```
#!/usr/bin/env python
# coding: utf-8

# In[5]:


import pandas as pd


# In[23]:


df = pd.read_csv(r"C:\Users\EBI\OneDrive\Documents\DATA ANALYSIS
PROJECT\INDIA BOOTCAMP\PYTHON\Zomato.csv")
df.head(2)


# In[24]:


df.dtypes


# In[25]:


df['rate'] = df['rate'].str.strip(r"/5")
df.head(2)


# In[6]:


# number of restaurant
len(df['name'].unique() )


# In[7]:


#How many restaurants fall under each listed_in(type) category?

df['listed_in(type)'].value_counts()


# In[11]:


df[['listed_in(type)', 'name']].head(7)


# In[26]:


# What is the average rating of restaurants that provide online ordering
vs. those that do not?
#rating.mean, df-name
#df[rate].mean()
```

```
#name_meanR = df['name'].

#convert rate to integer from object

df = df.astype({'rate': 'float'})
df.dtypes


# In[74]:


# What is the average rating of restaurants that provide online ordering
vs. those that do not?
df.groupby('online_order', )['rate'].mean()


# In[27]:


#What is the average cost (for two) of restaurants that allow table
booking vs. those that do not?df
#change column name
df.rename(columns = {'approx_cost(for two people)':'cost_per_2'},
inplace = True)
df.head(1)


# In[92]:


df.rename({'rate':'rating'}, axis=1,  inplace = True)


# In[94]:


df.head(1)


# In[95]:


#What is the average cost (for two) of restaurants that allow table
booking vs. those that do not?
df['cost_per_2'].mean()


# In[98]:


df.groupby('book_table')['cost_per_2'].mean()


# In[97]:


table_book
```

```
# In[28]:


#Find the restaurant(s) with the highest number of votes.
# votes .max
max_vote = df.nlargest(1, 'votes')[['name', 'votes']]
max_vote



# In[115]:


df['votes'].max()


# In[117]:


df.head(1)


# In[120]:


df.nlargest(3, 'cost_per_2')[['name', 'cost_per_2']]


# In[131]:


#rating#Find the restaurant(s) with the lowest rating but more than 200
votes.
#lowest rating = df.nsmallest(, 'rating')[['name', 'rating']]
# df[vote] > 200

lowest_rating = df[df['votes'] > 200].nsmallest(1, 'rating') [['name',
'votes', 'rating']]
lowest_rating


# In[31]:


lowest_rating = df[df['votes'] > 200][['name', 'votes']]
lowest_rating.head(1)


# In[34]:


# List all restaurants where the cost is above the overall ave
#avg cost = df[cost].mean()

avg_cost = df[df["cost_per_2"] > df["cost_per_2"].mean()][["name",
"cost_per_2"]]
avg_cost.head(10).sort_values(by="cost_per_2", ascending = True)
```

```
# In[3]:


df


# In[13]:


df.head(1)


# In[20]:


import pandas as pd


# In[42]:


# Find the most common cost value (mode) among all restaurants.
# df[cost_per_2].mode() and name

df['cost_per_2'].mode()


# In[57]:


#Create a new column rating_value (extract numeric value from rate).
Then, find the highest-rated restaurant under each listed_in(type).
#rating_value = rate
#df.nlargest(1, rate)
#df[listed_in(type)]
#df[["listed_in(type)", 'rate']]
df['rating_value'] = df['rate']
df.head(1)


# In[48]:


#Create a new column rating_value (extract numeric value from rate).
Then, find the highest-rated restaurant under each listed_in(type)
#group by - listed type
#rating_value : idxmax() to get index of max rating in each group
df.groupby ('listed_in(type)') ['rating_value'].idxmax()


# In[50]:


df['rating_value'].dtypes
```

```
# In[53]:


#use loc to fetch rows
highest_rated = df.loc[df.groupby ('listed_in(type)')
['rating_value'].idxmax()]
highest_rated [['listed_in(type)','rating_value']]


# In[63]:


df.loc[1, ['listed_in(type)','rating_value']].transpose


# In[69]:


#Find how many restaurants have a cost â‰¤500 and a rating â‰¥4.0.
#count of name
#cost <= 500
#rating >=4
name_cost_rate =df[(df['cost_per_2'] <= 500) & (df['rating_value']>=4) ]
name_cost_rate [['name', 'cost_per_2', 'rating_value']].head(5)


# In[97]:


# Find the restaurant(s) where cost is highest but rating is still below
3.5.
# Cost-df.nlargest(1, cost)
# df rating < 3.5
#df.nlargest(1, 'cost_per_2'). head(1)
df[df['rating_value']<3.5].nlargest(1, 'cost_per_2')[['name',
'cost_per_2', 'rating_value']]


# In[87]:


rate_=df[df['rating_value']<3.5].head(1)


# In[93]:


name_cost_rate= rate_.nlargest(1, 'cost_per_2')
name_cost_rate . head () [['name', 'cost_per_2', 'rating_value']]


# In[95]:


df.head(1)


# In[107]:
```

```
#Group restaurants by online_order and calculate: average votes, average
cost, and average rating.
# groupby online order
# average votes: votes.mean
# cost.mean
df['rate'].mean()
df['cost_per_2'].mean()
df['votes'].mean()
df.groupby('online_order'). mean (numeric_only= True)


# In[108]:


# Group restaurants by book_table and calculate the total number of
votes.
# df.groupby('book_table')
df.groupby('book_table')['votes'].sum (numeric_only= True)


# In[120]:


# Find the ratio of restaurants that allow both online_order and
book_table to total restaurants.
both_yes = df[(df['online_order'] == 'Yes') & (df['book_table']=='Yes'
)].shape[0]
both_yes


# In[118]:


denom = len(df['name'].unique())
denom


# In[121]:


ratio = both_yes/denom
ratio


# In[1]:


import pandas as pd


# In[2]:


df = pd.read_csv(r"C:\Users\EBI\OneDrive\Documents\DATA ANALYSIS
PROJECT\INDIA BOOTCAMP\PROJECT\orders.csv")
df.head(5)
```

```
# In[4]:


df.dtypes


# In[5]:


#change order date to date
#check for duplicates
# check null and na values
#create a sales column

df.info()


# In[31]:


df.isnull() . sum()


# In[50]:


df['Ship Mode'].value_counts()


# In[29]:


df.loc[110:118, ['Segment','Ship Mode']]


# In[44]:


x = df['Ship Mode'].mode()[0]
x


# In[84]:


df['Ship Mode'].fillna(x, inplace = True)


# In[85]:


df['Ship Mode'].isnull().sum()


# In[51]:
```

```
# In[52]:


df.isna().sum()


# In[59]:


#change date datatype from float to date
df["Order Date"]=pd.to_datetime (df["Order Date"], errors = 'coerce')


# In[60]:


df["Order Date"].dtypes


# In[61]:


df.info()


# In[69]:


df.duplicated().sum()


# In[68]:


df.head()


# In[71]:


df.describe()


# In[73]:


# add sales column = list price * qty * (1-disc/100)

df["Order Date"].isna().sum()


# In[74]:
```

```
df = pd.read_csv(r"C:\Users\EBI\OneDrive\Documents\DATA ANALYSIS
PROJECT\INDIA BOOTCAMP\PROJECT\orders.csv")
df.head(5)


# In[76]:


df["Order Date"] = pd.to_datetime(df["Order Date"],  errors = "coerce",
dayfirst = True)


# In[77]:


df.dtypes


# In[79]:


df["Order Date"] .isna().sum()


# In[81]:


x= df['Ship Mode'].mode() [0]
x


# In[86]:


df.isnull().sum()


# In[87]:


df.duplicated().sum()


# In[88]:


# add sales column = list price * qty * (1-disc/100)

df.head(3)


# In[90]:


# add sales column = list price * qty * (1-disc/100)
df['Sales'] = df['List Price'] * df['Quantity'] * (1- df['Discount
Percent']/100)
df.head(2)
```

```
# In[93]:


# 1.  What are the top 10 products by total sales revenue?
#       product id - sales . sort values (by = sales, ascending = false
top10= df[['Product Id','Sales']].sort_values(by = ['Sales'], ascending =
False)
top10.head(10)[['Product Id','Sales']]


# In[95]:


# 2.  Which products have the highest average order value?
 # avg order =  df[sales] .mean, nlargest(avg order, 1)
avg_order = df["Sales"] .mean()
avg_order


# In[104]:


avg_high_product = df.groupby("Product Id")["Sales"] .mean()
avg_high_product.head(10).sort_values( ascending = False)


# In[110]:


#3.   What is the overall total revenue per month?
# group by order date and sum revenue

sales_month = df.groupby(df["Order Date"].dt.strftime("%B %Y") )
["Sales"].sum()
sales_month


# In[ ]:




# In[ ]:
```