In [ ]:
```
## Loading Libraries
```

In [25]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
## Loading Datasets and Fetching data
```

In [ ]:
```
user_details = pd.read_excel(r"C:\Users\chara\Downloads\UserDetails.xlsx")
Cooking_session=pd.read_excel(r"C:\Users\chara\Downloads\Cookingsession.xlsx")
Order_details=pd.read_excel(r"C:\Users\chara\Downloads\order details.xlsx")
```

In [ ]:
```
# Checking Information
```

In [26]:
```
user_details.info
```

Out[26]:
```
<bound method DataFrame.info of   User ID      User Name  Age       Location Registration Date         Phone  \
0    U001  Alice Johnson   28       New York        2023-01-15  123-456-7890
1    U002      Bob Smith   35    Los Angeles        2023-02-20  987-654-3210
2    U003    Charlie Lee   42        Chicago        2023-03-10  555-123-4567
3    U004    David Brown   27  San Francisco        2023-04-05  444-333-2222
4    U005     Emma White   30        Seattle        2023-05-22  777-888-9999
5    U006    Frank Green   25         Austin        2023-06-15  888-777-6666
6    U007     Grace King   38         Boston        2023-07-02  999-888-7777
7    U008      Henry Lee   31          Miami        2023-08-11  101-202-3030
8    U009    Irene Moore   33         Dallas        2023-09-01  202-303-4040
9    U010     Jack White   29        Phoenix        2023-10-10  303-404-5050

                Email Favorite Meal  Total Orders
0    alice@email.com        Dinner            12
1      bob@email.com         Lunch             8
2  charlie@email.com     Breakfast            15
3    david@email.com        Dinner            10
4     emma@email.com         Lunch             9
5    frank@email.com        Dinner             7
6    grace@email.com     Breakfast            14
7    henry@email.com        Dinner             5
8    irene@email.com         Lunch             6
9     jack@email.com        Dinner             8  >
```

In [19]:
```
Cooking_session.info
```

Out[19]:
```
<bound method DataFrame.info of    Session ID User ID        Dish Name  Meal Type       Session Start  \
0        S001    U001        Spaghetti     Dinner 2024-12-01 19:00:00
1        S002    U002     Caesar Salad      Lunch 2024-12-01 12:00:00
2        S003    U003  Grilled Chicken     Dinner 2024-12-02 19:30:00
3        S004    U001         Pancakes  Breakfast 2024-12-02 07:30:00
4        S005    U004     Caesar Salad      Lunch 2024-12-03 13:00:00
5        S006    U002        Spaghetti     Dinner 2024-12-03 18:30:00
6        S007    U005  Grilled Chicken     Dinner 2024-12-04 18:00:00
7        S008    U003    Veggie Burger      Lunch 2024-12-04 13:30:00
8        S009    U001  Grilled Chicken     Dinner 2024-12-05 19:00:00
9        S010    U002          Oatmeal  Breakfast 2024-12-05 07:00:00
10       S011    U003         Pancakes  Breakfast 2024-12-06 08:00:00
11       S012    U004        Spaghetti     Dinner 2024-12-06 19:00:00
12       S013    U005     Caesar Salad      Lunch 2024-12-07 12:30:00
13       S014    U006  Grilled Chicken     Dinner 2024-12-07 18:00:00
14       S015    U007        Spaghetti     Dinner 2024-12-08 19:30:00
15       S016    U008    Veggie Burger      Lunch 2024-12-08 13:30:00

             Session End  Duration (mins)  Session Rating
0  2024-12-01 19:30:00               30             4.5
1  2024-12-01 12:20:00               20             4.0
2  2024-12-02 20:10:00               40             4.8
3  2024-12-02 08:00:00               30             4.2
4  2024-12-03 13:15:00               15             4.7
5  2024-12-03 19:00:00               30             4.3
6  2024-12-04 18:45:00               45             4.6
7  2024-12-04 13:50:00               20             4.4
8  2024-12-05 19:40:00               40             4.9
9  2024-12-05 07:10:00               10             4.1
10 2024-12-06 08:30:00               30             4.6
11 2024-12-06 19:40:00               40             4.7
12 2024-12-07 13:00:00               30             4.4
13 2024-12-07 18:45:00               45             4.8
14 2024-12-08 20:10:00               40             5.0
15 2024-12-08 13:50:00               20             4.3  >
```

In [24]:
```
Order_details.info
```

Out[24]:
```
<bound method DataFrame.info of     Order ID User ID Order Date  Meal Type        Dish Name Order Status  \
0       1001    U001 2024-12-01     Dinner        Spaghetti    Completed
1       1002    U002 2024-12-01      Lunch     Caesar Salad    Completed
2       1003    U003 2024-12-02     Dinner  Grilled Chicken     Canceled
3       1004    U001 2024-12-02  Breakfast         Pancakes    Completed
4       1005    U004 2024-12-03      Lunch     Caesar Salad    Completed
5       1006    U002 2024-12-03     Dinner        Spaghetti    Completed
6       1007    U005 2024-12-04     Dinner  Grilled Chicken    Completed
7       1008    U003 2024-12-04      Lunch    Veggie Burger     Canceled
8       1009    U001 2024-12-05     Dinner  Grilled Chicken    Completed
9       1010    U002 2024-12-05  Breakfast          Oatmeal    Completed
10      1011    U003 2024-12-06  Breakfast         Pancakes    Completed
11      1012    U004 2024-12-06     Dinner        Spaghetti    Completed
12      1013    U005 2024-12-07      Lunch     Caesar Salad    Completed
13      1014    U006 2024-12-07     Dinner  Grilled Chicken    Completed
14      1015    U007 2024-12-08     Dinner        Spaghetti    Completed
15      1016    U008 2024-12-08      Lunch    Veggie Burger    Completed

    Amount (USD) Time of Day  Rating Session ID
0           15.0       Night     5.0       S001
1           10.0         Day     4.0       S002
2           12.5       Night     NaN       S003
3            8.0     Morning     4.0       S004
4            9.0         Day     4.0       S005
5           14.0       Night     4.0       S006
6           13.5       Night     4.0       S007
7           11.0         Day     NaN       S008
8           12.0       Night     5.0       S009
9            7.0     Morning     4.0       S010
10           8.5     Morning     4.0       S011
11          12.5       Night     4.0       S012
12           9.0         Day     4.0       S013
13          13.0       Night     5.0       S014
14          14.0       Night     5.0       S015
15          11.0         Day     4.0       S016  >
```

In [162]:
```
user_details.describe()
```

Out[162]:

	age	total orders
count	10.000000	10.000000
mean	31.800000	9.400000
std	5.266245	3.339993
min	25.000000	5.000000
25%	28.250000	7.250000
50%	30.500000	8.500000
75%	34.500000	11.500000
max	42.000000	15.000000

In [161]:
```
Cooking_session.describe()
```

Out[161]:

	duration (mins)	session rating
count	16.000000	16.000000
mean	30.312500	4.518750
std	10.873324	0.292617
min	10.000000	4.000000
25%	20.000000	4.300000
50%	30.000000	4.550000
75%	40.000000	4.725000
max	45.000000	5.000000

In [164]:
```
Order_details.describe()
```

Out[164]:

	order id	amount (usd)
count	16.000000	16.000000
mean	1008.500000	11.250000
std	4.760952	2.435843
min	1001.000000	7.000000
25%	1004.750000	9.000000
50%	1008.500000	11.500000
75%	1012.250000	13.125000
max	1016.000000	15.000000

In [64]:
```
## Feteching Top 5
```

In [28]:
```
user_details.head()
```

Out[28]:

	User ID	User Name	Age	Location	Registration Date	Phone	Email	Favorite Meal	Total Orders
0	U001	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
1	U002	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	bob@email.com	Lunch	8
2	U003	Charlie Lee	42	Chicago	2023-03-10	555-123-4567	charlie@email.com	Breakfast	15
3	U004	David Brown	27	San Francisco	2023-04-05	444-333-2222	david@email.com	Dinner	10
4	U005	Emma White	30	Seattle	2023-05-22	777-888-9999	emma@email.com	Lunch	9

In [32]:
```
Cooking_session.head()
```

Out[32]:

	Session ID	User ID	Dish Name	Meal Type	Session Start	Session End	Duration (mins)	Session Rating
0	S001	U001	Spaghetti	Dinner	2024-12-01 19:00:00	2024-12-01 19:30:00	30	4.5
1	S002	U002	Caesar Salad	Lunch	2024-12-01 12:00:00	2024-12-01 12:20:00	20	4.0
2	S003	U003	Grilled Chicken	Dinner	2024-12-02 19:30:00	2024-12-02 20:10:00	40	4.8
3	S004	U001	Pancakes	Breakfast	2024-12-02 07:30:00	2024-12-02 08:00:00	30	4.2
4	S005	U004	Caesar Salad	Lunch	2024-12-03 13:00:00	2024-12-03 13:15:00	15	4.7

In [33]:
```
Order_details.head()
```

Out[33]:

	Order ID	User ID	Order Date	Meal Type	Dish Name	Order Status	Amount (USD)	Time of Day	Rating	Session ID
0	1001	U001	2024-12-01	Dinner	Spaghetti	Completed	15.0	Night	5.0	S001
1	1002	U002	2024-12-01	Lunch	Caesar Salad	Completed	10.0	Day	4.0	S002
2	1003	U003	2024-12-02	Dinner	Grilled Chicken	Canceled	12.5	Night	NaN	S003
3	1004	U001	2024-12-02	Breakfast	Pancakes	Completed	8.0	Morning	4.0	S004
4	1005	U004	2024-12-03	Lunch	Caesar Salad	Completed	9.0	Day	4.0	S005

In [ ]:
```
## Cleaning Data set
```

In [34]:
```
## removing duplicated
```

In [41]:
```
user_details.drop_duplicates(inplace=True)
Cooking_session.drop_duplicates(inplace=True)
Order_details.drop_duplicates(inplace=True)
```

In [42]:
```
## Handling Missing value
```

In [50]:
```
user_details.fillna('Unknown', inplace=True)
Cooking_session.fillna({'Duration (mins)': Cooking_session['Duration (mins)'].median(),'Session Rating': Cooking_session['Session Rating'].mean() }, inplace=Tru
Order_details.fillna({'Amount (USD)': Order_details['Amount (USD)'].median(),'Rating': 'Unknown'}, inplace=True)
```

In [51]:
```
## Checking date formates
```

In [54]:
```
Order_details['Order Date'] = pd.to_datetime(Order_details['Order Date'])
Cooking_session['Session Start'] = pd.to_datetime(Cooking_session['Session Start'])
```

In [ ]:
```
## Checking column names in each dataset
```

In [55]:
```
user_details.columns
```

Out[55]:
```
Index(['User ID', 'User Name', 'Age', 'Location', 'Registration Date', 'Phone',
       'Email', 'Favorite Meal', 'Total Orders'],
      dtype='object')
```

In [57]:
```
Cooking_session.columns
```

Out[57]:
```
Index(['Session ID', 'User ID', 'Dish Name', 'Meal Type', 'Session Start',
       'Session End', 'Duration (mins)', 'Session Rating'],
      dtype='object')
```

In [58]:
```
Order_details.columns
```

Out[58]:
```
Index(['Order ID', 'User ID', 'Order Date', 'Meal Type', 'Dish Name',
       'Order Status', 'Amount (USD)', 'Time of Day', 'Rating', 'Session ID'],
      dtype='object')
```

In [84]:
```
# Standardize column names to lowercase for consistency
user_details.columns = user_details.columns.str.strip().str.lower()
Cooking_session.columns = Cooking_session.columns.str.strip().str.lower()
Order_details.columns = Order_details.columns.str.strip().str.lower()
```

In [89]:
```
# Merge the datasets
# Merge CookingSessions and OrderDetails based on common keys
merged_data = pd.merge(Cooking_session, Order_details, on = 'session id', how = 'inner')
```

In [90]:
```
# Merge with UserDetails to get complete information
final_data = pd.merge(merged_data, user_details, left_on='user id_x', right_on='user id', how='inner')
```

In [93]:
```
final_data.head()
```

Out[93]:

	session id	user id_x	dish name_x	meal type_x	session start	session end	duration (mins)	session rating	order id	user id_y	...	rating	user id	user name	age	location	registration date	phone	email	favorite meal	total orders
0	S001	U001	Spaghetti	Dinner	2024-12-01 19:00:00	2024-12-01 19:30:00	30	4.5	1001	U001	...	5.0	U001	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
1	S004	U001	Pancakes	Breakfast	2024-12-02 07:30:00	2024-12-02 08:00:00	30	4.2	1004	U001	...	4.0	U001	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
2	S009	U001	Grilled Chicken	Dinner	2024-12-05 19:00:00	2024-12-05 19:40:00	40	4.9	1009	U001	...	5.0	U001	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
3	S002	U002	Caesar Salad	Lunch	2024-12-01 12:00:00	2024-12-01 12:20:00	20	4.0	1002	U002	...	4.0	U002	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	bob@email.com	Lunch	8
4	S006	U002	Spaghetti	Dinner	2024-12-03 18:30:00	2024-12-03 19:00:00	30	4.3	1006	U002	...	4.0	U002	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	bob@email.com	Lunch	8

5 rows × 26 columns

In [94]:
```
# Drop redundant columns after merge
final_data.drop(['user id_x', 'user id'], axis=1, inplace=True)
```

In [95]:
```
# Clean and transform data (Example: Handling missing values)
final_data.fillna({'rating': 0, 'session rating': 0}, inplace=True)
```

In [96]:
```
final_data.head()
```

Out[96]:

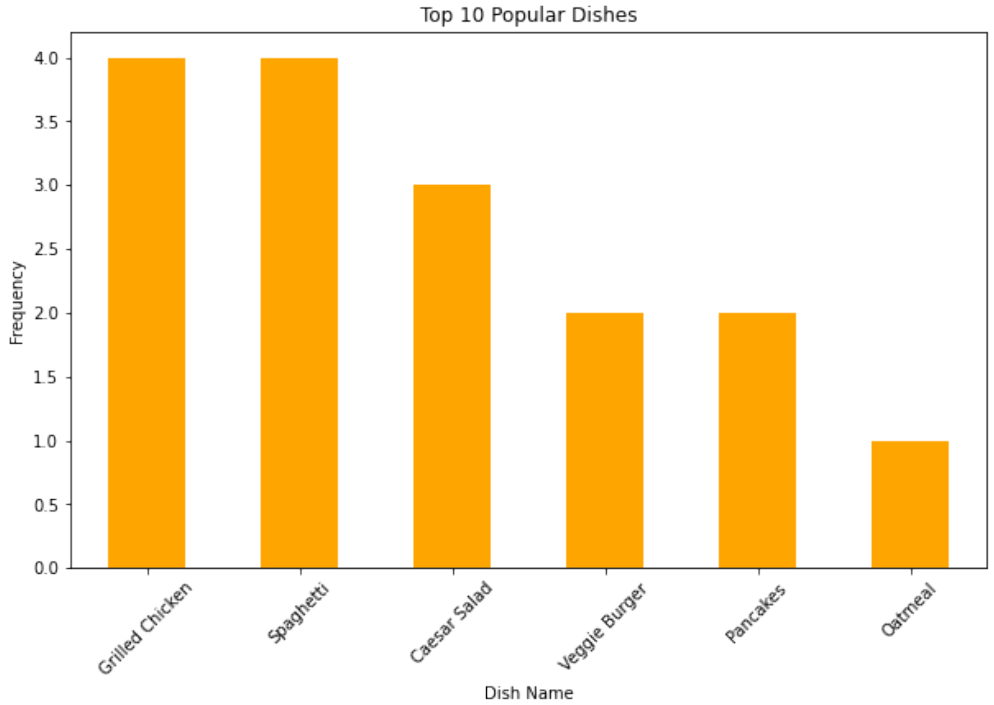
	session id	dish name_x	meal type_x	session start	session end	duration (mins)	session rating	order id	user id_y	order date	...	time of day	rating	user name	age	location	registration date	phone	email	favorite meal	total orders
0	S001	Spaghetti	Dinner	2024-12-01 19:00:00	2024-12-01 19:30:00	30	4.5	1001	U001	2024-12-01	...	Night	5.0	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
1	S004	Pancakes	Breakfast	2024-12-02 07:30:00	2024-12-02 08:00:00	30	4.2	1004	U001	2024-12-02	...	Morning	4.0	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
2	S009	Grilled Chicken	Dinner	2024-12-05 19:00:00	2024-12-05 19:40:00	40	4.9	1009	U001	2024-12-05	...	Night	5.0	Alice Johnson	28	New York	2023-01-15	123-456-7890	alice@email.com	Dinner	12
3	S002	Caesar Salad	Lunch	2024-12-01 12:00:00	2024-12-01 12:20:00	20	4.0	1002	U002	2024-12-01	...	Day	4.0	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	bob@email.com	Lunch	8
4	S006	Spaghetti	Dinner	2024-12-03 18:30:00	2024-12-03 19:00:00	30	4.3	1006	U002	2024-12-03	...	Night	4.0	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	bob@email.com	Lunch	8

5 rows × 24 columns

In [98]:
```
popular_dishes = final_data["dish name_x"].value_counts().head(10)
print("Top 10 Popular Dishes:")
print(popular_dishes)
```

```
Top 10 Popular Dishes:
Grilled Chicken    4
Spaghetti          4
Caesar Salad       3
Veggie Burger      2
Pancakes           2
Oatmeal            1
Name: dish name_x, dtype: int64
```

In [104]:
```
# Visualization: Top 10 Popular Dishes
plt.figure(figsize=(10, 6))
popular_dishes.plot(kind='bar', color='orange')
plt.title('Top 10 Popular Dishes')
plt.xlabel('Dish Name')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```



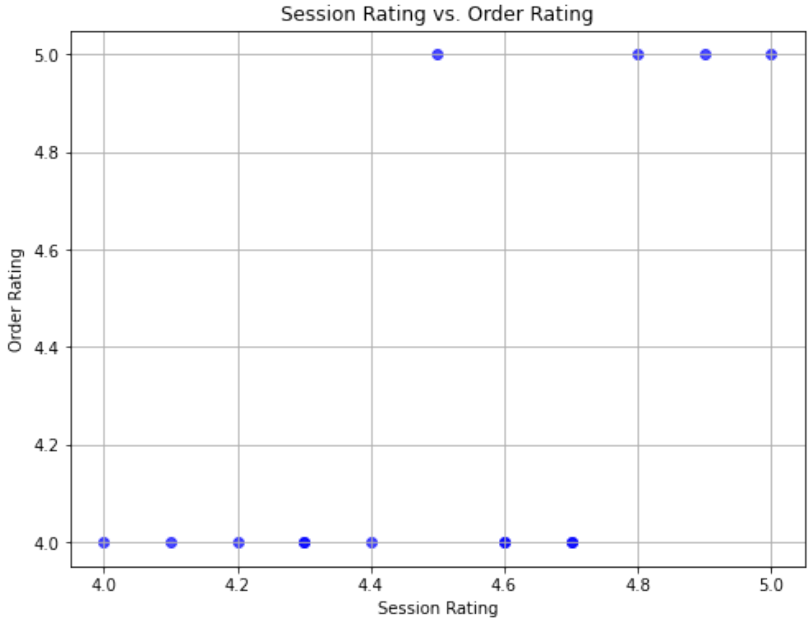
In [111]:
```
# Convert columns to numeric, forcing errors to NaN
final_data["session rating"] = pd.to_numeric(final_data["session rating"], errors="coerce")
final_data["rating"] = pd.to_numeric(final_data["rating"], errors="coerce")
final_data["amount (usd)"] = pd.to_numeric(final_data["amount (usd)"], errors="coerce")
```

In [132]:
```
# Analyze relationship between cooking sessions and user orders
session_order_correlation = final_data.groupby("user name").agg({
    "session rating": "mean",
    "rating": "mean",
    "amount (usd)": "sum"
}).reset_index()
session_order_correlation
```

Out[132]:

	user name	session rating	rating	amount (usd)
0	Alice Johnson	4.533333	4.666667	35.0
1	Bob Smith	4.133333	4.000000	31.0
2	Charlie Lee	4.600000	4.000000	32.0
3	David Brown	4.700000	4.000000	21.5
4	Emma White	4.500000	4.000000	22.5
5	Frank Green	4.800000	5.000000	13.0
6	Grace King	5.000000	5.000000	14.0
7	Henry Lee	4.300000	4.000000	11.0

In [133]:
```
plt.figure(figsize=(8, 6))
plt.scatter(final_data['session rating'], final_data['rating'], alpha=0.7, c='blue')
plt.title('Session Rating vs. Order Rating')
plt.xlabel('Session Rating')
plt.ylabel('Order Rating')
plt.grid()
plt.show()
```



In [131]:
```
# Filter completed orders
completed_orders = final_data[final_data['order status'] == 'Completed']

# Calculating average session rating and order rating for completed orders
average_session_rating_completed = completed_orders['session rating'].mean()
average_order_rating_completed = completed_orders['rating'].mean()

# Calculating average session duration for completed orders
average_duration_completed = completed_orders['duration (mins)'].mean()

# Group by Order Status to compare session ratings, order ratings, and durations
session_rating_by_order_status = final_data.groupby('order status')['session rating'].mean()
order_rating_by_order_status = final_data.groupby('order status')['rating'].mean()
duration_by_order_status = final_data.groupby('order status')['duration (mins)'].mean()

# Printing results
print("Average Session Rating for Completed Orders:", average_session_rating_completed)
print("Average Order Rating for Completed Orders:", average_order_rating_completed)
print("Average Session Duration for Completed Orders (mins):", average_duration_completed)
print("\nAverage Session Rating by Order Status:\n", session_rating_by_order_status)
print("\nAverage Order Rating by Order Status:\n", order_rating_by_order_status)
print("\nAverage Session Duration by Order Status (mins):\n", duration_by_order_status)
```

```
Average Session Rating for Completed Orders: 4.507142857142857
Average Order Rating for Completed Orders: 4.285714285714286
Average Session Duration for Completed Orders (mins): 30.357142857142858

Average Session Rating by Order Status:
 order status
Canceled     4.600000
Completed    4.507143
Name: session rating, dtype: float64

Average Order Rating by Order Status:
 order status
Canceled          NaN
Completed    4.285714
Name: rating, dtype: float64

Average Session Duration by Order Status (mins):
 order status
Canceled     30.000000
Completed    30.357143
Name: duration (mins), dtype: float64
```

In [134]:
```
final_data.columns
```

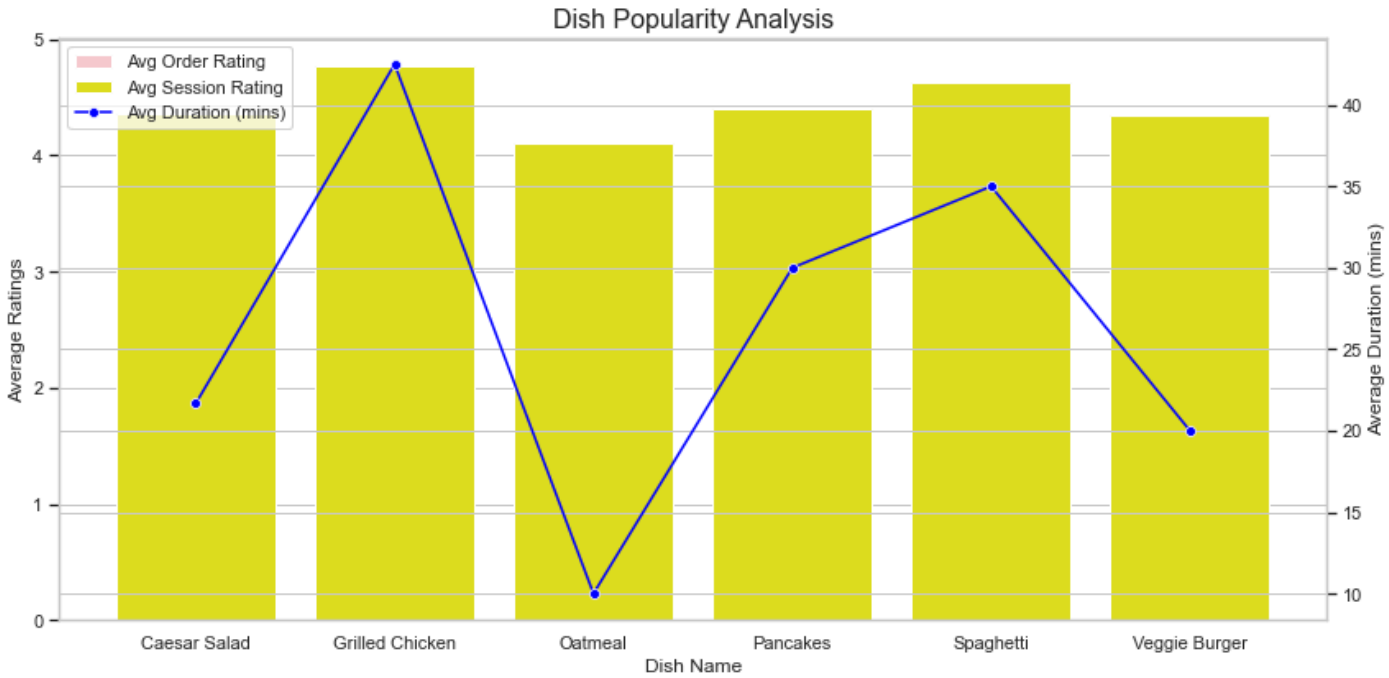
Out[134]:
```
Index(['session id', 'dish name_x', 'meal type_x', 'session start',
       'session end', 'duration (mins)', 'session rating', 'order id',
       'user id_y', 'order date', 'meal type_y', 'dish name_y', 'order status',
       'amount (usd)', 'time of day', 'rating', 'user name', 'age', 'location',
       'registration date', 'phone', 'email', 'favorite meal', 'total orders'],
      dtype='object')
```

In [137]:
```
# Filtering the DataFrame for relevant columns
dish_data = final_data[['dish name_x', 'rating', 'session rating', 'duration (mins)']]

# Calculating average ratings and session duration for each dish
dish_analysis = dish_data.groupby('dish name_x').agg({
    'rating': 'mean',
    'session rating': 'mean',
    'duration (mins)': 'mean',
    }).reset_index()

dish_analysis.columns = ['dish name_x', 'Avg Order Rating', 'Avg Session Rating', 'Avg Duration (mins)']

# Sorting dishes by average order rating, session rating, and completion rate
popular_dishes_df = dish_analysis.sort_values(by=['Avg Order Rating', 'Avg Session Rating'], ascending=False)

# Printing the resulting DataFrame
print("Popular Dishes DataFrame:\n", popular_dishes_df)
popular_dishes_df.to_excel('output.xlsx', index=False)
```

```
Popular Dishes DataFrame:
        dish name_x  Avg Order Rating  Avg Session Rating  Avg Duration (mins)
1  Grilled Chicken          4.666667            4.775000            42.500000
4        Spaghetti          4.500000            4.625000            35.000000
3         Pancakes          4.000000            4.400000            30.000000
0     Caesar Salad          4.000000            4.366667            21.666667
5    Veggie Burger          4.000000            4.350000            20.000000
2          Oatmeal          4.000000            4.100000            10.000000
```

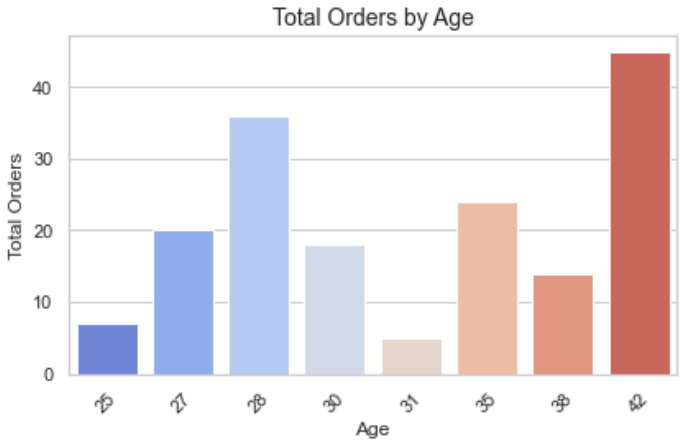
In [149]:

```
# Plotting
plt.figure(figsize=(12, 6))
sns.set(style="whitegrid")

# Bar plot for average order rating and session rating
ax = sns.barplot(x='dish name_x', y='Avg Order Rating', data=dish_analysis, color='pink', label='Avg Order Rating')
sns.barplot(x='dish name_x', y='Avg Session Rating', data=dish_analysis, color='yellow', label='Avg Session Rating')

# Create another Y-axis for duration
ax2 = ax.twinx()
sns.lineplot(x='dish name_x', y='Avg Duration (mins)', data=dish_analysis, color='blue', marker='o', label='Avg Duration (mins)', ax=ax2)

# Adding labels and title
ax.set_ylabel('Average Ratings')
ax2.set_ylabel('Average Duration (mins)')
ax.set_xlabel('Dish Name')
ax.set_title('Dish Popularity Analysis', fontsize=16)

# Adjusting legend
bar_legend = ax.get_legend_handles_labels()
line_legend = ax2.get_legend_handles_labels()
plt.legend(bar_legend[0] + line_legend[0], bar_legend[1] + line_legend[1], loc='upper left')

# Adjust X-axis for better readability
plt.xticks(rotation=45)

# Display the plot
plt.tight_layout()
plt.show()
```

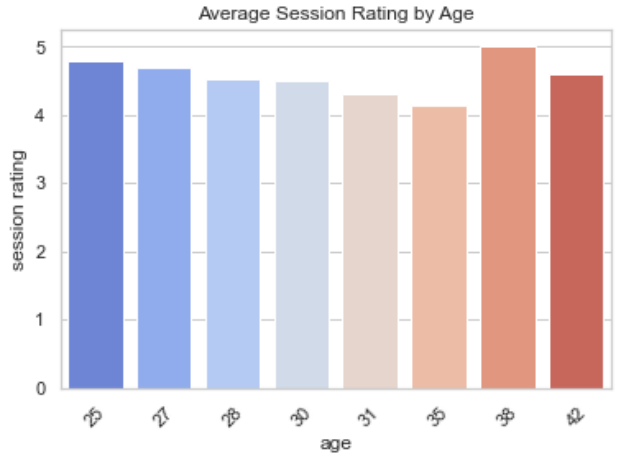


In [156]:
```
# Calculating total orders and average session rating by age
age_analysis = final_data.groupby('age').agg({
    'total orders': 'sum',
    'session rating': 'mean',
    'amount (usd)': 'mean'
}).reset_index()

# Calculating total orders and average session rating by location
location_analysis = final_data.groupby('location').agg({
    'total orders': 'sum',
    'session rating': 'mean',
    'amount (usd)': 'mean'
}).reset_index()
```

In [160]:
```
# Plotting total orders by age
plt.figure(figsize=(6, 4))
sns.set(style="whitegrid")
sns.barplot(x='age', y='total orders', data=age_analysis, palette='coolwarm')  # Changed to 'coolwarm'
plt.title('Total Orders by Age', fontsize=14)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Total Orders', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

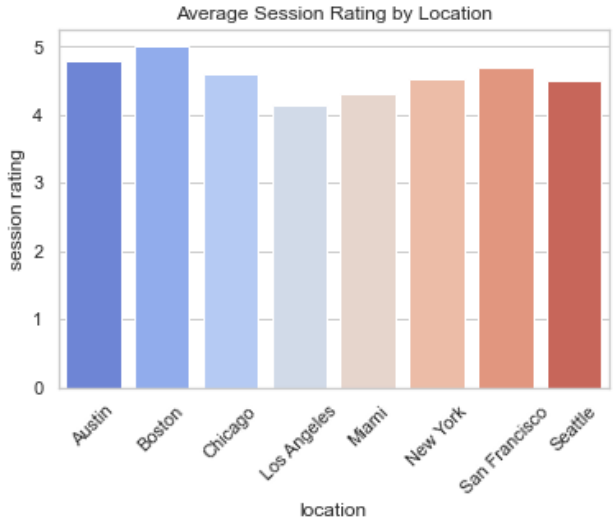


In [168]:
```
## average session rating by age
plt.figure(figsize=(6, 4))
sns.barplot(x='age', y='session rating', data=age_analysis, palette='coolwarm')
plt.title('Average Session Rating by Age')
plt.xticks(rotation=45)
plt.show()
```

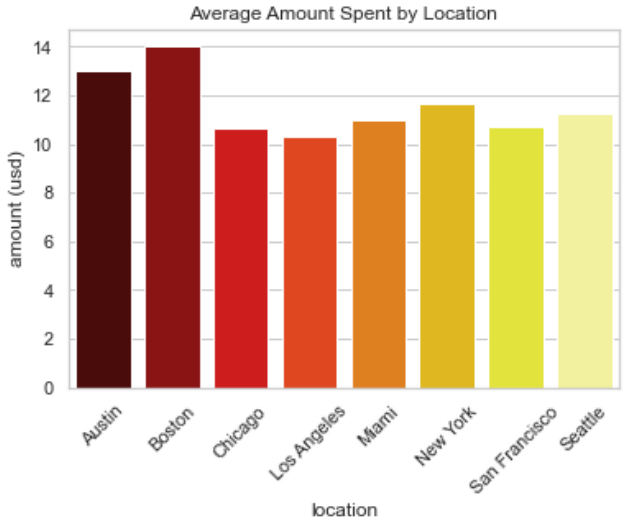


In [170]:
```
## Total orders by location
plt.figure(figsize=(6, 4))
sns.barplot(x='location', y='total orders', data=location_analysis, palette='coolwarm')
plt.title('Total Orders by Location')
plt.xticks(rotation=45)
plt.show()
```

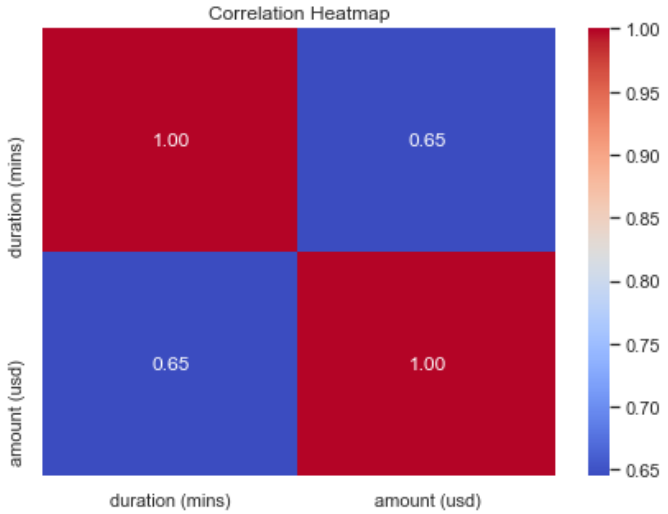
In [171]:
```
## Average Session Rating By Location
plt.figure(figsize=(6, 4))
sns.barplot(x='location', y='session rating', data=location_analysis, palette='coolwarm')
plt.title('Average Session Rating by Location')
plt.xticks(rotation=45)
plt.show()
```



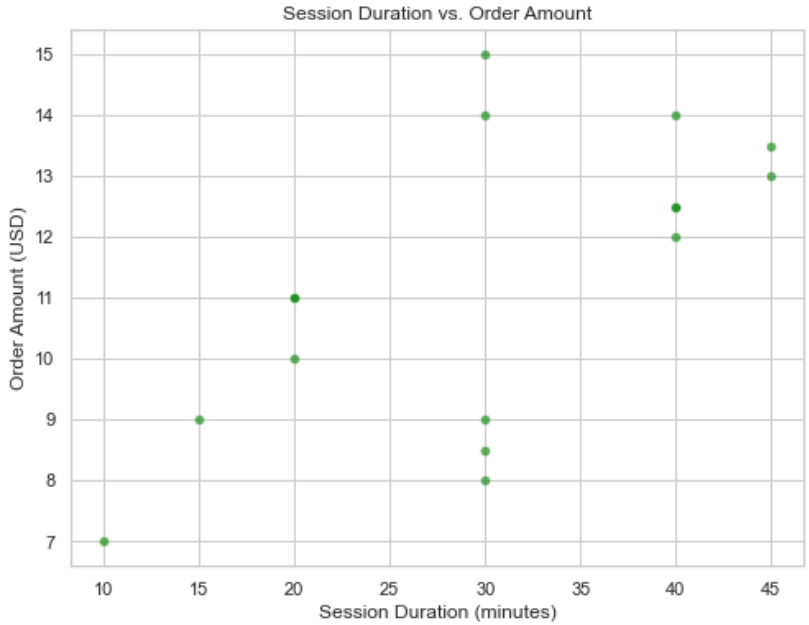
In [175]:
```
## Average Amopunt Spend By Loacation
plt.figure(figsize=(6, 4))
sns.barplot(x='location', y='amount (usd)', data=location_analysis, palette='hot')
plt.title('Average Amount Spent by Location')
plt.xticks(rotation=45)
plt.show()
```



In [179]:
```
# Heatmap showing correlation between different numerical columns
correlation_matrix = final_data[['duration (mins)', 'amount (usd)']].corr()
plt.figure(figsize=(7, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



In [180]:
```
# Scatter plot to see the relationship between cooking session duration and order amount
plt.figure(figsize=(8, 6))
sns.scatterplot(x='duration (mins)', y='amount (usd)', data=final_data, alpha=0.6, color='green')
plt.title('Session Duration vs. Order Amount')
plt.xlabel('Session Duration (minutes)')
plt.ylabel('Order Amount (USD)')
plt.show()
```



In [182]:
```
## Analysis by age
print("Analysis by Age:")
print(age_analysis)
## Analysis by location
print("\nAnalysis by Location:")
print(location_analysis)
age_analysis.to_excel("output.xlsx")
```

```
Analysis by Age:
   age  total orders  session rating  amount (usd)
0   25             7        4.800000     13.000000
1   27            20        4.700000     10.750000
2   28            36        4.533333     11.666667
3   30            18        4.500000     11.250000
4   31             5        4.300000     11.000000
5   35            24        4.133333     10.333333
6   38            14        5.000000     14.000000
7   42            45        4.600000     10.666667

Analysis by Location:
        location  total orders  session rating  amount (usd)
0         Austin             7        4.800000     13.000000
1         Boston            14        5.000000     14.000000
2        Chicago            45        4.600000     10.666667
3    Los Angeles            24        4.133333     10.333333
4          Miami             5        4.300000     11.000000
5       New York            36        4.533333     11.666667
6  San Francisco            20        4.700000     10.750000
7        Seattle            18        4.500000     11.250000
```