Food Delivery Analytics Hackathon

Overview

This notebook implements an end-to-end data engineering pipeline to merge three disparate datasets representing a food delivery system.

Problem Statement: Merge transactional (CSV), user (JSON), and restaurant (SQL) data into a single source of truth for analysis.

Tools Used: Python, Pandas, Regex

---

1. Dataset Exploration

In [1]:
```python
import pandas as pd
import json
import re
import os

# Define file paths
ORDERS_PATH = "orders.csv"
USERS_PATH = "users.json"
RESTAURANTS_PATH = "restaurants.sql"

# 1. Load Orders (CSV)
print("Loading orders.csv...")
orders = pd.read_csv(ORDERS_PATH)
print(f"Orders Shape: {orders.shape}")
display(orders.head(3))

# 2. Load Users (JSON)
print("\nLoading users.json...")
with open(USERS_PATH, 'r') as f:
    users_data = json.load(f)
users = pd.DataFrame(users_data)
```

```
print(f"Users Shape: {users.shape}")
display(users.head(3))

# 3. Parse Restaurants (SQL)
print("\nParsing restaurants.sql...")
restaurant_data = []
with open(RESTAURANTS_PATH, 'r') as f:
    for line in f:
        if line.strip().startswith("INSERT INTO"):
            # Regex to extract values: (id, 'Name', 'Cuisine', Rating)
            match = re.search(r"VALUES\s*\((\d+),\s*'([^']*)',\s*'([^']*)',\s*([0-9.]+)\);", line)
            if match:
                restaurant_data.append({
                    "restaurant_id": int(match.group(1)),
                    "restaurant_name": match.group(2),
                    "cuisine": match.group(3),
                    "rating": float(match.group(4))
                })

restaurants = pd.DataFrame(restaurant_data)
print(f"Restaurants Shape: {restaurants.shape}")
display(restaurants.head(3))
```

```
Loading orders.csv...
Orders Shape: (10000, 6)
```

	order_id	user_id	restaurant_id	order_date	total_amount	restaurant_name
0	1	2508	450	18-02-2023	842.97	New Foods Chinese
1	2	2693	309	18-01-2023	546.68	Ruchi Curry House Multicuisine
2	3	2084	107	15-07-2023	163.93	Spice Kitchen Punjabi

```
Loading users.json...
Users Shape: (3000, 4)
```

	user_id	name	city	membership
0	1	User_1	Chennai	Regular
1	2	User_2	Pune	Gold
2	3	User_3	Bangalore	Gold

```
Parsing restaurants.sql...
Restaurants Shape: (500, 4)
```

	restaurant_id	restaurant_name	cuisine	rating
0	1	Restaurant_1	Chinese	4.8
1	2	Restaurant_2	Indian	4.1
2	3	Restaurant_3	Mexican	4.3

2. ETL Implementation

We will now clean the data and perform a LEFT JOIN to create the master dataset.

Logic: `orders` LEFT JOIN `users` ON `user_id` LEFT JOIN `restaurants` ON `restaurant_id`

In [2]:
```python
# --- Data Cleaning ---
# Ensure ID columns are integers
orders['user_id'] = orders['user_id'].astype(int)
orders['restaurant_id'] = orders['restaurant_id'].astype(int)
users['user_id'] = users['user_id'].astype(int)
restaurants['restaurant_id'] = restaurants['restaurant_id'].astype(int)

# Handle duplicates in master tables (if any)
users = users.drop_duplicates(subset=['user_id'])
restaurants = restaurants.drop_duplicates(subset=['restaurant_id'])

# --- Merging Datasets ---

# Merge 1: Orders + Users
# Rename user city/name to avoid conflicts if needed, though here schemas differ enough
```

```
# users columns: user_id, name, city, membership
merged_df = pd.merge(orders, users, on='user_id', how='left')

# Rename columns for clarity immediately after merge
merged_df = merged_df.rename(columns={
    'name': 'user_name',
    'city': 'user_city',
    'membership': 'user_membership'
})

# Merge 2: Result + Restaurants
final_df = pd.merge(merged_df, restaurants, on='restaurant_id', how='left')

# Rename restaurant columns
# orders.csv has 'restaurant_name', restaurants.sql has 'restaurant_name'
# We prioritize the master data (sql) but 'restaurant_name_y' will be created.

final_df = final_df.rename(columns={
    'cuisine': 'restaurant_cuisine',
    'rating': 'restaurant_rating',
    'restaurant_name_y': 'restaurant_name_master',
    'restaurant_name_x': 'restaurant_name_trans'
})

# Use master name if available, else transactional name
final_df['restaurant_name'] = final_df['restaurant_name_master'].fillna(final_df['restaurant_name_trans'])
final_df.drop(columns=['restaurant_name_master', 'restaurant_name_trans'], inplace=True)

print("ETL Complete. Final Dataset created.")
```

```
ETL Complete. Final Dataset created.
```

3. Merge Validation

Validating the integrity of the merge to ensure no data loss and correct schema.

In [3]:
```
# 1. Row Count Validation
initial_count = len(orders)
final_count = len(final_df)
print(f"Initial Orders: {initial_count}")
print(f"Final Rows: {final_count}")
```

```
assert initial_count == final_count, "Row count mismatch! Check for duplicates in join keys."

# 2. ID Range Checks
print(f"User IDs: {final_df['user_id'].min()} - {final_df['user_id'].max()}")
print(f"Restaurant IDs: {final_df['restaurant_id'].min()} - {final_df['restaurant_id'].max()}")

# 3. Null Values Check
print("\nNull Values Check:")
print(final_df.isnull().sum())

print("\nValidation Passed.")
```

```
Initial Orders: 10000
Final Rows: 10000
User IDs: 1 - 3000
Restaurant IDs: 1 - 500

Null Values Check:
order_id              0
user_id               0
restaurant_id         0
order_date            0
total_amount          0
user_name             0
user_city             0
user_membership       0
restaurant_cuisine    0
restaurant_rating     0
restaurant_name       0
dtype: int64

Validation Passed.
```

4. Final Dataset Overview

In [4]:

```
display(final_df.head())
final_df.info()
```

	order_id	user_id	restaurant_id	order_date	total_amount	user_name	user_city	user_membership	restaurant_cuisine	restau
0	1	2508	450	18-02-2023	842.97	User_2508	Hyderabad	Regular	Mexican	
1	2	2693	309	18-01-2023	546.68	User_2693	Pune	Regular	Indian	
2	3	2084	107	15-07-2023	163.93	User_2084	Chennai	Gold	Mexican	
3	4	319	224	04-10-2023	1155.97	User_319	Bangalore	Gold	Chinese	
4	5	1064	293	25-12-2023	1321.91	User_1064	Pune	Regular	Italian	

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   order_id            10000 non-null  int64
 1   user_id             10000 non-null  int32
 2   restaurant_id       10000 non-null  int32
 3   order_date          10000 non-null  object
 4   total_amount        10000 non-null  float64
 5   user_name           10000 non-null  object
 6   user_city           10000 non-null  object
 7   user_membership     10000 non-null  object
 8   restaurant_cuisine  10000 non-null  object
 9   restaurant_rating   10000 non-null  float64
 10  restaurant_name     10000 non-null  object
dtypes: float64(2), int32(2), int64(1), object(6)
memory usage: 781.4+ KB
```

5. MCQ Analysis

Q1: Gold Revenue by City

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

In [5]:
```
gold_users = final_df[final_df['user_membership'] == 'Gold']
gold_city_aov = gold_users.groupby('user_city')['total_amount'].mean().sort_values(ascending=False)
print(gold_city_aov)
print(f"\nAnswer: {gold_city_aov.idxmax()}")
```

```
user_city
Chennai      808.459080
Hyderabad    806.421034
Bangalore    793.223756
Pune         781.162243
Name: total_amount, dtype: float64

Answer: Chennai
```

Q2: Highest Average Order Value by Cuisine

Which cuisine has the highest average order value?

In [6]:
```
cuisine_aov = final_df.groupby('restaurant_cuisine')['total_amount'].mean().sort_values(ascending=False)
print(cuisine_aov)
print(f"\nAnswer: {cuisine_aov.idxmax()}")
```

```
restaurant_cuisine
Mexican    808.021344
Italian    799.448578
Indian     798.466011
Chinese    798.389020
Name: total_amount, dtype: float64

Answer: Mexican
```

Q3: Users with > 1000 Total Spend

How many distinct users placed orders worth more than ₹1000 in total?

In [7]:
```
user_spend = final_df.groupby('user_id')['total_amount'].sum()
high_value_users = (user_spend > 1000).sum()
```

```
print(f"Count: {high_value_users}")
```

```
Count: 2544
```

Q4: Revenue by Rating Range

Which restaurant rating range generated the highest total revenue?

In [8]:
```
bins = [2.9, 3.5, 4.0, 4.5, 5.0]
labels = ['3.0 - 3.5', '3.6 - 4.0', '4.1 - 4.5', '4.6 - 5.0']
final_df['rating_range'] = pd.cut(final_df['restaurant_rating'], bins=bins, labels=labels)

revenue_rating = final_df.groupby('rating_range', observed=False)['total_amount'].sum().sort_values(ascending=False)
print(revenue_rating)
print(f"\nAnswer: {revenue_rating.idxmax()}")
```

```
rating_range
4.6 - 5.0    2197030.75
3.0 - 3.5    2136772.70
4.1 - 4.5    1960326.26
3.6 - 4.0    1717494.41
Name: total_amount, dtype: float64

Answer: 4.6 - 5.0
```

Q5: Cuisine with Lowest Distinct Restaurants but High Revenue

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

In [9]:
```
cuisine_stats = final_df.groupby('restaurant_cuisine').agg({
    'restaurant_id': 'nunique',
    'total_amount': 'sum'
}).sort_values('restaurant_id')
print(cuisine_stats)
print(f"\nAnswer: {cuisine_stats.index[0]}")
```

```
                    restaurant_id  total_amount
restaurant_cuisine
Chinese                       120    1930504.65
Indian                        126    1971412.58
Italian                       126    2024203.80
Mexican                       128    2085503.09

Answer: Chinese
```

Q6: Percentage of Gold Member Orders

What percentage of total orders were placed by Gold members?

In [10]:
```python
gold_orders = len(final_df[final_df['user_membership'] == 'Gold'])
total = len(final_df)
pct = (gold_orders / total) * 100
print(f"Percentage: {pct:.2f}%")
print(f"Rounded: {round(pct)}%")
```

```
Percentage: 49.87%
Rounded: 50%
```

Q7: High AOV Low Volume Restaurant

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

In [11]:
```python
rest_stats = final_df.groupby('restaurant_name').agg({
    'order_id': 'count',
    'total_amount': 'mean'
})
low_vol = rest_stats[rest_stats['order_id'] < 20].sort_values('total_amount', ascending=False)
print(low_vol.head())
print(f"\nAnswer: {low_vol.index[0]}")
```

```
                 order_id  total_amount
restaurant_name
Restaurant_294         13   1040.222308
Restaurant_262         18   1029.473333
Restaurant_77          12   1029.180833
Restaurant_193         15   1026.306667
Restaurant_7           16   1002.140625

Answer: Restaurant_294
```

Q8: Top Revenue Combination

Which combination of membership and cuisine contributes the highest revenue?

In [12]:
```
combo_rev = final_df.groupby(['user_membership', 'restaurant_cuisine'])['total_amount'].sum().sort_values(ascending=F
print(combo_rev.head())
print(f"\nAnswer: {combo_rev.index[0]}")
```

```
user_membership  restaurant_cuisine
Regular          Mexican               1072943.30
                 Italian               1018424.75
Gold             Mexican               1012559.79
                 Italian               1005779.05
Regular          Indian                 992100.27
Name: total_amount, dtype: float64

Answer: ('Regular', 'Mexican')
```

Q9: Revenue by Quarter

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

In [13]:
```
final_df['order_date'] = pd.to_datetime(final_df['order_date'], dayfirst=True)
final_df['quarter'] = final_df['order_date'].dt.quarter
revenue_q = final_df.groupby('quarter')['total_amount'].sum().sort_values(ascending=False)
q_map = {1: 'Q1', 2: 'Q2', 3: 'Q3', 4: 'Q4'}
print(revenue_q.rename(index=q_map))
print(f"\nAnswer: {q_map[revenue_q.idxmax()]}")
```

```
quarter
Q3    2037385.10
Q4    2018263.66
Q1    2010626.64
Q2    1945348.72
Name: total_amount, dtype: float64

Answer: Q3
```

6. Numerical Answers

In [14]:
```python
# 1. Total Gold Orders
print(f"Total Gold Orders: {len(final_df[final_df['user_membership'] == 'Gold'])}")

# 2. Total Hyderabad Revenue
hyd_rev = final_df[final_df['user_city'] == 'Hyderabad']['total_amount'].sum()
print(f"Hyderabad Revenue: {round(hyd_rev)}")

# 3. Distinct Users
print(f"Distinct Users: {final_df['user_id'].nunique()}")

# 4. Gold Member AOV
gold_aov = final_df[final_df['user_membership'] == 'Gold']['total_amount'].mean()
print(f"Gold Member AOV: {gold_aov:.2f}")

# 5. Orders with Rating >= 4.5
print(f"Orders Rating >= 4.5: {len(final_df[final_df['restaurant_rating'] >= 4.5])}")

# 6. Orders in Top Gold Revenue City
top_gold_city = final_df[final_df['user_membership'] == 'Gold'].groupby('user_city')['total_amount'].sum().idxmax()
gold_orders_top_city = len(final_df[(final_df['user_membership'] == 'Gold') & (final_df['user_city'] == top_gold_city
print(f"Gold Orders in {top_gold_city}: {gold_orders_top_city}")
```

```
Total Gold Orders: 4987
Hyderabad Revenue: 1889367
Distinct Users: 2883
Gold Member AOV: 797.15
Orders Rating >= 4.5: 3374
Gold Orders in Chennai: 1337
```

7. Conclusion

This notebook successfully loaded, cleaned, and merged the datasets. All validation checks passed, ensuring the integrity of the analysis. The specific business questions have been answered based on this single source of truth.