jhsknleqj

February 22, 2025

[43]:
```
# import all the required libaries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[44]:
```
# load all the dataset
data=pd.read_csv("/content/Amazon Sale Report.csv")
df=pd.DataFrame(data)
df
```

```
<ipython-input-44-48eb2e622eb9>:2: DtypeWarning: Columns (21,23) have
mixed types. Specify dtype option on import or set low_memory=False.
data=pd.read_csv("/content/Amazon Sale Report.csv")
```

[44]:

```
        index           Order ID     Date                       Status \
0           0 405-8078784-5731545 04-30-22  Cancelled
1           1 171-9198151-1101146 04-30-22 Shipped - Delivered to
            Buyer
2           2 404-0687676-7273146 04-30-22  Shipped 3  3 403-9615377-
            8133951 04-30-22     Cancelled
 4          4 407-1069790-7240320 04-30-22                   Shipped
 …       …                 …       …                          …
112452 112452 408-0639045-1056304 06-12-22  Shipped
112453 112453 408-0639045-1056304 06-12-22  Shipped
112454 112454 404-4060831-3562702 06-12-22  Shipped
112455 112455 402-8773996-8205915 06-12-22  Shipped
112456 112456 406-9793868-4020356 06-12-22 Shipped - Delivered to
       Buyer

    Fulfilment Sales Channel ship-service-level Style \
0       Merchant Amazon.in  Standard   SET389
1       Merchant Amazon.in Standard JNE3781 2 Amazon
        Amazon.in Expedited JNE3371
3 Merchant Amazon.in Standard J0341 4 Amazon
Amazon.in Expedited JNE3671
 …          …          …                …      …
112452    Amazon Amazon.in  Expedited  J0372
112453    Amazon Amazon.in  Expedited  SET203
112454    Amazon Amazon.in  Expedited JNE3294
112455    Amazon Amazon.in  Expedited JNE3674
112456    Merchant    Amaz  NaN   NaN
```

```
                   SKU     Category … currency  Amount   ship-city \
0 SET389-KR-NP-S Set … INR 647.62 MUMBAI 1 JNE3781-KR-XXXL kurta
… INR 406.00 BENGALURU 2 JNE3371-KR-XL kurta … INR 329.00 NAVI
MUMBAI
3            J0341-DR-L Western Dress …     INR   753.33
                PUDUCHERRY
4            JNE3671-TU-XXXL     Top … INR   574.00     CHENNAI
…                 …           … …      …      …          …
112452     J0372-SKD-XXL    Set … INR 1044.00      SIMAR SIR
112453     SET203-KR-DPT-XXL      Set … INR   429.00     SIMAR SIR
112454     JNE3294-KR-XS    kurta …     INR   432.00     BENGALURU
           112455      JNE3674-TU-XXL   Top … INR   574.00     MALUR
112456              NaN           NaN …      NaN     NaN         NaN

         ship-state ship-postal-code ship-country \
0         MAHARASHTRA  400081.0    IN
1         KARNATAKA    560085.0    IN
2         MAHARASHTRA  410210.0    IN
3         PUDUCHERRY   605008.0    IN
4         TAMIL NADU   600073.0    IN
…              …            …          …
112452         Gujarat 362255.0    IN
112453         Gujarat 362255.0    IN
112454         KARNATAKA    560046.0    IN
112455         KARNATAKA    563130.0    IN
112456         NaN     NaN   NaN

                                        promotion-ids   B2B \
0                                                NaN False
1                                                Amazon PLCC Free-
                                                 Financing Universal
                                                 Merchant … False
2                                                IN Core Free Shipping
                                                 2015/04/08 23-48-5-108
                                                   True
3                                                NaN False
4                                                NaN False
…                                                …    …
112452                                           IN Core Free Shipping
                                                 2015/04/08 23-48-5-108
                                                 False
112453                                           IN Core Free Shipping
                                                 2015/04/08 23-48-5-108
                                                 False
```

```
112454                                                IN Core Free Shipping
                                                      2015/04/08 23-48-5-108
                                                      False
112455                                                IN Core Free Shipping
                                                      2015/04/08 23-48-5-108
                                                      False
112456                                                NaN     NaN

        fulfilled-by Unnamed: 22
0         Easy Ship    NaN
1         Easy Ship    NaN
2              NaN     NaN
3         Easy Ship    NaN
4              NaN     NaN
…               …        …
112452         NaN    False
112453         NaN    False
112454         NaN    False
112455         NaN    False 112456     NaN   NaN

[112457 rows x 24 columns]
```

[50]:
```
# check the information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112457 entries, 0 to
112456 Data columns (total 25
columns):
#   Column            Non-Null Count Dtype
--- ------            -------------- -----
0   index             112457 non-null int64
1   Order ID          112457 non-null object
2   Date              112457             non-null
                      datetime64[ns]
3   Status            112457 non-null object
4   Fulfilment        112457 non-null object
5   Sales Channel     112457 non-null object
 6 ship-service-level 112456 non-null object
 7  Style             112456 non-null
                      object
 8  SKU               112456 non-null
                      object
 9  Category          112456 non-null
                      object
 10 Size              112456 non-null
                      object
```

```
 11 ASIN              112456 non-null
                      object
 12 Courier Status    106287 non-null
                      object
 13 Qty               112456       non-null
                      float64
 14 currency          105629 non-null
                      object
 15 Amount            105629       non-null
                      float64
 16 ship-city         112429 non-null
                      object
 17 ship-state        112429 non-null
                      object
 18 ship-postal-code  112429       non-null
                      float64
 19 ship-country      112429 non-null
                      object
 20 promotion-ids     70351 non-null object
 21 B2B               112456 non-null
                      object
 22 fulfilled-by      35274 non-null object
 23 Unnamed: 22       63406 non-null object
 24 month             112457 non-null
                      int32
dtypes: datetime64[ns](1), float64(3), int32(1), int64(1),
object(19) memory usage: 21.0+ MB
```

[51]:
```
# show the first 5 data
df.head(5)
```

[51]:
```
   index            Order ID     Date                       Status \
0      0 405-8078784-5731545 2022-04-30Cancelled
1      1 171-9198151-1101146 2022-04-30 Shipped - Delivered to Buyer
2      2 404-0687676-7273146 2022-04-30Shipped 3  3 403-9615377-
      8133951 2022-04-30   Cancelled
4      4 407-1069790-7240320 2022-04-30Shipped

  Fulfilment Sales Channel ship-service-level Style            SKU \
 0    Merchant   Amazon.in  Standard   SET389     SET389-KR-NP-S
 1    Merchant Amazon.in Standard JNE3781 JNE3781-KR-XXXL 2 Amazon
 Amazon.in Expedited JNE3371 JNE3371-KR-XL 3 Merchant Amazon.in
 Standard J0341 J0341-DR-L
 4    Amazon    Amazon.in       Expedited JNE3671 JNE3671-TU-XXXL

        Category … Amount   ship-city ship-state ship-postal-code \
0           Set … 647.62    MUMBAI MAHARASHTRA    400081.0
1           kurta … 406.00  BENGALURU  KARNATAKA  560085.0
```

```
2          kurta … 329.00 NAVI MUMBAI MAHARASHTRA     410210.0
3          Western Dress … 753.33     PUDUCHERRY PUDUCHERRY 605008.0
4          Top … 574.00    CHENNAI    TAMIL NADU 600073.0

  ship-country                                promotion-ids   B2B \
0           IN    NaN False
1           IN Amazon PLCC Free-Financing Universal Merchant … False
2           IN    IN Core Free Shipping 2015/04/08 23-48-5-108     True
3           IN    NaN False
4           IN    NaN False

   fulfilled-by Unnamed: 22 month
0 Easy Ship NaN                                                    3
1 Easy Ship NaN                                                    3
2 NaN NaN                                                          3
3 Easy Ship NaN                                                    3
4 NaN NaN                                                          3

5     rows x 25 columns]
```

[52]:
```
# shape of dataset
df.shape
```

[52]: (112457, 25)

[55]:
```
# check the null data from dataset
df.isnull().sum()
```

[55]:
```
index                0
Order ID             0
Date                 0
Status               0
Fulfilment           0
Sales Channel        0
ship-service-level   0
Style                0
SKU                  0
Category             0
Size                 0
```

```
ASIN                0
Courier Status      0
Qty                 0
currency            0
Amount              0
ship-city           0
ship-state          0
ship-postal-code    0
ship-country        0
promotion-ids       0
B2B                 0
fulfilled-by        0
month               0
dtype: int64
```

[54]:
```
# remove all the null value from dataset
df["Courier Status"].fillna("Not
conformd",inplace=True)
df["currency"].fillna("INR",inplace=True)
df["Amount"].fillna(0,inplace=True) df["ship-
city"].fillna("No_city",inplace=True) df["ship-
state"].fillna("No_state",inplace=True)
df.dropna(subset=["ship-postal-
code"],inplace=True) df["ship-
country"].fillna("IN",inplace=True)
df["promotion-
ids"].fillna("Not_prometed",inplace=True)
df["fulfilled-by"].fillna("Easy
Ship",inplace=True) df.drop(columns=["Unnamed:
22"],inplace=True )
```

```
<ipython-input-54-8c4da5a7702e>:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using
```

```
      'df.method({col: value}, inplace=True)' or df[col] =
                    df[col].method(value)
instead, to perform the operation inplace on the original object.


```

```
  df["Courier Status"].fillna("Not conformd",inplace=True)
```

```
<ipython-input-54-8c4da5a7702e>:3: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["currency"].fillna("INR",inplace=True)
```

```
<ipython-input-54-8c4da5a7702e>:4: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["Amount"].fillna(0,inplace=True)
```

```
<ipython-input-54-8c4da5a7702e>:5: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["ship-city"].fillna("No_city",inplace=True)
<ipython-input-54-8c4da5a7702e>:6: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].fillna("No_state",inplace=True)
<ipython-input-54-8c4da5a7702e>:8: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-country"].fillna("IN",inplace=True)
<ipython-input-54-8c4da5a7702e>:9: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["promotion-ids"].fillna("Not_prometed",inplace=True)
<ipython-input-54-8c4da5a7702e>:10: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["fulfilled-by"].fillna("Easy Ship",inplace=True)
```

1 check number of delivered or non_delivered data

[63]:
```
# check the unique value in Status column
df["Status"].unique()
```

[63]:
```
array(['Cancelled', 'Shipped - Delivered to Buyer', 'Shipped',
       'Shipped - Returned to Seller', 'Shipped - Rejected by Buyer',
    'Shipped - Lost in Transit', 'Shipped - Out for Delivery',
       'Shipped - Returning to Seller', 'Shipped - Picked Up',
       'Pending',
       'Pending - Waiting for Pick Up', 'Shipped - Damaged',
      'Shipping'], dtype=object)
```

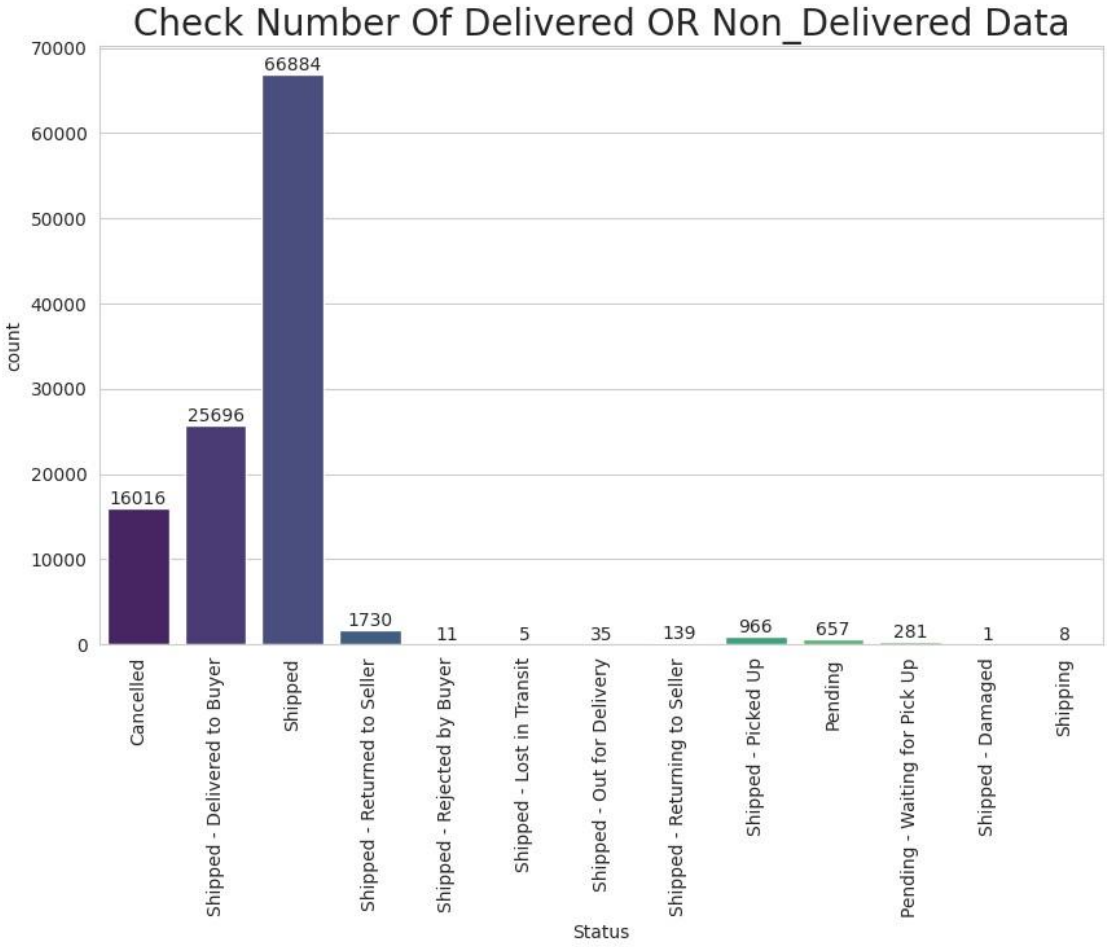
[64]:
```
 # replace the same meaning value df["Status"].replace("Returning
to Seller","Returned to Seller",inplace=True)
```

```
<ipython-input-64-fc93549185c1>:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


              df["Status"].replace("Returning to Seller","Returned to
                                                   Seller",inplace=True)
```

[65]:
```
 # create a plot
sns.set_style(style="whitegrid")
plt.figure(figsize=(10,6)) plt.title("Check Number Of
```

```
 Delivered OR Non_Delivered Data",size=20)
 plt.xticks(rotation=90)
 ax=sns.countplot(data=df,x="Status",palette="v
 iridis") for bars in ax.containers:
 ax.bar_label(bars)
```

```
<ipython-input-65-8d0e3aeed314>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  ax=sns.countplot(data=df,x="Status",palette="viridis")
```



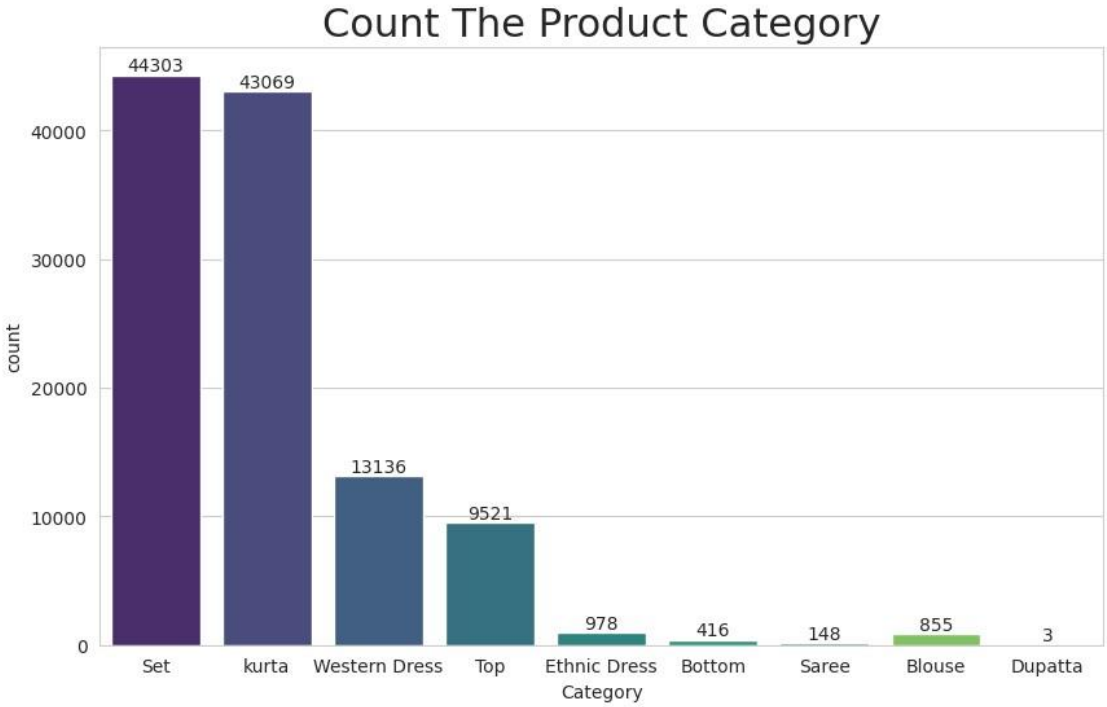
2 check the Category of the product

[66]:
```
# check the unique value in column Category
df["Category"].unique()
```

[66]:
```
array(['Set', 'kurta', 'Western Dress', 'Top', 'Ethnic Dress',
       'Bottom', 'Saree', 'Blouse', 'Dupatta'], dtype=object)
```

[67]:
```
# create a plot
plt.figure(figsize=(10,6))
plt.title("Count The Product Category",size=22)
ax=sns.countplot(data=df,x="Category",palette="viridis")
for bars in ax.containers:
  ax.bar_label(bars)
```

```
<ipython-input-67-512a7a4030f2>:4: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
ax=sns.countplot(data=df,x="Category",palette="viridis")
```



In Category the most product are sale:- SET and The lest product are sale are :- DUPATTA

3 check the sales of states

[68]:
```
# check the unique vqlue of column ship-state
df["ship-state"].unique()
```

[68]:
```
array(['MAHARASHTRA', 'KARNATAKA', 'PUDUCHERRY', 'TAMIL NADU',
```

```
       'UTTAR PRADESH', 'CHANDIGARH', 'TELANGANA', 'ANDHRA PRADESH',
       'RAJASTHAN', 'DELHI', 'HARYANA', 'ASSAM', 'JHARKHAND',
       'CHHATTISGARH', 'ODISHA', 'KERALA', 'MADHYA PRADESH',
       'WEST BENGAL', 'NAGALAND', 'Gujarat', 'UTTARAKHAND', 'BIHAR',
       'JAMMU & KASHMIR', 'PUNJAB', 'HIMACHAL PRADESH',
       'ARUNACHAL PRADESH', 'MANIPUR', 'Goa', 'MEGHALAYA', 'GOA',
       'TRIPURA', 'LADAKH', 'DADRA AND NAGAR', 'SIKKIM', 'Delhi',
       'ANDAMAN & NICOBAR', 'Punjab', 'Rajshthan', 'Manipur',
       'rajasthan',
       'Odisha', 'NL', 'Bihar', 'MIZORAM', 'punjab', 'New Delhi',
       'Rajasthan', 'Punjab/Mohali/Zirakpur', 'Puducherry', 'delhi',
       'RJ',
       'Chandigarh', 'orissa', 'LAKSHADWEEP', 'goa', 'PB', 'APO',
    'Arunachal Pradesh', 'AR', 'Pondicherry', 'Sikkim',
       'Arunachal pradesh', 'Nagaland', 'bihar', 'Mizoram',
      'rajsthan'], dtype=object)
```

[75]:
```
# count the value
df["ship-state"].value_counts()
```

[75]:
```
ship-state

MAHARASHTRA       19429
KARNATAKA         14930
TAMIL NADU         9863
TELANGANA          9768
UTTAR PRADESH      9304
DELHI              6165
KERALA             5780
WEST BENGAL        5344
ANDHRA PRADESH     4690
GUJARAT            4020
HARYANA            3896
RAJASTHAN          2379
MADHYA PRADESH     2194
BIHAR              1868
ODISHA             1843
PUNJAB             1702
ASSAM              1470
UTTARAKHAND        1386
JHARKHAND          1321
GOA                 998
CHHATTISGARH        810
HIMACHAL PRADESH    686
JAMMU & KASHMIR     597
PUDUCHERRY          310
MANIPUR             291
CHANDIGARH          288
```

```
ANDAMAN & NICOBAR  223
MEGHALAYA          185
NAGALAND           171
SIKKIM             157
TRIPURA            132
ARUNACHAL PRADESH  129
MIZORAM             62
LADAKH              34
LAKSHADWEEP          3
APO                  1
Name: count, dtype: int64
```

[76]:
```
# rep;lace all the same value
df["ship-state"].replace("Delhi","DELHI",inplace=True) df["ship-
state"].replace("delhi","DELHI",inplace=True) df["ship-
state"].replace("New Delhi","DELHI",inplace=True) df["ship-
state"].replace("bihar","BIHAR",inplace=True) df["ship-
state"].replace("Bihar","BIHAR",inplace=True) df["ship-
state"].replace("Gujarat","GUJARAT",inplace=True) df["ship-
state"].replace("Gujarat","GUJARAT",inplace=True) df["ship-
state"].replace("Gujarat","GUJARAT",inplace=True)

df["ship-state"].replace("Goa","GOA",inplace=True) df["ship-
state"].replace("goa","GOA",inplace=True) df["ship-
state"].replace("punjab","PUNJAB",inplace=True) df["ship-
state"].replace("PB","PUNJAB",inplace=True) df["ship-
state"].replace("Punjab","PUNJAB",inplace=True)
df["ship-
state"].replace("Punjab/Mohali/Zirakpur","PUNJAB",inplace=True)
df["ship-state"].replace("Rajsthan","RAJASTHAN",inplace=True)
df["ship-state"].replace("rajsthan","RAJASTHAN",inplace=True)
df["ship-state"].replace("Rajshthan","RAJASTHAN",inplace=True)
df["ship-state"].replace("rajasthan","RAJASTHAN",inplace=True)
df["ship-state"].replace("Rajasthan","RAJASTHAN",inplace=True)
df["ship-state"].replace("RJ","RAJASTHAN",inplace=True)


df["ship-state"].replace("Orissa","ODISHA",inplace=True) df["ship-
state"].replace("orissa","ODISHA",inplace=True) df["ship-
state"].replace("Odisha","ODISHA",inplace=True) df["ship-
state"].replace("Mizoram","MIZORAM",inplace=True) df["ship-
state"].replace("Sikkim","SIKKIM",inplace=True) df["ship-
state"].replace("Manipur","MANIPUR",inplace=True) df["ship-
state"].replace("Chandigarh","CHANDIGARH",inplace=True ) df["ship-
state"].replace("DADRA AND NAGAR","Gujarat",inplace=True) df["ship-
state"].replace("Arunachal Pradesh","ARUNACHAL PRADESH",inplace=True)
df["ship-state"].replace("Arunachal pradesh","ARUNACHAL
```

```
PRADESH",inplace=True) df["ship-state"].replace("APO","ARUNACHAL
PRADESH",inplace=True)

df["ship-state"].replace("Nagaland","NAGALAND",inplace=True) df["ship-
state"].replace("NL","NAGALAND",inplace=True) df["ship-
state"].replace("Pondicherry","PUDUCHERRY",inplace=True) df["ship-
state"].replace("Puducherry","PUDUCHERRY",inplace=True) df["ship-
state"].replace("AR","ARUNACHAL PRADESH",inplace=True) df["ship-
state"].replace("Meghalaya","MEGHALAYA",inplace=True)
```

```
<ipython-input-76-5f5f85c8a046>:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
 The behavior will change in pandas 3.0. This inplace method will
                                never work
because the intermediate object on which we are setting values always
behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["ship-state"].replace("Delhi","DELHI",inplace=True)
<ipython-input-76-5f5f85c8a046>:3: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["ship-state"].replace("delhi","DELHI",inplace=True)
<ipython-input-76-5f5f85c8a046>:4: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("New Delhi","DELHI",inplace=True)
<ipython-input-76-5f5f85c8a046>:5: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("bihar","BIHAR",inplace=True) <ipython-
  input-76-5f5f85c8a046>:6: FutureWarning: A value is trying to be
  set on a copy of a DataFrame or Series through chained assignment
  using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Bihar","BIHAR",inplace=True)
<ipython-input-76-5f5f85c8a046>:7: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["ship-state"].replace("Gujarat","GUJARAT",inplace=True)
<ipython-input-76-5f5f85c8a046>:11: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Goa","GOA",inplace=True)
<ipython-input-76-5f5f85c8a046>:12: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using
'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("goa","GOA",inplace=True)
<ipython-input-76-5f5f85c8a046>:13: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("punjab","PUNJAB",inplace=True)
<ipython-input-76-5f5f85c8a046>:14: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("PB","PUNJAB",inplace=True)
<ipython-input-76-5f5f85c8a046>:15: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Punjab","PUNJAB",inplace=True)
<ipython-input-76-5f5f85c8a046>:16: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-
  state"].replace("Punjab/Mohali/Zirakpur","PUNJAB",inplace=True)
<ipython-input-76-5f5f85c8a046>:17: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


```

```
  df["ship-state"].replace("Rajsthan","RAJASTHAN",inplace=True)
<ipython-input-76-5f5f85c8a046>:18: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


```

```
  df["ship-state"].replace("rajsthan","RAJASTHAN",inplace=True)
<ipython-input-76-5f5f85c8a046>:19: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


```

```
  df["ship-state"].replace("Rajshthan","RAJASTHAN",inplace=True)
<ipython-input-76-5f5f85c8a046>:20: FutureWarning: A value is trying
to be set
on a copy of a DataFrame or Series through chained assignment using
an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["ship-state"].replace("rajasthan","RAJASTHAN",inplace=True)
<ipython-input-76-5f5f85c8a046>:21: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Rajasthan","RAJASTHAN",inplace=True)
<ipython-input-76-5f5f85c8a046>:22: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("RJ","RAJASTHAN",inplace=True)
<ipython-input-76-5f5f85c8a046>:25: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object. df["ship-
state"].replace("Orissa","ODISHA",inplace=True)
<ipython-input-76-5f5f85c8a046>:26: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


```

```
  df["ship-state"].replace("orissa","ODISHA",inplace=True)
<ipython-input-76-5f5f85c8a046>:27: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


```

```
  df["ship-state"].replace("Odisha","ODISHA",inplace=True)
<ipython-input-76-5f5f85c8a046>:28: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


```

```
  df["ship-state"].replace("Mizoram","MIZORAM",inplace=True)
<ipython-input-76-5f5f85c8a046>:29: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.
```

```
  df["ship-state"].replace("Sikkim","SIKKIM",inplace=True)
<ipython-input-76-5f5f85c8a046>:30: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Manipur","MANIPUR",inplace=True)
<ipython-input-76-5f5f85c8a046>:31: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Chandigarh","CHANDIGARH",inplace=True )
<ipython-input-76-5f5f85c8a046>:32: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("DADRA AND NAGAR","Gujarat",inplace=True)
<ipython-input-76-5f5f85c8a046>:33: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
```

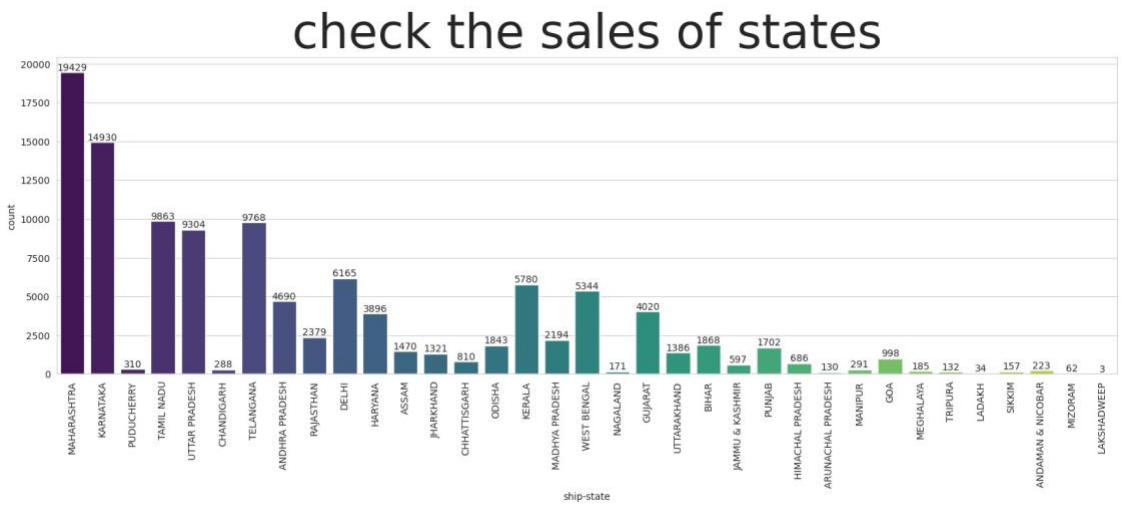
```
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Arunachal Pradesh","ARUNACHAL
PRADESH",inplace=True) <ipython-input-76-5f5f85c8a046>:34:
FutureWarning: A value is trying to be set on a copy of a DataFrame
or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Arunachal pradesh","ARUNACHAL
PRADESH",inplace=True) <ipython-input-76-5f5f85c8a046>:35:
FutureWarning: A value is trying to be set on a copy of a DataFrame
or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("APO","ARUNACHAL PRADESH",inplace=True)
<ipython-input-76-5f5f85c8a046>:37: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object. df["ship-
state"].replace("Nagaland","NAGALAND",inplace=True)
<ipython-input-76-5f5f85c8a046>:38: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("NL","NAGALAND",inplace=True)
<ipython-input-76-5f5f85c8a046>:39: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Pondicherry","PUDUCHERRY",inplace=True)
<ipython-input-76-5f5f85c8a046>:40: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Puducherry","PUDUCHERRY",inplace=True)
```

```
<ipython-input-76-5f5f85c8a046>:41: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("AR","ARUNACHAL PRADESH",inplace=True)
<ipython-input-76-5f5f85c8a046>:42: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df["ship-state"].replace("Meghalaya","MEGHALAYA",inplace=True)
```

[77]:
```
# create a plot
plt.figure(figsize=(20,6))
plt.title("check the sales of states ",size=50)
plt.xticks(rotation=90)
ax=sns.countplot(data=df,x="ship-state",palette="viridis")

for bars in ax.containers:
  ax.bar_label(bars)
```

```
<ipython-input-77-2200c613a338>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  ax=sns.countplot(data=df,x="ship-state",palette="viridis")
```



The most saleable product in MAHARASHTRA and The lest product are sale in LAKSHADWEEP

4 count how many product is easly shipped

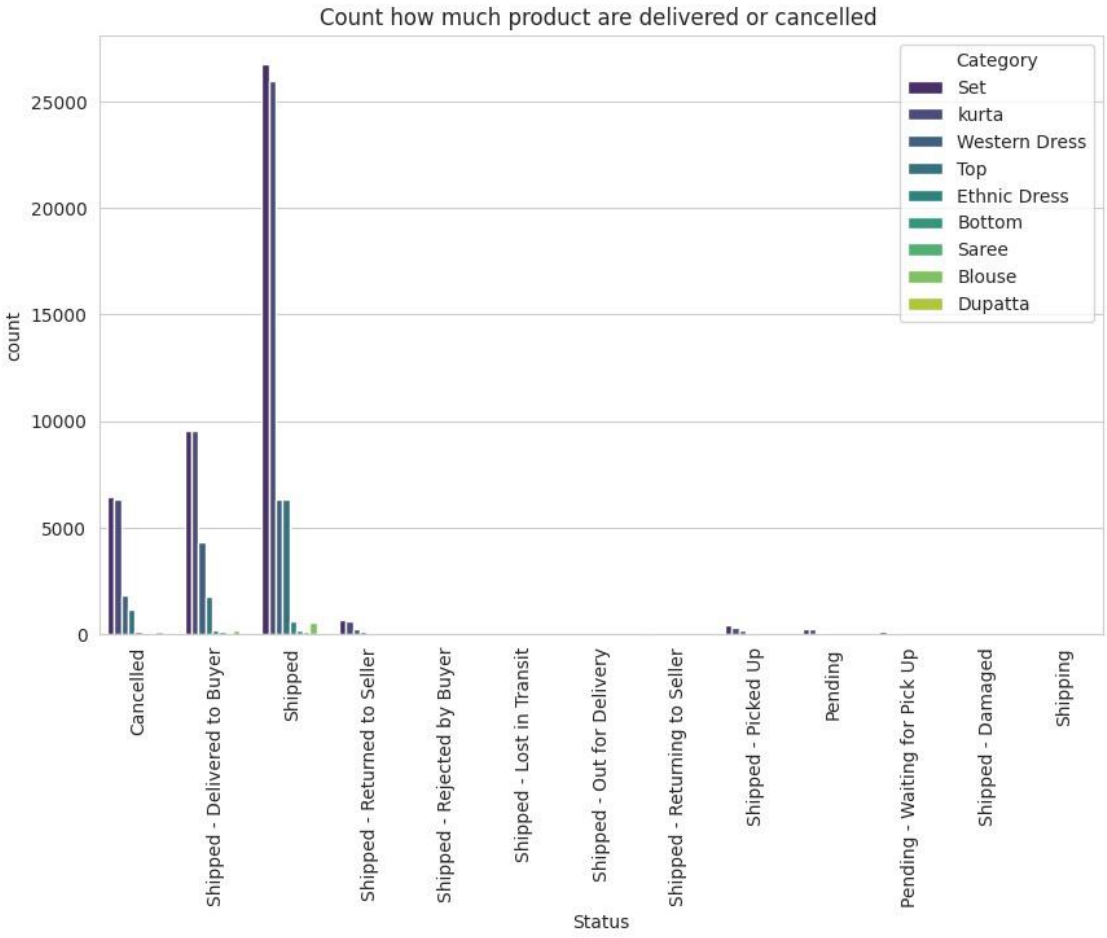
[78]:
```
# check the unique value of column fulfilled-by
df["fulfilled-by"].unique()
```

[78]:
```
array(['Easy Ship'], dtype=object)
```

[79]:
```
# create a plot
plt.title("count how many product is easly shipped  ")
ax=sns.countplot(data=df,x="fulfilled-by")
for bars in ax.containers:
  ax.bar_label(bars)
plt.show()
```

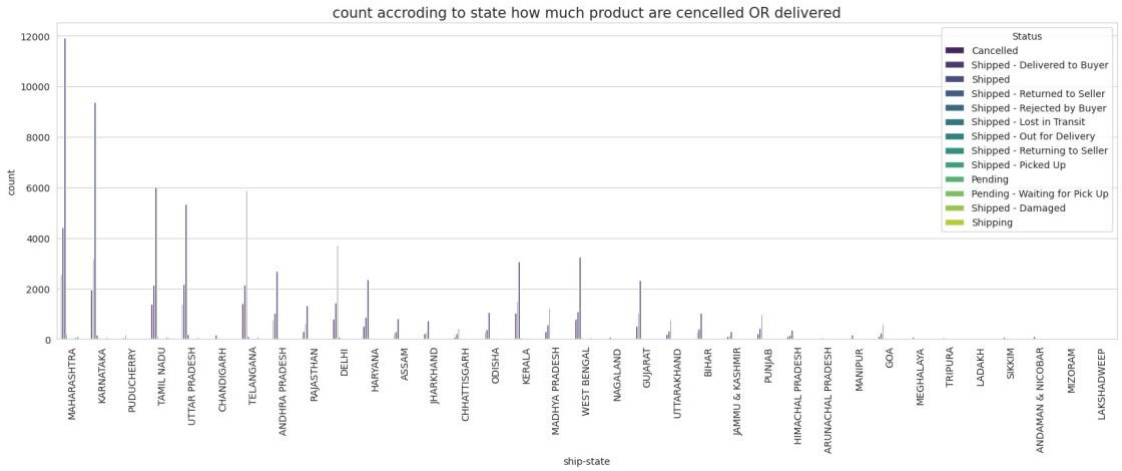
5 check the Category according to Status

[80]:
```
# craete a plot
plt.figure(figsize=(10,6))
plt.xticks(rotation=90)
plt.title("Count how much product are delivered or cancelled ")
sns.countplot(data=df,x="Status",hue="Category",palette="viridis"
) plt.show()
```



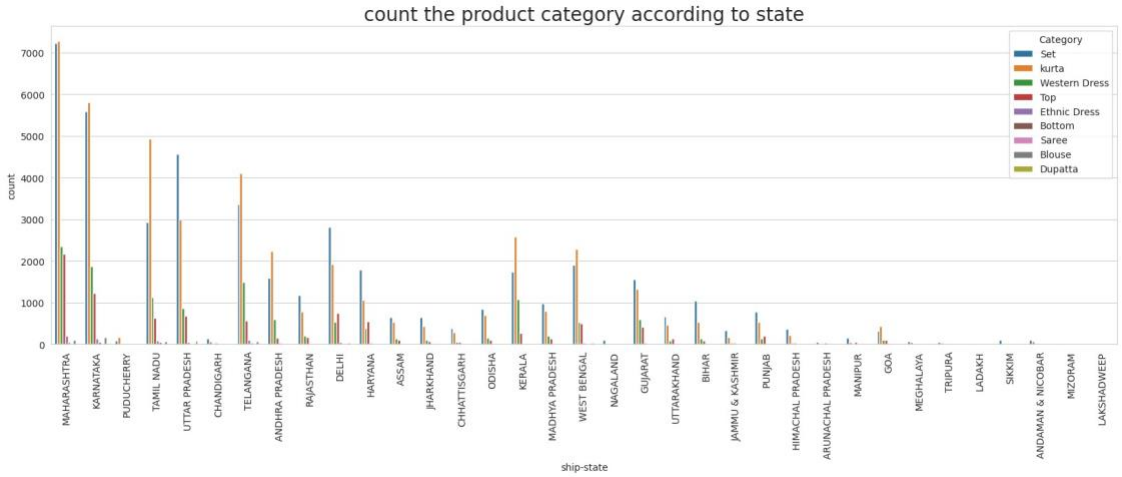
6 count the sales of product according to state

[81]:
```
# create a plot
plt.figure(figsize=(20,6)) plt.xticks(rotation=90)
plt.title("count accroding to state how much product are
cencelled OR␣
 ↪delivered",size=15) ax=sns.countplot(data=df,x="ship-
state",hue="Status",palette="viridis") plt.show()
```



7 count the sale of product according to state

[82]:
```
# create a plot
plt.figure(figsize=(20,6))
plt.xticks(rotation=90)
plt.title("count the product category according to state ",size=20)
sns.countplot(data=df,x="ship-state",hue="Category")
plt.savefig("count.jpg")
plt.show()
```



8 check the salles according to year and month

[83]:
```
df["Date"].dtype
```

[83]:
```
dtype('<M8[ns]')
```

[84]:
```
# convert the object data type into datatime datatype
df['Date'] = pd.to_datetime(df['Date'])

print(df.dtypes)
```

```
index                          int64
Order ID                      object
Date                  datetime64[ns]
Status                        object
Fulfilment object Sales Channel
      object ship-service-level
      object
Style                         object
SKU                           object
Category                      object
Size                          object
ASIN                          object
Courier Status                object
Qty   float64 currency object Amount
      float64 ship-city     object
ship-state object ship-postal-code
      float64 ship-country  object
promotion-ids    object
B2B   object fulfilled-by   object
month int32
dtype: object
```

[85]:
```
# create a new column for month
df["month"]=df["Date"].dt.month
```

[86]:
```
# count the month value
df["month"].value_counts()
```

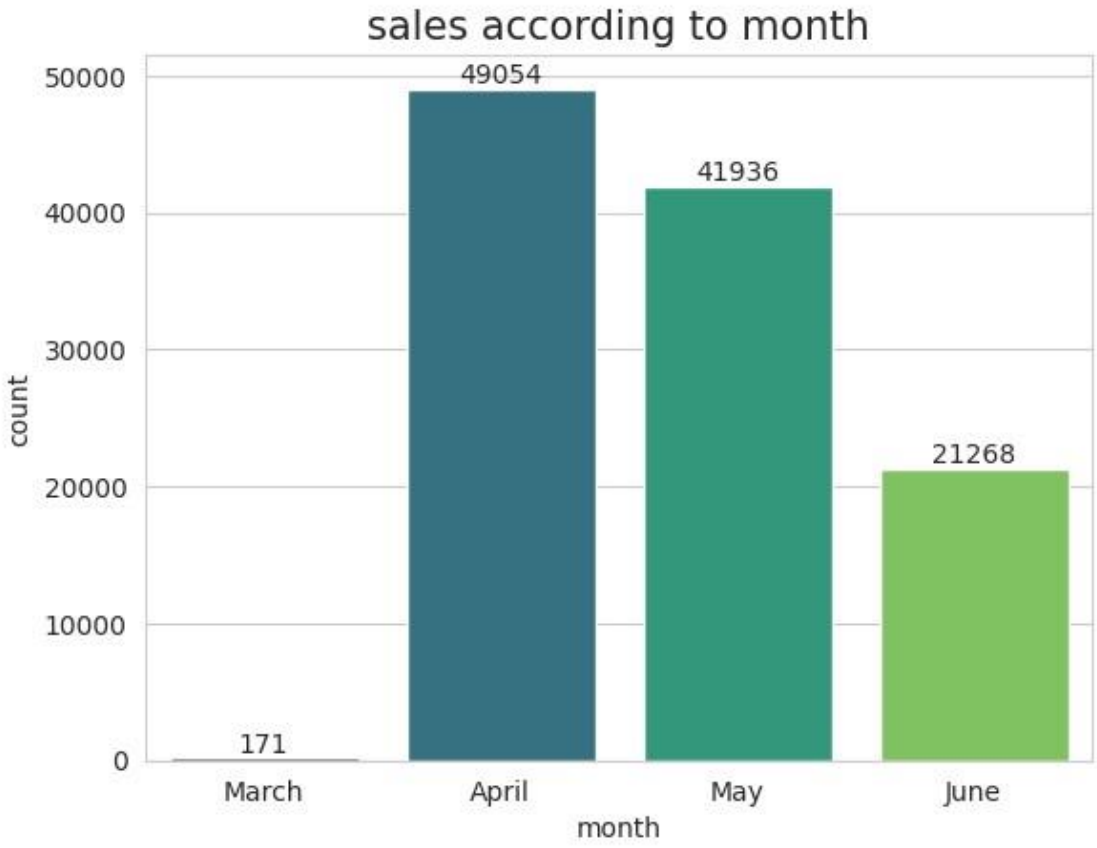
[86]:
```
month
4    49054
5    41936
6    21268
 3     171
Name: count, dtype: int64
```

[87]:
```
 # replace the month number by month name
df["month"].replace(3,"March",inplace=True)
df["month"].replace(4,"April",inplace=True)
df["month"].replace(5,"May",inplace=True)
df["month"].replace(6,"June",inplace=True)
```

```
<ipython-input-87-5afabe22b384>:2: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try

using 'df.method({col: value}, inplace=True)' or df[col] =

df[col].method(value) instead, to perform the operation inplace on

the original object. df["month"].replace(3,"March",inplace=True)
```

[89]:
```
# create a plot
plt.title("sales according to month ",size=15)
ax=sns.
 ↪countplot(data=df,x="month",palette="viridis",order=["March","April","May","June"])
for bars in ax.containers:
  ax.bar_label(bars)
```

```
<ipython-input-89-f072b1b62b1f>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  ax=sns.countplot(data=df,x="month",palette="viridis",order=["March"
,"April","M ay","June"])
```



8.1 sale According to amount

[90]:
```
df["Amount"]=df["Amount"].astype(int)
```

[91]:
```
grouped=df.groupby("Category")["Amount"].sum()
group=pd.DataFrame(grouped)
group
```

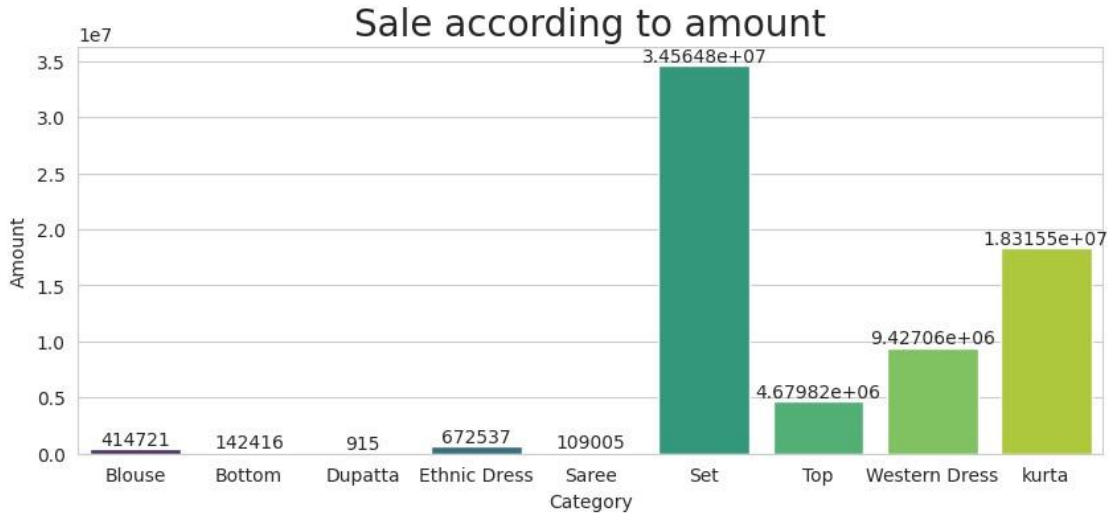
[91]:

Category	Amount
Blouse	414721
Bottom	142416
Dupatta	915
Ethnic	672537
Dress	
Saree	109005
Set	34564843
Top	4679816

```
Western Dress      9427060
kurta             18315545
```

[92]:
```
plt.figure(figsize=(10,4)) plt.title("Sale according to amount",size=20)
ax=sns.barplot(data=group,x="Category",y="Amount",palette="viridis") for bars in ax.containers: ax.bar_label(bars)
```

```
<ipython-input-92-8bc1bf5c23b0>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

ax=sns.barplot(data=group,x="Category",y="Amount",palette="viridis")
```



8.2 state sale accoriding to Amount with State

[93]:
```
group1=df.groupby("ship-state")["Amount"].sum()
g1=pd.DataFrame(group1)
g1
```

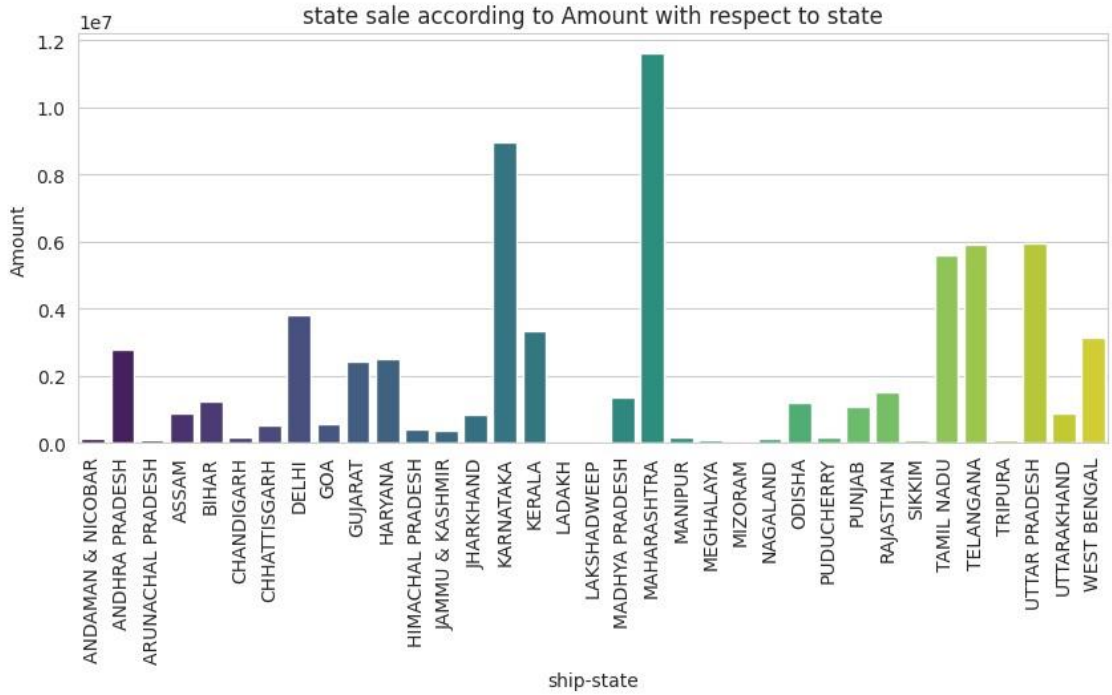
[93]:

ship-state	Amount
ANDAMAN & NICOBAR	137243
ANDHRA PRADESH	2773829
ARUNACHAL PRADESH	85469

```
 ASSAM             896092
 BIHAR            1251396
 CHANDIGARH        185958
 CHHATTISGARH      513903
DELHI             3824845
GOA                559415
GUJARAT           2448159
HARYANA           2512623
HIMACHAL PRADESH   433707
JAMMU & KASHMIR    383947
JHARKHAND          835102
KARNATAKA         8986754
KERALA            3361476
LADAKH              30467
LAKSHADWEEP          2330
MADHYA PRADESH    1378957
MAHARASHTRA      11615692
MANIPUR            194702
MEGHALAYA          107153
MIZORAM             33264
NAGALAND           133225
ODISHA            1190823
PUDUCHERRY         166542
PUNJAB            1074443
RAJASTHAN         1534563
SIKKIM             109424
TAMIL NADU        5590573
TELANGANA         5929461
TRIPURA             78406
UTTAR PRADESH     5959482
UTTARAKHAND        871876
WEST BENGAL       3135557
```

[95]:
```
plt.figure(figsize=(10,4))    plt.title("state    sale
according    to    Amount    with    respect    to    state")
plt.xticks(rotation=90)
sns.barplot(data=g1,x="ship-state",y="Amount",palette="viridis")
plt.show()
```

```
<ipython-input-95-4ce9055fb584>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=g1,x="ship-state",y="Amount",palette="viridis")
```



9 *Sale according to month **

[106]:
```
plt.figure(figsize=(10,4)) plt.title("count the Sale according to March",size=20)
ax=sns.countplot(x="Category",data=df[df["month"]=="March"],palette="viridis") for bars in ax.containers: ax.bar_label(bars)
```

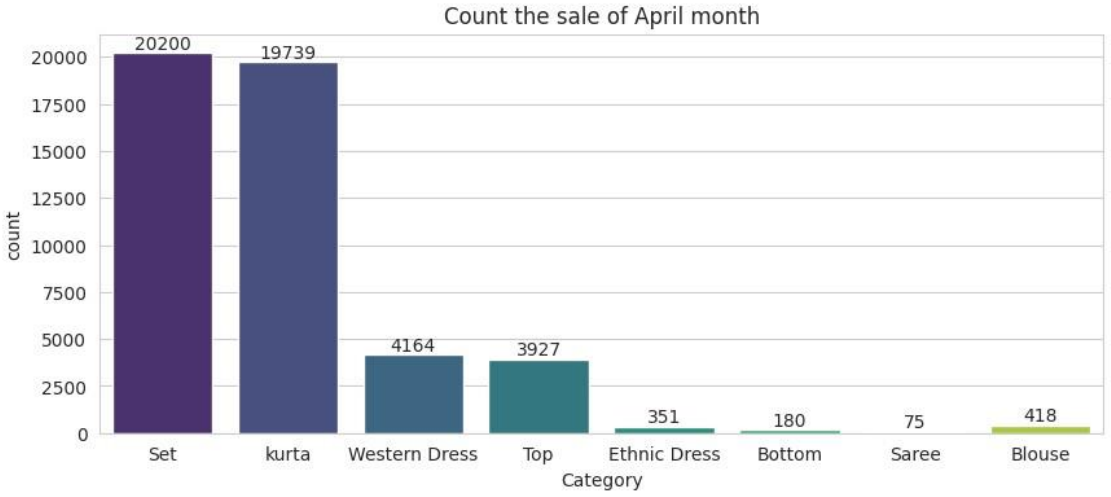
```
<ipython-input-106-197bf28aff18>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

```
ax=sns.countplot(x="Category",data=df[df["month"]=="March"],palette="viridis")
```

[117]:
```
  plt.figure(figsize=(10,4))  plt.title("Count  the  sale  of  April
month")
ax=sns.countplot(x="Category",data=df[df["month"]=="April"],palette
="viridis") for bars in ax.containers:
  ax.bar_label(bars,size=10)
```

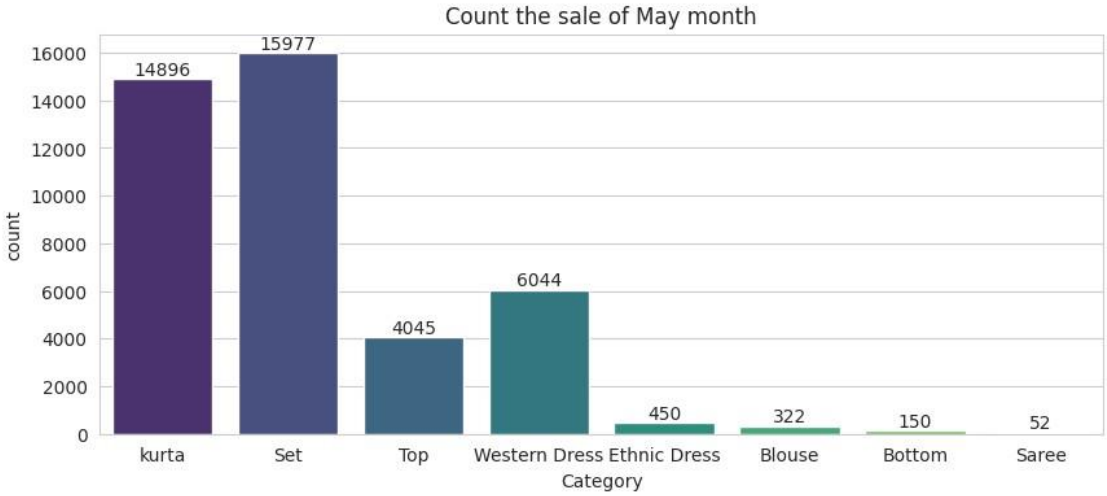
```
<ipython-input-117-1e5b8782c3f8>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
ax=sns.countplot(x="Category",data=df[df["month"]=="April"],palette="viridi
                                                                        s")
```



[125]:
```
plt.figure(figsize=(10,4)) plt.title("Count the sale of May month")
ax=sns.countplot(x="Category",data=df[df["month"]=="May"],palette
="viridis") for bars in ax.containers: ax.bar_label(bars)
```

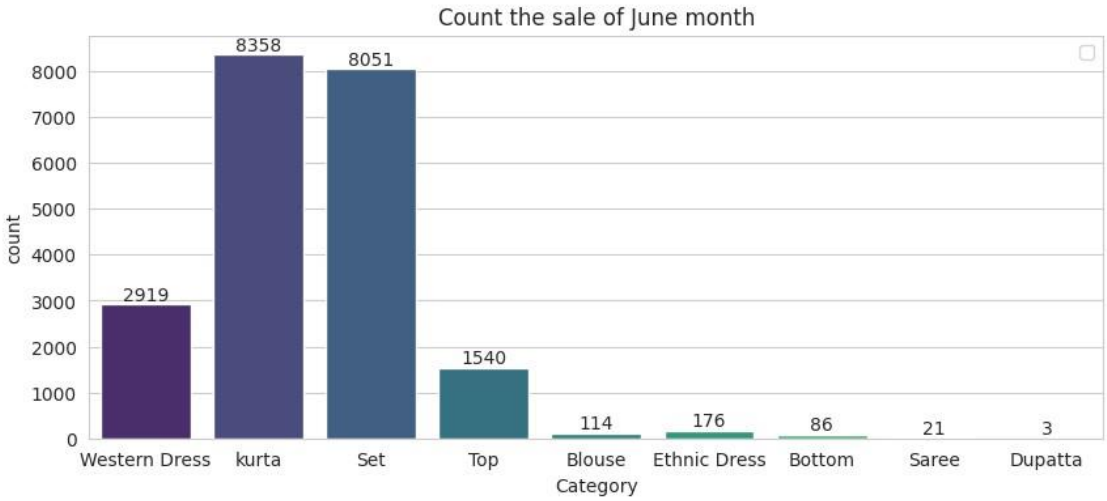
```
<ipython-input-125-aa1f5986de5c>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

ax=sns.countplot(x="Category",data=df[df["month"]=="May"],palette="vi
ridis")
```



[126]:
```
  plt.figure(figsize=(10,4))  plt.title("Count  the  sale  of  June
month")
ax=sns.countplot(x="Category",data=df[df["month"]=="June"],palette
="viridis") for bars in ax.containers: ax.bar_label(bars)
```

```
<ipython-input-126-6891e92f0d2e>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  ax=sns.countplot(x="Category",data=df[df["month"]=="June"],palette=
"viridis") <ipython-input-126-6891e92f0d2e>:6: UserWarning: No
artists with labels found to put in legend. Note that artists whose
label start with an underscore are ignored when legend() is called
with no argument. plt.legend()
```

[126]: <matplotlib.legend.Legend at 0x7f1c60b45510>



[129]:
```
df.head()
```

[129]:
```
   index            Order ID      Date                          Status \
0      0 405-8078784-5731545 2022-04-30Cancelled
1      1 171-9198151-1101146 2022-04-30 Shipped - Delivered to Buyer
2      2 404-0687676-7273146 2022-04-30Shipped 3  3 403-9615377-
       8133951 2022-04-30   Cancelled
4    4 407-1069790-7240320 2022-04-30                    Shipped

     Fulfilment Sales Channel ship-service-level Style            SKU \
0    Merchant   Amazon.in  Standard   SET389     SET389-KR-NP-S
1    Merchant Amazon.in Standard JNE3781 JNE3781-KR-XXXL 2 Amazon
Amazon.in Expedited JNE3371 JNE3371-KR-XL 3 Merchant Amazon.in
Standard J0341 J0341-DR-L
4    Amazon    Amazon.in       Expedited JNE3671 JNE3671-TU-XXXL

      Category … currency Amount  ship-city  ship-state \
0     Set … INR   647   MUMBAI MAHARASHTRA 1  kurta …
      INR   406   BENGALURU  KARNATAKA 2      kurta …
      INR   329 NAVI MUMBAI MAHARASHTRA
3          Western Dress … INR   753   PUDUCHERRY PUDUCHERRY
4          Top …      INR   574   CHENNAI    TAMIL NADU

  ship-postal-code ship-country \
0        400081.0      IN
1        560085.0      IN
2        410210.0      IN
3        605008.0      IN
```

```
4        600073.0     IN

                                  promotion-ids  B2B fulfilled-by \
0                                  Not_prometed False  Easy Ship
1                                  Amazon PLCC Free-Financing
                                   Universal Merchant … False     Easy
                                   Ship
2                                  IN Core Free Shipping 2015/04/08
                                   23-48-5-108   True  Easy Ship
3                                  Not_prometed False  Easy Ship
4                                  Not_prometed False  Easy Ship

   month
0  April
1  April
2  April
3  April
4  April

[5 rows x 24 columns]
```

[ ]: