Grandma's Nest -- Bed and Breakfast Database

Team4: He, Huiyi | Lin, Yuanjun | Yan, Yihan | Yang, Guang | Zhou, Yihan | Zhu, Honglin

Relational Schema (Revised)

Customer(Customer_ID, Customer_Name, HasPet)

Customer_Attendance(Event_ID[F], Customer_ID[F])

Order_Reserved_By(Customer_ID[F], Reservation Order_ID[F])

Feedback(FID, Customer_ID[F], Score, Content, Received_Time, RO_ID[F])

Inventory_Item(ItemType_ID[F], Item_ID, Item_Name, Date_of_Purchase, Supplyer_ID[F], PO_ID[F])

Perishable_Item(ItemType_ID[F], Item_ID, Expiration_Date)

Recyclable_Item(ItemType_ID[F], Item_ID, Availability)

Supplier_History(ItemType_ID[F], Supplyer_ID, ContractFromDate, ContractToDate)

Service_Order(Service_ID, Room_ID, CreatedAt, FulfilledAt)

Item_IsAssignedTo_ServiceOrder(ItemType_ID[F], Item_ID[F], Service_ID[F])

Worker_IsAssignedTo_ServiceOrder(Employee_ID[F], Service_ID[F])

Reservation_Order(RO_ID, PayerID[F], Room_ID[F], Length, Quantity, Employee_ID[F])

Guest_Stay(RO_ID[F], Customer_ID[F])

Employee(Employee ID, EmployeeType)

Event(Event_ID, Date, Title, Employee_ID[F])

EventCustomizedBy(Customer_ID[F], Event_ID[F], Description)

Worker_IsAssignedTo_Event (Employee_ID[F], Event_ID[F])

Event_IsPlannedBy_EventPlanner (Employee_ID[F], Event_ID[F])

Promotion_IsDesignedBy_MP(Employee_ID[F],Scheme_ID[F])

Promotion(Scheme ID)

Promotion_Platform(Scheme ID[F] ,Platform)

Purchase_Order(PO_ID[F], Employee_ID[F])

Room_Type(Room_Type_ID, Base_Price)

Room(Room ID, Room_Type_ID[F], Employee_ID[F], RO_ID[F], Condition)

Room_Usage_History(Room_ID[F], RO_ID[F], Start, End)

Room_Price_History(Room_ID[F], Start, Price)

Queries

1. Customer Segmentation based on their preferences and behaviors, in order to personalized marketing and service improvements. Analyzing factors like visits-season, average spending, preferred room type, booking source, etc.
   Math model: Clustering
2. Demand Forecasting and Inventory Management according to seasons and events, which allow us to optimize inventory for popular amenities and also prevent stock outs or overstock situations.
   Math Model: Regression
3. Customer feedback analysis. Through analyzing customers scorings and comments to determine overall customers satisfaction, and make changes in service quality. Also we can do queries on customer satisfaction on their requirement of customized events.
   Math Model: NLP

1. Customer Segmentation based on their preferences and behaviors, in order to personalized marketing and service improvements. Analyzing factors like visits-season, average spending, preferred room type, booking source, etc.
Math model: Clustering

```
WITH Customer_Statistics AS (
    SELECT
        c.Customer_ID,
        AVG(ro.Length * ro.Quantity * rt.Base_Price) AS Avg_Spending,
        COUNT(ro.RO_ID) AS Total_Visits,
        (SELECT rt.Room_Type_ID
         FROM Reservation_Order ro2
         JOIN Room_Type rt ON ro2.RoomType_ID = rt.Room_Type_ID
         WHERE ro2.PayerID = c.Customer_ID
         GROUP BY rt.Room_Type_ID
         ORDER BY COUNT(rt.Room_Type_ID) DESC
         LIMIT 1) AS Preferred_Room_Type,
        (SELECT strftime('%m', gs.Check_In_Date)
         FROM Guest_Stay gs
         WHERE gs.Customer_ID = c.Customer_ID
         GROUP BY strftime('%m', gs.Check_In_Date)
         ORDER BY COUNT(gs.Check_In_Date) DESC
         LIMIT 1) AS Preferred_Booking_Season
    FROM Customer c
    JOIN Reservation_Order ro ON c.Customer_ID = ro.PayerID
    JOIN Room_Type rt ON ro.RoomType_ID = rt.Room_Type_ID
    GROUP BY c.Customer_ID
)
SELECT * FROM Customer_Statistics;
```

Then outfile into csv and use Python for analysis.

Explanation:

- **Step 1:** Calculate the average spending, total visits, preferred room type, preferred booking source, and preferred booking season for each customer.
- **Step 2:** Use this aggregated data for customer segmentation using a clustering algorithm (e.g., K-means).
- **Tools and Method:** Apply **K-means clustering** in Python or R to segment customers based on the extracted features.
- **Business Value:** Identifying different customer segments helps in crafting personalized marketing strategies, which can increase customer satisfaction and loyalty.

Visualization:

- Use a scatter plot or radar chart to visualize the customer segments and their key characteristics.

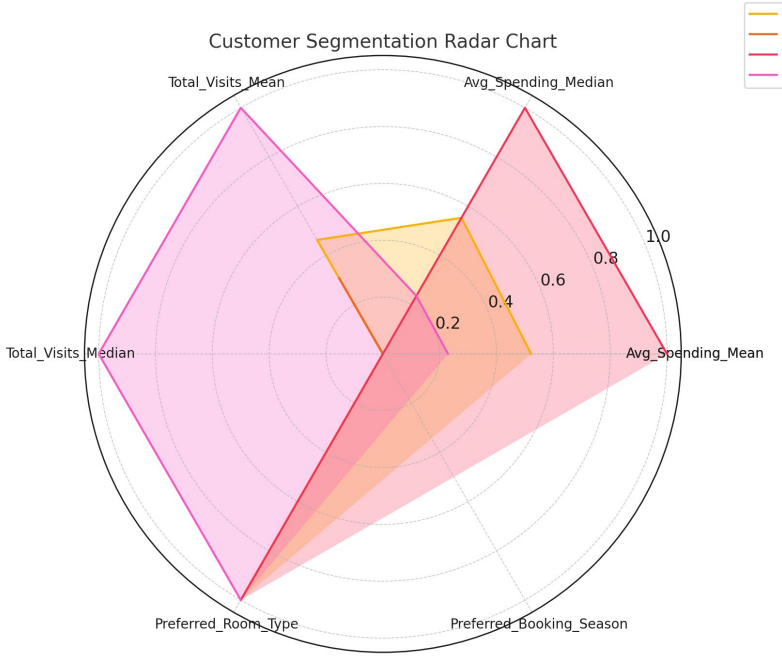
Customer Segmentation: Key clusters

Cluster 0:
- High Spending but Low Visits
- Preferred Room Type: 5
- Marketing Strategy: Exclusive perks and luxury branding

Cluster 1:
- Low Spending and Moderate Visits
- Preferred Room Type: 4
- Marketing Strategy: Discounts and loyalty programs

Cluster 2:
- Elite High Spenders
- Preferred Room Type: 5
- Marketing Strategy: VIP programs and personalized services

Cluster 3:
- Moderate Spending and Visits
- Preferred Room Type: 5
- Marketing Strategy: Family packages and repeat booking rewar



2. Demand Forecasting and Inventory Management according to seasons and events, which allow us to optimize inventory for popular amenities and also prevent stock outs or overstock situations.
Math Model: Regression

```
WITH Event_Present_Dates AS
(SELECT DISTINCT Date FROM Event)
SELECT *, (Service_Order.CreatedAt in Event_Present_Dates)
FROM Item_IsAssignedTo_ServiceOrder, Service_Order
WHERE Item_IsAssignedTo_ServiceOrder.Service_ID =
Service_Order.Service_ID;
```

Then outfile into csv and use Python for analysis.

Explanation:

- **Step 1:** Calculate historical demand for each item type based on monthly sales data.
- **Step 2:** Incorporate additional sales impact from upcoming events.
- **Step 3:** Estimate the predicted demand using regression analysis.
- **Tools and Method:** Use **Time Series Analysis (ARIMA model)** or **Linear Regression** in Python or R for demand forecasting.
- **Business Value:** Accurate demand forecasts help in inventory planning, reducing costs associated with stockouts or excess inventory, and improving customer satisfaction.

Visualization:

- Use a time series plot to show historical demand trends and predicted future demand.
- Create a bar chart comparing predicted demand vs. current stock levels.

Demand Forecasting and Inventory Management

– Create 4 tables

- **Item**
  (Item_ID, Item_Name, Item_Type)

  Item_Type: Housekeeping, Amenities, Guest Services, Event Supplies, Health

- **Service_Order**
  (Service_Order_ID, Service_Order_Date)
- **Item_IsAssignedTo_ServiceOrder**
  (Service_Order_ID, Item_ID, Quantity)
- **Event**
  (Event_ID, Event_Name, Event_Date, Event_Type)

  Event_Type: Grandma's, Customized

```
%sql
-- Create table for Items
DROP TABLE IF EXISTS Item;
CREATE TABLE Item (
    Item_ID INT,
    Item_Name STRING,
    Item_Type STRING
)
USING DELTA;

-- Create table for Service Orders
DROP TABLE IF EXISTS Service_Order;
CREATE TABLE Service_Order (
    Service_Order_ID INT,
    Service_Order_Date DATE
)
USING DELTA;

-- Create table to link Items with Service Orders
DROP TABLE IF EXISTS Item_IsAssignedTo_ServiceOrder;
CREATE TABLE Item_IsAssignedTo_ServiceOrder (
    Service_Order_ID INT,
    Item_ID INT,
    Quantity INT
)
USING DELTA;

-- Create table for Events
DROP TABLE IF EXISTS Event;
CREATE TABLE Event (
    Event_ID INT,
    Event_Name STRING,
    Event_Date DATE,
    Event_Type STRING
)
USING DELTA;
```

Demand Forecasting and Inventory Management

– Extract data to csv

Identify Seasonal and Event-Driven Demand

- Matches service order dates with event dates to uncover patterns
- Links item usage to specific seasons and event types

Optimize Inventory for Popular Items

- Highlights high-demand items during events, enabling better stock planning
- Reduces risk of stock-outs by forecasting spikes during events

Avoid Overstock Situations

- Identifies low-demand periods, preventing excess inventory and storage costs

Data-Driven Reordering

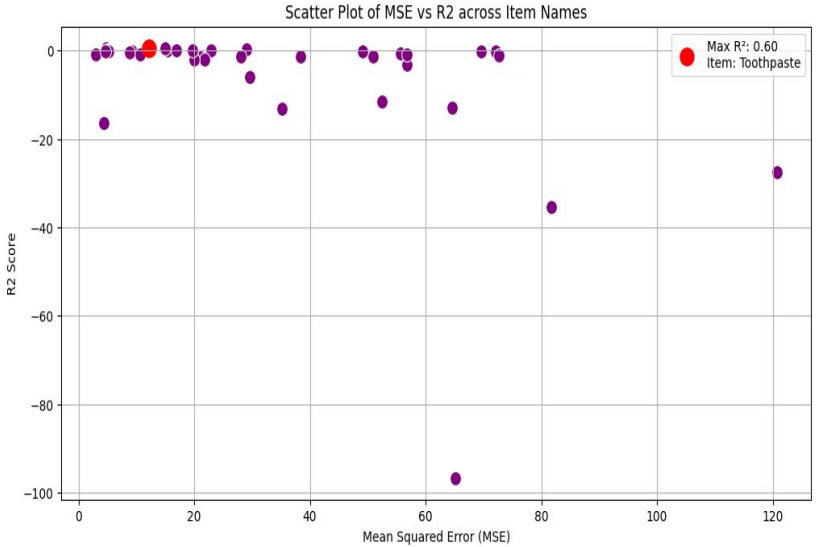
- Tracks item quantities per order, helping set precise reorder points

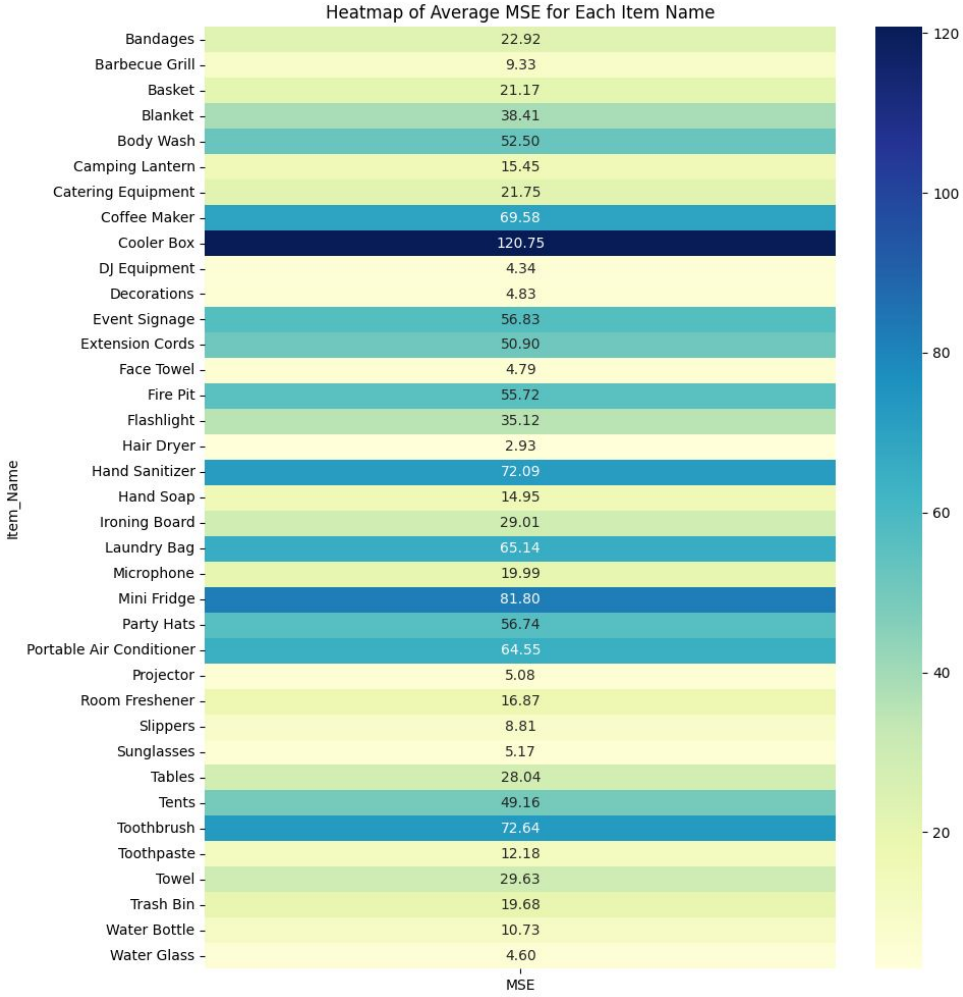
```
%sql
WITH Event_Present_Dates AS (
    SELECT DISTINCT Event_Date AS Date
    FROM Event
)
SELECT
    SO.Service_Order_Date,
    I.Item_Name,
    I.Item_Type,
    ISAS.Quantity,
    E.Event_Name,
    E.Event_Type,
    (CASE WHEN SO.Service_Order_Date IN (SELECT Date FROM Event_Present_Dates) THEN TRUE ELSE FALSE END) AS Is_Event_Day
FROM
    Item_IsAssignedTo_ServiceOrder ISAS
JOIN
    Service_Order SO
ON
    ISAS.Service_Order_ID = SO.Service_Order_ID
JOIN
    Item I
ON
    ISAS.Item_ID = I.Item_ID
LEFT JOIN
    Event E
ON
    SO.Service_Order_Date = E.Event_Date
ORDER BY
    Is_Event_Day DESC, SO.Service_Order_Date;
```

	Service_Order_Date	Item_Name	Item_Type	Quantity	Event_Name	Event_Type	Is_Event_Day
1	2022-03-25	Barbecue Grill	Event Supplies	12	Cherry Blossom Festival	Grandma's	true
2	2022-04-15	Robe	Guest Services	6	Spring Art Show	Grandma's	true
3	2022-08-25	DJ Equipment	Event Supplies	15	Music in the Park	Grandma's	true
4	2022-12-10	Blanket	Housekeeping	21	Christmas Market Fair	Grandma's	true
5	2022-12-15	Body Wash	Amenities	19	Winter Wonderland Ball	Grandma's	true
6	2022-12-20	Coffee Maker	Guest Services	6	Winter Holiday Ice Skating	Grandma's	true
7	2023-02-18	Tents	Event Supplies	18	Mardi Gras Party	Grandma's	true
8	2023-03-10	Face Towel	Amenities	10	Spring Blossom Festival	Grandma's	true
9	2023-04-25	Camping Lantern	Event Supplies	9	Spring Picnic	Grandma's	true
10	2023-06-21	Slippers	Guest Services	20	Summer Music Jam	Grandma's	true

Demand Forecasting and Inventory Management

– Apply Ridge Regression





3. Customer feedback analysis. Through analyzing customers scorings and comments to determine overall customers satisfaction, and make changes in service quality. Also we can do queries on customer satisfaction on their requirement of customized events.

Math Model: NLP

```
SELECT *
FROM Feedback_Summary, Reservation_Order
WHERE Feedback_Summary.RO_ID =
Reservation_Order.RO_ID;
```

Then outfile into csv and use Python for analysis.

Explanation:

- **Step 1:** Aggregate feedback data by calculating average ratings and concatenating customer comments.
- **Step 2:** Analyze feedback specifically related to customized events.
- **Step 3:** Use Natural Language Processing (NLP) to extract common themes and sentiments from the comments.
- **Tools and Method:** Apply **Sentiment Analysis** and **Topic Modeling (e.g., LDA)** in Python (using libraries like NLTK or SpaCy).
- **Business Value:** Understanding customer sentiment can help prioritize service improvements, address common issues, and enhance the quality of customized events.

Visualization:

- Use a word cloud to display common themes from customer comments.
- Create a sentiment distribution plot showing positive, neutral, and negative feedback.

Customer Feedback Analysis

– Create 5 tables

- **Customer**
  (Customer_ID, Customer_Name, HasPet)
- **Employee**
  (Employee ID, EmployeeType)
- **Room Type**
  (Room_Type_ID, Base_Price)

```
(1, 150.00),  -- Single Room
(2, 200.00),  -- Double Room
(3, 250.00),  -- Suite
(4, 175.00),  -- Deluxe Room
(5, 300.00);  -- Presidential Suite
```

- **Reservation Order**
  (RO_ID, PayerIDF, Room_IDF, Length, Quantity, Employee_IDF)
- **Feedback Summary**
  (FID, Customer_IDF, Score, Content, Received_Time, RO_IDF)

```
CREATE TABLE Customer (
    CUSTOMER_ID INT AUTO_INCREMENT PRIMARY KEY,
    CUSTOMER_NAME VARCHAR(100) NOT NULL,
    HasPet BOOLEAN DEFAULT FALSE
);

CREATE TABLE Employee (
    Employee_ID INT AUTO_INCREMENT PRIMARY KEY,
    EmployeeType ENUM('Worker', 'Marketing_Personnel', 'Event_Planner', 'Clerk') NOT NULL
);

CREATE TABLE Room_Type (
    Room_Type_ID INT AUTO_INCREMENT PRIMARY KEY,      -- Unique ID for room type
    Base_Price DECIMAL(10, 2) NOT NULL               -- Base price for the room type
);

CREATE TABLE Reservation_Order (
    RO_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Reservation Order ID
    PayerID INT NOT NULL,                  -- Payer ID
    RoomType_ID INT NOT NULL,              -- Room Type ID
    Length INT NOT NULL CHECK (Length > 0),            -- Length of the stay
    Quantity INT NOT NULL CHECK (Quantity > 0),                -- Number of rooms reserved
    Employee_ID INT NOT NULL,               -- Employee handling the reservation
    FOREIGN KEY (PayerID) REFERENCES Customer(Customer_ID),
    FOREIGN KEY (RoomType_ID) REFERENCES Room_Type(Room_Type_ID),
    FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID)
);

CREATE TABLE Feedback_Summary (
    FID INT AUTO_INCREMENT PRIMARY KEY,    -- Feedback ID
    CUSTOMER_ID INT NOT NULL,
    RO_ID INT NOT NULL,                    -- Reservation Order ID (Foreign Key)
    SCORE INT CHECK (SCORE BETWEEN 1 AND 5),  -- Feedback score
    SUMMARY TEXT,                          -- Summary of the feedback
    RECEIVED_TIME DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (RO_ID) REFERENCES Reservation_Order (RO_ID)
);
```

Customer Feedback Analysis

– Extract data to csv

Feedback Distribution by Room Type

Analyze how room type affects customer feedback scores.

```
SELECT
    r.RoomType_ID,
    rt.Base_Price AS Room_Base_Price,
    AVG(f.SCORE) AS Avg_Score,
    COUNT(f.FID) AS Total_Feedbacks
FROM Feedback_Summary f
JOIN Reservation_Order r ON f.RO_ID = r.RO_ID
JOIN Room_Type rt ON r.RoomType_ID = rt.Room_Type_ID
GROUP BY r.RoomType_ID, rt.Base_Price
ORDER BY Avg_Score DESC;
```

RoomType_ID	Room_Base_Price	Avg_Score	Total_Feedbacks
1	150.00	5.0000	10
3	250.00	4.7000	10
2	200.00	4.2000	10
5	300.00	3.9000	10
4	175.00	3.0000	10

Customer With and Without Pets

Compare the average feedback score of customers with pets versus those without pets

Result: Pet-friendly features seem to contribute positively to customer satisfaction

```
SELECT
    c.HasPet AS Has_Pet,
    AVG(f.SCORE) AS Avg_Feedback_Score,
    COUNT(f.FID) AS Total_Feedbacks
FROM Feedback_Summary f
JOIN Customer c ON f.CUSTOMER_ID = c.CUSTOMER_ID
GROUP BY c.HasPet;
```

Has_Pet	Avg_Feedback_Score	Total_Feedbacks
0	4.0400	25
1	4.2800	25

Customer Feedback Analysis

– Extract data to csv

Service Performance Analysis

Analyze feedback grouped by employees handling the reservations.

```
SELECT
    e.Employee_ID,
    e.EmployeeType,
    AVG(f.SCORE) AS Avg_Score,
    COUNT(f.FID) AS Total_Feedbacks
FROM Feedback_Summary f
JOIN Reservation_Order r ON f.RO_ID = r.RO_ID
JOIN Employee e ON r.Employee_ID = e.Employee_ID
GROUP BY e.Employee_ID, e.EmployeeType
ORDER BY Avg_Score DESC;
```

Employee_ID	EmployeeType	Avg_Score	Total_Feedbacks
1	Worker	5.0000	10
3	Event_Planner	5.0000	10
2	Marketing_Personnel	4.0000	10
5	Worker	4.0000	10
4	Clerk	3.0000	10

Sentiment Analysis

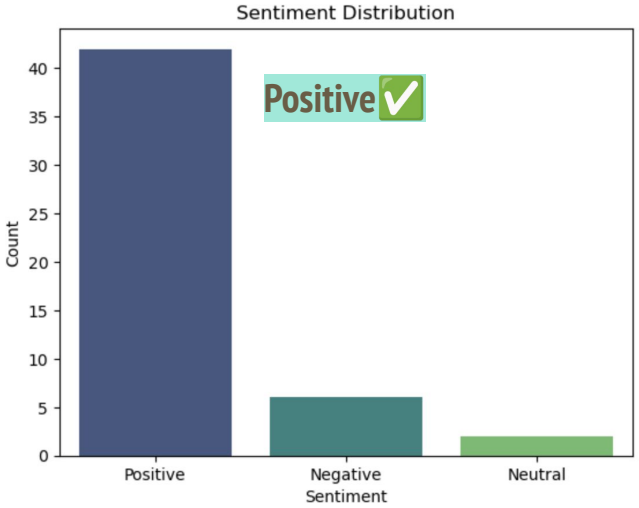
Perform deeper analysis on feedback comments using natural language processing.

```
SELECT
    c.CUSTOMER_NAME AS Customer_Name,
    f.SUMMARY AS Feedback_Comment,
    f.SCORE AS Feedback_Score,
    r.RoomType_ID AS Room_Type,
    r.Length AS Stay_Length,
    r.Quantity AS Rooms_Reserved,
    f.RECEIVED_TIME AS Feedback_Time
FROM Feedback_Summary f
JOIN Customer c ON f.CUSTOMER_ID = c.CUSTOMER_ID
JOIN Reservation_Order r ON f.RO_ID = r.RO_ID
ORDER BY f.RECEIVED_TIME;
```

Customer_Name	Feedback_Comment	Feedback_Score	Room_Type	Stay_Length	Rooms_Reserved	Feedback_Time
John Doe	Amazing service and clean room!	5	1	3	1	2024-12-01 10:00:00
Jane Smith	Perfect for pets! My dog loved the open space.	5	2	2	1	2024-12-01 10:00:00
Michael Brown	Okay stay, but the bathroom was dirty.	3	4	1	3	2024-12-01 16:30:00
David Wilson	Absolutely amazing, will come again!	5	1	3	1	2024-12-02 10:00:00
Laura Harris	The service was slow.	3	4	1	3	2024-12-02 11:30:00
Emily Johnson	Comfortable stay, but the room was a bit noisy.	4	3	5	2	2024-12-02 12:00:00
Sarah Davis	Good experience overall, but could use more pe…	4	5	4	2	2024-12-02 12:00:00
Mary Lewis	Best hotel stay I have ever had!	5	1	3	1	2024-12-02 12:30:00

Customer Feedback Analysis

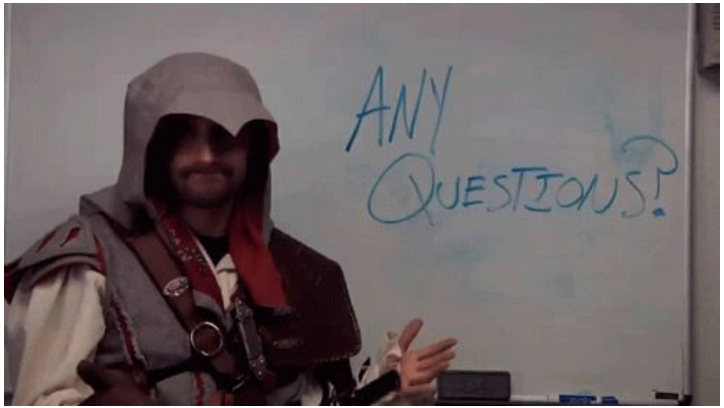
– Apply NLP to the data obtained



“Room”, “Stay”, “Great”, “Good”

```
Topic 1:
perfect experience hotel wonderful loved stay definitely return absolutely fantastic
Topic 2:
room excellent okay service comfortable stay small highly bit amazing
Topic 3:
stay good room great nice noisy overall breakfast better use
```

Thanks!



He, Huiyi | Lin, Yuanjun | Yan, Yihan | Yang, Guang | Zhou, Yihan | Zhu, Honglin

Grandma's Nest – Bed and Breakfast Database

Grandma's Nest is a boutique bed and breakfast that embodies the warmth and care of a grandmother's home. Located in a tranquil oceanside neighborhood, it offers a cozy atmosphere with personalized service, themed rooms inspired by local culture, and an authentic connection to the community. Guests can enjoy daily-changing breakfasts, relax in the garden, and take in city views, all while feeling at home. Its pet-friendly policies and proximity to local attractions make it a perfect choice for travelers seeking comfort and exploration.

This project aims to develop a database system to enhance the operations of Grandma's Nest by managing employee roles, customer interactions, event planning, and resource management. By integrating these components, the system will streamline operations, support data-driven decision-making, and uphold the personalized service that defines Grandma's Nest. This report details the database design, including the entity-relationship (ER) model, relational schema, and analytical methods for customer behavior, demand forecasting, and service quality improvement.

I. ER Model Explanation

The EER model integrates employee roles, customer interactions, resource management, event planning, and financial transactions, providing a detailed view of how different components interact within the business framework. First of all, the core of the model are Employees, which are categorized into specialized roles such as Workers, Event Planners, and Marketing Personnel. This classification enhances clarity and ensures that specific responsibilities are associated with the appropriate employee type.

The model also takes Customers as a central entity, allowing the business to track their interactions, preferences, and feedback. Customers are connected to events, room reservations, and promotions, reflecting their engagement across various touchpoints. The inclusion of customer-specific attributes, such as the option to track pets, demonstrates attention to personalization and detailed service offerings. Customers can attend events, reserve rooms, and provide feedback, creating a feedback loop that supports continuous improvement. Moreover, the integration of customer data into the broader system is expected to target marketing and enhance overall service quality.

Events play a significant role in the system and are categorized into Grandma's Events and Customized Events. This subclassing, facilitated by the d,t hierarchy, allows the model to differentiate between standard recurring events and those tailored to specific customer needs. Events require various resources, such as inventory items and employees, further integrating them with other entities in the system. For instance, an event may involve room bookings, promotional campaigns, or even inventory orders, demonstrating the interconnectedness of the model. Moreover, Inventory Management enables the tracking of items used in events and services. Associated with their type and supplier history, the business has a detailed record of resource utilization and procurement.

The model also includes a detailed representation of Room Management, with rooms being associated with attributes such as Room Type, Condition, and Base Price. This structure enables efficient room reservation tracking and ensures that customers can book rooms that meet their specific needs. The integration of reservation details, such as the length of stay and number of rooms, into the Reservation Order entity connects the customer experience with operational data.

Finally, another notable aspect of the model is its focus on Promotions. These are associated with marketing platforms and schemes, reflecting the role of targeted campaigns in attracting customers. Marketing Personnel design these promotions, creating a clear relationship between employee activities and customer engagement strategies. By capturing the platform and scheme details for each promotion, our business can analyze the effectiveness of its marketing campaigns and adjust strategies accordingly.

II. Relational Schema

Customer Management

1. Customer (Customer_ID, Customer_Name, HasPet): Tracks customer information, including whether they own pets.
2. Customer_Attendance (Event_ID^F, Customer_ID^F): Records customer participation in events.

Employee Management

3. Employee (Employee_ID, EmployeeType): Represents employees and their roles, categorized by enumerated types (e.g., Worker, Marketing Personnel, Event Planner, Clerk).

Event and Promotion Management

4. Event (Event_ID, Date, Title, Employee_ID^F): Captures event details and associates them with the responsible employee.
5. EventCustomizedBy (Customer_ID^F, Event_ID^F, Description): Links customers with customizations for specific events.
6. Worker_IsAssignedTo_Event (Employee_ID^F, Event_ID^F): Tracks worker assignments to events.
7. Event_IsPlannedBy_EventPlanner (Employee_ID, Event_ID): Links event planners to the events they organize.
8. Promotion (Scheme_ID): Stores promotion-related details.
9. Promotion_IsDesignedBy_MP (Employee_ID^F, Scheme_ID^F): Links marketing personnel to the promotions they create.
10. Promotion_Platform (Scheme_ID^F, Platform): Specifies the platform where a promotion is hosted.

Inventory Management

11. Inventory_Item (ItemType_ID^F, Item_ID, Item_Name, Date_of_Purchase, Supplier_ID^F, PO_ID^F): Tracks inventory items and their suppliers.
12. Perishable_Item (ItemType_ID^F, Item_ID, Expiration_Date): Extends Inventory_Item for perishable goods.
13. Recyclable_Item (ItemType_ID^F, Item_ID, Availability): Extends Inventory_Item for recyclable goods.
14. Supplier_History (ItemType_ID^F, Supplier_ID, ContractFromDate, ContractToDate): Records supplier contract history.

Service Management

15. Service_Order (Service_ID, Room_ID, CreatedAt, FulfilledAt): Manages service orders associated with rooms.
16. Item_IsAssignedTo_ServiceOrder (ItemType_ID^F, Item_ID^F, Service_ID^F): Links inventory items to service orders.
17. Worker_IsAssignedTo_ServiceOrder (Employee_ID^F, Service_ID^F): Tracks workers fulfilling service orders.

Reservation and Room Management

18. Reservation_Order (RO_ID, PayerID^F, Room_ID^F, Length, Quantity, Employee_ID^F): Manages room reservation orders, specifying the employee handling them.
19. Guest_Stay (RO_ID^F, Customer_ID^F): Links reservations to guest stays.
20. Room (Room_ID, Room_Type_ID^F, Employee_ID^F, RO_ID^F, Condition): Represents rooms and their current state.
21. Room_Type (Room_Type_ID, Base_Price): Categorizes rooms by type and price.
22. Room_Usage_History (Room_ID^F, RO_ID^F, Start, End): Tracks room usage over time.
23. Room_Price_History (Room_Type_ID^F, Start, Price): Stores historical pricing for room types.

Feedback and Procurement

24. Feedback (FID, Customer_ID^F, Score, Content, Received_Time, RO_ID^F): Captures customer feedback on reservations.
25. Purchase_Order (PO_ID^F, Employee_ID^F): Manages purchase orders for inventory procurement.

III. Queries and Methods

a) Customer Clustering

Purpose: The objective of this analysis was to better understand customer behavior and preferences by creating customer segmentation based on their historical reservation and spending data. Insights into spending habits, visitation frequency and room type preferences provide a foundation for targeted marketing and personalized service enhancements.

Method: The analysis began by retrieving comprehensive data from the Customer, Reservation_Order, and Payment tables using SQL queries to ensure all relevant customer information was included. A custom SQL query was then designed to retrieve key statistics for each customer. These statistics included their average spending, calculated as the average product of room length, quantity, and base price; the total number of visits, represented by the count of their reservations; the most frequently booked room type, determined by aggregating room preferences; and the preferred booking season, identified as the month in which they most commonly checked in.

Missing data are then addressed by filling with 0 to preserve the 'missing' as a piece of information. Then, measures such as average spending, total visits, preferred room type, and preferred booking season were selected as inputs to a K-means clustering model. The model, with a predetermined number of four clusters, grouped customers based on these attributes to identify distinct behavioral patterns.

The results of the clustering were visualized using a radar plot, which can show the six dimensions with respect to the four clusters with adequate clarity. This visualization provided an intuitive way to understand the segmentation and highlighted meaningful distinctions between the customer groups.

Results: The clustering analysis identified four distinct customer segments with unique spending and visitation patterns. Cluster 0 represents high spenders with low visit frequencies, favoring Room Type 5. Ideal for exclusive perks and luxury branding. Cluster 1 represents low spenders with moderate visit frequencies, preferring Room Type 4. Discounts and loyalty programs can boost engagement and spending. Cluster 2 represents elite high spenders consistently choosing Room Type 5. Best engaged through VIP programs and personalized services. Cluster 3 represents moderate spenders with moderate visit frequencies, also favoring Room Type 5. Family packages and rewards for repeat bookings are effective.

The visualization highlighted distinct separations among clusters, confirming the effectiveness of segmentation features. These insights provide actionable strategies for targeted marketing and can inform machine learning models for further analysis.

b) Demand Forecasting and Inventory Management

Purpose: This analysis supports demand forecasting and inventory management by leveraging historical sales data to identify demand trends influenced by seasonal patterns and events. The goal is to estimate future demand for various items during specific periods, optimizing inventory levels, preventing stockouts, and minimizing overstock situations.

Method: We created four relational tables in Databricks: Item, Service_Order, Item_IsAssignedTo_ServiceOrder, and Event, populated with realistic sample data (50 rows for Item, 300 for Service_Order and Item_IsAssignedTo_ServiceOrder, and 100 for Event). The dataset spans all seasons from 2022 to 2024, aligning event dates with service orders to analyze how events influence item demand.

SQL queries cross-referenced Service_Order_Date and Event_Date to identify overlaps where events affected item demand, leveraging fields like Item_Name, Item_Type, and Quantity, along with derived columns such as Is_Event_Day. The results, including event-driven trends, were exported as a CSV file for further analysis in Python.

In Python, data preprocessing converted dates to datetime format, added derived columns for year-month and time windows, and incorporated event-driven activity counts. Ridge regression models were trained for each item using service order dates, event activity counts, and order quantities. Model performance was evaluated using Mean Squared Error (MSE) and R^2, with results stored for further review.

Visualization included scatter plots to show the distribution of MSE and R^2 values across items, box plots to illustrate MSE variability, and heatmaps for a comparative view of prediction errors. An interactive Plotly chart enabled detailed evaluation of true vs. predicted values for selected items, providing actionable insights into model performance and event-driven demand patterns.

Results: This analysis provided actionable insights into how specific events and seasonal factors influence item demand, enabling more accurate forecasting. Items with low MSE and high R^2 scores were identified as having strong predictive performance, supporting efficient inventory planning. Conversely, items with higher errors were flagged for further investigation or alternative modeling approaches. The heatmap of average MSE highlighted items requiring prioritization in inventory strategies, while scatter plots revealed predictable items, helping to reduce uncertainty.

Interactive analysis further supported detailed evaluations of specific items, allowing for proactive adjustments to inventory levels based on forecast accuracy. By integrating historical data, SQL-based analysis, and predictive modeling, this approach enhances demand forecasting and inventory management, ensuring operational efficiency and cost savings.

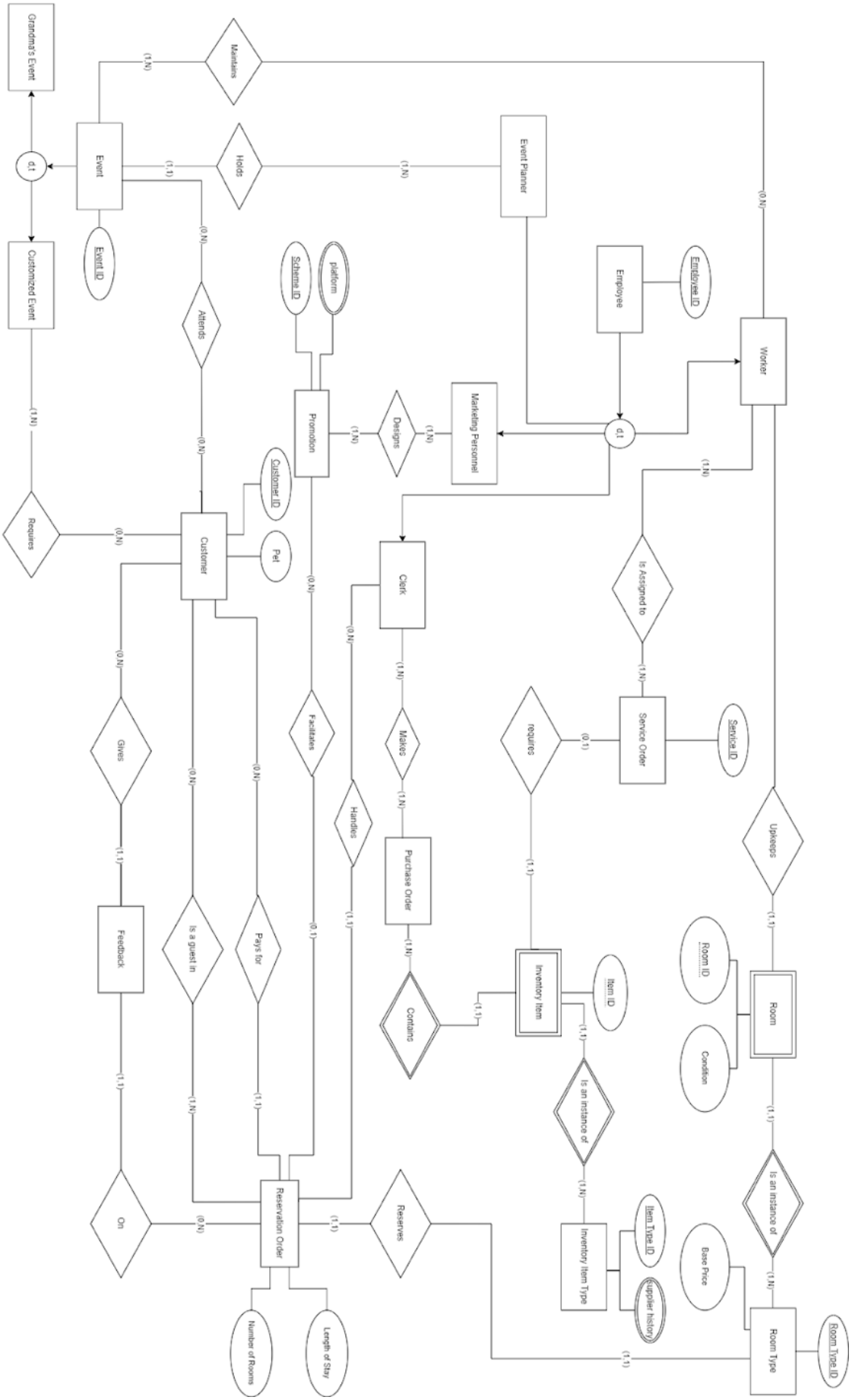
c) Customer Feedback Analysis

Purpose: This analysis evaluates customer feedback to understand satisfaction, identify service improvements, and address specific needs for customized events. Using Natural Language Processing (NLP), it extracts themes and sentiments from comments to enhance service quality.

Method: The analysis began by creating tables with randomly generated customer feedback data in SQL, including key tables such as Feedback_Summary, Reservation_Order, and Room_Type, each containing 50 items. SQL queries were then used to extract and analyze specific aspects of customer feedback that we are interested in. The first query examined feedback distribution by room type by joining Feedback_Summary, Reservation_Order, and Room_Type tables, grouping the data by room type, and calculating average feedback scores and total counts. The second query compared customer satisfaction between those with and without pets by joining Feedback_Summary and Customer tables, grouping by the HasPet attribute. The third query assessed employee performance by joining Feedback_Summary, Reservation_Order, and Employee tables, grouping the data by employee ID and type, and calculating average feedback scores. Lastly, detailed feedback comments and reservation information were extracted for sentiment analysis by joining Feedback_Summary, Customer, and Reservation_Order tables and ordering the results by feedback submission time. All outputs were exported to CSV files for further analysis and visualization using Python.

Results: First, the feedback distribution by room type table revealed that rooms with higher base prices generally received higher scores, which indicates a positive correlation between room quality and customer satisfaction. Furthermore, pet-friendly services appeared to enhance satisfaction levels among customers with pets. Analysis of employee performance showed that workers and event planners consistently received higher satisfaction ratings than other roles, highlighting their effective service delivery. Beyond the direct table insights, sentiment analysis and topic modeling provided deeper understanding. Sentiment analysis confirmed that most feedback was overwhelmingly positive. In the word cloud visualization, it emphasized frequent terms like "room," "stay," "great," and "good," pointing to a strong focus on room quality and the overall stay experience. Topic modeling identified three key themes: positive experiences with rooms and services, opportunities for improvement, such as noise reduction and enhanced service, and highlights of exceptional stays and individual preferences. In conclusion, this comprehensive analysis not only identifies strengths and areas for improvement but also provides actionable insights to enhance service quality and boost customer satisfaction.

IV. ER Model



1. "Grandma's Event" and "Customized Event" are subclasses of "Event (Event ID)," allowing us to track regular events like movie nights and BBQ party hosted by Grandma's nest, and also customer-hosted events. These entities go beyond basic attributes by analyzing attendance cost, and other factors to help us design future events based on customer preferences. It can ensures our events are optimized to better meet guest expectations and enhance overall experience. Thus, we think there is no problem for these subclasses.
2. We set `Employee` as a superclass because directly linking employees to each entity (e.g., `Event`, `Promotion`, `Service Order`) would make the database structure unclear. Different roles handle different services or activities, so by subclassing `Employee` (e.g., `Worker`, `Event Planner`), we can clearly show which positions are connected to which services or activities. This not only makes the database structure more organized but also enhances scalability for adding new roles or services in the future.

V. Appendix

```sql
CREATE DATABASE Grandma;

USE Grandma;


DROP TABLE IF EXISTS Feedback_Summary;
DROP TABLE IF EXISTS Reservation_Order;
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS Employee;
DROP TABLE IF EXISTS Room_Type;


CREATE TABLE Customer (
    CUSTOMER_ID INT AUTO_INCREMENT PRIMARY KEY,
    CUSTOMER_NAME VARCHAR(100) NOT NULL,
    HasPet BOOLEAN DEFAULT FALSE
);

CREATE TABLE Employee (
    Employee_ID INT AUTO_INCREMENT PRIMARY KEY,
    EmployeeType ENUM('Worker', 'Marketing_Personnel', 'Event_Planner', 'Clerk') NOT NULL
);

CREATE TABLE Room_Type (
    Room_Type_ID INT AUTO_INCREMENT PRIMARY KEY,      -- Unique ID for room type
    Base_Price DECIMAL(10, 2) NOT NULL               -- Base price for the room type
);

CREATE TABLE Reservation_Order (
    RO_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Reservation Order ID
    PayerID INT NOT NULL,                  -- Payer ID
    RoomType_ID INT NOT NULL,              -- Room Type ID
    Length INT NOT NULL CHECK (Length > 0),            -- Length of the stay
    Quantity INT NOT NULL CHECK (Quantity > 0),                -- Number of rooms reserved
    Employee_ID INT NOT NULL,               -- Employee handling the reservation
    FOREIGN KEY (PayerID) REFERENCES Customer(Customer_ID),
    FOREIGN KEY (RoomType_ID) REFERENCES Room_Type(Room_Type_ID),
    FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID)
);

CREATE TABLE Feedback_Summary (
    FID INT AUTO_INCREMENT PRIMARY KEY,    -- Feedback ID
    CUSTOMER_ID INT NOT NULL,
    RO_ID INT NOT NULL,                    -- Reservation Order ID (Foreign Key)
    SCORE INT CHECK (SCORE BETWEEN 1 AND 5),  -- Feedback score
    SUMMARY TEXT,                          -- Summary of the feedback
    RECEIVED_TIME DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (RO_ID) REFERENCES Reservation_Order (RO_ID)
);


INSERT INTO Customer (CUSTOMER_NAME, HasPet) VALUES
('John Doe', FALSE),
('Jane Smith', TRUE),
('Emily Johnson', FALSE),
('Michael Brown', FALSE),
('Sarah Davis', TRUE),
('David Wilson', FALSE),
('Jessica Lee', TRUE),
('Daniel White', FALSE),
('Laura Harris', TRUE),
('James Clark', FALSE),
('Mary Lewis', TRUE),
```

```
    ('Robert Walker', FALSE),
    ('Susan Hall', TRUE),
    ('William Allen', FALSE),
    ('Linda Young', TRUE),
    ('Joseph King', FALSE),
    ('Nancy Scott', TRUE),
    ('Charles Green', FALSE),
    ('Patricia Adams', TRUE),
    ('Christopher Nelson', FALSE),
    ('Betty Carter', TRUE),
    ('Thomas Mitchell', FALSE),
    ('Dorothy Perez', TRUE),
    ('Daniel Roberts', FALSE),
    ('Helen Evans', TRUE),
    ('Matthew Collins', FALSE),
    ('Lisa Stewart', TRUE),
    ('Mark Sanchez', FALSE),
    ('Angela Morris', TRUE),
    ('Steven Rivera', FALSE),
    ('Karen Cooper', TRUE),
    ('Paul Richardson', FALSE),
    ('Nancy Diaz', TRUE),
    ('George Bennett', FALSE),
    ('Elizabeth Wood', TRUE),
    ('Edward Turner', FALSE),
    ('Sarah Perez', TRUE),
    ('James Scott', FALSE),
    ('Helen Bailey', TRUE),
    ('Andrew Murphy', FALSE),
    ('Betty Cox', TRUE),
    ('Ryan Brooks', FALSE),
    ('Nancy Price', TRUE),
    ('Brian King', FALSE),
    ('Sandra Wright', TRUE),
    ('Jack Williams', FALSE),
    ('Kimberly Martinez', TRUE),
    ('Oliver Jackson', TRUE),
    ('Sophia Miller', FALSE),
    ('Benjamin Moore', TRUE),
    ('Charlotte Taylor', FALSE),
    ('Lucas Anderson', TRUE),
    ('Amelia Thomas', FALSE),
    ('Ethan Martin', TRUE),
    ('Isabella Garcia', FALSE),
    ('Mason Martinez', TRUE),
    ('Mia Hernandez', FALSE),
    ('Logan Robinson', TRUE),
    ('Harper Young', FALSE),
    ('Elijah Lee', TRUE),
    ('Avery Allen', FALSE),
    ('Alexander Walker', TRUE),
    ('Evelyn King', FALSE),
    ('James Wright', TRUE),
    ('Lily Scott', FALSE),
    ('Henry Adams', TRUE);


INSERT INTO Employee (Employee_ID, EmployeeType)
VALUES
(1, 'Worker'),
(2, 'Marketing_Personnel'),
(3, 'Event_Planner'),
(4, 'Clerk'),
```

```
    (5, 'Worker'),
    (6, 'Marketing_Personnel'),
    (7, 'Event_Planner'),
    (8, 'Clerk'),
    (9, 'Worker'),
    (10, 'Marketing_Personnel'),
    (11, 'Event_Planner'),
    (12, 'Clerk'),
    (13, 'Worker'),
    (14, 'Marketing_Personnel'),
    (15, 'Event_Planner'),
    (16, 'Clerk'),
    (17, 'Worker'),
    (18, 'Marketing_Personnel'),
    (19, 'Event_Planner'),
    (20, 'Clerk'),
    (21, 'Worker'),
    (22, 'Marketing_Personnel'),
    (23, 'Event_Planner'),
    (24, 'Clerk'),
    (25, 'Worker'),
    (26, 'Marketing_Personnel'),
    (27, 'Event_Planner'),
    (28, 'Clerk'),
    (29, 'Worker'),
    (30, 'Marketing_Personnel'),
    (31, 'Event_Planner'),
    (32, 'Clerk'),
    (33, 'Worker'),
    (34, 'Marketing_Personnel'),
    (35, 'Event_Planner'),
    (36, 'Clerk'),
    (37, 'Worker'),
    (38, 'Marketing_Personnel'),
    (39, 'Event_Planner'),
    (40, 'Clerk'),
    (41, 'Worker'),
    (42, 'Marketing_Personnel'),
    (43, 'Event_Planner'),
    (44, 'Clerk'),
    (45, 'Worker'),
    (46, 'Marketing_Personnel'),
    (47, 'Event_Planner'),
    (48, 'Clerk'),
    (49, 'Worker'),
    (50, 'Marketing_Personnel');


    INSERT INTO Room_Type (Room_Type_ID, Base_Price)
    VALUES
    (1, 150.00),  -- Single Room
    (2, 200.00),  -- Double Room
    (3, 250.00),  -- Suite
    (4, 175.00),  -- Deluxe Room
    (5, 300.00);  -- Presidential Suite

    INSERT INTO Reservation_Order (PayerID, RoomType_ID, Length, Quantity, Employee_ID) VALUES
    (1, 1, 3, 1, 1),
    (2, 2, 2, 1, 2),
    (3, 3, 5, 2, 3),
    (4, 4, 1, 3, 4),
    (5, 5, 4, 2, 5),
    (6, 1, 3, 1, 1),
```

```
    (7, 2, 2, 1, 2),
    (8, 3, 5, 2, 3),
    (9, 4, 1, 3, 4),
    (10, 5, 4, 2, 5),
    (11, 1, 3, 1, 1),
    (12, 2, 2, 1, 2),
    (13, 3, 5, 2, 3),
    (14, 4, 1, 3, 4),
    (15, 5, 4, 2, 5),
    (16, 1, 3, 1, 1),
    (17, 2, 2, 1, 2),
    (18, 3, 5, 2, 3),
    (19, 4, 1, 3, 4),
    (20, 5, 4, 2, 5),
    (21, 1, 3, 1, 1),
    (22, 2, 2, 1, 2),
    (23, 3, 5, 2, 3),
    (24, 4, 1, 3, 4),
    (25, 5, 4, 2, 5),
    (26, 1, 3, 1, 1),
    (27, 2, 2, 1, 2),
    (28, 3, 5, 2, 3),
    (29, 4, 1, 3, 4),
    (30, 5, 4, 2, 5),
    (31, 1, 3, 1, 1),
    (32, 2, 2, 1, 2),
    (33, 3, 5, 2, 3),
    (34, 4, 1, 3, 4),
    (35, 5, 4, 2, 5),
    (36, 1, 3, 1, 1),
    (37, 2, 2, 1, 2),
    (38, 3, 5, 2, 3),
    (39, 4, 1, 3, 4),
    (40, 5, 4, 2, 5),
    (41, 1, 3, 1, 1),
    (42, 2, 2, 1, 2),
    (43, 3, 5, 2, 3),
    (44, 4, 1, 3, 4),
    (45, 5, 4, 2, 5),
    (46, 1, 3, 1, 1),
    (47, 2, 2, 1, 2),
    (48, 3, 5, 2, 3),
    (49, 4, 1, 3, 4),
    (50, 5, 4, 2, 5);


INSERT INTO Feedback_Summary (CUSTOMER_ID, RO_ID, SCORE, SUMMARY, RECEIVED_TIME) VALUES
(1, 1, 5, 'Amazing service and clean room!', '2024-12-01 10:00:00'),
(2, 2, 5, 'Perfect for pets! My dog loved the open space.', '2024-12-01 10:00:00'),
(3, 3, 4, 'Comfortable stay, but the room was a bit noisy.', '2024-12-02 12:00:00'),
(4, 4, 3, 'Okay stay, but the bathroom was dirty.', '2024-12-01 16:30:00'),
(5, 5, 4, 'Good experience overall, but could use more pet-friendly amenities.', '2024-12-02 12:00:00'),
(6, 6, 5, 'Absolutely amazing, will come again!', '2024-12-02 10:00:00'),
(7, 7, 5, 'Fantastic! The staff were very accommodating to my cat.', '2024-12-03 14:00:00'),
(8, 8, 3, 'Decent stay, but the staff were slow.', '2024-12-04 16:00:00'),
(9, 9, 3, 'The service was slow.', '2024-12-02 11:30:00'),
(10, 10, 3, 'The room was fine, but no dedicated pet facilities.', '2024-12-04 16:00:00'),
(11, 11, 5, 'Best hotel stay I have ever had!', '2024-12-02 12:30:00'),
(12, 12, 4, 'Nice but could use a better mattress.', '2024-12-02 13:00:00'),
(13, 13, 5, 'Perfect place to stay for a weekend getaway.', '2024-12-02 13:30:00'),
(14, 14, 3, 'The room had an unpleasant smell.', '2024-12-02 14:00:00'),
(15, 15, 4, 'Nice location and pet-friendly, but a bit pricey.', '2024-12-05 18:00:00'),
(16, 16, 5, 'Highly recommended, excellent service!', '2024-12-02 15:00:00'),
```

```
    (17, 17, 4, 'Had a good experience, but the Wi-Fi was slow.', '2024-12-02 15:30:00'),
    (18, 18, 5, 'Perfect for a business trip.', '2024-12-02 16:00:00'),
    (19, 19, 3, 'Average stay, not bad but not great.', '2024-12-02 16:30:00'),
    (20, 20, 4, 'Nice stay, but the breakfast was lacking variety.', '2024-12-02 17:00:00'),
    (21, 21, 5, 'Loved everything about the room and staff!', '2024-12-02 17:30:00'),
    (22, 22, 4, 'Very comfortable, but room could have been cleaner.', '2024-12-02 18:00:00'),
    (23, 23, 5, 'Wonderful hotel, will stay again.', '2024-12-02 18:30:00'),
    (24, 24, 3, 'Too noisy outside, couldn't sleep well.', '2024-12-02 19:00:00'),
    (25, 25, 4, 'Great room, very spacious.', '2024-12-02 19:30:00'),
    (26, 26, 5, 'Amazing stay, excellent service and comfort.', '2024-12-02 20:00:00'),
    (27, 27, 4, 'Good stay, but I expected more variety in the food.', '2024-12-02 20:30:00'),
    (28, 28, 5, 'Perfect place for a family vacation.', '2024-12-02 21:00:00'),
    (29, 29, 3, 'The room was too cold, even with the heater on.', '2024-12-02 21:30:00'),
    (30, 30, 4, 'Room was okay, but the lighting was a bit dim.', '2024-12-02 22:00:00'),
    (31, 31, 5, 'Wonderful experience, will definitely return!', '2024-12-02 22:30:00'),
    (32, 32, 4, 'Good stay, but the pillows were too hard.', '2024-12-02 23:00:00'),
    (33, 33, 5, 'Excellent! Room was clean and well-maintained.', '2024-12-02 23:30:00'),
    (34, 34, 3, 'Noisy guests disturbed the peace.', '2024-12-03 00:00:00'),
    (35, 35, 4, 'Great staff, but the room temperature was hard to adjust.', '2024-12-03 00:30:00'),
    (36, 36, 5, 'Highly recommend, great for a short stay!', '2024-12-03 01:00:00'),
    (37, 37, 4, 'Good stay overall, but could use better shower pressure.', '2024-12-03 01:30:00'),
    (38, 38, 5, 'I had a fantastic time, everything was perfect.', '2024-12-03 02:00:00'),
    (39, 39, 3, 'The room was smaller than expected.', '2024-12-03 02:30:00'),
    (40, 40, 4, 'Great hotel, but the air conditioning was loud.', '2024-12-03 03:00:00'),
    (41, 41, 5, 'Excellent, had a wonderful time!', '2024-12-03 03:30:00'),
    (42, 42, 4, 'The room was comfortable, but needs a few upgrades.', '2024-12-03 04:00:00'),
    (43, 43, 5, 'Fantastic stay, loved everything.', '2024-12-03 04:30:00'),
    (44, 44, 3, 'Room was okay but lacked amenities.', '2024-12-03 05:00:00'),
    (45, 45, 4, 'Nice stay, but the bed was a bit uncomfortable.', '2024-12-03 05:30:00'),
    (46, 46, 5, 'Absolutely perfect! Will definitely return.', '2024-12-03 06:00:00'),
    (47, 47, 4, 'Good stay, but check-in took too long.', '2024-12-03 06:30:00'),
    (48, 48, 5, 'A beautiful experience, I will be back!', '2024-12-03 07:00:00'),
    (49, 49, 3, 'Okay stay, but the pool was too small.', '2024-12-03 07:30:00'),
    (50, 50, 4, 'Great stay overall, just wished the breakfast was included.', '2024-12-03 08:00:00');
```

```sql
-- 1. Feedback Distribution by Room Type
-- Analyze how room type affects customer feedback scores.
SELECT
    r.RoomType_ID,
    rt.Base_Price AS Room_Base_Price,
    AVG(f.SCORE) AS Avg_Score,
    COUNT(f.FID) AS Total_Feedbacks
FROM Feedback_Summary f
JOIN Reservation_Order r ON f.RO_ID = r.RO_ID
JOIN Room_Type rt ON r.RoomType_ID = rt.Room_Type_ID
GROUP BY r.RoomType_ID, rt.Base_Price
ORDER BY Avg_Score DESC;

-- 2. Customers With and Without Pets
-- Compare the average feedback score of customers with pets versus those without pets.
SELECT
    c.HasPet AS Has_Pet,
    AVG(f.SCORE) AS Avg_Feedback_Score,
    COUNT(f.FID) AS Total_Feedbacks
FROM Feedback_Summary f
JOIN Customer c ON f.CUSTOMER_ID = c.CUSTOMER_ID
GROUP BY c.HasPet;

-- 3. Service Performance Analysis
-- Analyze feedback grouped by employees handling the reservations.
SELECT
    e.Employee_ID,
    e.EmployeeType,
    AVG(f.SCORE) AS Avg_Score,
    COUNT(f.FID) AS Total_Feedbacks
FROM Feedback_Summary f
JOIN Reservation_Order r ON f.RO_ID = r.RO_ID
JOIN Employee e ON r.Employee_ID = e.Employee_ID
GROUP BY e.Employee_ID, e.EmployeeType
ORDER BY Avg_Score DESC;

-- 4. Sentiment Analysis
-- Perform deeper analysis on feedback comments using natural language processing (NLP).
SELECT
    c.CUSTOMER_NAME AS Customer_Name,
    f.SUMMARY AS Feedback_Comment,
    f.SCORE AS Feedback_Score,
    r.RoomType_ID AS Room_Type,
    r.Length AS Stay_Length,
    r.Quantity AS Rooms_Reserved,
    f.RECEIVED_TIME AS Feedback_Time
FROM Feedback_Summary f
JOIN Customer c ON f.CUSTOMER_ID = c.CUSTOMER_ID
JOIN Reservation_Order r ON f.RO_ID = r.RO_ID
ORDER BY f.RECEIVED_TIME;
```

Customer Segmentation

December 12, 2024

[ ]:
```
# Query full data for each table and display it to the user

# Fetching full data for each table
customer_data = pd.read_sql_query("SELECT * FROM Customer;", conn)
reservation_order_data = pd.read_sql_query("SELECT * FROM Reservation_Order;",␣
 ↪conn)
payment_data = pd.read_sql_query("SELECT * FROM Payment;", conn)
```

[ ]:
```
# Correcting and re-implementing the SQL query for customer statistics
customer_statistics_query = """
WITH Customer_Statistics AS (
    SELECT
        c.Customer_ID,
        AVG(ro.Length * ro.Quantity * rt.Base_Price) AS Avg_Spending,
        COUNT(ro.RO_ID) AS Total_Visits,
        (SELECT rt.Room_Type_ID
         FROM Reservation_Order ro2
         JOIN Room_Type rt ON ro2.RoomType_ID = rt.Room_Type_ID
         WHERE ro2.PayerID = c.Customer_ID
         GROUP BY rt.Room_Type_ID
         ORDER BY COUNT(rt.Room_Type_ID) DESC
         LIMIT 1) AS Preferred_Room_Type,
        (SELECT strftime('%m', gs.Check_In_Date)
         FROM Guest_Stay gs
         WHERE gs.Customer_ID = c.Customer_ID
         GROUP BY strftime('%m', gs.Check_In_Date)
         ORDER BY COUNT(gs.Check_In_Date) DESC
         LIMIT 1) AS Preferred_Booking_Season
    FROM Customer c
    JOIN Reservation_Order ro ON c.Customer_ID = ro.PayerID
    JOIN Room_Type rt ON ro.RoomType_ID = rt.Room_Type_ID
    GROUP BY c.Customer_ID
)
SELECT * FROM Customer_Statistics;
"""

# Executing the query and fetching results into a DataFrame
```

```
customer_statistics_df = pd.read_sql_query(customer_statistics_query, conn)

# Displaying the result to the user
tools.display_dataframe_to_user(name="Customer Statistics for Segmentation",␣
 ↪dataframe=customer_statistics_df)
```

[ ]:
```
# Handling NULL values in Preferred_Booking_Season and Preferred_Room_Type␣
 ↪columns
customer_statistics_df['Preferred_Booking_Season'] =␣
 ↪customer_statistics_df['Preferred_Booking_Season'].fillna(0).astype(int)
customer_statistics_df['Preferred_Room_Type'] =␣
 ↪customer_statistics_df['Preferred_Room_Type'].fillna(0).astype(int)

# Selecting features for clustering again
features = customer_statistics_df[['Avg_Spending', 'Total_Visits',␣
 ↪'Preferred_Room_Type', 'Preferred_Booking_Season']]

# Applying K-means clustering
kmeans = KMeans(n_clusters=4, random_state=42)
customer_statistics_df['Cluster'] = kmeans.fit_predict(features)

# Visualizing the clusters using a scatter plot (Avg_Spending vs Total_Visits)
plt.figure(figsize=(10, 6))
plt.scatter(customer_statistics_df['Avg_Spending'],␣
 ↪customer_statistics_df['Total_Visits'],
            c=customer_statistics_df['Cluster'], cmap='viridis', alpha=0.7)
plt.colorbar(label='Cluster')
plt.xlabel('Average Spending')
plt.ylabel('Total Visits')
plt.title('Customer Segmentation: Avg Spending vs Total Visits')
plt.grid(True)
plt.show()

# Displaying the clustered data to the user
tools.display_dataframe_to_user(name="Customer Segmentation Clusters with␣
 ↪Resolved Issues", dataframe=customer_statistics_df)
```

[ ]:
```
# Identifying insights and summarizing clusters with descriptive statistics

# Grouping data by clusters to understand customer behavior in each cluster
cluster_summary = customer_statistics_df.groupby('Cluster').agg({
    'Avg_Spending': ['mean', 'median', 'max', 'min'],
    'Total_Visits': ['mean', 'median', 'max', 'min'],
    'Preferred_Room_Type': lambda x: x.mode()[0],  # Most common room type
    'Preferred_Booking_Season': lambda x: x.mode()[0],  # Most common booking␣
 ↪season
```

```
}).reset_index()

# Renaming columns for better readability
cluster_summary.columns = [
    'Cluster',
    'Avg_Spending_Mean', 'Avg_Spending_Median', 'Avg_Spending_Max',␣
 ↪'Avg_Spending_Min',
    'Total_Visits_Mean', 'Total_Visits_Median', 'Total_Visits_Max',␣
 ↪'Total_Visits_Min',
    'Preferred_Room_Type', 'Preferred_Booking_Season'
]

# Displaying the summary of customer segments to the user
import ace_tools as tools; tools.display_dataframe_to_user(name="Customer␣
 ↪Segmentation Cluster Summary", dataframe=cluster_summary)
```

[ ]:
```
# Fixing the radar chart preparation process

# Normalizing the data again, ensuring valid column selection
cluster_summary_normalized = cluster_summary.copy()
numerical_columns = [
    'Avg_Spending_Mean', 'Avg_Spending_Median',
    'Total_Visits_Mean', 'Total_Visits_Median',
    'Preferred_Room_Type', 'Preferred_Booking_Season'
]

# Normalize only the numerical columns
cluster_summary_normalized[numerical_columns] =␣
 ↪cluster_summary_normalized[numerical_columns].apply(
    lambda x: (x - x.min()) / (x.max() - x.min())
)

# Repeating the first column for radar plot closure
values_list = cluster_summary_normalized[numerical_columns].values
categories = numerical_columns + [numerical_columns[0]]

# Radar chart setup
fig, ax = plt.subplots(figsize=(10, 8), subplot_kw=dict(polar=True))

for i in range(len(values_list)):
    values = np.append(values_list[i], values_list[i][0])  # Repeat first value
    ax.plot(
        np.linspace(0, 2 * np.pi, len(categories)),
        values,
        label=f"Cluster {cluster_summary_normalized.iloc[i, 0]}"
    )
    ax.fill(
```

```
        np.linspace(0, 2 * np.pi, len(categories)),
        values,
        alpha=0.25
    )

# Customizing the radar chart
ax.set_xticks(np.linspace(0, 2 * np.pi, len(categories)))
ax.set_xticklabels(categories, fontsize=10)
ax.set_title("Customer Segmentation Radar Chart", fontsize=14)
ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))
plt.show()
```

[ ]:
```
# Visualizing customer segments using a scatter plot with Avg_Spending vs␣
↪Total_Visits

plt.figure(figsize=(10, 6))

# Scatter plot of Avg_Spending vs Total_Visits, colored by cluster
scatter = plt.scatter(
    customer_statistics_df['Avg_Spending'],
    customer_statistics_df['Total_Visits'],
    c=customer_statistics_df['Cluster'],
    cmap='viridis',
    alpha=0.7
)

# Adding plot details
plt.colorbar(scatter, label='Cluster')
plt.xlabel('Average Spending')
plt.ylabel('Total Visits')
plt.title('Customer Segmentation: Avg Spending vs Total Visits')
plt.grid(True)
plt.show()
```

Query 2 Data Generation

December 12, 2024

[ ]:
```
%sql
-- Create table for Items
DROP TABLE IF EXISTS Item;
CREATE TABLE Item (
    Item_ID INT,
    Item_Name STRING,
    Item_Type STRING
)
USING DELTA;

-- Create table for Service Orders
DROP TABLE IF EXISTS Service_Order;
CREATE TABLE Service_Order (
    Service_Order_ID INT,
    Service_Order_Date DATE
)
USING DELTA;

-- Create table to link Items with Service Orders
DROP TABLE IF EXISTS Item_IsAssignedTo_ServiceOrder;
CREATE TABLE Item_IsAssignedTo_ServiceOrder (
    Service_Order_ID INT,
    Item_ID INT,
    Quantity INT
)
USING DELTA;

-- Create table for Events
DROP TABLE IF EXISTS Event;
CREATE TABLE Event (
    Event_ID INT,
    Event_Name STRING,
    Event_Date DATE,
    Event_Type STRING
)
USING DELTA;
```

[ ]:
```
%sql
INSERT INTO Item VALUES
(1, 'Pillow', 'Housekeeping'),
(2, 'Blanket', 'Housekeeping'),
(3, 'Water Glass', 'Amenities'),
(4, 'Shampoo', 'Amenities'),
(5, 'Slippers', 'Guest Services'),
(6, 'Kettle', 'Guest Services'),
(7, 'Body Wash', 'Amenities'),
(8, 'Flashlight', 'Event Supplies'),
(9, 'Towel', 'Amenities'),
(10, 'Camping Lantern', 'Event Supplies'),
(11, 'Tea Bag', 'Amenities'),
(12, 'Mini Fridge', 'Guest Services'),
(13, 'Hand Soap', 'Amenities'),
(14, 'Ironing Board', 'Guest Services'),
(15, 'Projector', 'Event Supplies'),
(16, 'Hair Dryer', 'Guest Services'),
(17, 'Tissues', 'Amenities'),
(18, 'Laundry Bag', 'Housekeeping'),
(19, 'Microphone', 'Event Supplies'),
(20, 'Robe', 'Guest Services'),
(21, 'Face Towel', 'Amenities'),
(22, 'Stage', 'Event Supplies'),
(23, 'Hand Sanitizer', 'Amenities'),
(24, 'Cups', 'Amenities'),
(25, 'Portable Air Conditioner', 'Event Supplies'),
(26, 'Lotion', 'Amenities'),
(27, 'Catering Equipment', 'Event Supplies'),
(28, 'Coffee Maker', 'Guest Services'),
(29, 'Cooler Box', 'Event Supplies'),
(30, 'Room Freshener', 'Amenities'),
(31, 'Extension Cords', 'Event Supplies'),
(32, 'Laundry Detergent', 'Guest Services'),
(33, 'Barbecue Grill', 'Event Supplies'),
(34, 'Event Signage', 'Event Supplies'),
(35, 'DJ Equipment', 'Event Supplies'),
(36, 'Tables', 'Event Supplies'),
(37, 'Toothbrush', 'Amenities'),
(38, 'Toothpaste', 'Amenities'),
(39, 'Water Bottle', 'Amenities'),
(40, 'Fire Pit', 'Event Supplies'),
(41, 'Bandages', 'Health'),
(42, 'Sunglasses', 'Guest Services'),
(43, 'Party Hats', 'Event Supplies'),
(44, 'Tents', 'Event Supplies'),
(45, 'Decorations', 'Event Supplies'),
```

```
(46, 'Chairs', 'Event Supplies'),
(47, 'Basket', 'Amenities'),
(48, 'Serving Tray', 'Guest Services'),
(49, 'Room Key', 'Guest Services'),
(50, 'Trash Bin', 'Housekeeping');
```

[ ]:
```
%sql
INSERT INTO Service_Order VALUES
-- Dates for 2022
(1, '2022-01-05'), (2, '2022-01-08'), (3, '2022-01-12'), (4, '2022-01-15'), (5,␣
 ↪'2022-01-20'),
(6, '2022-02-01'), (7, '2022-02-02'), (8, '2022-02-07'), (9, '2022-02-10'),␣
 ↪(10, '2022-02-15'),
(11, '2022-02-22'), (12, '2022-02-25'), (13, '2022-03-05'), (14, '2022-03-12'),␣
 ↪(15, '2022-03-15'),
(16, '2022-03-22'), (17, '2022-03-25'), (18, '2022-04-10'), (19, '2022-04-15'),␣
 ↪(20, '2022-04-20'),
(21, '2022-05-01'), (22, '2022-05-03'), (23, '2022-05-10'), (24, '2022-05-15'),␣
 ↪(25, '2022-05-20'),
(26, '2022-06-05'), (27, '2022-06-07'), (28, '2022-06-15'), (29, '2022-06-20'),␣
 ↪(30, '2022-06-25'),
(31, '2022-07-01'), (32, '2022-07-03'), (33, '2022-07-10'), (34, '2022-07-12'),␣
 ↪(35, '2022-07-18'),
(36, '2022-07-25'), (37, '2022-08-01'), (38, '2022-08-05'), (39, '2022-08-10'),␣
 ↪(40, '2022-08-15'),
(41, '2022-08-17'), (42, '2022-08-25'), (43, '2022-09-01'), (44, '2022-09-03'),␣
 ↪(45, '2022-09-08'),
(46, '2022-09-12'), (47, '2022-09-17'), (48, '2022-09-20'), (49, '2022-09-25'),␣
 ↪(50, '2022-10-01'),
(51, '2022-10-05'), (52, '2022-10-10'), (53, '2022-10-12'), (54, '2022-10-15'),␣
 ↪(55, '2022-10-19'),
(56, '2022-10-25'), (57, '2022-11-01'), (58, '2022-11-03'), (59, '2022-11-05'),␣
 ↪(60, '2022-11-10'),
(61, '2022-11-12'), (62, '2022-11-18'), (63, '2022-11-22'), (64, '2022-11-27'),␣
 ↪(65, '2022-12-01'),
(66, '2022-12-05'), (67, '2022-12-07'), (68, '2022-12-10'), (69, '2022-12-12'),␣
 ↪(70, '2022-12-15'),
(71, '2022-12-18'), (72, '2022-12-20'), (73, '2022-12-25'), (74, '2022-12-27'),␣
 ↪(75, '2022-12-29'),

-- Dates for 2023
(76, '2023-01-05'), (77, '2023-01-08'), (78, '2023-01-13'), (79, '2023-01-15'),␣
 ↪(80, '2023-01-20'),
(81, '2023-01-25'), (82, '2023-01-30'), (83, '2023-02-05'), (84, '2023-02-08'),␣
 ↪(85, '2023-02-15'),
```

```
(86, '2023-02-18'), (87, '2023-02-23'), (88, '2023-03-02'), (89, '2023-03-03'),␣
 ↪(90, '2023-03-05'),
(91, '2023-03-10'), (92, '2023-03-12'), (93, '2023-03-18'), (94, '2023-03-19'),␣
 ↪(95, '2023-03-25'),
(96, '2023-03-30'), (97, '2023-04-02'), (98, '2023-04-04'), (99, '2023-04-07'),␣
 ↪(100, '2023-04-12'),
(101, '2023-04-15'), (102, '2023-04-18'), (103, '2023-04-22'), (104,␣
 ↪'2023-04-25'), (105, '2023-05-01'),
(106, '2023-05-05'), (107, '2023-05-07'), (108, '2023-05-12'), (109,␣
 ↪'2023-05-15'), (110, '2023-05-18'),
(111, '2023-05-22'), (112, '2023-05-29'), (113, '2023-06-01'), (114,␣
 ↪'2023-06-04'), (115, '2023-06-06'),
(116, '2023-06-10'), (117, '2023-06-13'), (118, '2023-06-15'), (119,␣
 ↪'2023-06-18'), (120, '2023-06-21'),
(121, '2023-06-25'), (122, '2023-07-01'), (123, '2023-07-04'), (124,␣
 ↪'2023-07-06'), (125, '2023-07-08'),
(126, '2023-07-10'), (127, '2023-07-15'), (128, '2023-07-16'), (129,␣
 ↪'2023-07-23'), (130, '2023-07-25'),
(131, '2023-08-01'), (132, '2023-08-06'), (133, '2023-08-09'), (134,␣
 ↪'2023-08-12'), (135, '2023-08-15'),
(136, '2023-08-19'), (137, '2023-08-22'), (138, '2023-08-25'), (139,␣
 ↪'2023-08-28'), (140, '2023-09-01'),
(141, '2023-09-02'), (142, '2023-09-05'), (143, '2023-09-10'), (144,␣
 ↪'2023-09-13'), (145, '2023-09-17'),
(146, '2023-09-18'), (147, '2023-09-20'), (148, '2023-09-23'), (149,␣
 ↪'2023-09-25'), (150, '2023-09-30'),
(151, '2023-10-03'), (152, '2023-10-05'), (153, '2023-10-07'), (154,␣
 ↪'2023-10-10'), (155, '2023-10-12'),
(156, '2023-10-16'), (157, '2023-10-18'), (158, '2023-10-20'), (159,␣
 ↪'2023-10-25'), (160, '2023-10-28'),
(161, '2023-11-01'), (162, '2023-11-04'), (163, '2023-11-10'), (164,␣
 ↪'2023-11-12'), (165, '2023-11-14'),
(166, '2023-11-18'), (167, '2023-11-20'), (168, '2023-11-25'), (169,␣
 ↪'2023-11-27'), (170, '2023-12-01'),
(171, '2023-12-03'), (172, '2023-12-05'), (173, '2023-12-06'), (174,␣
 ↪'2023-12-10'), (175, '2023-12-12'),
(176, '2023-12-13'), (177, '2023-12-17'), (178, '2023-12-20'), (179,␣
 ↪'2023-12-25'), (180, '2023-12-29'),

-- Dates for 2024
(181, '2024-01-03'), (182, '2024-01-04'), (183, '2024-01-07'), (184,␣
 ↪'2024-01-12'), (185, '2024-01-14'),
(186, '2024-01-17'), (187, '2024-01-18'), (188, '2024-01-19'), (189,␣
 ↪'2024-01-23'), (190, '2024-01-25'),
```

```
(191, '2024-01-29'), (192, '2024-02-01'), (193, '2024-02-04'), (194,␣
 ↪'2024-02-08'), (195, '2024-02-10'),
(196, '2024-02-11'), (197, '2024-02-16'), (198, '2024-02-17'), (199,␣
 ↪'2024-02-18'), (200, '2024-02-19'),
(201, '2024-02-22'), (202, '2024-02-25'), (203, '2024-02-28'), (204,␣
 ↪'2024-03-01'), (205, '2024-03-02'),
(206, '2024-03-06'), (207, '2024-03-09'), (208, '2024-03-12'), (209,␣
 ↪'2024-03-14'), (210, '2024-03-16'),
(211, '2024-03-18'), (212, '2024-03-21'), (213, '2024-03-23'), (214,␣
 ↪'2024-03-25'), (215, '2024-03-31'),
(216, '2024-04-01'), (217, '2024-04-03'), (218, '2024-04-04'), (219,␣
 ↪'2024-04-08'), (220, '2024-04-09'),
(221, '2024-04-10'), (222, '2024-04-13'), (223, '2024-04-16'), (224,␣
 ↪'2024-04-18'), (225, '2024-04-21'),
(226, '2024-04-24'), (227, '2024-04-26'), (228, '2024-04-29'), (229,␣
 ↪'2024-05-02'), (230, '2024-05-05'),
(231, '2024-05-07'), (232, '2024-05-09'), (233, '2024-05-13'), (234,␣
 ↪'2024-05-15'), (235, '2024-05-18'),
(236, '2024-05-22'), (237, '2024-05-25'), (238, '2024-05-28'), (239,␣
 ↪'2024-05-30'), (240, '2024-06-01'),
(241, '2024-06-02'), (242, '2024-06-03'), (243, '2024-06-06'), (244,␣
 ↪'2024-06-07'), (245, '2024-06-08'),
(246, '2024-06-10'), (247, '2024-06-12'), (248, '2024-06-14'), (249,␣
 ↪'2024-06-16'), (250, '2024-06-18'),
(251, '2024-06-20'), (252, '2024-06-22'), (253, '2024-06-24'), (254,␣
 ↪'2024-06-26'), (255, '2024-06-28'),
(256, '2024-07-01'), (257, '2024-07-03'), (258, '2024-07-04'), (259,␣
 ↪'2024-07-06'), (260, '2024-07-07'),
(261, '2024-07-10'), (262, '2024-07-12'), (263, '2024-07-13'), (264,␣
 ↪'2024-07-15'), (265, '2024-07-17'),
(266, '2024-07-19'), (267, '2024-07-20'), (268, '2024-07-22'), (269,␣
 ↪'2024-07-25'), (270, '2024-07-28'),
(271, '2024-07-30'), (272, '2024-08-01'), (273, '2024-08-03'), (274,␣
 ↪'2024-08-05'), (275, '2024-08-07'),
(276, '2024-08-09'), (277, '2024-08-10'), (278, '2024-08-12'), (279,␣
 ↪'2024-08-14'), (280, '2024-08-16'),
(281, '2024-08-18'), (282, '2024-08-20'), (283, '2024-08-22'), (284,␣
 ↪'2024-08-24'), (285, '2024-08-26'),
(286, '2024-08-28'), (287, '2024-08-30'), (288, '2024-09-01'), (289,␣
 ↪'2024-09-03'), (290, '2024-09-05'),
(291, '2024-09-07'), (292, '2024-09-09'), (293, '2024-09-11'), (294,␣
 ↪'2024-09-13'), (295, '2024-09-15'),
(296, '2024-09-17'), (297, '2024-09-19'), (298, '2024-09-21'), (299,␣
 ↪'2024-09-23'), (300, '2024-09-25'),
```

```
(301, '2024-10-01'), (302, '2024-10-03'), (303, '2024-10-06'), (304,␣
 ↪'2024-10-07'), (305, '2024-10-10'),
(306, '2024-10-12'), (307, '2024-10-14'), (308, '2024-10-16'), (309,␣
 ↪'2024-10-18'), (310, '2024-10-20'),
(311, '2024-10-22'), (312, '2024-10-25'), (313, '2024-10-28'), (314,␣
 ↪'2024-10-30'), (315, '2024-11-01'),
(316, '2024-11-03'), (317, '2024-11-05'), (318, '2024-11-07'), (319,␣
 ↪'2024-11-09'), (320, '2024-11-12'),
(321, '2024-11-14'), (322, '2024-11-16'), (323, '2024-11-18'), (324,␣
 ↪'2024-11-20'), (325, '2024-11-22'),
(326, '2024-11-24'), (327, '2024-11-26'), (328, '2024-11-28'), (329,␣
 ↪'2024-11-30'), (330, '2024-12-02'),
(331, '2024-12-04'), (332, '2024-12-06'), (333, '2024-12-08'), (334,␣
 ↪'2024-12-10'), (335, '2024-12-12'),
(336, '2024-12-14'), (337, '2024-12-16'), (338, '2024-12-18'), (339,␣
 ↪'2024-12-20'), (340, '2024-12-22'),
(341, '2024-12-24'), (342, '2024-12-26'), (343, '2024-12-28'), (344,␣
 ↪'2024-12-30');
```

[ ]:
```
%sql
INSERT INTO Item_IsAssignedTo_ServiceOrder VALUES
(1, 8, 10),
(2, 10, 4),
(3, 19, 8),
(4, 22, 17),
(5, 23, 25),
(6, 25, 9),
(7, 27, 7),
(8, 29, 5),
(9, 33, 12),
(10, 36, 10),
(11, 40, 18),
(12, 42, 5),
(13, 45, 9),
(14, 48, 14),
(15, 50, 10),
(16, 34, 5),
(17, 33, 12),
(18, 29, 10),
(19, 20, 6),
(20, 25, 6),
(21, 38, 18),
(22, 27, 6),
(23, 36, 7),
(24, 46, 9),
(25, 37, 8),
```

```
(26, 27, 5),
(27, 22, 14),
(28, 12, 17),
(29, 15, 10),
(30, 47, 7),
(31, 41, 15),
(32, 11, 14),
(33, 34, 18),
(34, 44, 5),
(35, 10, 20),
(36, 18, 24),
(37, 50, 16),
(38, 43, 17),
(39, 19, 9),
(40, 8, 11),
(41, 39, 14),
(42, 35, 15),
(43, 3, 19),
(44, 9, 20),
(45, 26, 18),
(46, 7, 21),
(47, 41, 20),
(48, 44, 7),
(49, 16, 16),
(50, 28, 22),
(51, 30, 14),
(52, 33, 7),
(53, 21, 18),
(54, 9, 16),
(55, 6, 25),
(56, 11, 8),
(57, 40, 15),
(58, 24, 12),
(59, 23, 11),
(60, 17, 22),
(61, 32, 7),
(62, 43, 5),
(63, 5, 19),
(64, 18, 17),
(65, 16, 13),
(66, 41, 6),
(67, 13, 18),
(68, 2, 21),
(69, 36, 10),
(70, 7, 19),
(71, 26, 5),
(72, 28, 6),
```

```
(73, 50, 9),
(74, 42, 8),
(75, 12, 20),
(76, 19, 14),
(77, 25, 22),
(78, 14, 24),
(79, 46, 18),
(80, 20, 9),
(81, 3, 20),
(82, 15, 10),
(83, 29, 19),
(84, 38, 22),
(85, 16, 16),
(86, 44, 18),
(87, 10, 7),
(88, 41, 17),
(89, 50, 16),
(90, 30, 8),
(91, 21, 10),
(92, 35, 9),
(93, 4, 11),
(94, 29, 12),
(95, 38, 21),
(96, 23, 14),
(97, 28, 17),
(98, 13, 22),
(99, 27, 9),
(100, 8, 13),
(101, 12, 10),
(102, 34, 18),
(103, 44, 6),
(104, 10, 9),
(105, 42, 10),
(106, 45, 14),
(107, 49, 7),
(108, 19, 18),
(109, 37, 21),
(110, 50, 5),
(111, 30, 20),
(112, 18, 13),
(113, 2, 12),
(114, 41, 10),
(115, 25, 23),
(116, 34, 7),
(117, 22, 5),
(118, 3, 14),
(119, 29, 19),
```

```
(120, 5, 20),
(121, 17, 9),
(122, 47, 8),
(123, 24, 15),
(124, 44, 5),
(125, 39, 17),
(126, 8, 9),
(127, 27, 12),
(128, 36, 5),
(129, 40, 9),
(130, 10, 15),
(131, 29, 8),
(132, 16, 10),
(133, 44, 20),
(134, 31, 7),
(135, 47, 18),
(136, 23, 11),
(137, 15, 6),
(138, 39, 16),
(139, 35, 20),
(140, 27, 13),
(141, 43, 7),
(142, 13, 22),
(143, 25, 5),
(144, 7, 16),
(145, 33, 10),
(146, 50, 7),
(147, 41, 8),
(148, 18, 19),
(149, 21, 13),
(150, 3, 10),
(151, 31, 18),
(152, 19, 9),
(153, 7, 14),
(154, 40, 20),
(155, 30, 13),
(156, 12, 5),
(157, 22, 14),
(158, 47, 6),
(159, 18, 23),
(160, 39, 11),
(161, 16, 15),
(162, 50, 18),
(163, 25, 10),
(164, 30, 5),
(165, 5, 18),
(166, 41, 22),
```

```
(167, 12, 19),
(168, 2, 17),
(169, 7, 11),
(170, 8, 14),
(171, 24, 10),
(172, 36, 12),
(173, 49, 13),
(174, 13, 14),
(175, 27, 7),
(176, 44, 21),
(177, 10, 19),
(178, 31, 15),
(179, 14, 12),
(180, 33, 8),
(181, 50, 10),
(182, 25, 13),
(183, 7, 16),
(184, 41, 19),
(185, 19, 13),
(186, 45, 9),
(187, 3, 17),
(188, 47, 11),
(189, 14, 13),
(190, 42, 12),
(191, 9, 10),
(192, 31, 9),
(193, 26, 6),
(194, 37, 18),
(195, 34, 9),
(196, 8, 17),
(197, 16, 14),
(198, 21, 13),
(199, 47, 15),
(200, 32, 11),
(201, 18, 10),
(202, 28, 14),
(203, 6, 12),
(204, 29, 10),
(205, 42, 5),
(206, 33, 12),
(207, 9, 8),
(208, 43, 14),
(209, 15, 11),
(210, 24, 18),
(211, 34, 7),
(212, 10, 14),
(213, 27, 19),
```

```
(214, 5, 15),
(215, 41, 7),
(216, 14, 20),
(217, 37, 13),
(218, 19, 14),
(219, 21, 9),
(220, 2, 11),
(221, 6, 20),
(222, 30, 14),
(223, 36, 15),
(224, 16, 5),
(225, 8, 21),
(226, 50, 16),
(227, 13, 18),
(228, 31, 5),
(229, 23, 9),
(230, 34, 10),
(231, 47, 17),
(232, 41, 10),
(233, 27, 13),
(234, 50, 5),
(235, 30, 10),
(236, 10, 17),
(237, 44, 11),
(238, 35, 20),
(239, 39, 9),
(240, 18, 14),
(241, 5, 13),
(242, 42, 10),
(243, 3, 14),
(244, 8, 13),
(245, 37, 8),
(246, 47, 10),
(247, 41, 16),
(248, 19, 10),
(249, 31, 14),
(250, 44, 7),
(251, 10, 15),
(252, 35, 14),
(253, 18, 22),
(254, 32, 9),
(255, 27, 17),
(256, 7, 11),
(257, 30, 12),
(258, 5, 19),
(259, 39, 15),
(260, 41, 18),
```

```
(261, 8, 11),
(262, 10, 20),
(263, 21, 7),
(264, 44, 19),
(265, 50, 12),
(266, 37, 20),
(267, 19, 11),
(268, 15, 9),
(269, 2, 22),
(270, 13, 8),
(271, 31, 16),
(272, 28, 9),
(273, 30, 14),
(274, 38, 7),
(275, 42, 18),
(276, 5, 22),
(277, 19, 12),
(278, 47, 15),
(279, 21, 9),
(280, 44, 5),
(281, 9, 18),
(282, 31, 17),
(283, 36, 13),
(284, 10, 19),
(285, 47, 18),
(286, 21, 6),
(287, 2, 17),
(288, 14, 20),
(289, 33, 12),
(290, 39, 9),
(291, 19, 5),
(292, 25, 16),
(293, 8, 10),
(294, 42, 7),
(295, 36, 20),
(296, 7, 22),
(297, 5, 18),
(298, 27, 14),
(299, 37, 10),
(300, 8, 13),
(301, 16, 11),
(302, 30, 17),
(303, 43, 16),
(304, 34, 14),
(305, 3, 16),
(306, 23, 9),
(307, 18, 11),
```

```
(308, 39, 7),
(309, 45, 10),
(310, 32, 18),
(311, 36, 16),
(312, 41, 7),
(313, 19, 10),
(314, 8, 22),
(315, 45, 9),
(316, 27, 10),
(317, 33, 18),
(318, 10, 12),
(319, 44, 5),
(320, 18, 17),
(321, 12, 14),
(322, 50, 15),
(323, 30, 6),
(324, 34, 8),
(325, 28, 7),
(326, 47, 17),
(327, 16, 13),
(328, 2, 18),
(329, 40, 6),
(330, 9, 15),
(331, 14, 11),
(332, 35, 14),
(333, 38, 10),
(334, 33, 15),
(335, 5, 16),
(336, 48, 14),
(337, 50, 13),
(338, 21, 7),
(339, 13, 5),
(340, 27, 10),
(341, 41, 18),
(342, 15, 7),
(343, 34, 9),
(344, 9, 16);
```

[ ]:
```
%sql
INSERT INTO Event VALUES
(1, 'Christmas Party', '2024-12-24', 'Grandma\'s'),
(2, 'New Year Celebration', '2024-12-31', 'Grandma\'s'),
(3, 'Skiing Season', '2024-12-20', 'Grandma\'s'),
(4, 'Beach Festival', '2024-06-15', 'Customized'),
(5, 'Spring Carnival', '2024-04-10', 'Customized'),
(6, 'Autumn Gala', '2024-10-05', 'Customized'),
(7, 'Summer Concert', '2024-08-12', 'Customized'),
```

```
(8, 'Mountain Hike Event', '2024-09-20', 'Customized'),
(9, 'Thanksgiving Dinner', '2024-11-28', 'Grandma\'s'),
(10, 'Valentine\'s Day Gala', '2024-02-14', 'Grandma\'s'),
(11, 'Corporate Retreat', '2024-11-01', 'Customized'),
(12, 'Eco Awareness Festival', '2024-09-15', 'Customized'),
(13, 'Wine and Food Tasting', '2024-05-10', 'Customized'),
(14, 'Cultural Parade', '2024-07-04', 'Customized'),
(15, 'Winter Wonderland', '2024-12-10', 'Grandma\'s'),
(16, 'Charity Marathon', '2024-10-22', 'Customized'),
(17, 'Halloween Party', '2024-10-31', 'Grandma\'s'),
(18, 'Horse Riding Show', '2024-07-20', 'Customized'),
(19, 'Bird Watching Camp', '2024-06-20', 'Customized'),
(20, 'Rock Concert', '2024-08-30', 'Customized'),
(21, 'Photography Tour', '2024-05-25', 'Customized'),
(22, 'Yoga Festival', '2024-07-18', 'Customized'),
(23, 'Beach Clean-Up Drive', '2024-08-22', 'Customized'),
(24, 'Fishing Derby', '2024-06-05', 'Customized'),
(25, 'Cycling Marathon', '2024-07-30', 'Customized'),
(26, 'Meditation Retreat', '2024-09-10', 'Customized'),
(27, 'Tennis Championship', '2024-07-08', 'Customized'),
(28, 'Hot Air Balloon Ride', '2024-01-10', 'Customized'),
(29, 'Scuba Diving Adventure', '2024-05-14', 'Customized'),
(30, 'Snowboarding Festival', '2024-12-14', 'Customized'),
(31, 'Brewery Tour', '2024-06-22', 'Customized'),
(32, 'Cultural Dance Fest', '2024-09-18', 'Customized'),
(33, 'Outdoor Film Screening', '2024-07-28', 'Customized'),
(34, 'Stargazing Night', '2024-08-18', 'Customized'),
(35, 'Music Festival', '2024-08-05', 'Customized'),
(36, 'Wildlife Safari', '2024-08-25', 'Customized'),
(37, 'Craft Workshop', '2024-11-04', 'Customized'),
(38, 'Farm Visit', '2024-10-12', 'Customized'),
(39, 'Art Exhibition', '2024-11-20', 'Customized'),
(40, 'Live Theatre', '2024-10-18', 'Customized'),
(41, 'Holiday Market', '2024-12-22', 'Customized'),
(42, 'Snowshoeing Tour', '2024-12-08', 'Grandma\'s'),
(43, 'Farmers Market Fair', '2024-09-05', 'Customized'),
(44, 'Zip Lining Adventure', '2024-09-12', 'Customized'),
(45, 'Orchestra Night', '2024-11-12', 'Customized'),
(46, 'Dance Workshop', '2024-10-15', 'Customized'),
(47, 'Stargazing Adventure', '2024-01-18', 'Customized'),
(48, 'Flower Exhibition', '2024-03-21', 'Customized'),
(49, 'Picnic Party', '2024-05-01', 'Customized'),
(50, 'Harvest Festival', '2024-09-30', 'Customized'),
(51, 'New Year Countdown', '2022-12-31', 'Grandma\'s'),
(52, 'Spring Blossom Festival', '2023-03-10', 'Grandma\'s'),
(53, 'Summer Music Jam', '2023-06-21', 'Grandma\'s'),
(54, 'Autumn Harvest Feast', '2023-09-05', 'Grandma\'s'),
```

```
(55, 'Winter Ice Sculpture Display', '2023-12-22', 'Grandma\'s'),
(56, 'Valentine\'s Day Dinner', '2023-02-14', 'Grandma\'s'),
(57, 'Beach Volleyball Tournament', '2023-06-07', 'Grandma\'s'),
(58, 'Jazz Festival', '2023-07-15', 'Grandma\'s'),
(59, 'Fall Nature Hike', '2023-10-10', 'Grandma\'s'),
(60, 'Halloween Haunted House', '2023-10-31', 'Grandma\'s'),
(61, 'Christmas Tree Lighting', '2023-12-01', 'Grandma\'s'),
(62, 'Easter Egg Hunt', '2023-04-09', 'Grandma\'s'),
(63, 'Independence Day Parade', '2023-07-04', 'Grandma\'s'),
(64, 'Wine Harvest Tour', '2023-09-20', 'Grandma\'s'),
(65, 'Candlelight Concert', '2023-11-15', 'Grandma\'s'),
(66, 'Mardi Gras Party', '2023-02-18', 'Grandma\'s'),
(67, 'Spring Picnic', '2023-04-25', 'Grandma\'s'),
(68, 'Summer Outdoor Movie Night', '2023-08-02', 'Grandma\'s'),
(69, 'Autumn Wine Tasting', '2023-10-05', 'Grandma\'s'),
(70, 'Winter Ski Tournament', '2023-12-05', 'Grandma\'s'),
(71, 'Charity Gala', '2023-11-30', 'Grandma\'s'),
(72, 'St. Patrick\'s Day Parade', '2023-03-17', 'Grandma\'s'),
(73, 'Food Truck Festival', '2023-05-20', 'Grandma\'s'),
(74, 'Mountain Climbing Expedition', '2023-08-15', 'Grandma\'s'),
(75, 'Biking Challenge', '2023-09-10', 'Grandma\'s'),
(76, 'Folk Music Festival', '2023-07-02', 'Grandma\'s'),
(77, 'Hot Air Balloon Festival', '2023-09-25', 'Grandma\'s'),
(78, 'Science Expo', '2023-06-30', 'Grandma\'s'),
(79, 'Yoga Retreat', '2023-05-30', 'Grandma\'s'),
(80, 'Christmas Market', '2023-12-18', 'Grandma\'s'),
(81, 'Summer Campfire Night', '2023-06-28', 'Grandma\'s'),
(82, 'Cultural Night', '2023-08-10', 'Grandma\'s'),
(83, 'Football Tournament', '2023-10-12', 'Grandma\'s'),
(84, 'New Year\'s Day Parade', '2022-01-01', 'Grandma\'s'),
(85, 'Spring Art Show', '2022-04-15', 'Grandma\'s'),
(86, 'Summer BBQ Bash', '2022-06-18', 'Grandma\'s'),
(87, 'Autumn Gala Dinner', '2022-10-08', 'Grandma\'s'),
(88, 'Winter Holiday Ice Skating', '2022-12-20', 'Grandma\'s'),
(89, 'Easter Parade', '2022-04-17', 'Grandma\'s'),
(90, 'Fourth of July Fireworks', '2022-07-04', 'Grandma\'s'),
(91, 'Music in the Park', '2022-08-25', 'Grandma\'s'),
(92, 'Oktoberfest Celebration', '2022-09-30', 'Grandma\'s'),
(93, 'Christmas Market Fair', '2022-12-10', 'Grandma\'s'),
(94, 'Halloween Carnival', '2022-10-31', 'Grandma\'s'),
(95, 'Winter Wonderland Ball', '2022-12-15', 'Grandma\'s'),
(96, 'New Year's Eve Gala', '2022-12-31', 'Grandma\'s'),
(97, 'Cherry Blossom Festival', '2022-03-25', 'Grandma\'s'),
(98, 'Outdoor Sports Challenge', '2022-05-18', 'Grandma\'s'),
(99, 'Seaside Festival', '2022-07-07', 'Grandma\'s'),
(100, 'Cultural Dance Show', '2022-09-11', 'Grandma\'s');
```

[ ]:
```
%sql
WITH Event_Present_Dates AS (
    SELECT DISTINCT Event_Date AS Date
    FROM Event
)
SELECT
    SO.Service_Order_Date,
    I.Item_Name,
    I.Item_Type,
    ISAS.Quantity,
    E.Event_Name,
    E.Event_Type,
    (CASE WHEN SO.Service_Order_Date IN (SELECT Date FROM Event_Present_Dates)␣
 ↪THEN TRUE ELSE FALSE END) AS Is_Event_Day
FROM
    Item_IsAssignedTo_ServiceOrder ISAS
JOIN
    Service_Order SO
ON
    ISAS.Service_Order_ID = SO.Service_Order_ID
JOIN
    Item I
ON
    ISAS.Item_ID = I.Item_ID
LEFT JOIN
    Event E
ON
    SO.Service_Order_Date = E.Event_Date
ORDER BY
    Is_Event_Day DESC, SO.Service_Order_Date;
```

[ ]:
```

```

Regression

December 12, 2024

[ ]:
```
# Query full data for each table and display it to the user

# Fetching full data for each table
customer_data = pd.read_sql_query("SELECT * FROM Customer;", conn)
reservation_order_data = pd.read_sql_query("SELECT * FROM Reservation_Order;",␣
 ↪conn)
payment_data = pd.read_sql_query("SELECT * FROM Payment;", conn)
```

[ ]:
```
# Correcting and re-implementing the SQL query for customer statistics
customer_statistics_query = """
WITH Customer_Statistics AS (
    SELECT
        c.Customer_ID,
        AVG(ro.Length * ro.Quantity * rt.Base_Price) AS Avg_Spending,
        COUNT(ro.RO_ID) AS Total_Visits,
        (SELECT rt.Room_Type_ID
         FROM Reservation_Order ro2
         JOIN Room_Type rt ON ro2.RoomType_ID = rt.Room_Type_ID
         WHERE ro2.PayerID = c.Customer_ID
         GROUP BY rt.Room_Type_ID
         ORDER BY COUNT(rt.Room_Type_ID) DESC
         LIMIT 1) AS Preferred_Room_Type,
        (SELECT strftime('%m', gs.Check_In_Date)
         FROM Guest_Stay gs
         WHERE gs.Customer_ID = c.Customer_ID
         GROUP BY strftime('%m', gs.Check_In_Date)
         ORDER BY COUNT(gs.Check_In_Date) DESC
         LIMIT 1) AS Preferred_Booking_Season
    FROM Customer c
    JOIN Reservation_Order ro ON c.Customer_ID = ro.PayerID
    JOIN Room_Type rt ON ro.RoomType_ID = rt.Room_Type_ID
    GROUP BY c.Customer_ID
)
SELECT * FROM Customer_Statistics;
"""

# Executing the query and fetching results into a DataFrame
```

```
customer_statistics_df = pd.read_sql_query(customer_statistics_query, conn)

# Displaying the result to the user
tools.display_dataframe_to_user(name="Customer Statistics for Segmentation",␣
 ↪dataframe=customer_statistics_df)
```

[ ]:
```
# Handling NULL values in Preferred_Booking_Season and Preferred_Room_Type␣
 ↪columns
customer_statistics_df['Preferred_Booking_Season'] =␣
 ↪customer_statistics_df['Preferred_Booking_Season'].fillna(0).astype(int)
customer_statistics_df['Preferred_Room_Type'] =␣
 ↪customer_statistics_df['Preferred_Room_Type'].fillna(0).astype(int)

# Selecting features for clustering again
features = customer_statistics_df[['Avg_Spending', 'Total_Visits',␣
 ↪'Preferred_Room_Type', 'Preferred_Booking_Season']]

# Applying K-means clustering
kmeans = KMeans(n_clusters=4, random_state=42)
customer_statistics_df['Cluster'] = kmeans.fit_predict(features)

# Visualizing the clusters using a scatter plot (Avg_Spending vs Total_Visits)
plt.figure(figsize=(10, 6))
plt.scatter(customer_statistics_df['Avg_Spending'],␣
 ↪customer_statistics_df['Total_Visits'],
            c=customer_statistics_df['Cluster'], cmap='viridis', alpha=0.7)
plt.colorbar(label='Cluster')
plt.xlabel('Average Spending')
plt.ylabel('Total Visits')
plt.title('Customer Segmentation: Avg Spending vs Total Visits')
plt.grid(True)
plt.show()

# Displaying the clustered data to the user
tools.display_dataframe_to_user(name="Customer Segmentation Clusters with␣
 ↪Resolved Issues", dataframe=customer_statistics_df)
```

[ ]:
```
# Identifying insights and summarizing clusters with descriptive statistics

# Grouping data by clusters to understand customer behavior in each cluster
cluster_summary = customer_statistics_df.groupby('Cluster').agg({
    'Avg_Spending': ['mean', 'median', 'max', 'min'],
    'Total_Visits': ['mean', 'median', 'max', 'min'],
    'Preferred_Room_Type': lambda x: x.mode()[0],  # Most common room type
    'Preferred_Booking_Season': lambda x: x.mode()[0],  # Most common booking␣
 ↪season
```

```
}).reset_index()

# Renaming columns for better readability
cluster_summary.columns = [
    'Cluster',
    'Avg_Spending_Mean', 'Avg_Spending_Median', 'Avg_Spending_Max',␣
 ↪'Avg_Spending_Min',
    'Total_Visits_Mean', 'Total_Visits_Median', 'Total_Visits_Max',␣
 ↪'Total_Visits_Min',
    'Preferred_Room_Type', 'Preferred_Booking_Season'
]

# Displaying the summary of customer segments to the user
import ace_tools as tools; tools.display_dataframe_to_user(name="Customer␣
 ↪Segmentation Cluster Summary", dataframe=cluster_summary)
```

[ ]:
```
# Fixing the radar chart preparation process

# Normalizing the data again, ensuring valid column selection
cluster_summary_normalized = cluster_summary.copy()
numerical_columns = [
    'Avg_Spending_Mean', 'Avg_Spending_Median',
    'Total_Visits_Mean', 'Total_Visits_Median',
    'Preferred_Room_Type', 'Preferred_Booking_Season'
]

# Normalize only the numerical columns
cluster_summary_normalized[numerical_columns] =␣
 ↪cluster_summary_normalized[numerical_columns].apply(
    lambda x: (x - x.min()) / (x.max() - x.min())
)

# Repeating the first column for radar plot closure
values_list = cluster_summary_normalized[numerical_columns].values
categories = numerical_columns + [numerical_columns[0]]

# Radar chart setup
fig, ax = plt.subplots(figsize=(10, 8), subplot_kw=dict(polar=True))

for i in range(len(values_list)):
    values = np.append(values_list[i], values_list[i][0])  # Repeat first value
    ax.plot(
        np.linspace(0, 2 * np.pi, len(categories)),
        values,
        label=f"Cluster {cluster_summary_normalized.iloc[i, 0]}"
    )
    ax.fill(
```

```
        np.linspace(0, 2 * np.pi, len(categories)),
        values,
        alpha=0.25
    )

# Customizing the radar chart
ax.set_xticks(np.linspace(0, 2 * np.pi, len(categories)))
ax.set_xticklabels(categories, fontsize=10)
ax.set_title("Customer Segmentation Radar Chart", fontsize=14)
ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))
plt.show()
```

[ ]:
```
# Visualizing customer segments using a scatter plot with Avg_Spending vs␣
↪Total_Visits

plt.figure(figsize=(10, 6))

# Scatter plot of Avg_Spending vs Total_Visits, colored by cluster
scatter = plt.scatter(
    customer_statistics_df['Avg_Spending'],
    customer_statistics_df['Total_Visits'],
    c=customer_statistics_df['Cluster'],
    cmap='viridis',
    alpha=0.7
)

# Adding plot details
plt.colorbar(scatter, label='Cluster')
plt.xlabel('Average Spending')
plt.ylabel('Total Visits')
plt.title('Customer Segmentation: Avg Spending vs Total Visits')
plt.grid(True)
plt.show()
```

Sentiment_Analysis

December 12, 2024

[2]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from textblob import TextBlob
import seaborn as sns
```

[6]:
```
data = pd.read_csv('/Users/mac/Desktop/Data/Sentiment_Analysis.csv')
data.head()
```

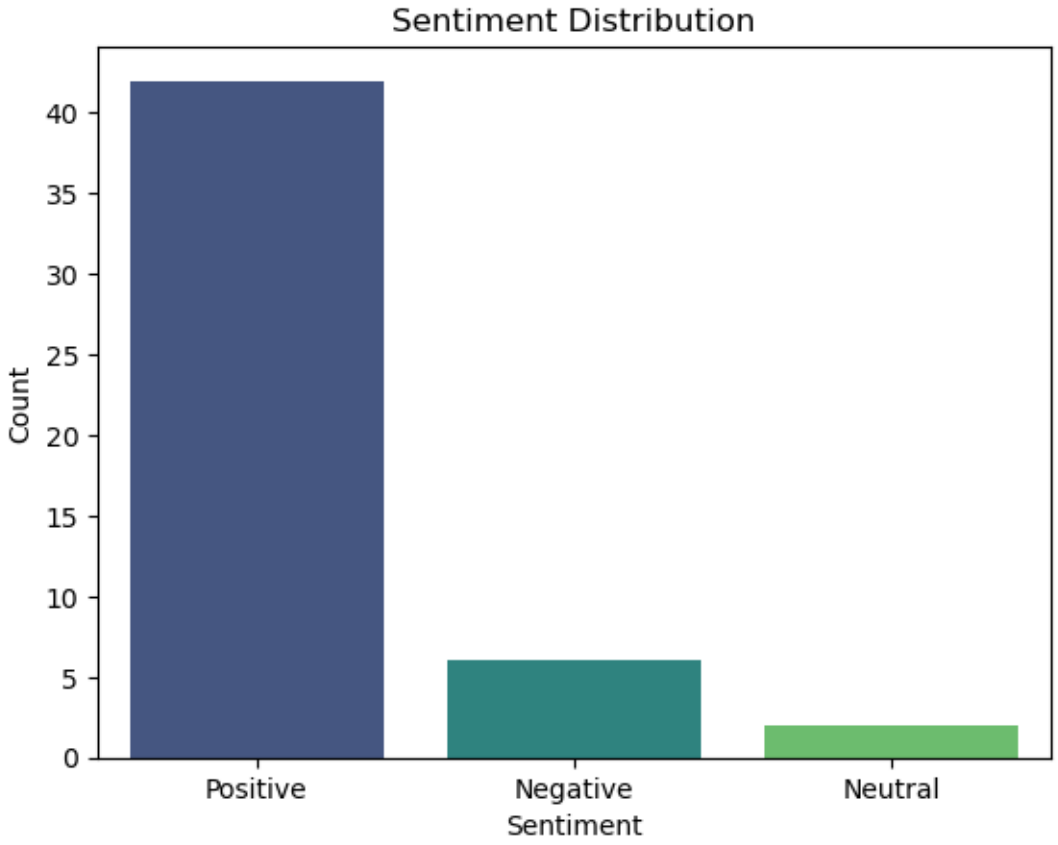
[6]:

	Customer_Name	Feedback_Comment	Feedback_Score	Room_Type	Stay_Length	Rooms_Reserved	Feedback_Time
0	John Doe	Great experience, loved the room and service!	5	1	3	1	2024-12-01 14:30:00
1	Jane Smith	Good stay, but the room was a little small.	4	2	2	1	2024-12-01 15:00:00
2	Emily Johnson	Excellent! Everything was perfect.	5	3	5	2	2024-12-01 16:00:00
3	Michael Brown	Okay stay, but the bathroom was dirty.	3	4	1	3	2024-12-01 16:30:00
4	Sarah Davis	Nice room, but the AC wasn't working properly.	4	5	4	2	2024-12-01 17:00:00

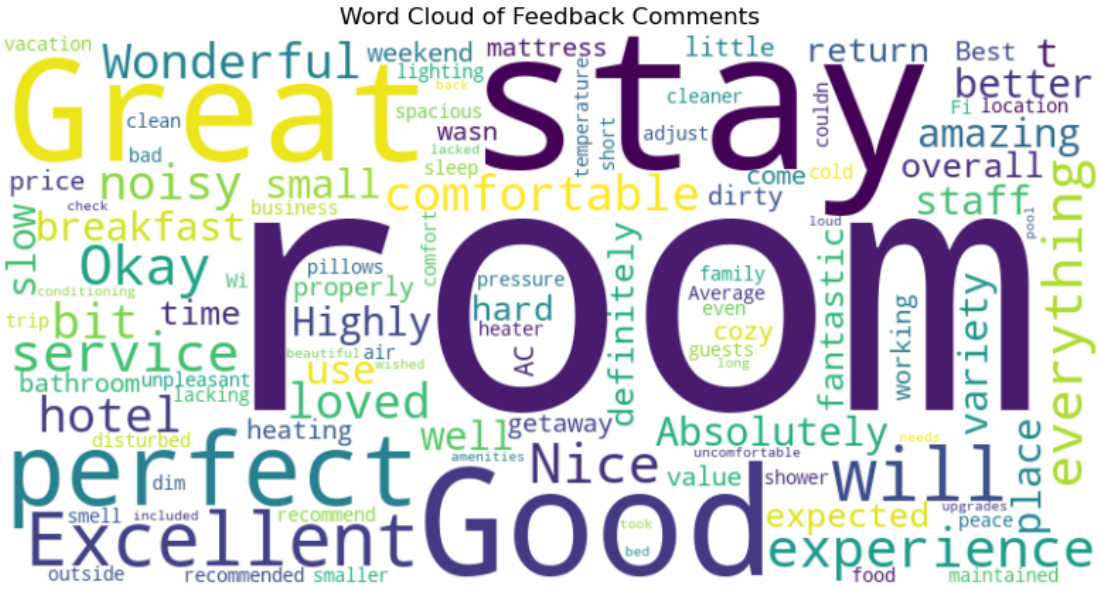
[7]:
```
# Step 1: Sentiment Analysis
def analyze_sentiment(comment):
    analysis = TextBlob(comment)
    if analysis.sentiment.polarity > 0:
        return 'Positive'
    elif analysis.sentiment.polarity == 0:
        return 'Neutral'
    else:
        return 'Negative'

data['Sentiment'] = data['Feedback_Comment'].apply(analyze_sentiment)
```

```
# Visualize sentiment distribution
sns.countplot(x='Sentiment', data=data, palette='viridis')
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```



[ ]:
```
# Step 2: Word Cloud
all_comments = " ".join(data['Feedback_Comment'])
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(all_comments)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Feedback Comments')
plt.show()
```



[9]:
```
# Step 3: Topic Modeling (LDA)
vectorizer = CountVectorizer(stop_words='english')
x_counts = vectorizer.fit_transform(data['Feedback_Comment'])
lda = LatentDirichletAllocation(n_components=3, random_state=42)  # 3 topics
lda.fit(x_counts)

# Display topics
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx + 1}: ")
        print(" ".join([feature_names[i] for i in topic.argsort()[:
 ↪-no_top_words - 1:-1]]))

no_top_words = 10
feature_names = vectorizer.get_feature_names_out()
display_topics(lda, feature_names, no_top_words)
```

```
Topic 1:
perfect experience hotel wonderful loved stay definitely return absolutely
fantastic
Topic 2:
room excellent okay service comfortable stay small highly bit amazing
Topic 3:
stay good room great nice noisy overall breakfast better use
```

[ ]: