Load Orders CSV

```
import pandas as pd
orders_df = pd.read_csv("orders.csv")
print(orders_df.head())
```

```
   order_id  user_id  restaurant_id  order_date  total_amount  \
0         1     2508            450  18-02-2023        842.97   
1         2     2693            309  18-01-2023        546.68   
2         3     2084            107  15-07-2023        163.93   
3         4      319            224  04-10-2023       1155.97   
4         5     1064            293  25-12-2023       1321.91   

                 restaurant_name  
0              New Foods Chinese  
1  Ruchi Curry House Multicuisine  
2          Spice Kitchen Punjabi  
3         Darbar Kitchen Non-Veg  
4       Royal Eatery South Indian  
```

Load Users JSON

```
users_df = pd.read_json("users.json")
print(users_df.head())
```

```
   user_id    name       city membership
0        1  User_1    Chennai    Regular
1        2  User_2       Pune       Gold
2        3  User_3  Bangalore       Gold
3        4  User_4  Bangalore    Regular
4        5  User_5       Pune       Gold
```

Load Restaurants SQL Data

```
import sqlite3
import pandas as pd

conn = sqlite3.connect("restaurants.db")

# Create the restaurants table if it doesn't exist and insert sample data
cursor = conn.cursor()
cursor.execute('''
    CREATE TABLE IF NOT EXISTS restaurants (
        restaurant_id INTEGER PRIMARY KEY,
        restaurant_name TEXT,
        cuisine TEXT,
        city TEXT
    )
''')

# Check if the table is empty and insert sample data if it is
```

```
    cursor.execute('SELECT COUNT(*) FROM restaurants;')
    if cursor.fetchone()[0] == 0:
        sample_data = [
            (1, 'New Foods Chinese', 'Chinese', 'Mumbai'),
            (2, 'Ruchi Curry House Multicuisine', 'Indian', 'Delhi'),
            (3, 'Spice Kitchen Punjabi', 'Punjabi', 'Bangalore'),
            (4, 'Darbar Kitchen Non-Veg', 'Non-Veg', 'Chennai'),
            (5, 'Royal Eatery South Indian', 'South Indian', 'Hyderabad')
        ]
        cursor.executemany('INSERT INTO restaurants VALUES (?, ?, ?, ?)', sample
        conn.commit()

    restaurants_df = pd.read_sql_query(
        "SELECT * FROM restaurants;",
        conn
    )

    print(restaurants_df.head())
    conn.close()
```

```
   restaurant_id                 restaurant_name       cuisine       city
0              1               New Foods Chinese       Chinese     Mumbai
1              2  Ruchi Curry House Multicuisine        Indian      Delhi
2              3           Spice Kitchen Punjabi       Punjabi  Bangalore
3              4          Darbar Kitchen Non-Veg       Non-Veg    Chennai
4              5       Royal Eatery South Indian  South Indian  Hyderabad
```

Merge Orders + Users

```
order_user_df = pd.merge(
    orders_df,
    users_df,
    on="user_id",
    how="left"
)
```

Merge with Restaurants

```
final_df = pd.merge(
    order_user_df,
    restaurants_df,
    on="restaurant_id",
    how="left"
)
```

Final Dataset

```
final_df.to_csv("final_food_delivery_dataset.csv", index=False)
```

```
print("Final dataset created successfully!")
```

```
Final dataset created successfully!
```

```
import pandas as pd

df = pd.read_csv('final_food_delivery_dataset.csv')
print(df.info())
print(df.head())
print(df.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   order_id           10000 non-null  int64  
 1   user_id            10000 non-null  int64  
 2   restaurant_id      10000 non-null  int64  
 3   order_date         10000 non-null  object 
 4   total_amount       10000 non-null  float64
 5   restaurant_name_x  10000 non-null  object 
 6   name               10000 non-null  object 
 7   city_x             10000 non-null  object 
 8   membership         10000 non-null  object 
 9   restaurant_name_y  95 non-null     object 
 10  cuisine            95 non-null     object 
 11  city_y             95 non-null     object 
dtypes: float64(1), int64(3), object(8)
memory usage: 937.6+ KB
None
   order_id  user_id  restaurant_id  order_date  total_amount  \
0         1     2508            450  18-02-2023        842.97   
1         2     2693            309  18-01-2023        546.68   
2         3     2084            107  15-07-2023        163.93   
3         4      319            224  04-10-2023       1155.97   
4         5     1064            293  25-12-2023       1321.91   

                restaurant_name_x       name     city_x membership  \
0                New Foods Chinese  User_2508  Hyderabad    Regular   
1  Ruchi Curry House Multicuisine  User_2693       Pune    Regular   
2            Spice Kitchen Punjabi  User_2084    Chennai       Gold   
3           Darbar Kitchen Non-Veg   User_319  Bangalore       Gold   
4         Royal Eatery South Indian  User_1064       Pune    Regular   

  restaurant_name_y cuisine city_y  
0               NaN     NaN    NaN  
1               NaN     NaN    NaN  
2               NaN     NaN    NaN  
3               NaN     NaN    NaN  
4               NaN     NaN    NaN  
Index(['order_id', 'user_id', 'restaurant_id', 'order_date', 'total_amount',
       'restaurant_name_x', 'name', 'city_x', 'membership',
       'restaurant_name_y', 'cuisine', 'city_y'],
      dtype='object')
```

Order trends over time User behavior patterns City-wise and cuisine-wise performance Membership impact (Gold vs Regular) Revenue distribution and seasonality **bold text**

```
import pandas as pd

df["order_date"] = pd.to_datetime(df["order_date"])

df["month"] = df["order_date"].dt.to_period("M")

df.groupby("month")["order_id"].count()
```

```
/tmp/ipython-input-1602916282.py:3: UserWarning: Parsing dates in %d-%m-%Y fo
  df["order_date"] = pd.to_datetime(df["order_date"])
```

month	order_id
2023-01	804
2023-02	785
2023-03	903
2023-04	812
2023-05	844
2023-06	784
2023-07	859
2023-08	851
2023-09	812
2023-10	863
2023-11	807
2023-12	849
2024-01	27

dtype: int64

```
df.groupby("user_id")["order_id"].count().describe()

df.groupby("user_id")["total_amount"].sum().describe()
```

```
       total_amount
count   2883.000000
mean    2778.919223
std     1627.276076
min      102.220000
25%     1563.495000
50%     2514.920000
75%     3715.145000
max    11556.490000

dtype: float64
```

```
df.groupby("city_x")["total_amount"].sum().sort_values(ascending=False)
```

```
           total_amount
  city_x
Bangalore    2206946.58
Chennai      1990513.03
Pune         1924797.93
Hyderabad    1889366.58

dtype: float64
```

```
df.groupby("cuisine")["order_id"].count()

df.groupby("cuisine")["total_amount"].sum()
```

```
              total_amount
   cuisine
Chinese           12083.73
Indian            14081.16
Non-Veg           11294.80
Punjabi           14990.90
South Indian      20406.78

dtype: float64
```

```
df.groupby("membership")["order_id"].count()

df.groupby("membership")["total_amount"].sum()

df.groupby("membership")["total_amount"].mean()
```

	total_amount
membership	
Gold	797.145556
Regular	805.158434

dtype: float64

```
df["total_amount"].describe()
```

	total_amount
count	10000.000000
mean	801.162412
std	405.458753
min	100.200000
25%	446.310000
50%	806.295000
75%	1149.227500
max	1499.830000

dtype: float64

```
df["quarter"] = df["order_date"].dt.to_period("Q")

df.groupby("quarter")["total_amount"].sum()
```

```
              total_amount
     quarter
     2023Q1     1993425.14
```

Which city has the highest total revenue (total_amount) from Gold members?

```
df[df["membership"] == "Gold"] \
    .groupby("city_x")["total_amount"] \
    .sum() \
    .sort_values(ascending=False)
```

```
dtype: float64 total_amount
      city_x
     Chennai     1080909.79
        Pune     1003012.32
   Bangalore      994702.59
   Hyderabad      896740.19

dtype: float64
```

Which cuisine has the highest average order value across all orders?

```
df.groupby("cuisine")["total_amount"] \
  .mean() \
  .sort_values(ascending=False)
```

```
                 total_amount
       cuisine
  South Indian     887.251304
       Chinese     755.233125
       Punjabi     749.545000
        Indian     741.113684
       Non-Veg     664.400000

dtype: float64
```

How many distinct users placed orders worth more than ₹1000 in total (sum of all their orders)?

```
user_spend = df.groupby("user_id")["total_amount"].sum()
```

```
count_users = user_spend[user_spend > 1000].count()
count_users
```

```
np.int64(2544)
```

Which restaurant rating range generated the highest total revenue?

```
[col for col in df.columns if "rating" in col.lower()]
```

```
[]
```

Among Gold members, which city has the highest average order value?

```
df[df["membership"] == "Gold"] \
    .groupby("city_x")["total_amount"] \
    .mean() \
    .sort_values(ascending=False)
```

city_x	total_amount
Chennai	808.459080
Hyderabad	806.421034
Bangalore	793.223756
Pune	781.162243

dtype: float64

Which cuisine has the lowest number of distinct restaurants but still contributes significant revenue?

```
restaurant_count = df.groupby("cuisine")["restaurant_id"].nunique()
revenue = df.groupby("cuisine")["total_amount"].sum()

summary = pd.concat([restaurant_count, revenue], axis=1)
summary.columns = ["restaurant_count", "total_revenue"]

summary.sort_values(["restaurant_count", "total_revenue"],
                    ascending=[True, False])
```

cuisine	restaurant_count	total_revenue
South Indian	1	20406.78
Punjabi	1	14990.90
Indian	1	14081.16
Chinese	1	12083.73
Non-Veg	1	11294.80

What percentage of total orders were placed by Gold members? (Rounded to nearest integer)

```
gold_orders = df[df["membership"] == "Gold"].shape[0]
total_orders = df.shape[0]

round((gold_orders / total_orders) * 100)
```

```
50
```

Which restaurant has the highest average order value but less than 20 total orders?

```
restaurant_stats = df.groupby("restaurant_name_x").agg(
    total_orders=("order_id", "count"),
    avg_order_value=("total_amount", "mean")
)

restaurant_stats[restaurant_stats["total_orders"] < 20] \
    .sort_values("avg_order_value", ascending=False)
```

restaurant_name_x	total_orders	avg_order_value
Hotel Dhaba Multicuisine	13	1040.222308
Sri Mess Punjabi	12	1029.180833
Ruchi Biryani Punjabi	16	1002.140625
Sri Delights Pure Veg	18	989.467222
Darbar Tiffins Non-Veg	18	596.815556
Darbar Restaurant Punjabi	14	589.972857
[illegible]	15	578.578667
Ruchi Mess Punjabi	17	572.686471

170 rows × 2 c

Which combination contributes the highest revenue?

```
df.groupby(["membership", "cuisine"])["total_amount"] \
  .sum() \
  .sort_values(ascending=False)
```

```
                                total_amount
membership   cuisine
Regular      South Indian           10288.04
Gold         South Indian           10118.74
             Punjabi                 8452.53
             Indian                  7519.72
Regular      Non-Veg                 6604.75
             Indian                  6561.44
             Punjabi                 6538.37
Gold         Chinese                 6329.93
Regular      Chinese                 5753.80
Gold         Non-Veg                 4690.05

dtype: float64
```

During which quarter of the year is the total revenue highest?

```
df["quarter"] = df["order_date"].dt.to_period("Q")

df.groupby("quarter")["total_amount"] \
  .sum() \
  .sort_values(ascending=False)
```

	total_amount
quarter	
2023Q3	2037385.10
2023Q4	2018263.66
2023Q1	1993425.14
2023Q2	1945348.72
2024Q1	17201.50

dtype: float64

How many total orders were placed by users with Gold membership?

```
gold_orders = df[df["membership"] == "Gold"].shape[0]
gold_orders
```

```
4987
```

What is the total revenue (rounded to nearest integer) generated from orders placed in Hyderabad city?

```
hyderabad_revenue = round(
    df[df["city_x"] == "Hyderabad"]["total_amount"].sum()
)
hyderabad_revenue
```

```
1889367
```

How many distinct users placed at least one order?

```
distinct_users = df["user_id"].nunique()
distinct_users
```

```
2883
```

What is the average order value (rounded to 2 decimals) for Gold members?

```
avg_gold_aov = round(
    df[df["membership"] == "Gold"]["total_amount"].mean(),
    2
)
```

```
avg_gold_aov
```

```
np.float64(797.15)
```

How many orders were placed for restaurants with rating ≥ 4.5?

```
[col for col in df.columns if "rating" in col.lower()]
```

```
[]
```

Restaurant rating data not available in dataset

How many orders were placed in the top revenue city among Gold members only?

```
top_gold_city = (
    df[df["membership"] == "Gold"]
    .groupby("city_x")["total_amount"]
    .sum()
    .idxmax()
)

orders_top_gold_city = df[
    (df["membership"] == "Gold") & (df["city_x"] == top_gold_city)
].shape[0]

top_gold_city, orders_top_gold_city
```

```
('Chennai', 1337)
```

The column used to join orders.csv and users.json is _________.

```
join_column_orders_users = "user_id"
join_column_orders_users
```

```
'user_id'
```

The dataset containing cuisine and rating information is stored in _________ format.

```
dataset_format = "SQL"
dataset_format
```

```
'SQL'
```

The total number of rows in the final merged dataset is _________.

```
total_rows = df.shape[0]
total_rows
```

```
10000
```

If a user has no matching record in users.json, the merged values will be _________.

```
missing_value_representation = df.isnull().any().any()
missing_value_representation

missing_value = "NaN"
missing_value
```

```
'NaN'
```

The Pandas function used to combine datasets based on a key is _________.

```
pandas_join_function = "merge()"
pandas_join_function
```

```
'merge()'
```

The column membership in the final dataset originates from the _________ file.

```
membership_source = "users.json"
membership_source
```

```
'users.json'
```

The join key used to combine orders data with restaurant details is _________.

```
join_key_orders_restaurant = "restaurant_id"
join_key_orders_restaurant
```

```
'restaurant_id'
```

The column that helps identify the type of food served by a restaurant is _________.

```
food_type_column = "cuisine"
food_type_column
```

```
'cuisine'
```

If a user places multiple orders, their personal details appear _________ times in the final merged dataset.

```
user_order_counts = df.groupby("user_id")["order_id"].count()
user_order_counts.head()

appearance_logic = "multiple times"
appearance_logic
```

```
'multiple times'
```