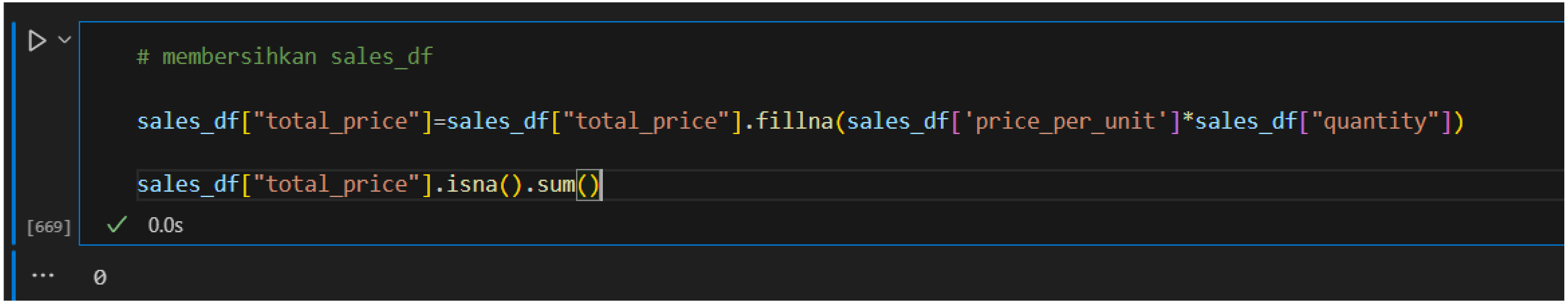
```
print("\nsales_df\n",sales_df.describe())
```



```
sales_df
          sales_id     order_id   product_id  price_per_unit    quantity  \
count  5000.000000  5000.000000  5000.000000     5000.000000  5000.00000
mean   2499.500000   503.038200   634.053200      103.501600     1.99240
std    1443.520003   285.964418   363.255794        9.195004     0.80751
min       0.000000     1.000000     1.000000       90.000000     1.00000
25%    1249.750000   258.000000   323.000000       95.000000     1.00000
50%    2499.500000   504.500000   635.000000      102.000000     2.00000
75%    3749.250000   749.000000   951.000000      112.000000     3.00000
max    4999.000000   999.000000  1259.000000      119.000000     3.00000

       total_price
count  4981.000000
mean    206.307368
std      86.352449
min      90.000000
25%     112.000000
50%     204.000000
75%     285.000000
max     357.000000
```

Cleaning – sales_df

1. Impute missing value di kolom total_price dengan mengalikan kolom price_per_unit *quantity



```
# membersihkan sales_df

sales_df["total_price"]=sales_df["total_price"].fillna(sales_df['price_per_unit']*sales_df["quantity"])

sales_df["total_price"].isna().sum()
```

[669] ✓ 0.0s

```
0
```