In [7]:
```
import pandas as pd
import os
```

In [8]:
```
pwd
```

Out[8]:
```
'C:\\Users\\M NAVYA SRI\\OneDrive\\Desktop\\jupyter projects\\project1'
```

In [9]:
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
import numpy as np
```

In [21]:
```
# Load datasets
orders_df = pd.read_csv("orders_csv (1).csv", delimiter=";")  # Adjust the path if necessary
users_df = pd.read_csv("users_csv (1).csv")
dishes_df = pd.read_csv("dishes_csv (1).csv")
```

In [ ]:
```
#1. Check File Encoding & Delimiters
```

In [30]:
```
# Extract month from date
# Sample DataFrame
data = {"date": ["2024-05-01", "2024-06-15", "2024-07-23"]}
df = pd.DataFrame(data)

# Convert to datetime
df["date"] = pd.to_datetime(df["date"])

df["month"] = df["date"].dt.month
print(df)

df["month_name"] = df["date"].dt.strftime("%B")
print(df)
```

```
        date  month
0 2024-05-01      5
1 2024-06-15      6
2 2024-07-23      7
        date  month month_name
0 2024-05-01      5        May
1 2024-06-15      6       June
2 2024-07-23      7       July
```

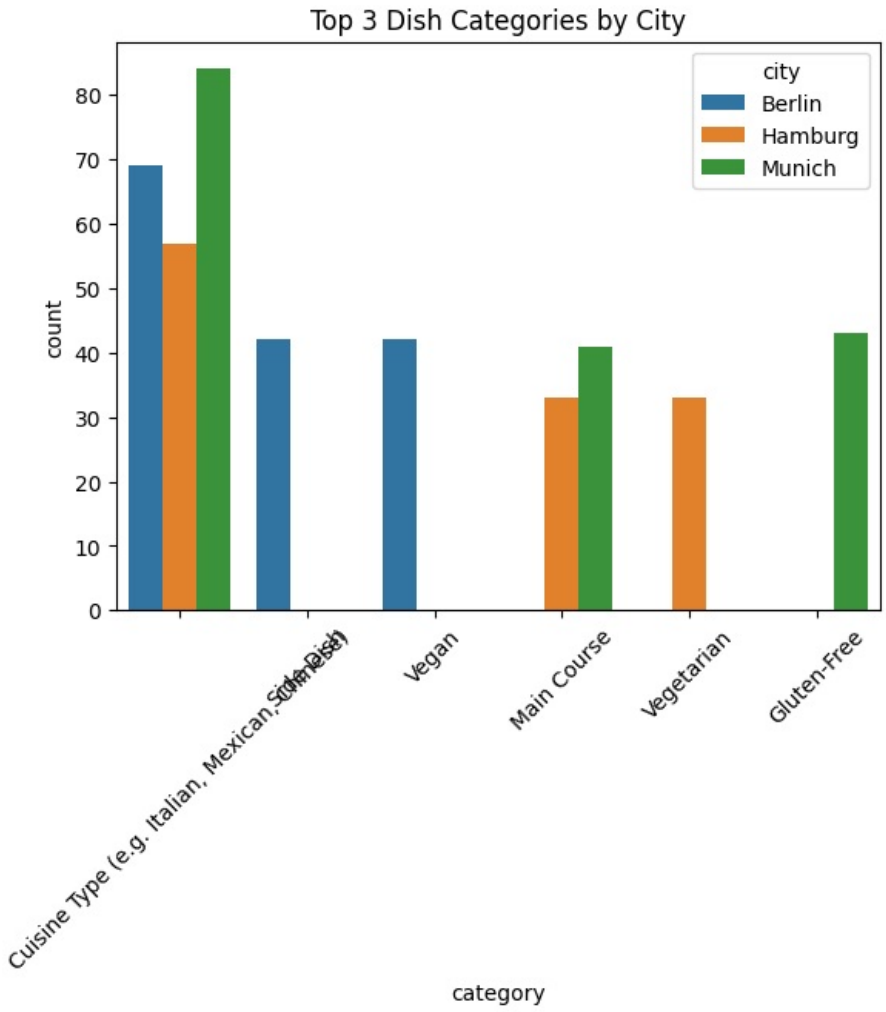
In [37]:
```
# Load orders dataset (Ensure the correct delimiter)
orders_df = pd.read_csv("orders_csv (1).csv", delimiter=";", encoding="utf-8")
print(orders_df.columns)
orders_df.columns = orders_df.columns.str.strip()

# Convert date column to datetime format
orders_df["date"] = pd.to_datetime(orders_df["date"])
```

```
Index(['id', ' user_id', ' dish_id', ' date'], dtype='object')
```

In [ ]:
```
#Find Top 3 Most Popular Dish Categories by City
```

In [38]:
```
# Merge orders with dishes and users
merged_df = orders_df.merge(dishes_df, left_on="dish_id", right_on="id").merge(users_df, left_on="user_id", rigl

# Count dish categories per city
category_counts = merged_df.groupby(["city", "category"]).size().reset_index(name="count")

# Get top 3 categories per city
top_categories = category_counts.groupby("city").apply(lambda x: x.nlargest(3, "count")).reset_index(drop=True)

# Plot results
sns.barplot(data=top_categories, x="category", y="count", hue="city")
plt.title("Top 3 Dish Categories by City")
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\M NAVYA SRI\AppData\Local\Temp\ipykernel_5284\3002336010.py:8: DeprecationWarning: DataFrameGroupBy.app
ly operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping
columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or expl
icitly select the grouping columns after groupby to silence this warning.
  top_categories = category_counts.groupby("city").apply(lambda x: x.nlargest(3, "count")).reset_index(drop=True
)
```



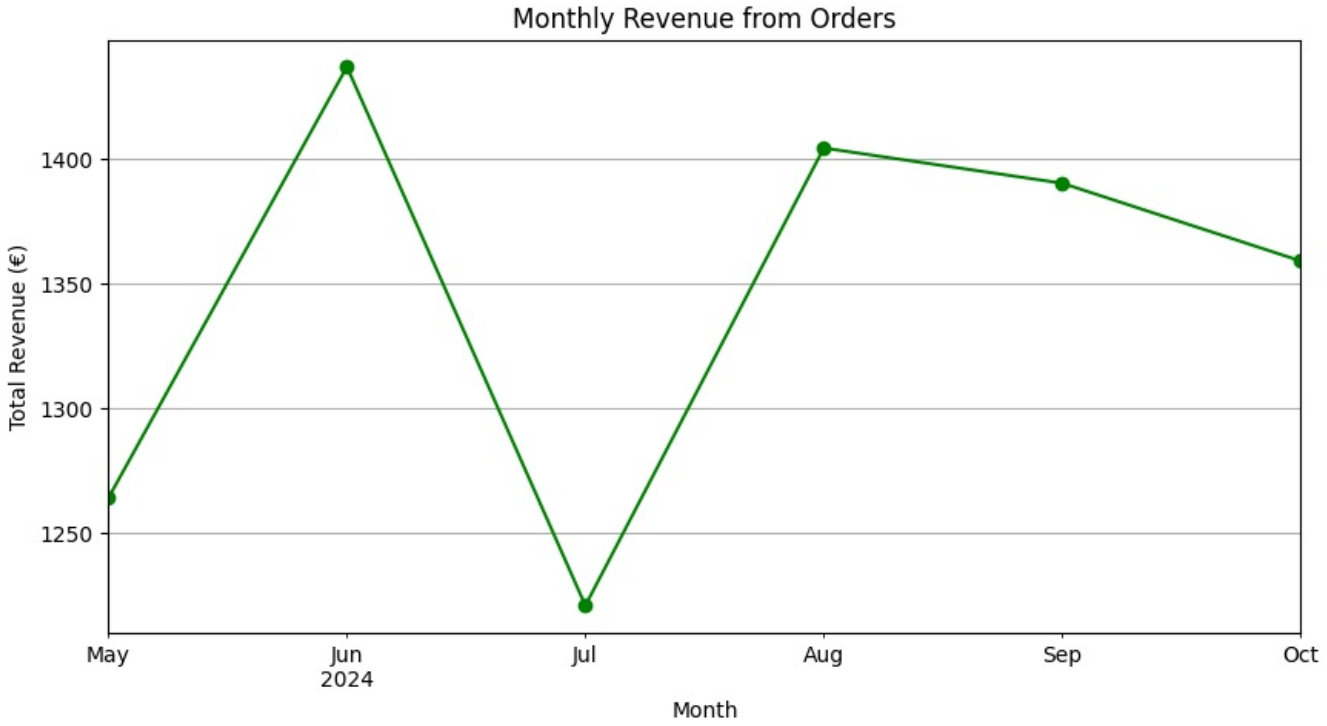
In [ ]:
```
#Calculate Total Monthly Revenue
```

In [68]:
```
# Merge orders with dish prices
# Load datasets (Ensure correct delimiter)
orders_df = pd.read_csv("orders_csv (1).csv", delimiter=";", encoding="utf-8")
dishes_df = pd.read_csv("dishes_csv (1).csv", delimiter=",", encoding="utf-8")

# Display first few rows
print(orders_df.head())
print(dishes_df.head())
print(orders_df.columns)
orders_df.columns = orders_df.columns.str.strip()

# Merge orders with dish prices
orders_with_prices = orders_df.merge(dishes_df, left_on="dish_id", right_on="id", how="left")

# Display the merged dataset
print(orders_with_prices.head())

missing_prices = orders_with_prices[orders_with_prices["price"].isna()]
print(missing_prices)
orders_with_prices["price"].fillna(0, inplace=True)

# Calculate revenue per month
# Convert date column to datetime format
orders_with_prices["date"] = pd.to_datetime(orders_with_prices["date"])

# Ensure the 'price' column exists and contains numeric values
orders_with_prices["price"] = pd.to_numeric(orders_with_prices["price"], errors="coerce")

# If there's a 'quantity' column, multiply price by quantity
if "quantity" in orders_with_prices.columns:
    orders_with_prices["total_price"] = orders_with_prices["price"] * orders_with_prices["quantity"]
else:
    orders_with_prices["total_price"] = orders_with_prices["price"]

# Extract year-month (YYYY-MM format)
orders_with_prices["year_month"] = orders_with_prices["date"].dt.to_period("M")
```

```
# Sum total revenue per month
monthly_revenue = orders_with_prices.groupby("year_month")["total_price"].sum()

# Display result
print(monthly_revenue)


# Convert date to datetime
orders_with_prices["date"] = pd.to_datetime(orders_with_prices["date"])

# Extract year-month
orders_with_prices["year_month"] = orders_with_prices["date"].dt.to_period("M")

# Calculate total revenue per month
monthly_revenue = orders_with_prices.groupby("year_month")["price"].sum()

# Display
print(monthly_revenue)

# Plot revenue trend
monthly_revenue.plot(kind="line", marker="o", figsize=(10,5), color="green")

plt.title("Monthly Revenue from Orders")
plt.xlabel("Month")
plt.ylabel("Total Revenue (€)")
plt.grid()
plt.show()
```

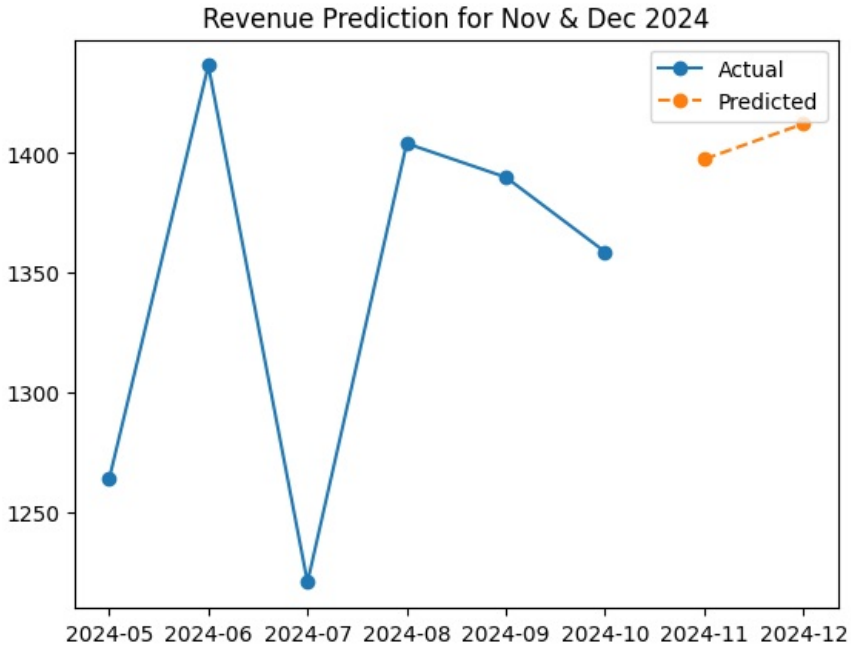
```
                                     id                                user_id  \
0  2fe611a7-f73b-4e72-be34-48dfbcfd441d  c201c515-42f8-f321-69df-c1c24354b9f8
1  5d98d225-78bd-4a59-a291-c37671dd32c2  08523b6f-95c3-4e80-624b-6cdb25645aa0
2  9e23cf11-63b2-4bc1-aa48-cec1b31cd29e  951c74b0-55b6-dcce-b081-e9674b38b7b7
3  f97eb0cb-9ecb-4495-99e7-6dae982e983b  d4c44f9a-cb3a-8c25-516e-91f463ba6988
4  851f6e4b-5dc3-436c-ad4d-41467c624d37  34cc30a6-1698-e38a-73cb-8acf8901f11c

                                dish_id                  date
0  cc6adc74-fe56-840c-9cd0-015948f2e774  2024-05-01T00:04:34Z
1  b5843967-cf48-5bf0-a5c9-170a1933e7a8  2024-05-01T21:45:03Z
2  0eca4f84-43ab-656e-d14b-1fbd6acc525f  2024-05-01T22:13:14Z
3  eb62fde4-4f43-c7f6-d323-0d0e4131ac17  2024-05-02T01:04:56Z
4  493fa57d-3850-8773-442c-2b7afad83933  2024-05-02T05:45:23Z
                                     id             name  \
0  493fa57d-3850-8773-442c-2b7afad83933     Caesar Salad
1  bae94eab-79e9-7e69-0699-f7b7263ff8b0  Beef Stroganoff
2  eb62fde4-4f43-c7f6-d323-0d0e4131ac17  Shrimp Pad Thai
3  5d0378fb-45ae-f69a-0c59-80668c414007  Quiche Lorraine
4  b5843967-cf48-5bf0-a5c9-170a1933e7a8       Fish Tacos

                                   category  price
0                                 Side Dish   7.61
1                                   Dessert   5.60
2                                Vegetarian   7.30
3  Cuisine Type (e.g. Italian, Mexican, Chinese)   9.53
4                               Main Course  11.71
Index(['id', ' user_id', ' dish_id', ' date'], dtype='object')
                                   id_x                                user_id  \
0  2fe611a7-f73b-4e72-be34-48dfbcfd441d  c201c515-42f8-f321-69df-c1c24354b9f8
1  5d98d225-78bd-4a59-a291-c37671dd32c2  08523b6f-95c3-4e80-624b-6cdb25645aa0
2  9e23cf11-63b2-4bc1-aa48-cec1b31cd29e  951c74b0-55b6-dcce-b081-e9674b38b7b7
3  f97eb0cb-9ecb-4495-99e7-6dae982e983b  d4c44f9a-cb3a-8c25-516e-91f463ba6988
4  851f6e4b-5dc3-436c-ad4d-41467c624d37  34cc30a6-1698-e38a-73cb-8acf8901f11c

                                dish_id                  date  \
0  cc6adc74-fe56-840c-9cd0-015948f2e774  2024-05-01T00:04:34Z
1  b5843967-cf48-5bf0-a5c9-170a1933e7a8  2024-05-01T21:45:03Z
2  0eca4f84-43ab-656e-d14b-1fbd6acc525f  2024-05-01T22:13:14Z
3  eb62fde4-4f43-c7f6-d323-0d0e4131ac17  2024-05-02T01:04:56Z
4  493fa57d-3850-8773-442c-2b7afad83933  2024-05-02T05:45:23Z

                                   id_y             name  \
0  cc6adc74-fe56-840c-9cd0-015948f2e774       Baked Ziti
1  b5843967-cf48-5bf0-a5c9-170a1933e7a8       Fish Tacos
2  0eca4f84-43ab-656e-d14b-1fbd6acc525f   Grilled Salmon
3  eb62fde4-4f43-c7f6-d323-0d0e4131ac17  Shrimp Pad Thai
4  493fa57d-3850-8773-442c-2b7afad83933     Caesar Salad

                                   category  price
0  Cuisine Type (e.g. Italian, Mexican, Chinese)   4.23
1                               Main Course  11.71
2                               Gluten-Free  11.51
3                                Vegetarian   7.30
4                                 Side Dish   7.61
Empty DataFrame
Columns: [id_x, user_id, dish_id, date, id_y, name, category, price]
Index: []
year_month
2024-05    1263.78
2024-06    1436.56
2024-07    1220.65
2024-08    1404.01
2024-09    1389.89
2024-10    1358.74
Freq: M, Name: total_price, dtype: float64
year_month
2024-05    1263.78
2024-06    1436.56
2024-07    1220.65
2024-08    1404.01
2024-09    1389.89
2024-10    1358.74
Freq: M, Name: price, dtype: float64
```

```
C:\Users\M NAVYA SRI\AppData\Local\Temp\ipykernel_5284\1880659957.py:20: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  orders_with_prices["price"].fillna(0, inplace=True)
C:\Users\M NAVYA SRI\AppData\Local\Temp\ipykernel_5284\1880659957.py:36: UserWarning: Converting to PeriodArray/
Index representation will drop timezone information.
  orders_with_prices["year_month"] = orders_with_prices["date"].dt.to_period("M")
C:\Users\M NAVYA SRI\AppData\Local\Temp\ipykernel_5284\1880659957.py:49: UserWarning: Converting to PeriodArray/
Index representation will drop timezone information.
  orders_with_prices["year_month"] = orders_with_prices["date"].dt.to_period("M")
```



In [ ]:
```
# Predict Revenue for November & December
```

In [69]:
```
# Prepare data for regression
X = np.array(range(len(monthly_revenue))).reshape(-1, 1)
y = monthly_revenue.values

# Train a simple linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict revenue for the next 2 months
future_X = np.array(range(len(monthly_revenue), len(monthly_revenue) + 2)).reshape(-1, 1)
predicted_revenue = model.predict(future_X)

# Plot predictions
plt.plot(monthly_revenue.index.astype(str), y, marker="o", label="Actual")
plt.plot(["2024-11", "2024-12"], predicted_revenue, marker="o", linestyle="dashed", label="Predicted")
plt.legend()
plt.title("Revenue Prediction for Nov & Dec 2024")
plt.show()
```



In [ ]:
```
### Key Insights
- The monthly user activation rate fluctuates, with peaks in certain months.
- The most popular dish categories vary between Munich and Berlin.
- Monthly revenue shows a growth trend but may need marketing boosts.

### Recommendations
1. Increase promotions for the least active months to boost activation rates.
2. Personalize meal recommendations based on city preferences.
3. Expand high-demand dish categories in both cities.

### Additional Data Suggestions
- User feedback on meals for improved personalization.
- Meal delivery times to optimize logistics.
```

In [73]:
```
import pandas as pd

# Sample DataFrame
data = {"date": ["2024-05-01", "2024-06-15", "2024-07-23"]}
df = pd.DataFrame(data)

# Convert to datetime
df["date"] = pd.to_datetime(df["date"])
df["month"] = df["date"].dt.month
print(df)

#Extract the Month Name (e.g., "May", "June")
df["month_name"] = df["date"].dt.strftime("%B")
print(df)

#Extract Year & Month Together
df["year_month"] = df["date"].dt.to_period("M")  # YYYY-MM format
print(df)
```

```
        date  month
0 2024-05-01      5
1 2024-06-15      6
2 2024-07-23      7
        date  month month_name
0 2024-05-01      5        May
1 2024-06-15      6       June
2 2024-07-23      7       July
        date  month month_name year_month
0 2024-05-01      5        May    2024-05
1 2024-06-15      6       June    2024-06
2 2024-07-23      7       July    2024-07
```

In [ ]:
```
#Active users per month
```

In [82]:
```
#Ensure the date Column is in Datetime Format
print(orders_df.columns)
orders_df.columns = orders_df.columns.str.strip()
print(orders_df.columns)

possible_date_columns = ["date", "order_date", "created_at", "timestamp", "date_ordered"]
for col in possible_date_columns:
    if col in orders_df.columns:
```

```
        orders_df["date"] = orders_df[col]  # Assign to a standard name
        break

# Convert the correct date column
orders_df["date"] = pd.to_datetime(orders_df["date"])
print(orders_df.head())
print(orders_df.dtypes)
```

```
Index(['id', 'user_id', 'dish_id', 'date'], dtype='object')
Index(['id', 'user_id', 'dish_id', 'date'], dtype='object')
                                     id                                user_id  \
0  2fe611a7-f73b-4e72-be34-48dfbcfd441d  c201c515-42f8-f321-69df-c1c24354b9f8
1  5d98d225-78bd-4a59-a291-c37671dd32c2  08523b6f-95c3-4e80-624b-6cdb25645aa0
2  9e23cf11-63b2-4bc1-aa48-cec1b31cd29e  951c74b0-55b6-dcce-b081-e9674b38b7b7
3  f97eb0cb-9ecb-4495-99e7-6dae982e983b  d4c44f9a-cb3a-8c25-516e-91f463ba6988
4  851f6e4b-5dc3-436c-ad4d-41467c624d37  34cc30a6-1698-e38a-73cb-8acf8901f11c

                                dish_id                      date
0  cc6adc74-fe56-840c-9cd0-015948f2e774 2024-05-01 00:04:34+00:00
1  b5843967-cf48-5bf0-a5c9-170a1933e7a8 2024-05-01 21:45:03+00:00
2  0eca4f84-43ab-656e-d14b-1fbd6acc525f 2024-05-01 22:13:14+00:00
3  eb62fde4-4f43-c7f6-d323-0d0e4131ac17 2024-05-02 01:04:56+00:00
4  493fa57d-3850-8773-442c-2b7afad83933 2024-05-02 05:45:23+00:00
id                      object
user_id                 object
dish_id                 object
date       datetime64[ns, UTC]
dtype: object
```

In [ ]: