1. Install Required Libraries

!pip install pandas numpy matplotlib seaborn xlwt

2. Specify File Path

In [4]:
```python
file_path = r"C:\Assignment\Data Analyst Intern Assignment - Excel.xlsx"
```

3. Load Datasets

In [7]:
```python
import pandas as pd
user_details = pd.read_excel(file_path, sheet_name='UserDetails.csv')
cooking_sessions = pd.read_excel(file_path, sheet_name='CookingSessions.csv')
order_details = pd.read_excel(file_path, sheet_name='OrderDetails.csv')
```

4. Preview User Details Dataset

In [8]:
```python
user_details.head(20)
```

Out[8]:

	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
5	U006	Frank Green	25	Austin	2023-06-15	888-777-6666	frank@email.com	Dinner	7
6	U007	Grace King	38	Boston	2023-07-02	999-888-7777	grace@email.com	Breakfast	14
7	U008	Henry Lee	31	Miami	2023-08-11	101-202-3030	henry@email.com	Dinner	5
8	U009	Irene Moore	33	Dallas	2023-09-01	202-303-4040	irene@email.com	Lunch	6
9	U010	Jack White	29	Phoenix	2023-10-10	303-404-5050	jack@email.com	Dinner	8

5. Check for Missing Values in All Datasets

In [10]:
```python
print(user_details.isnull().sum())
print(cooking_sessions.isnull().sum())
print(order_details.isnull().sum())
```

```
User ID              0
User Name            0
Age                  0
Location             0
Registration Date    0
Phone                0
Email                0
Favorite Meal        0
Total Orders         0
dtype: int64
Session ID         0
User ID            0
Dish Name          0
Meal Type          0
Session Start      0
Session End        0
Duration (mins)    0
Session Rating     0
dtype: int64
Order ID        0
User ID         0
Order Date      0
Meal Type       0
Dish Name       0
Order Status    0
Amount (USD)    0
Time of Day     0
Rating          2
Session ID      0
dtype: int64
```

6. Handle Missing Values

In [11]:
```python
# Example: Fill missing values in Age with the average age
order_details['Rating'].fillna(order_details['Rating'].mean(), inplace=True)
```

7. Merge Cooking Sessions with User Details

In [12]:
```python
merged_data_1 = pd.merge(cooking_sessions, user_details, on='User ID', how='left')
```

8. Merge All Datasets

In [13]:
```python
final_data = pd.merge(merged_data_1, order_details, on='Session ID', how='left')
```

9. Identify and Drop Duplicate Columns

In [14]:
```python
# Check for duplicate columns by comparing the content
for col1 in final_data.columns:
    for col2 in final_data.columns:
        if col1 != col2 and final_data[col1].equals(final_data[col2]):
            print(f"Duplicate column found: {col1} and {col2}")

# Drop duplicate columns
# Example: If 'User ID_y' is identical to 'User ID_x', drop 'User ID_y'
final_data = final_data.drop(columns=['User ID_x'], errors='ignore')
final_data = final_data.drop(columns=['Dish Name_x'], errors='ignore')
final_data = final_data.drop(columns=['Meal Type_x'], errors='ignore')


# Verify the updated columns
print(final_data.columns)
```

```
Duplicate column found: User ID_x and User ID_y
Duplicate column found: Dish Name_x and Dish Name_y
Duplicate column found: Meal Type_x and Meal Type_y
Duplicate column found: User ID_y and User ID_x
Duplicate column found: Meal Type_y and Meal Type_x
Duplicate column found: Dish Name_y and Dish Name_x
Index(['Session ID', 'Session Start', 'Session End', 'Duration (mins)',
       'Session Rating', 'User Name', 'Age', 'Location', 'Registration Date',
       'Phone', 'Email', 'Favorite Meal', 'Total Orders', 'Order ID',
       'User ID_y', 'Order Date', 'Meal Type_y', 'Dish Name_y', 'Order Status',
       'Amount (USD)', 'Time of Day', 'Rating'],
      dtype='object')
```

10. Rename Columns for Consistency

In [15]:
```python
final_data = final_data.rename(columns={
    'Dish Name_y': 'Dish_Name',
    'Meal Type_y': 'Meal_Type',
    'User ID_y': 'User_ID'
})
print(final_data.columns)
```

```
Index(['Session ID', 'Session Start', 'Session End', 'Duration (mins)',
       'Session Rating', 'User Name', 'Age', 'Location', 'Registration Date',
       'Phone', 'Email', 'Favorite Meal', 'Total Orders', 'Order ID',
       'User_ID', 'Order Date', 'Meal_Type', 'Dish_Name', 'Order Status',
       'Amount (USD)', 'Time of Day', 'Rating'],
      dtype='object')
```

11. Check for Missing Values in Merged Dataset

In [16]:
```python
print(final_data.isnull().sum())
```

```
Session ID           0
Session Start        0
Session End          0
Duration (mins)      0
Session Rating       0
User Name            0
Age                  0
Location             0
Registration Date    0
Phone                0
Email                0
Favorite Meal        0
Total Orders         0
Order ID             0
User_ID              0
Order Date           0
Meal_Type            0
Dish_Name            0
Order Status         0
Amount (USD)         0
Time of Day          0
Rating               0
dtype: int64
```

12. Data Cleaning and Preprocessing

In [17]:
```python
final_data['Registration Date'] = pd.to_datetime(final_data['Registration Date'], errors='coerce')
final_data['Order Date'] = pd.to_datetime(final_data['Order Date'], errors='coerce')
final_data['Session Start'] = pd.to_datetime(final_data['Session Start'], errors='coerce')
final_data['Session End'] = pd.to_datetime(final_data['Session End'], errors='coerce')
```

In [18]:
```python
final_data['Age'] = pd.to_numeric(final_data['Age'], errors='coerce')
final_data['Duration (mins)'] = pd.to_numeric(final_data['Duration (mins)'], errors='coerce')
```

In [19]:
```python
final_data['Session_Date'] = pd.to_datetime(final_data['Session Start'], errors='coerce').dt.date
final_data['Order_Date'] = pd.to_datetime(final_data['Order Date'], errors='coerce').dt.date
```

13. Save Merged Dataset

In [20]:
```python
final_data.to_excel(r"C:\Assignment\Merged_Dataset.xlsx", index=False)
print("Merged dataset saved successfully!")
```

```
Merged dataset saved successfully!
```

14. Load Merged Dataset

In [21]:
```python
file_path = r"C:\Assignment\Merged_Dataset.xlsx"
Merged_Dataset = pd.read_excel(file_path)
```

In [23]:
```python
Merged_Dataset.head(10)
```

Out[23]:

	Session ID	Session Start	Session End	Duration (mins)	Session Rating	User Name	Age	Location	Registration Date	Phone	...	User_ID	Order Date	Meal_Type	Dish_Name	Order Status	Amount (USD)	Time of Day	Rating	Session_Date	Order_Date
0	S001	2024-12-01 19:00:00	2024-12-01 19:30:00	30	4.5	Alice Johnson	28	New York	2023-01-15	123-456-7890	...	U001	2024-12-01	Dinner	Spaghetti	Completed	15.0	Night	5.000000	2024-12-01	2024-12-01
1	S002	2024-12-01 12:00:00	2024-12-01 12:20:00	20	4.0	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	...	U002	2024-12-01	Lunch	Caesar Salad	Completed	10.0	Day	4.000000	2024-12-01	2024-12-01
2	S003	2024-12-02 19:30:00	2024-12-02 20:10:00	40	4.8	Charlie Lee	42	Chicago	2023-03-10	555-123-4567	...	U003	2024-12-02	Dinner	Grilled Chicken	Canceled	12.5	Night	4.285714	2024-12-02	2024-12-02
3	S004	2024-12-02 07:30:00	2024-12-02 08:00:00	30	4.2	Alice Johnson	28	New York	2023-01-15	123-456-7890	...	U001	2024-12-02	Breakfast	Pancakes	Completed	8.0	Morning	4.000000	2024-12-02	2024-12-02
4	S005	2024-12-03 13:00:00	2024-12-03 13:15:00	15	4.7	David Brown	27	San Francisco	2023-04-05	444-333-2222	...	U004	2024-12-03	Lunch	Caesar Salad	Completed	9.0	Day	4.000000	2024-12-03	2024-12-03
5	S006	2024-12-03 18:30:00	2024-12-03 19:00:00	30	4.3	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	...	U002	2024-12-03	Dinner	Spaghetti	Completed	14.0	Night	4.000000	2024-12-03	2024-12-03
6	S007	2024-12-04 18:00:00	2024-12-04 18:45:00	45	4.6	Emma White	30	Seattle	2023-05-22	777-888-9999	...	U005	2024-12-04	Dinner	Grilled Chicken	Completed	13.5	Night	4.000000	2024-12-04	2024-12-04
7	S008	2024-12-04 13:30:00	2024-12-04 13:50:00	20	4.4	Charlie Lee	42	Chicago	2023-03-10	555-123-4567	...	U003	2024-12-04	Lunch	Veggie Burger	Canceled	11.0	Day	4.285714	2024-12-04	2024-12-04
8	S009	2024-12-05 19:00:00	2024-12-05 19:40:00	40	4.9	Alice Johnson	28	New York	2023-01-15	123-456-7890	...	U001	2024-12-05	Dinner	Grilled Chicken	Completed	12.0	Night	5.000000	2024-12-05	2024-12-05
9	S010	2024-12-05 07:00:00	2024-12-05 07:10:00	10	4.1	Bob Smith	35	Los Angeles	2023-02-20	987-654-3210	...	U002	2024-12-05	Breakfast	Oatmeal	Completed	7.0	Morning	4.000000	2024-12-05	2024-12-05

10 rows × 24 columns

15. Popular Dishes Analysis

In [24]:
```python
popular_dishes = Merged_Dataset['Dish_Name'].value_counts().head(10)
popular_dishes.head(10)
```

Out[24]:
```
Dish_Name
Spaghetti          4
Grilled Chicken    4
Caesar Salad       3
Pancakes           2
Veggie Burger      2
Oatmeal            1
Name: count, dtype: int64
```

16. Demographic Influences Analysis

In [25]:
```python
age_meal_type = Merged_Dataset.groupby('Age')['Meal_Type'].value_counts().unstack().fillna(0)
location_meal_type = Merged_Dataset.groupby('Location')['Meal_Type'].value_counts().unstack().fillna(0)
age_meal_type.head(10)
```

Out[25]:

Meal_Type	Breakfast	Dinner	Lunch
Age			
25	0.0	1.0	0.0
27	0.0	1.0	1.0
28	1.0	2.0	0.0
30	0.0	1.0	1.0
31	0.0	0.0	1.0
35	1.0	1.0	1.0
38	0.0	1.0	0.0
42	1.0	1.0	1.0

In [26]:
```python
location_meal_type.head(10)
```

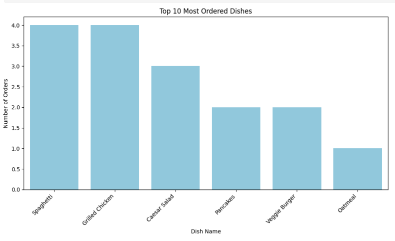
Out[26]:

Meal_Type	Breakfast	Dinner	Lunch
Location			
Austin	0.0	1.0	0.0
Boston	0.0	1.0	0.0
Chicago	1.0	1.0	1.0
Los Angeles	1.0	1.0	1.0
Miami	0.0	0.0	1.0
New York	1.0	2.0	0.0
San Francisco	0.0	1.0	1.0
Seattle	0.0	1.0	1.0

17. Visualize Top Dishes

In [27]:
```python
import matplotlib.pyplot as plt
import seaborn as sns


plt.figure(figsize=(10, 6))
sns.barplot(x=popular_dishes.index, y=popular_dishes.values, color='skyblue')  # Use a single color
plt.xticks(rotation=45, ha='right')
plt.title('Top 10 Most Ordered Dishes')
plt.xlabel("Dish Name")
plt.ylabel("Number of Orders")
plt.tight_layout()
plt.show()
print("Bar chart showing the frequency of the top 10 most ordered dishes.")
```

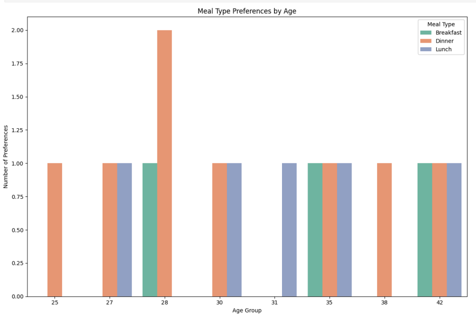


```
Bar chart showing the frequency of the top 10 most ordered dishes.
```

18. Visualize Meal Type Preferences by Age

In [30]:
```python
age_meal_type_simplified = age_meal_type.reset_index().melt(id_vars='Age', var_name='Meal Type', value_name='Count')

plt.figure(figsize=(12, 8))
sns.barplot(x='Age', y='Count', hue='Meal Type', data=age_meal_type_simplified, palette='Set2')
plt.title('Meal Type Preferences by Age')
plt.xlabel('Age Group')
plt.ylabel('Number of Preferences')
plt.legend(title='Meal Type')
plt.tight_layout()
plt.show()
print("Grouped bar chart showing meal type preferences across different age groups.")
```



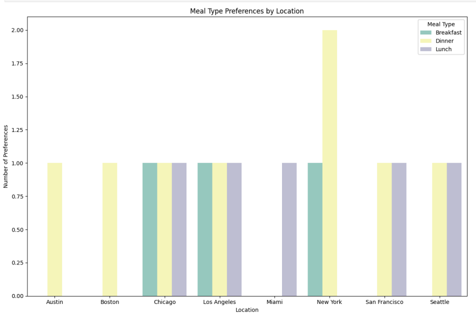
```
Grouped bar chart showing meal type preferences across different age groups.
```

19. Visualize Meal Type Preferences by Location

In [31]:
```python
location_meal_type_simplified = location_meal_type.reset_index().melt(id_vars='Location', var_name='Meal Type', value_name='Count')

plt.figure(figsize=(12, 8))
sns.barplot(x='Location', y='Count', hue='Meal Type', data=location_meal_type_simplified, palette='Set3')
plt.title('Meal Type Preferences by Location')
plt.xlabel('Location')
plt.ylabel('Number of Preferences')
plt.legend(title='Meal Type')
plt.tight_layout()
plt.show()
print("Grouped bar chart showing meal type preferences across different locations.")

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```



```
Grouped bar chart showing meal type preferences across different locations.
```

20. Analyze Order Status

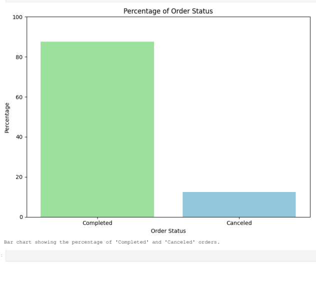
In [32]:
```python
if 'Order Status' in Merged_Dataset.columns:

    status_counts = Merged_Dataset['Order Status'].value_counts()
    status_percentages = Merged_Dataset['Order Status'].value_counts(normalize=True) * 100


    status_summary = pd.DataFrame({
        'Order Status': status_counts.index,
        'Count': status_counts.values,
        'Percentage': status_percentages.values
    })


    plt.figure(figsize=(8, 6))
    sns.barplot(
        x='Order Status',
        y='Percentage',
        data=status_summary,
        hue='Order Status',
        palette=['lightgreen', 'skyblue']
    )
    plt.title('Percentage of Order Status')
    plt.xlabel('Order Status')
    plt.ylabel('Percentage')
    plt.ylim(0, 100)  # Set y-axis to percentage range
    plt.tight_layout()
    plt.show()

    print("Bar chart showing the percentage of 'Completed' and 'Canceled' orders.")
else:
    print("Error: 'Order Status' column not found in the dataset.")
```



```
Bar chart showing the percentage of 'Completed' and 'Canceled' orders.
```

In [ ]: