RESTAURANT ORDERS DATA ANALYSIS

PROJECT DOCUMENTATION

AND

USER MANUAL

DATE: 24-01-2020

ANIKET GAIKWAD

Email ID: aniketng21p@gmail.com

CONTENTS

- **PROBLEM STATEMENT:**

Restaurant management has become a crucial task for the restaurant owners. The means of publicizing their products and services have encountered a drastic change in the past 15 years. In order to increase sales and gain more profit, the restaurant owners have to focus their attention on the following aspects:

- Location
- Smart Menu Planning
- Guest Experience
- Sales and Marketing
- Staff Management
- Customer Relation Management

In the work sample, we have the aim to find valuable insights from the obtained data and presenting them to the restaurant owner using different graphical representation.

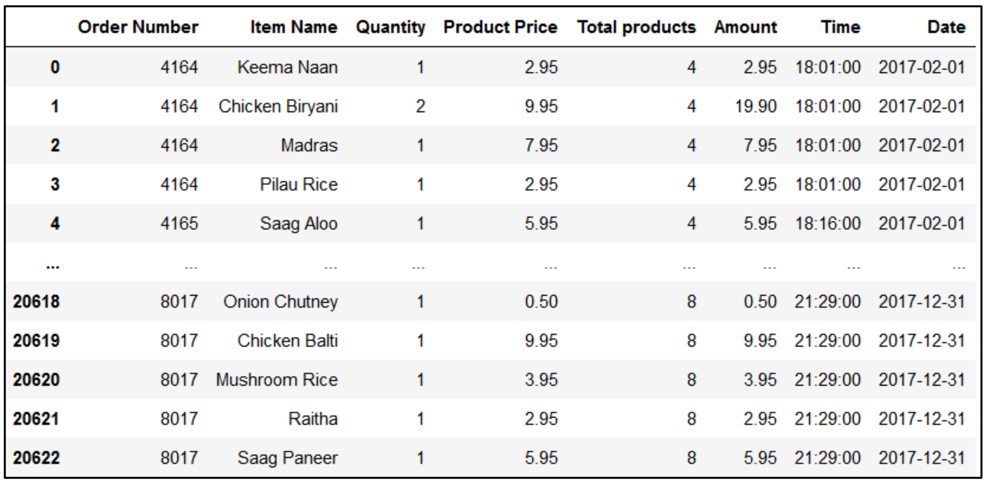
- **DATA SOURCE:**

The restaurant orders data set was obtained from kaggle. Here's the link to the dataset:

https://www.kaggle.com/henslersoftware/19560-indian-takeaway-orders

The Dataset contains the Order ID, Order Date, Food Item Names, Quantity and their Product Price.

DATASET VIEW:



	Order Number	Item Name	Quantity	Product Price	Total products	Amount	Time	Date
0	4164	Keema Naan	1	2.95	4	2.95	18:01:00	2017-02-01
1	4164	Chicken Biryani	2	9.95	4	19.90	18:01:00	2017-02-01
2	4164	Madras	1	7.95	4	7.95	18:01:00	2017-02-01
3	4164	Pilau Rice	1	2.95	4	2.95	18:01:00	2017-02-01
4	4165	Saag Aloo	1	5.95	4	5.95	18:16:00	2017-02-01
...	...	...	...	...	...	...	...	...
20618	8017	Onion Chutney	1	0.50	8	0.50	21:29:00	2017-12-31
20619	8017	Chicken Balti	1	9.95	8	9.95	21:29:00	2017-12-31
20620	8017	Mushroom Rice	1	3.95	8	3.95	21:29:00	2017-12-31
20621	8017	Raitha	1	2.95	8	2.95	21:29:00	2017-12-31
20622	8017	Saag Paneer	1	5.95	8	5.95	21:29:00	2017-12-31

- **<u>TOOLS USED</u>:**
    - **PROGRAMMING LANGUAGE:** Python 3.7
    - **WEB APPLICATION:** Jupyter Notebook

- **<u>DATA PREPROCESSING</u>:**

To find essential insights which will provide the restaurant owner with an in-depth view of his business, we have to generate some important features that are not provided in the raw dataset.

- **CLASSIFYING FOOD ITEMS:**

The dataset has 177 distinct food items. These food items are labelled into four different categories:

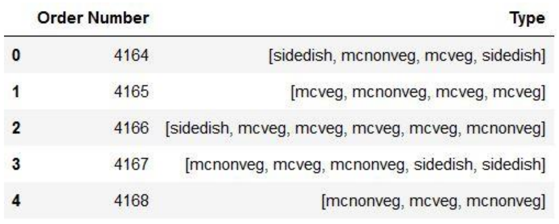
1. **MCVEG** (MAIN COURSE VEGETARIAN DISH)
2. **MCNONVEG** (MAIN COURSE NON-VEGETARIAN DISH)
3. **BEVERAGES**
4. **SIDEDISH** (STARTERS, BREADS, RICE, PULAO, RAITAS, BIRYANIS)

- **COMPUTING TOTAL ORDER AMOUNT AND LIST OF FOOD ITEMS IN A SPECIFIC ORDER:**

We use the 'Quantity' and 'Product Price' to compute the total order amount. We also combine all the food items from the same order and store it in a list.

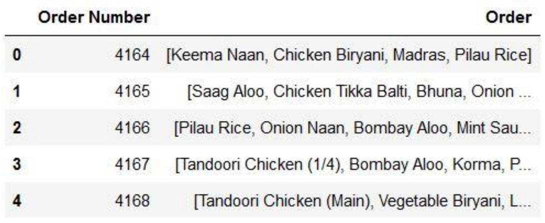
- **GENERATING DIFFERENT DATASETS WITH SPECIFIC COLUMNS:**

1. **ORDERTYPE:**



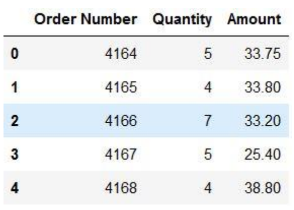
	Order Number	Type
0	4164	[sidedish, mcnonveg, mcveg, sidedish]
1	4165	[mcveg, mcnonveg, mcveg, mcveg]
2	4166	[sidedish, mcveg, mcveg, mcveg, mcveg, mcnonveg]
3	4167	[mcnonveg, mcveg, mcnonveg, sidedish, sidedish]
4	4168	[mcnonveg, mcveg, mcnonveg]

2. **ORDERDESC:**



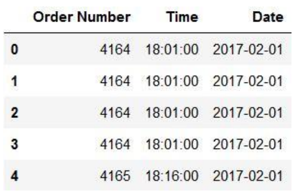
	Order Number	Order
0	4164	[Keema Naan, Chicken Biryani, Madras, Pilau Rice]
1	4165	[Saag Aloo, Chicken Tikka Balti, Bhuna, Onion ...
2	4166	[Pilau Rice, Onion Naan, Bombay Aloo, Mint Sau...
3	4167	[Tandoori Chicken (1/4), Bombay Aloo, Korma, P...
4	4168	[Tandoori Chicken (Main), Vegetable Biryani, L...

3. **QUANTAMT:**



	Order Number	Quantity	Amount
0	4164	5	33.75
1	4165	4	33.80
2	4166	7	33.20
3	4167	5	25.40
4	4168	4	38.80

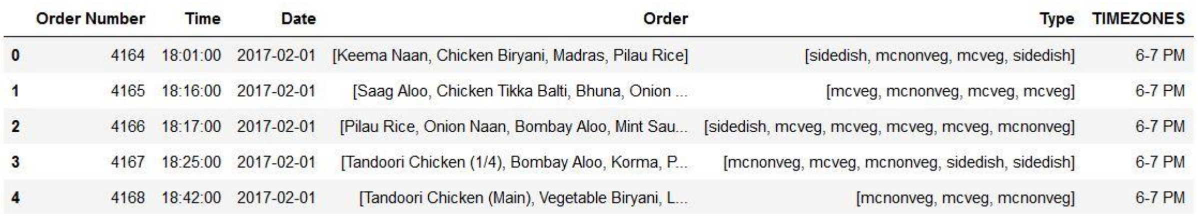
4. **ORDERDATE:**



	Order Number	Time	Date
0	4164	18:01:00	2017-02-01
1	4164	18:01:00	2017-02-01
2	4164	18:01:00	2017-02-01
3	4164	18:01:00	2017-02-01
4	4165	18:16:00	2017-02-01

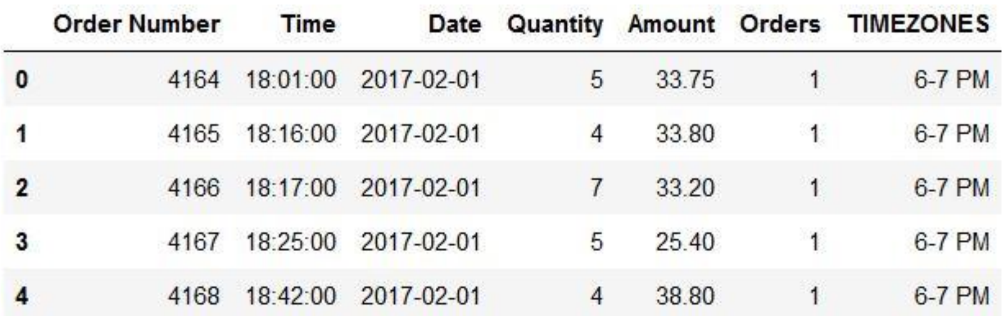
- **GENERATING TWO MASTER DATAFRAMES WITH DIFFERENT SETS OF COLUMNS:**

1. **COMPLETE_DATA:** Contains the order number, list of food items as well as their categories, time zone information.



	Order Number	Time	Date	Order	Type	TIMEZONES
0	4164	18:01:00	2017-02-01	[Keema Naan, Chicken Biryani, Madras, Pilau Rice]	[sidedish, mcnonveg, mcveg, sidedish]	6-7 PM
1	4165	18:16:00	2017-02-01	[Saag Aloo, Chicken Tikka Balti, Bhuna, Onion ...	[mcveg, mcnonveg, mcveg, mcveg]	6-7 PM
2	4166	18:17:00	2017-02-01	[Pilau Rice, Onion Naan, Bombay Aloo, Mint Sau...	[sidedish, mcveg, mcveg, mcveg, mcveg, mcnonveg]	6-7 PM
3	4167	18:25:00	2017-02-01	[Tandoori Chicken (1/4), Bombay Aloo, Korma, P...	[mcnonveg, mcveg, mcnonveg, sidedish, sidedish]	6-7 PM
4	4168	18:42:00	2017-02-01	[Tandoori Chicken (Main), Vegetable Biryani, L...	[mcnonveg, mcveg, mcnonveg]	6-7 PM

2. **COMPLETE_DATA1:** Contains numeric values such as order number, quantity, amount, orders and time zone information.



	Order Number	Time	Date	Quantity	Amount	Orders	TIMEZONES
0	4164	18:01:00	2017-02-01	5	33.75	1	6-7 PM
1	4165	18:16:00	2017-02-01	4	33.80	1	6-7 PM
2	4166	18:17:00	2017-02-01	7	33.20	1	6-7 PM
3	4167	18:25:00	2017-02-01	5	25.40	1	6-7 PM
4	4168	18:42:00	2017-02-01	4	38.80	1	6-7 PM

- **USER DRIVEN PROGRAM:**

USER INPUT FOR DATA:

The user should enter the 'Start Date' and 'End Date'. The program will only consider the data included in the specified date range by the user.

```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

FUNCTION DESCRIPTION:

1. getweeklyanalysis(df):

PARAMETERS PASSED:

df - Dataframe containing 'Date', 'Quantity', 'Orders' and 'Amount' Columns.

USAGE:

Used to obtain the total earning in a week. The function analyses the data fed by the user and generate a bar chart indicating the total sales amount in that week.

ALGORITHM:

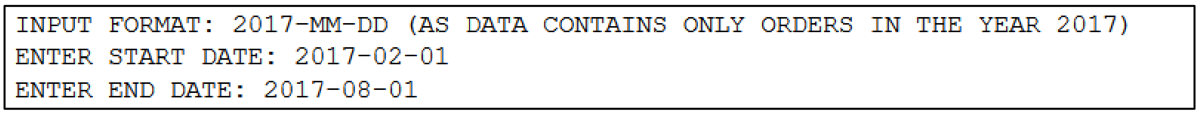
1. Obtain 'Date', 'Quantity', 'Orders' and 'Amount' Columns from the user input data and store in Data Frame 'df'.
2. Create a 'Week' column which will have the output of parsing the week number from the 'Date' column.
3. Use the groupby clause on 'Week' column and store the result in Data Frame 'df'.
4. Compute 'Averageno_items_per_order' by dividing 'Quantity' with 'Orders'.
5. Reset the index for Data Frame 'df' and declare variable 'filter'=0.
6. While duration is not equal to 4, the program will ask the user for input for the following:

```
FILTER BY:
1.AMOUNT (ENTER 1)
2.ORDERS (ENTER 2)
3.NO. OF ITEMS PER ORDER (ENTER 3)
4.SKIP THIS CHART (ENTER 4)
```

- If the user inputs filter= 1, the program will plot a bar chart with 'Week Number' on the x-axis and 'Amount' on the y-axis.
- If the user inputs filter= 2, the program will plot a bar chart with 'Week Number' on the x-axis and 'Orders' on the y-axis.
- If the user inputs filter= 3, the program will plot a bar chart with 'Week Number' on the x-axis and 'Averageno_items_per_order' on the y-axis.
- Or Else exit the function.

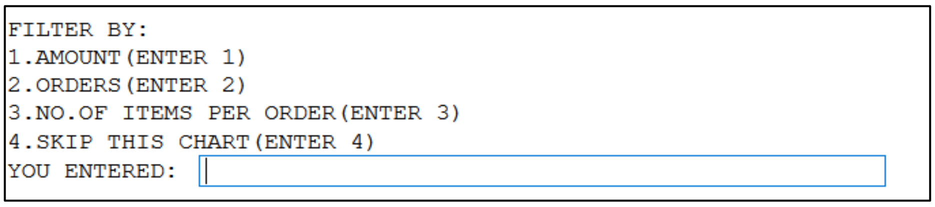
USAGE EXAMPLE:

- Suppose the user enters the data range given below:



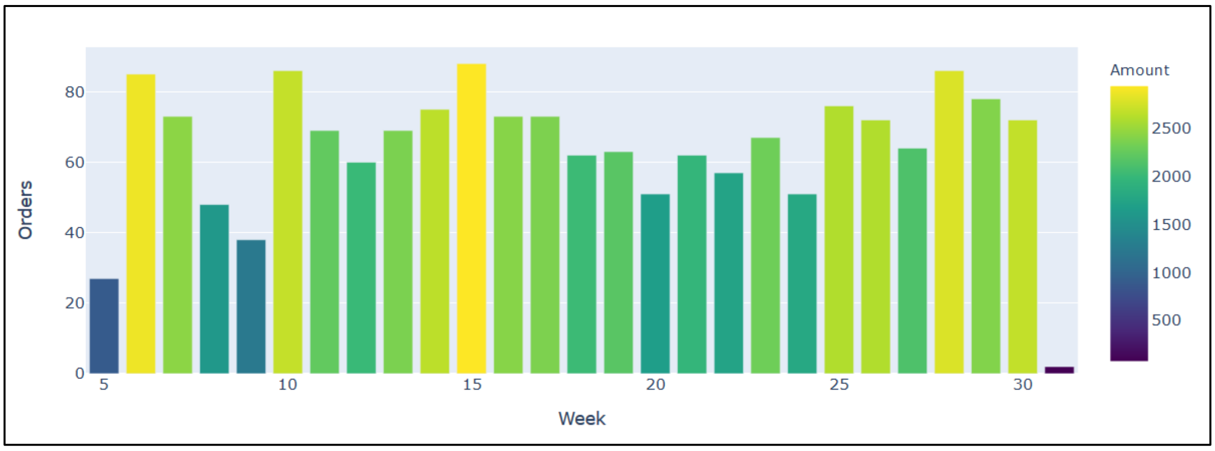
```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

- User will be asked to filter the bar chart according to the filters available.



```
FILTER BY:
1.AMOUNT(ENTER 1)
2.ORDERS(ENTER 2)
3.NO.OF ITEMS PER ORDER(ENTER 3)
4.SKIP THIS CHART(ENTER 4)
YOU ENTERED:
```

- A Bar Chart will be plot according to the user input. Suppose the user enters '2', the program will display a bar chart with 'Orders' on the y-axis.



- The user will be asked to enter a different value for filter again OR to exit by entering '4'.

- **INSIGHTS FOUND FROM THE ABOVE PLOT:**
- User can have a clear view of weekly earning, order count as well as average number of items in an order.
- User can obtain specific weeks which have high sales and improve the offers provided accordingly to maximise the profit.

2. gemonthlyanalysis(df):

PARAMETERS PASSED:

df - Dataframe containing 'Date', 'Quantity', 'Orders' and 'Amount' Columns.

USAGE:

Used to obtain the total earning in a month. The function analyses the data fed by the user and generate a bar chart indicating the total sales amount in that month.

ALGORITHM:

1. Obtain 'Date', 'Quantity', 'Orders' and 'Amount' Columns from the user input data and store in Data Frame 'df'.
2. Create a 'Month' column which will have the output of parsing the month from the 'Date' column.
3. Use the groupby clause on 'Month' column and store the result in Data Frame 'df'.
4. Compute 'Averageno_items_per_order' by dividing 'Quantity' with 'Orders'.
5. Reset the index for Data Frame 'df' and declare variable 'filter'=0.
6. While duration is not equal to 4, the program will ask the user for input for the following:

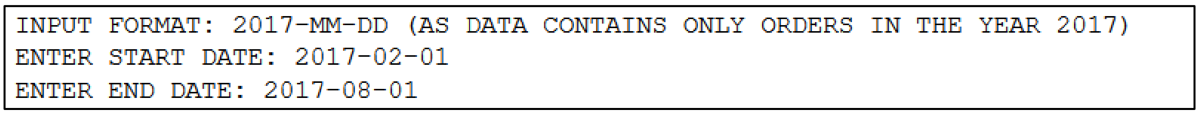
```
FILTER BY:
1.AMOUNT (ENTER 1)
2.ORDERS (ENTER 2)
3.NO. OF ITEMS PER ORDER (ENTER 3)
4.SKIP THIS CHART (ENTER 4)
```

   - If the user inputs filter'= 1, the program will plot a bar chart with 'Month' on the x-axis and 'Amount' on the y-axis.
   - If the user inputs filter'= 2, the program will plot a bar chart with 'Month' on the x-axis and 'Orders' on the y-axis.

- If the user inputs filter'= 3, the program will plot a bar chart with 'Month' on the x-axis and 'Averageno_items_per_order' on the y-axis.
- Or Else exit the function.

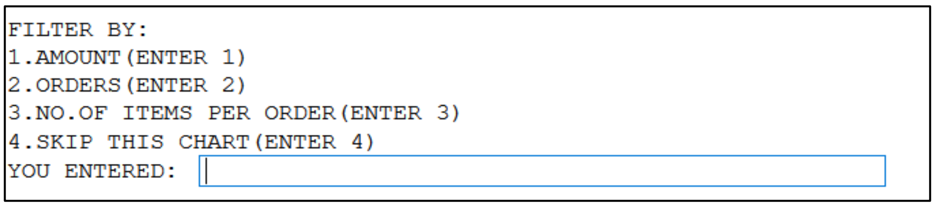
USAGE EXAMPLE:

- Suppose the user enters the data range given below:



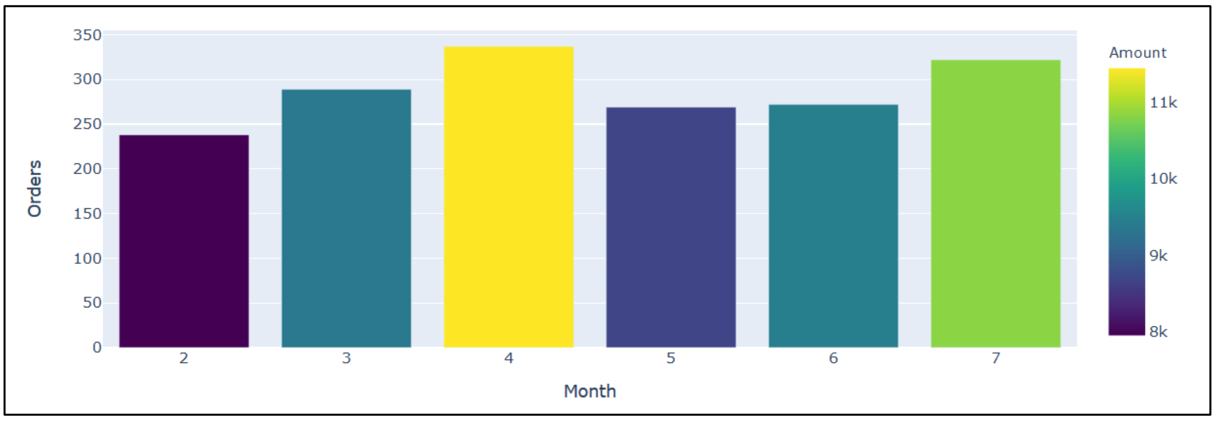
```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

- User will be asked to filter the bar chart according to the filters available.



```
FILTER BY:
1.AMOUNT(ENTER 1)
2.ORDERS(ENTER 2)
3.NO.OF ITEMS PER ORDER(ENTER 3)
4.SKIP THIS CHART(ENTER 4)
YOU ENTERED:
```

- A Bar Chart will be plot according to the user input. Suppose the user enters '2', the program will display a bar chart with 'Orders' on the y-axis.



- The user will be asked to enter a different value for filter again OR to exit by entering '4'.

- **INSIGHTS FOUND FROM THE ABOVE PLOT:**
- User can have a clear view of monthly earning, monthly order count as well as average number of items in an order.
- User can obtain specific months which have high sales and improve the offers provided accordingly to maximise the profit.

3. getdailyanalysis(df):

PARAMETERS PASSED:

df - Dataframe containing ‘Date’, ‘Quantity’, ‘Orders’ and ‘Amount’ Columns.

USAGE:

Used to obtain the total earning in a week. The function analyses the data fed by the user and generate a bar chart indicating the total sales amount in that week.

ALGORITHM:

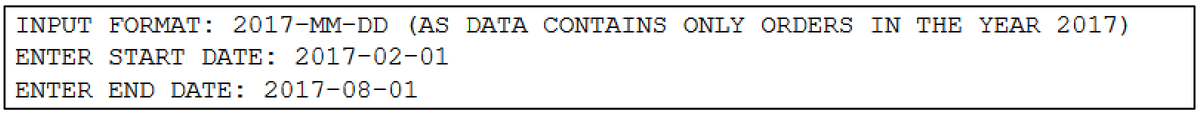
1. Obtain ‘Date’, ‘Quantity’, ‘Orders’ and ‘Amount’ Columns from the user input data and store in Data Frame ‘df’.
2. Use the groupby clause on ‘Date’ column with sum() as the aggregate function and store the result in Data Frame ‘df’.
3. Compute 'Averageno_items_per_order' by dividing ‘Quantity’ with ‘Orders’.
4. Reset the index for Data Frame ‘df’ and declare variable ‘filter’=0.
5. While duration is not equal to 4, the program will ask the user for input for the following:

```
FILTER BY:
1.AMOUNT (ENTER 1)
2.ORDERS (ENTER 2)
3.NO. OF ITEMS PER ORDER (ENTER 3)
4.SKIP THIS CHART (ENTER 4)
```

   - If the user inputs filter’= 1, the program will plot a bar chart with ‘Date’ on the x-axis and ‘Amount’ on the y-axis.
   - If the user inputs filter’= 2, the program will plot a bar chart with ‘Date’ on the x-axis and ‘Orders on the y-axis.
   - If the user inputs filter’= 3, the program will plot a bar chart with ‘Date’ on the x-axis and 'Averageno_items_per_order' on the y-axis.
   - Or Else exit the function.

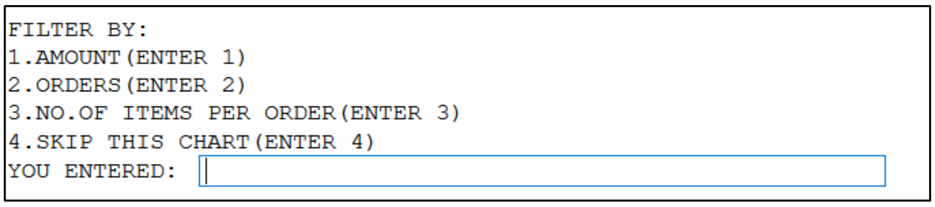
USAGE EXAMPLE:

- Suppose the user enters the data range given below:



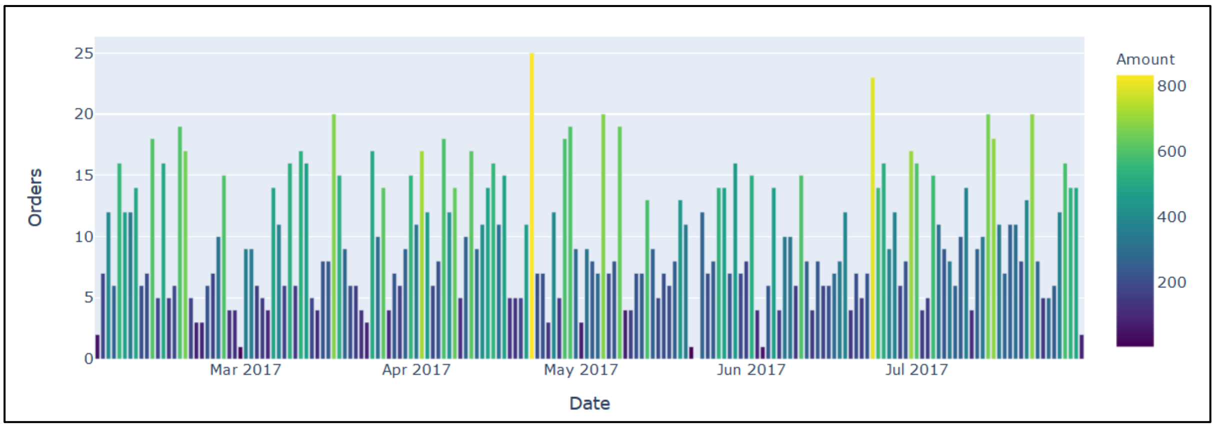
```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

- User will be asked to filter the bar chart according to the filters available.



```
FILTER BY:
1.AMOUNT(ENTER 1)
2.ORDERS(ENTER 2)
3.NO.OF ITEMS PER ORDER(ENTER 3)
4.SKIP THIS CHART(ENTER 4)
YOU ENTERED:
```

- A Bar Chart will be plot according to the user input. Suppose the user enters '2', the program will display a bar chart with 'Orders' on the y-axis and 'Date' on the x-axis.



- The user will be asked to enter a different value for filter again OR to exit by entering '4'.

- **INSIGHTS FOUND FROM THE ABOVE PLOT:**
- User can have a clear view of daily earning, order count as well as average number of items in an order.
- User can obtain specific days which have high sales and improve the offers provided accordingly to maximise the profit.

4. gethourlyorders(df):

PARAMETERS PASSED:

df - Dataframe containing 'Date', 'Quantity', 'Orders' and 'Amount' Columns.

USAGE:

Used to obtain the total count of orders between different time zones defined.

ALGORITHM:

1. Ask the user whether he wants to view the plot or not.
2. If user input is 'Yes', continue.
   - Obtain 'Date', 'Quantity', 'Time zone' and 'Amount' Columns from the user input data and store in Data Frame 'df'.
   - Use the groupby clause on 'Date' and 'Time zone' column and store the result in Data Frame 'df'.
   - Count the total no. of orders in a specific Time zone and store it in the 'Order Count' column.
   - Use the groupby clause on 'Time zones' column with sum() as the aggregate function and store it in Data Frame 'df'.
   - Plot the bar chart with 'Time zones' on x-axis and 'Order Count' on the y-axis.
3. Else exit the function.

USAGE EXAMPLE:

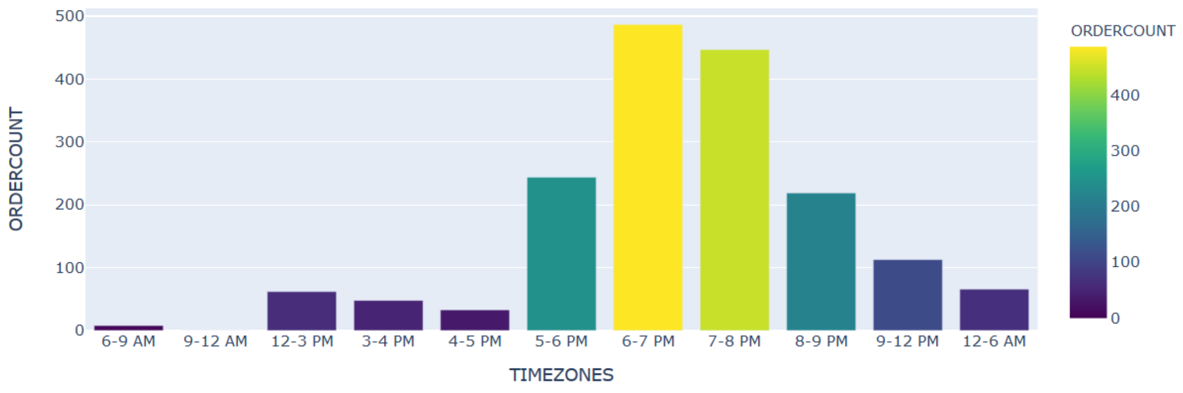
- Suppose the user enters the data range given below:

```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

- The user will be asked whether he/she wants to plot the Bar Chart? Suppose the user enters 1:

```
HERE IS THE TIME WHEN MOST OF YOUR ORDERS ARE RECIEVED:
DO YOU WANT TO VIEW THE PLOT?
1.YES(ENTER 1)
2.NO(ENTER 2)
ENTER YOUR INPUT: 1
```

- A bar chart will be displayed with 'Time zones' on the x-axis and 'Order Count' on the y-axis.



- **INSIGHTS FOUND FROM THE ABOVE PLOT:**
- User can have a clear picture, from which time zone does he/she is getting the most orders.
- User can work on the time zones where he is lacking in orders by providing offers in that time zone.

5. getmostordereditem(df):

PARAMETERS PASSED:

df - Dataframe containing 'Date', 'Order' ,'type' and 'Time zone' Columns.

USAGE:

Used to obtain the most ordered food item in a specified time zone entered by the user.

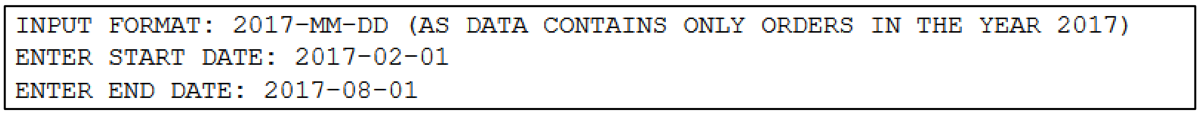
ALGORITHM:

1. Print Available Time zones for user assistance.
2. Obtain the Time zone from user and store it in variable 'zone'.
3. Filter the dataset according to the Time zone.
4. Use the groupby clause on 'Time zones' and 'Order' and store the output in a list named 'type'.
5. Obtain the list of all food items for that Time zone in a list 'time_items'. This list contains list of small lists, each containing names of food items of a specific order.
6. If length of the list 'time_items' is Zero, Exit the function displaying a message.

7. Else use the reduce function for converting the list of small lists into one large list which is stored in ‘total_items’.
8. Find out the total distinct food items present in ‘total_items’.
9. Compute the occurrence of each distinct food item in the ‘total_items’. Store the result the Data Frame ‘most_occ_item’.
10. Sort the values according to ‘Occurrence’ in the Data Frame ‘most_occ_item’.
11. Assign its category to every food item and store it in ‘type’ column.
12. Filter out all food items having category as ‘side dish’.
13. Select the first 20 rows of the Data Frame ‘most_occ_item’.

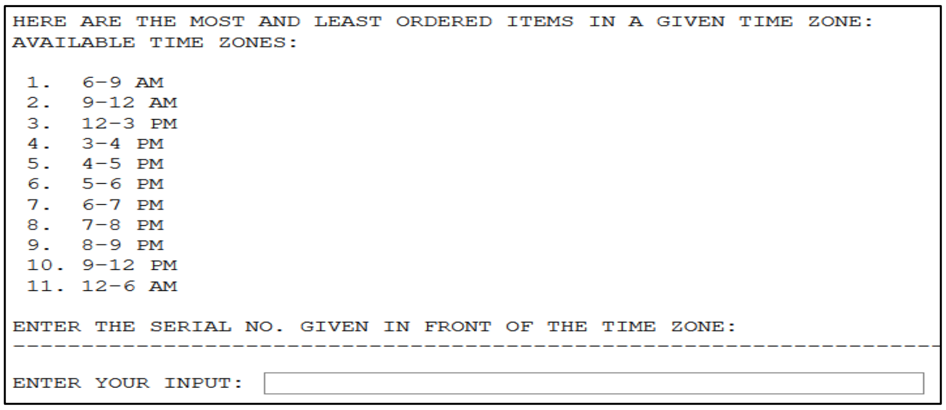
USAGE EXAMPLE:

- Suppose the user enters the data range given below:



```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

- User should select the desired time zone. Suppose the user enters 7.



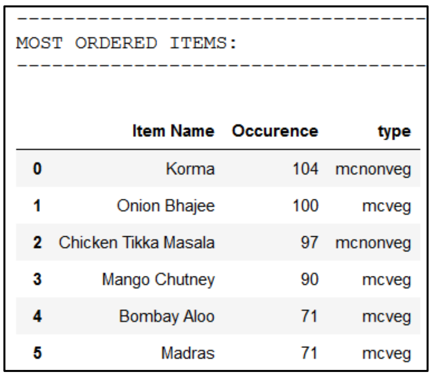
```
HERE ARE THE MOST AND LEAST ORDERED ITEMS IN A GIVEN TIME ZONE:
AVAILABLE TIME ZONES:

 1.  6-9 AM
 2.  9-12 AM
 3.  12-3 PM
 4.  3-4 PM
 5.  4-5 PM
 6.  5-6 PM
 7.  6-7 PM
 8.  7-8 PM
 9.  8-9 PM
 10. 9-12 PM
 11. 12-6 AM

ENTER THE SERIAL NO. GIVEN IN FRONT OF THE TIME ZONE:
-------------------------------------------------------------------------

ENTER YOUR INPUT:
```

- The most ordered food item in the selected time zone will be displayed.



```
------------------------------------
MOST ORDERED ITEMS:
------------------------------------
```

	Item Name	Occurence	type
0	Korma	104	mcnonveg
1	Onion Bhajee	100	mcveg
2	Chicken Tikka Masala	97	mcnonveg
3	Mango Chutney	90	mcveg
4	Bombay Aloo	71	mcveg
5	Madras	71	mcveg

- **INSIGHTS FOUND FROM THE ABOVE PLOT:**
- User can find out the most ordered food item in the given time zone.
- User can prepare the most selling food items before forth, hence provides the user with an opportunity to increase sales.

6. getcombo(df):

PARAMETERS PASSED:

df - Dataframe containing ‘Date’, ‘Order’ ,‘type’ and ‘Time zone’ Columns.

USAGE:

Used to obtain the most ordered food item in a specified time zone entered by the user.

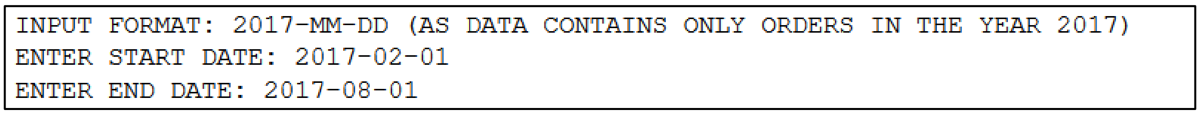
ALGORITHM:

1. Print Available Time zones for user assistance.
2. Obtain the Time zone from user and store it in variable ‘zone’.
3. Filter the dataset according to the Time zone.
4. Print all the side dishes available for user assistance.
5. Obtain input for the side dish for which we are going to find combinations for.
6. Iterate through the whole data and label the ‘Status’ column as ‘Found’ or ‘Not Found’.
7. Select rows having ‘Status’ column as ‘Found’.
8. Use the groupby clause on ‘Time zones’ and ‘Order’ and store the output in a list named ‘type’.
9. Obtain the list of all food items for that Time zone in a list ‘val’. This list contains list of small lists, each containing names of food items of a specific order.
10. If length of the list ‘count’ is Zero, Exit the function displaying a message.
11. Else use the reduce function for converting the list of small lists into one large list which is stored in ‘total_item’.
12. Find out the total distinct food items present in ‘total_item’.
13. Compute the occurrence of each distinct food item in the ‘total_item’. Store the result the Data Frame ‘combo_final’.
14. Assign its category to every food item and store it in ‘type’ column.

15. Filter out all food items having category as ‘side dish’.
16. Select the first 10 rows of the Data Frame ‘combo_final’.

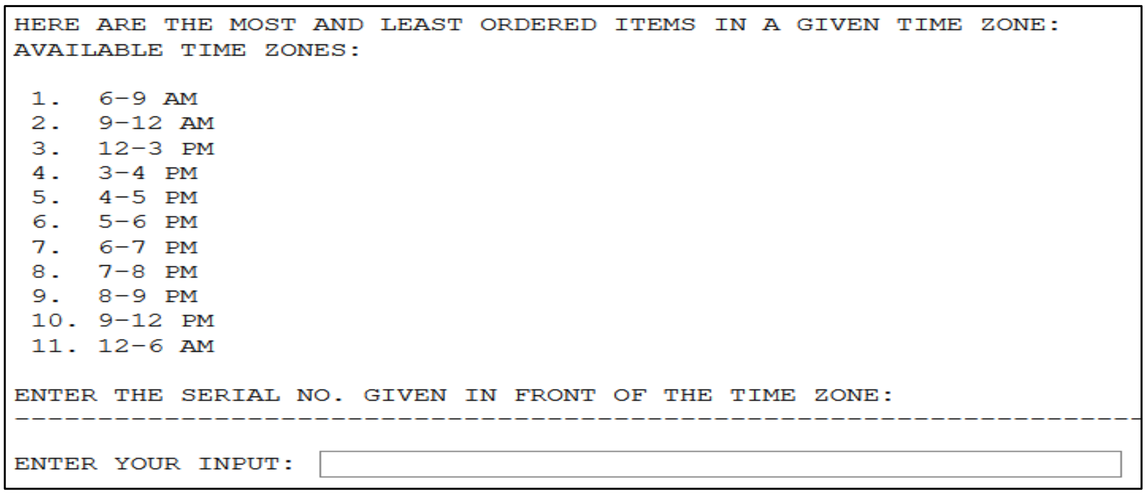
USAGE EXAMPLE:

- Suppose the user enters the data range given below:



```
INPUT FORMAT: 2017-MM-DD (AS DATA CONTAINS ONLY ORDERS IN THE YEAR 2017)
ENTER START DATE: 2017-02-01
ENTER END DATE: 2017-08-01
```

- User should select the desired time zone. Suppose the user enters 7.



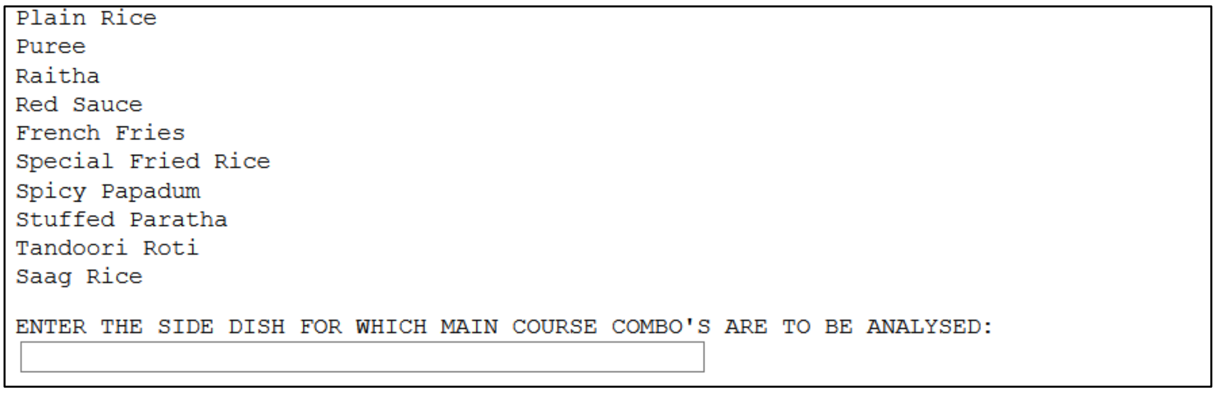
```
HERE ARE THE MOST AND LEAST ORDERED ITEMS IN A GIVEN TIME ZONE:
AVAILABLE TIME ZONES:

 1.  6-9 AM
 2.  9-12 AM
 3.  12-3 PM
 4.  3-4 PM
 5.  4-5 PM
 6.  5-6 PM
 7.  6-7 PM
 8.  7-8 PM
 9.  8-9 PM
 10. 9-12 PM
 11. 12-6 AM

ENTER THE SERIAL NO. GIVEN IN FRONT OF THE TIME ZONE:
--------------------------------------------------------------------

ENTER YOUR INPUT:
```

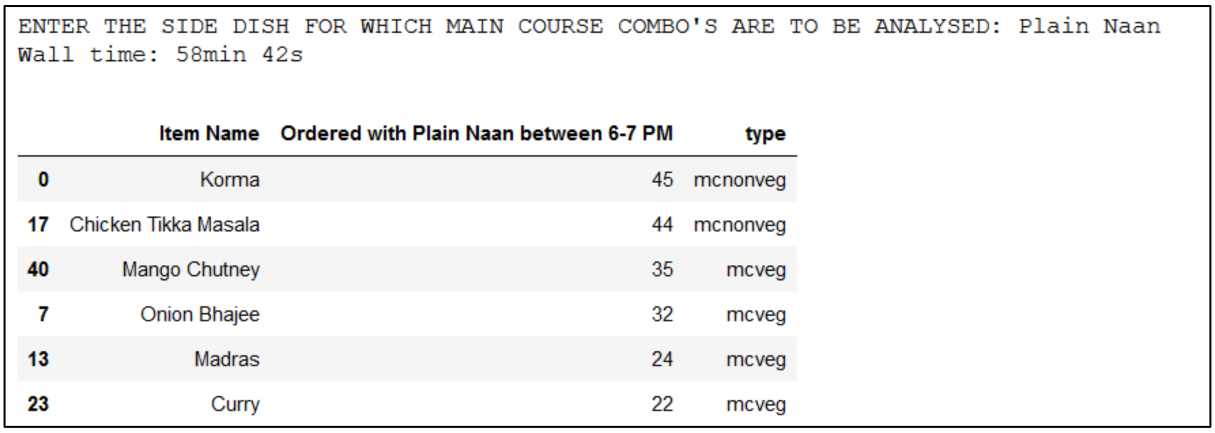
- User then should enter the side dish for which he wants to find combinations for. Suppose the user enters ‘Plain Naan’:



```
Plain Rice
Puree
Raitha
Red Sauce
French Fries
Special Fried Rice
Spicy Papadum
Stuffed Paratha
Tandoori Roti
Saag Rice

ENTER THE SIDE DISH FOR WHICH MAIN COURSE COMBO'S ARE TO BE ANALYSED:
```

- The program will show the most ordered main course items with the ‘Plain Naan’ during the specified time zone entered by the user.



```
ENTER THE SIDE DISH FOR WHICH MAIN COURSE COMBO'S ARE TO BE ANALYSED: Plain Naan
Wall time: 58min 42s
```

	Item Name	Ordered with Plain Naan between 6-7 PM	type
0	Korma	45	mcnonveg
17	Chicken Tikka Masala	44	mcnonveg
40	Mango Chutney	35	mcveg
7	Onion Bhajee	32	mcveg
13	Madras	24	mcveg
23	Curry	22	mcveg

- **INSIGHTS FOUND FROM THE ABOVE PLOT:**

▪ User can find out the best-selling combination in the given time zone.
▪ He can add these effective combo's in the restaurant menu.
▪ User can prepare the most selling food items before forth, hence provides the user with an opportunity to increase sales.

- **TIME COMPLEXITY OF THE ALGORITHM:**

MODULE	TIME UTILISED
RAW DATA LOADING	**1500 -1700 uSEC**
DATA PRE-PROCESSING	**650-750 uSEC**
USER-DRIVEN PROGRAM	**DEPENDANT ON USER RESPONSE TIME**

- **WHAT MORE CAN BE DONE:**

1. INVENTORY TRACKING AND ANALYSIS:

▪ It feels the worst when you are having loads of orders coming in and your **ingredients run out of stock**.
▪ This eventually results in a **huge decline in sales** and also **causes harm to customer relations.**
▪ To avoid this from happening, we can keep track of the available quantity of all the ingredients required.

- An automatic suggestion model will **send a notification** to the restaurant owner, whenever there is ingredient shortage in the inventory.

2. CUSTOMER RELATION MANAGEMENT:

- Keeping a personalised observation on the customers may help the restaurant to **predict customer behaviour**.
- This will empower the system to suggest the customers with their most ordered food items which may lead to increased order count.
- This will also **strengthen customer relationship** and **increase loyalty**.

3. SALES FORECASTING:

- Sales forecasting can be done with the help of statistical modelling as well as deep learning techniques such as:
    - **ARIMA** (Auto Regressive Integrated Moving Average)
    - **SARIMA** (Seasonal Auto Regressive Integrated Moving Average)
    - **LSTM-RNN** (Long Short-Term Memory Recurrent Neural Network)
    - **PROPHET MODELLING**

4. CUSTOMER FEEDBACK ANALYSIS:

- Customer Feedback is an essential part of analysis as it provides the flaws or the mistakes done by the restaurant in cooking, serving, hospitality of the customer.
- **Customer Sentiment Analysis** can be performed with the help of NLP (NATURAL LANGUAGE PROCESSING) libraries such as TextBlob etc.

• CONCLUSION:

The work sample had the aim to analyse a simple data set containing features such as Item names, their quantity, price, date and time of purchase. Various basic insights were obtained which can be used by the business owners to enhance their services and increase sales.

THANK YOU