REPORT

IT-300

BUSINESS INTELLIGENCE AND DATABASE MANAGEMENT SYSTEMS

Business Intelligence Research Restaurants & Customers Rating Data in Mexico

Authors:
Rayen Nasraoui
Rayen Chtioui

Submitted to:
Prof. Ons Abdelkhalek

Contents

List of Figures

1 Introduction

This business intelligence project is focused on analyzing restaurant and consumer rating data. The study aims to find out how different factors, such as personality, activity, marital status, and restaurant types influence consumer ratings.

By understanding these relationships, we can gain valuable insights into consumer behavior and preferences, which can help restaurants improve their offerings and ultimately drive more business. Through this project, we aim to improve overall customer satisfaction and build better strategies for the restaurant industry.

2 Implementation

2.1 Data Gathering

We extracted the Restaurants and Consumers ratings dataset in Mexico from the UCI Machine Learning Repository. This is a link to the dataset.

2.2 Data Preparation

For the data preparation, we used Python to manipulate data and configure it for the data warehouse. At first, we converted all the data into one format (JSON) from different formats (CSV, Excel, JSON). We changed the structure of the data so that it could be easily manipulated and managed. At first, we created a script that allows for cleaning and manipulation of `chefmozhours4.csv` file because it contains days where some restaurants recorded to work for different hours in a day

```
132103|11:00-16:00,16:00-13:00,16:00-12:00,16:00-21:00,|Sun,
132103|11:00-16:00,16:00-13:00,16:00-12:00,16:00-21:00,|Mon,Tue,Wed,Thu,Fri,
132030|12:00-15:00,15:00-21:00,|Sun,
132030|12:00-15:00,15:00-21:00,|Mon,Tue,Wed,Thu,Fri,
```

Such type of data might be confusing to use and they are a minority on the dataset. So they are considered outliers and have been removed to allow for simple and understandable data manipulation. The next step would be to create usable data for the data warehouse. For that, we used Pandas to transform data into a tabular structure and create relationships between them. `etl.ipynb` includes all the transformations done to the dataset to export them into the data warehouse.
Attached here is the code for some examples of the manipulations done:

```
user_table.replace("?",np.nan)
result_user=user_table.groupby('userID', as_index=False).agg({
    'latitude': 'mean',
    'longitude': 'mean',
    'drink_level': 'first',
    'dress_preference': 'first',
    'ambience': 'first',
    'transport': 'first',
    'marital_status': 'first',
    'hijos': 'first',
    'birth_year': 'mean',
    'interest': 'first',
    'personality': 'first',
    'religion': 'first',
    'activity': 'first',
    'color': 'first',
    'weight': 'mean',
    'budget': 'first',
    'height': 'mean',
    'Rcuisine': lambda x: list(set(x)),
    'Upayment': lambda x: list(set(x))
}).assign(birth_year = lambda x: x.birth_year.astype(int))
```

```
# Specific values to remove
outliers = ['Australian', 'Austrian', 'Basque', 'British', 'Burmese', 'Cajun-Creole',
    'Canadian', 'Chilean', 'Cuban', 'Dim_Sum', 'Doughnuts', 'Eclectic', 'Filipino',
    'Fusion', 'Hawaiian', 'Hungarian', 'Indian-Pakistani', 'Indigenous', 'Indonesian',
    'Irish', 'Israeli', 'Jamaican', 'Kosher', 'Lebanese', 'Malaysian', 'Moroccan',
    'North_African', 'Pacific_Northwest', 'Pacific_Rim', 'Peruvian', 'Polynesian',
    'Portuguese', 'Romanian', 'Russian-Ukrainian', 'Scandinavian', 'Southeast_Asian',
    'Swiss', 'Tapas', 'Tea_House', 'Tibetan',
    'Tunisian','Middle_Eastern','Moroccan','Organic-Healthy','Persian','Peruvian',
    'Polish','Polynesian','Portuguese','Regional','Romanian','Russian-Ukrainian',
    'Scandinavian','Seafood','Soup','Southeast_Asian','Southern','Southwestern','Spanish','Steaks',
    'Sushi','Swiss','Tapas','Tea_House','Tex-Mex','Thai','Tibetan',
    'Tunisian','Turkish','Vegetarian','Vietnamese']

# Iterating through the rows of the dataframe
for index, row in result_user.iterrows():
    for outlier in outliers:
        if outlier in row['Rcuisine']:
            row['Rcuisine'].remove(outlier)
result_user_cuisine = result_user[['userID','Rcuisine']].copy()
result_user = result_user.drop('Rcuisine', axis=1)
result_user_cuisine = result_user_cuisine.explode('Rcuisine')
result_user_payment = result_user[['userID','Upayment']].copy()
result_user = result_user.drop('Upayment', axis=1)
result_user_payment = result_user_payment.explode('Upayment')
result_user_cuisine = result_user_cuisine.dropna(axis=0)
display(result_user)
display(result_user_cuisine)
display(result_user_payment)
```

```
    valid_user_ids = set(result_user['userID'])
valid_place_ids = set(result_restaurent['placeID'])
valid_rows = rating.apply(lambda x: x['userID'] in valid_user_ids and x['placeID'] in
    valid_place_ids, axis=1)
display(valid_rows)

filtered_result_df = rating[valid_rows]
display(filtered_result_df)
```

```
    restaurent_table.replace("?",np.nan).head(20)
result_restaurent = restaurent_table.groupby('placeID', as_index=False).agg({
    'Rpayment': lambda x: list(set(x)),
    'Rcuisine': lambda x: list(set(x)),
    'hours': 'first',
    'days': 'first',
    'parking_lot': 'first'
})
result_restaurent_cuisine = result_restaurent[['placeID','Rcuisine']].copy()
result_restaurent = result_restaurent.drop('Rcuisine', axis=1)
result_restaurent_cuisine = result_restaurent_cuisine.explode('Rcuisine')
result_restaurent_payment = result_restaurent[['placeID','Rpayment']].copy()
result_restaurent = result_restaurent.drop('Rpayment', axis=1)
result_restaurent_payment = result_restaurent_payment.explode('Rpayment')
result_restaurent_work_hours = result_restaurent[['placeID','hours','days']].copy()
result_restaurent = result_restaurent.drop(['hours','days'], axis=1)
result_restaurent_work_hours = result_restaurent_work_hours.explode(['hours','days'])
display(result_restaurent_cuisine)
display(result_restaurent_payment)
display(result_restaurent_work_hours)
display(result_restaurent)
```

2.3 Data Storage

2.3.1 Storage

For data storage, we employed the use of SQLAlchemy to map tables in our code and PostgreSQL as the database management system. To facilitate seamless communication between the two, we utilized psycopg2, a Python library. We have 2 main tables in the database, **Customers**, and **Restaurants**. the other tables are **Ratings**, **Restaurant Cuisine**, **Restaurant Payment**, **Restaurant Work Hours**, **Customer Cuisine**, **Customer Payment**. Attached here is the code for the classes:

```
from sqlalchemy import Boolean, Column, ForeignKey, Integer, String, dialects
from sqlalchemy.orm import relationship
from database import Base
class User(Base):
    __tablename__ = 'users'
    userID = Column(String, primary_key=True)
    drink_level = Column(String)
    dress_preference = Column(String)
    ambience = Column(String)
    transport = Column(String)
    marital_status = Column(String)
    hijos = Column(String)
    birth_year = Column(Integer)
    interest = Column(String)
    personality = Column(String)
    religion = Column(String)
    activity = Column(String)
    color = Column(String)
    budget = Column(String)
    latitude = Column(Float)
    longitude = Column(Float)
    weight = Column(Float)
    height = Column(Float)

class Restaurent(Base):
    __tablename__ = 'restaurents'
    placeID = Column(String, primary_key=True)
    parking_lot = Column(String)

class Rating(Base):
    __tablename__ = 'ratings'
    userID = Column(String, ForeignKey(
        'users.userID', ondelete='CASCADE'), primary_key=True)
    placeID = Column(String, ForeignKey(
        'restaurents.placeID', ondelete='CASCADE'), primary_key=True)
    rating = Column(Integer)
    food_rating = Column(Integer)
    service_rating = Column(Integer)
    user = relationship("User", backref="ratings")
    restaurent = relationship("Restaurent", backref="ratings")

class UserPayment(Base):
    __tablename__ = 'user_payment'
    userID = Column(String, ForeignKey(
        'users.userID', ondelete='CASCADE'), primary_key=True)
    Upayment = Column(String, primary_key=True)
    user = relationship("users", backref="user_payment")

class UserCuisine(Base):
    __tablename__ = 'user_cuisine'
    userID = Column(String, ForeignKey(
        'users.userID', ondelete='CASCADE'), primary_key=True)
    Rcuisine = Column(String, primary_key=True)
```

```
    user = relationship("users", backref="user_cuisine")

class RestaurentHours(Base):
    __tablename__ = 'restaurant_hours'
    placeID = Column(String, ForeignKey(
        'restaurents.placeID', ondelete='CASCADE'), primary_key=True)
    hours = Column(String, primary_key=True)
    days = Column(String, primary_key=True)
    restaurent = relationship("restaurents", backref="restaurant_hours")

class RestaurentPayment(Base):
    __tablename__ = 'restaurant_payment'
    placeID = Column(String, ForeignKey(
        'restaurents.placeID', ondelete='CASCADE'), primary_key=True)
    Rpayment = Column(String, primary_key=True)
    restaurent = relationship("restaurents", backref="restaurant_payment")


class RestaurentCuisine(Base):
    __tablename__ = 'restaurant_cuisine'
    placeID = Column(String, ForeignKey(
        'restaurents.placeID', ondelete='CASCADE'), primary_key=True)
    Rcuisine = Column(String, primary_key=True)
    restaurent = relationship("restaurents", backref="restaurant_cuisine")
```

- **Customers**: Represents the Customer informations.
- **Restaurants**: Have two columns the placeID and parking lot type.
- **Ratings**: Represents the food rating, service rating, and overall rating created by a Customer of a restaurant.
- **Restaurant Cuisine**: Represents the type of cuisines a restaurant have.
- **Restaurant Payment**: Represents the type of payment a restaurant accepts.
- **Restaurant Work Hours**: Represents the work hours of a restaurant.
- **Customer Cuisine**: Represents the type of cuisine a Customer prefers.
- **Customer Payment**: Represents the payment type a Customer prefers.

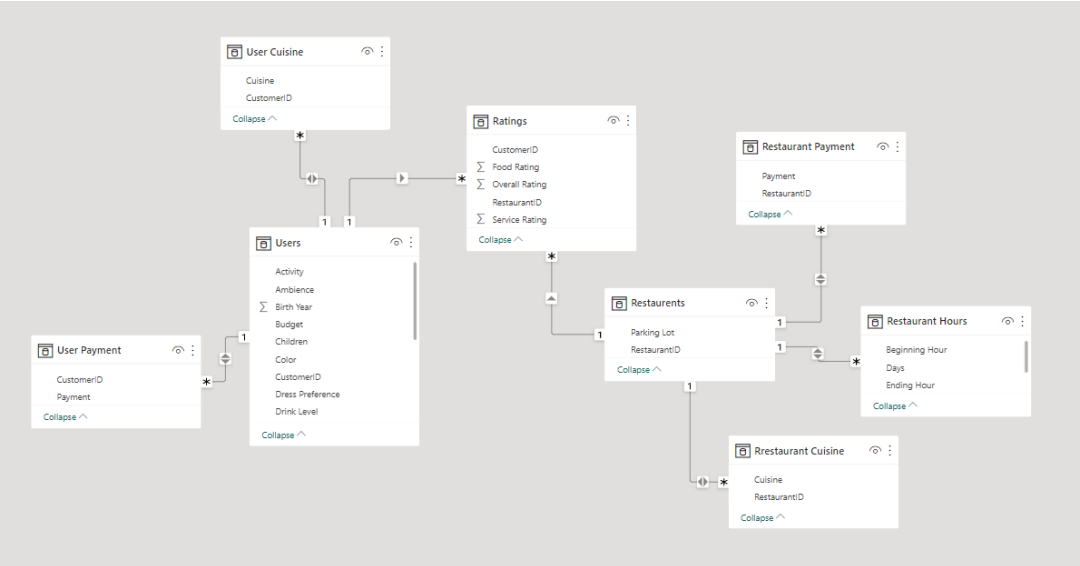

Figure 1: Data Warehouse Schema

2.3.2 Fact

The fact in this data is the ratings given by the user to each restaurant. The types of ratings include:

- **Food Rating**
- **Service Rating**
- **Overall Rating**

2.3.3 Dimensions

Dimensions included in this data are:

- **Restaurants**
- **Users**

Dimensions that are derived from the customer dimensions include:

- **Payment Method**
- **Users' Preferred Cuisine**

Dimensions that are derived from the restaurant's dimensions include:

- **Working Days and hours**
- **Restaurant's Cuisine**

In the end, we made the data warehouse schema to be a Snowflake Schema because of many to many relationships between customers' and restaurants' values in the customer and the existence of the working hours and days values which needs to be separated into a table by itself.

2.4 Data Visualization

For the Data visualization, we used many metrics to understand the data and get insights from it.

- The most-rated cuisine is Mexican followed by Coffee Shops, American, Cafeteria and Japanese food.
- Food rating and service rating are close to having a linear relationship.
- Most customers in the dataset live in San Luis Potosí, Ciudad Victoria, and Cuernavaca.
- The average weight of customers is 65.45 KG and it ranges from 40 to 120 KG.
- The average height of customers is 1.68 M and it ranges from 1.2 M to 2 M.
- Most of the customers are students.
- Most of the customers have a medium budget in all three areas available.
- Most of the customers want to go with their families to restaurants.
- The most used method of payment is cash.
- Customers prefer restaurants that work for long hours (6 to 15 hours and 24/7 restaurants)

3 Conclusion

In conclusion, The study proved that most of the customers in the dataset are living in San Luis Potosí, Ciudad Victoria, and Cuernavaca. They are mostly students who have a medium budget and prefer Mexican food with their families. This would give us a good idea of how to manage targeting customers when an investor wants to invest in the restaurant market in these locations in Mexico. They should advertise with these criteria to make their business go up fast.