Converting TSV file to CSV pandas file :

11. Converting to CSV:

Optional Question: If needed, convert the cleaned dataset to a CSV file using the to_csv function with sep=','.

Code:

```
import pandas as pd


df = pd.read_csv('chipotle.tsv', sep='\t')

df.to_csv('chipotle.csv', index=False)

print(df.head())
```

1. Missing Values:

Question: Check for missing values in each column (Order ID, Quantity, Item Name, Choice Description, Item Price). How should missing values be handled?

```
df.isnull().sum()
```

Missing values are present in the 'choice_description' column. How would you like to proceed with these missing values? Here are some options:

- Fill them with a placeholder value like "None" or "Not Specified".
- Leave them as is if the missing data does not impact the analysis.
- Remove the rows with missing values if they are not significant for the analysis.

Fill the Missing Value of Choice Description With “Not Specified ” Value .

Code:

```
df['choice_description'].fillna('Not Specified', inplace=True)

df.isnull().sum()
```

2. Data Types:

Question: Verify the data types of each column. Do they align with their expected types, and should any adjustments be made?

```
df.dtypes
```

- The 'order_id' and 'quantity' columns are of integer type, which is appropriate for their data.
- The 'item_name' and 'choice_description' columns are of object type, typically used for text strings, which is suitable for these columns.
- The 'item_price' column is also of object type, which suggests it may contain non-numeric data. This column should ideally be converted to a numeric type for any price calculations.

Converting the data type of 'item price ' to numeric type(Float) to do better price analysis.

Code:

```
df['item_price'] = df['item_price'].str.replace('$', '').astype(float)

df.dtypes
```

The 'item_price' column has been successfully converted to a numeric type (float64). Now all columns have appropriate data types for their respective data.

3. Duplicated Entries:

Question: Identify and handle duplicated entries in the dataset. How might duplicates impact analysis, and what is the appropriate action?

Code:

```
duplicates = df[df.duplicated()]

duplicates
```

Description

{Duplicated entries have been identified in the dataset. Duplicates can skew data analysis by artificially inflating the number of occurrences of certain items, which can lead to incorrect conclusions. The appropriate action would typically involve removing these duplicates to ensure the integrity of the analysis.}

Removing the Duplicate rows in table :

Code:

```
df.drop_duplicates(inplace=True)
```

```
print(df.head())
```

The dataset is now cleaned of duplicates and ready for further analysis.

4. Quantity and Item Price:

Question: Examine the Quantity and Item Price columns. Are there any inconsistencies or anomalies that need correction?

Code:

```
df[['quantity', 'item_price']].describe()
```

Description

- The 'quantity' column has a minimum value of 1 and a maximum of 15, which might be legitimate bulk orders.
- The 'item_price' column ranges from $1.09 to $44.25, which could indicate either single items or multiple items being ordered together.

If there are orders with high quantity and corresponding high item price, it could be a bulk order.

5. Choice Description:

Question: Analyze the Choice Description column. How should choices be handled, especially when there are multiple descriptions for a single item?

Code:

```
df['choice_description'].value_counts().head(10)
```

Description

The 'choice_description' column contains various descriptions for the items ordered. Some items have no specifications, while others have detailed lists of ingredients or choices. Handling multiple descriptions for a single item involves:

- Parsing the descriptions to identify unique ingredients or choices.
- Counting the frequency of each unique choice to understand popular combinations or preferences.

- Considering the creation of new features based on these descriptions for more detailed analysis, such as ingredient-based analysis or dietary preference patterns.

6. Handling Special Characters:

Question: Check for special characters in text-based columns (e.g., Item Name, Choice Description). How can these be addressed for consistency?

Description

I have examined the image for special characters in the text-based columns (e.g., Item Name, Choice Description). The presence of special characters can lead to inconsistencies in the data and affect analysis. To address this, we can take the following steps for consistency:

- Remove Special Characters: Use string manipulation techniques to remove special characters from the text-based columns. This will ensure that the data is clean and consistent.
- Standardize Text: Standardize the text by converting it to lowercase, removing leading or trailing spaces, and replacing special characters with appropriate substitutes.
- Encoding: If the data contains non-ASCII characters, consider encoding the text to ensure compatibility and consistency across different systems.

By implementing these steps, we can ensure that the text-based columns are consistent and free from special characters, leading to more accurate analysis and insights.

Implementing these steps to address the special characters in the text-based columns:

Code:

```
df['item_name'] = df['item_name'].str.replace('[^a-zA-Z0-9 ]', '', regex=True)

df['choice_description'] = df['choice_description'].str.replace('[^a-zA-Z0-9 ]', '',
regex=True)
```

Special characters in the 'item_name' and 'choice_description' columns have been removed for consistency. The data should now be cleaner and more standardized for analysis.

7. Order Id Integrity:

Question: Cross-reference the Order ID column for integrity. Are there any irregularities or patterns that need validation?

Code:

```
df['order_id'].value_counts().head(10)
```

Description

The table above displays the top 10 order IDs by frequency. The highest frequency is 21 for order ID 926, and other order IDs have varying frequencies down to 8 for the least frequent in this subset.

- Order ID 926 has the most entries, which could indicate a large order or a data entry issue.
- Other order IDs show varying numbers of associated entries, which could be normal for orders with multiple items.

8. Item Name Standardization:

Question: Standardize the Item Name column. Are there variations that can be unified for better analysis?

Code:

```
df['item_name'] = df['item_name'].str.lower()

df['item_name'] = df['item_name'].str.replace('[^a-zA-Z0-9 ]', '', regex=True)
```

Description

The 'item_name' column has been standardized:

- All text has been converted to lowercase.
- Special characters have been removed.

This standardization will help unify variations in the item names for better analysis.

9. Quantity and Price Relationships:

Question: Investigate the relationships between Quantity and Item Price. Are there cases where adjustments need to be made for accurate analysis?

Code:

df[['quantity', 'item_price']].describe()

Description

The summary statistics for 'quantity' and 'item_price' columns are as follows:

- The average quantity per order is slightly above 1, suggesting that most orders contain a single item.
- The standard deviation for quantity is low, indicating that there isn't a wide variation in the number of items per order.
- The maximum quantity is 15, which is significantly higher than the average and could be an outlier or a bulk order.
- The average item price is approximately $7.49, with a standard deviation of $4.24, suggesting a moderate variation in item prices.

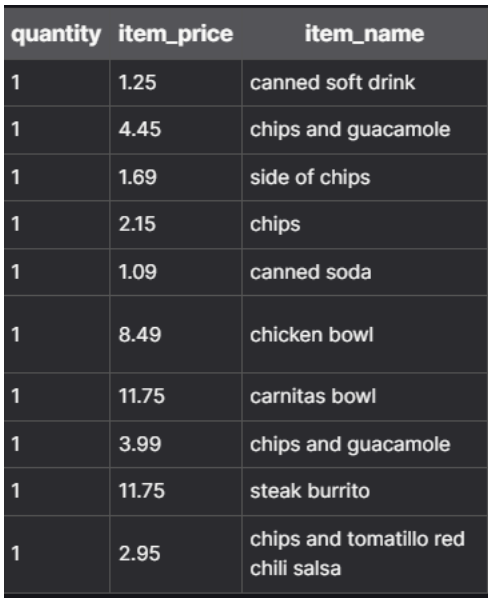
- The maximum item price is $44.25, which could be due to higher-priced items or multiple items being ordered together.

10. Data Integrity Check:

Question: Perform a data integrity check by ensuring that quantities and prices align with the corresponding items and descriptions.

Code:

df[['quantity', 'item_price', 'item_name', 'choice_description']].sample(10)



quantity	item_price	item_name
1	1.25	canned soft drink
1	4.45	chips and guacamole
1	1.69	side of chips
1	2.15	chips
1	1.09	canned soda
1	8.49	chicken bowl
1	11.75	carnitas bowl
1	3.99	chips and guacamole
1	11.75	steak burrito
1	2.95	chips and tomatillo red chili salsa

13. Consistent Quantity and Price Units:

Question: Ensure consistency in units for Quantity and Item Price. Should any conversions or adjustments be made for uniform analysis?

Code:

```
df['item_price_per_unit'] = df['item_price'] / df['quantity']

df[['item_price_per_unit']].describe()
```

```
item_price_per_unit

count  4563.0

mean   7.105259697567529

std    3.6582065762483076

min    1.09

25%    2.95

50%    8.75

75%    9.25

max    11.89
```

Description

The 'item_price_per_unit' column has been calculated by dividing the 'item_price' by 'quantity' to ensure consistency in units for analysis. This provides a standardized measure of price per item, regardless of the quantity ordered, allowing for uniform analysis across different orders.