OPIM 5272

Phase 2: Monarch Tourism Company

Team 4

(Team Members: Neehar Namjoshi, Vinay Reddy, Brenda Rivadeneyra, Daniel Partida, Hanish Chalicham)

Business Imperative

Monarch Tourism Company is dedicated to providing exceptional service to businesses in the tourism industry. It offers a range of services that help businesses manage their bookings, reservations, and customer experience. The system is designed to be user-friendly and intuitive, making it easy for businesses to get up and running quickly. In addition, we provide real-time travel information and itinerary planning tools that can help tourists make the most of their trip.

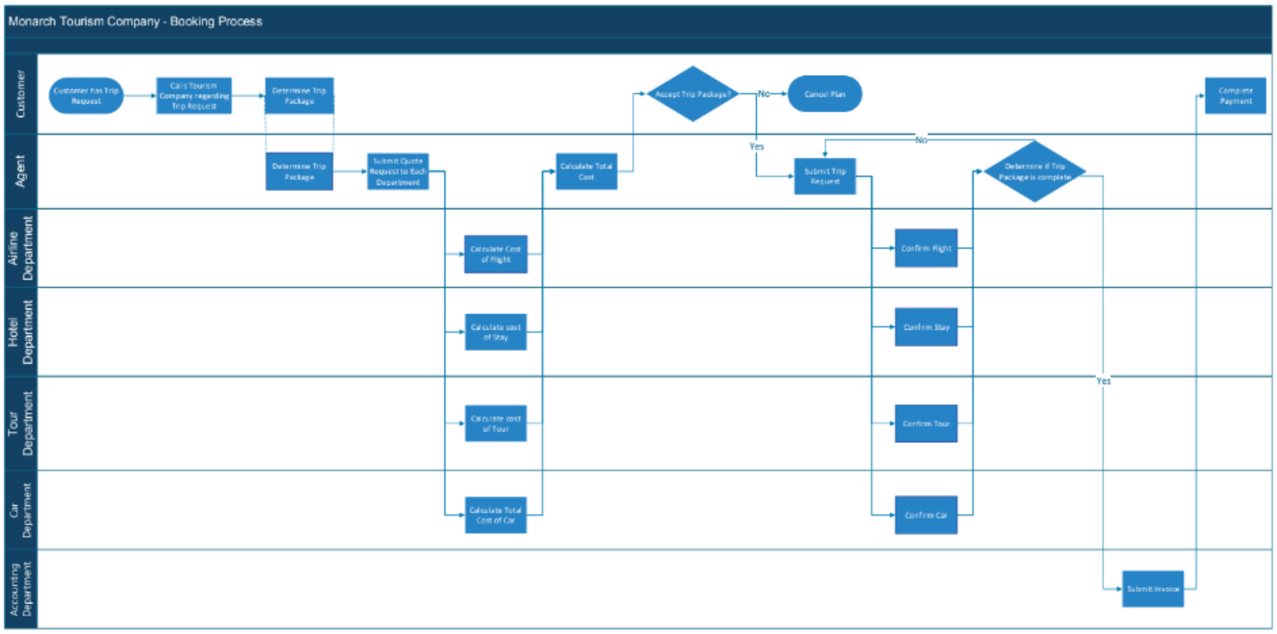
Data security is of paramount importance in today's world, and Monarch Tourism Company recognizes this. We have implemented robust security measures to protect sensitive information and ensure the privacy of our customers. We also offer customization options that allow businesses to tailor their services to the customer’s unique needs and preferences. We integrate with other systems and platforms to provide a seamless experience for customers and offer excellent customer support to help businesses and tourists resolve any issues or problems they may encounter.

The company's system streamlines and automates many processes, such as bookings, reservations, and payment processing, reducing the risk of errors and saving time for both businesses and tourists. The database design they offer allows users to easily search for destinations, activities, accommodations, and transportation options, view reviews from other customers, and make bookings for their chosen options. It is designed to be simple for application administrators to update and administer the database.

Monarch Tourism Company's goal is to make the tourism industry as seamless and stress-free as possible. We achieve this by offering a range of user-friendly, customizable services, and integrated with other systems and platforms. They prioritize data security and offer excellent customer support to ensure businesses and tourists have a positive experience.

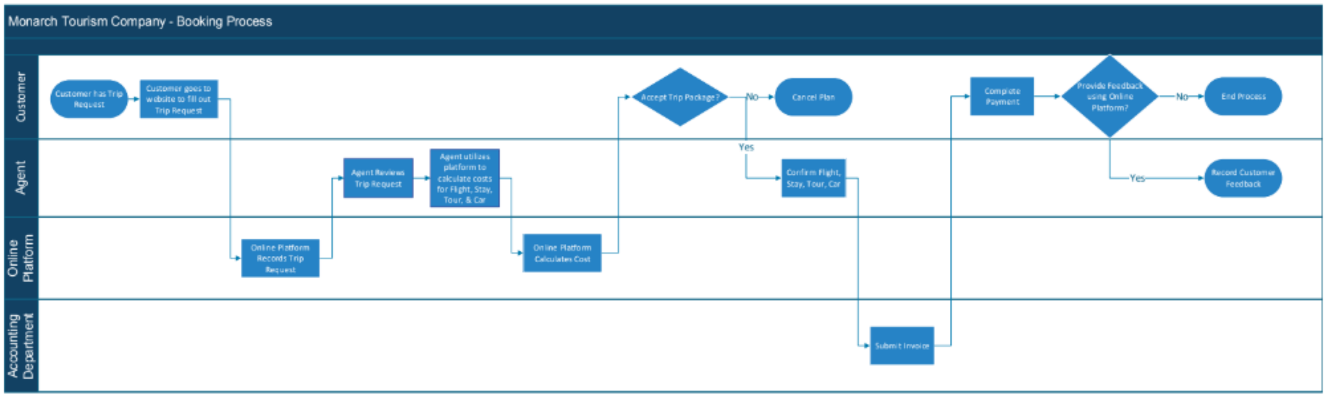
TO-BE Swimlane

Firstly, here is the corrected Phase 1 AS-IS Swimlane:



The original AS-IS Swimlane had too many diamond shape decisions, even when it was unneeded. Therefore, for the corrected AS-IS, the diamond shape decision blocks were changed to rectangular action blocks.

Below is the TO-BE Swimlane:


Monarch Tourism Company - Booking Process
Customer
Agent
Online Platform
Accounting Department
Customer has Trip Request
Customer goes to website to fill out Trip Request
Online Platform Records Trip Request
Agent Reviews Trip Request
Agent utilizes platform to calculate costs for Flight, Stay, Tour, & Car
Online Platform Calculates Cost
Accept Trip Package?
No
Cancel Plan
Yes
Confirm Flight, Stay, Tour, Car
Submit Invoice
Complete Payment
Provide Feedback using Online Platform?
No
End Process
Yes
Record Customer Feedback

The TO-BE Swimlane has the following improvements as compared to the AS-IS Swimlane.

Manual Tasks

- Customer no longer has to travel to the business location

- Customer uses the online platform to complete trip request

-Agent uses the online platform instead of the 3 departments here to calculate the cost of the trip package

Project Tracking

- The Trip Request is now tracked using the online platform

Agent Security

- The Online platform eliminates need for paper records

Customer Experience

- Customer can pick their own communications right from the start

Customer Access

- Customer can use the only platform

Customer Reviews

- Customer has option to provide feedback after completing the payment

Table creation code (DDL SQL script)

```
CREATE TABLE Customer (
customer_id int(10) Primary Key,
first_name varchar(64),
last_name varchar(64),
email_id varchar(255),
address varchar(255),
contact_number varchar(32));
```

The above code creates a table named "Customer". The table has six columns: "customer_id", "first_name", "last_name", "email_id", "address", and "contact_number". The "customer_id" column is an integer data type and is designated as the Primary Key for the table. This means that each row in the table will have a unique value for this column and it will be used to identify each customer record. The "first_name" and "last_name" columns are both varchar data types and these columns will store the first and last names of the customer. The "email_id" column is also a varchar data type and will store the email address of the customer. The "address" column is also a varchar data type and will store the address of the customer. The "contact_number" column is a varchar data type and will store the phone number of the customer.

```
CREATE TABLE Destination (
dest_id int(10) PRIMARY KEY,
dest_name varchar(255),
dest_description varchar(255),
dest_location varchar(255));
```

The above code creates a table named "Destination". The table has four columns: "dest_id", "dest_name", "dest_description", and "dest_location". The "dest_id" column is an integer data type with a maximum length of 10 digits and is designated as the Primary Key for the table. This means that each row in the table will have a unique value for this column and it will be used to identify each destination record. The "dest_name" column is a varchar data type with a maximum length of 255 characters and will store the name of the destination. The "dest_description" column is also a varchar data type with a maximum length of 255 characters and will store a brief description of the destination. The "dest_location" column is a varchar data type with a maximum length of 255 characters and will store the location of the destination.

```
CREATE TABLE Activity (

activity_id int(10) PRIMARY KEY,

dest_id int(10),

activity_name varchar(64),

activity_description varchar(255),

activity_duration int(10),

activity_price decimal(10,2));
```

The above code creates a table named "Activity". The table has six columns: "activity_id", "dest_id", "activity_name", "activity_description", "activity_duration", and "activity_price". The "activity_id" column is an integer data type with a maximum length of 10 digits and is designated as the Primary Key for the table. This means that each row in the table will have a unique value for this column and it will be used to identify each activity record. The "dest_id" column is also an integer data type with a maximum length of 10 digits and will store the ID of the destination that the activity is associated with. This column can be used to link activity records with their corresponding destination records in the "Destination" table. The "activity_name" column is a varchar data type with a maximum length of 64 characters and will store the name of the activity. The "activity_description" column is a varchar data type with a maximum length of 255 characters and will store a brief description of the activity. The "activity_duration" column is an integer data type with a maximum length of 10 digits and will store the duration of the activity in minutes, hours, or days. The "activity_price" column is a decimal data type with a maximum length of 10 digits and a scale of 2 and will store the price of the activity as a decimal value with two decimal places.

```
ALTER TABLE Activity

ADD CONSTRAINT fk FOREIGN KEY (dest_id) REFERENCES Destination(dest_id);
```

The above code adds a foreign key constraint to the "Activity" table. The "ALTER TABLE" keyword specifies that an existing table will be modified. The "ADD CONSTRAINT" keyword indicates that a new constraint will be added to the table. The "fk" is an identifier for the foreign key constraint. The "FOREIGN KEY" keyword specifies that the "dest_id" column in the "Activity" table is a foreign key column that will reference the "dest_id" column in the "Destination" table. The "REFERENCES" keyword indicates that the referenced table is the "Destination" table and the referenced column is the "dest_id" column.

```
CREATE TABLE Hotel_Accommodation (

accommodation_id int(10) PRIMARY KEY,

accommodation_name varchar(255),

accommodation_description varchar(255),

accommodation_price decimal(10,2),

accommodation_location varchar(255),

accommodation_booking_date date default sysdate());
```

The above code creates a new table called "Hotel_Accommodation" with five columns. "accommodation_id" is an integer data type with a maximum length of 10, which will be the primary key for the table. "accommodation_name" is a string data type with a maximum length of 255, which will store the name of the hotel accommodation. "accommodation_description" is a string data type with a maximum length of 255, which will store the description of the hotel accommodation. "accommodation_price" is a decimal data type with a precision of 10 and a scale of 2, which will store the price of the hotel accommodation. "accommodation_location" is a string data type with a maximum length of 255, which will store the location of the hotel accommodation. Additionally, the "accommodation_booking_date" column is defined with a default value of "sysdate()", which means that the current date will be automatically inserted into this column when a new record is inserted into the table.

```
CREATE TABLE Transportation (

transportation_id int(10) PRIMARY KEY,

transportation_type varchar(64),

departure_time time,
arrival_time time,
from_date date,
price decimal (10,2),
end_date date);
```

The above code creates a new table called "Transportation"and has seven column. "transportation_id" is an integer data type with a maximum length of 10, which will be the primary key for the table. "transportation_type" is a string data type with a maximum length of 64, which will store the type of transportation, such as "bus", "train", or "flight". "departure_time" is a time data type, which will store the time of departure for the transportation. "arrival_time" is a time data type, which will store the time of arrival for the transportation. "from_date" is a date data type, which will store the date when the transportation starts. "price" is a decimal data type with a precision of 10 and a scale of 2, which will store the price of the transportation. "end_date" is a date data type, which will store the date when the transportation ends.

```
CREATE TABLE Review (
review_id int(10) PRIMARY KEY,
customer_id int(10),
review_rating float(10),
review_comments varchar(255));
```

The above code creates a table named "Review" with four columns: "review_id", "customer_id", "review_rating", and "review_comments". The "review_id" column is the primary key of the table and has a data type of int(10). The "customer_id" column has a data type of int(10) and is used to link the review to a specific customer in the "Customer" table. The "review_rating" column has a data type of float(10) and stores the rating given by the customer for the destination or activity. The "review_comments" column has a data type of varchar(255) and allows the customer to provide comments about their experience. This table is likely intended to collect and store feedback from customers about various destinations and activities.

```
ALTER TABLE Review
```

ADD CONSTRAINT fk_2 FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);

The above code adds a foreign key constraint named "fk_2" to the "Review" table's "customer_id" column, referencing the "customer_id" column in the "Customer" table. This constraint ensures that each value in the "customer_id" column in the "Review" table exists as a primary key value in the "Customer" table. The foreign key constraint helps maintain referential integrity between the two tables, ensuring that data is consistent and accurate across both tables.

CREATE TABLE Car_Booking (

car_booking_id int(10) PRIMARY KEY,

transportation_id int(10),

cab_location varchar(64),

cab_type varchar(64));

The above code creates a table named "Car_Booking" with four columns: "car_booking_id", "transportation_id", "cab_location", and "cab_type". The "car_booking_id" column is the primary key of the table, which uniquely identifies each record in the table. The "transportation_id" column is a foreign key that references the "transportation_id" column in the "Transportation" table. It is used to associate each car booking with a transportation option, such as a rental car or a taxi. The "cab_location" column is a varchar data type that stores the location of the car booking, such as the pickup location or the destination. The "cab_type" column is a varchar data type that stores the type of cab or car booked, such as a sedan, SUV, or luxury car.

ALTER TABLE Car_Booking

ADD CONSTRAINT fk_3 FOREIGN KEY (transportation_id) REFERENCES Transportation(transportation_id);

The above code adds a foreign key constraint named "fk_3" to the "Car_Booking" table. The constraint links the "transportation_id" column of the "Car_Booking" table to the "transportation_id" column of the "Transportation" table, enforcing referential integrity between the two tables. Specifically, it ensures that any value entered into the "transportation_id" column of the "Car_Booking" table must already exist in the "transportation_id" column of the "Transportation" table, or else the insertion/update will fail.

CREATE TABLE Flight_Booking (

flight_booking_id int(10) PRIMARY KEY,

```
transportation_id int(10),

flight_number varchar(64),

flight_from varchar(64),

flight_to varchar(64),

airline_name varchar(64),

number_of_tickets int(15));
```

The above code creates a table named Flight_Booking with columns including flight_booking_id, transportation_id, flight_number, flight_from, flight_to, airline_name, and number_of_tickets. The flight_booking_id column is the primary key and will contain unique values for each flight booking. The transportation_id column is a foreign key that references the transportation_id column of the Transportation table. The flight_number, flight_from, flight_to, and airline_name columns store information about the flight, including the flight number, the departure location, the destination location, and the airline name, respectively. The number_of_tickets column stores the number of tickets booked for the flight.

```
ALTER TABLE Flight_Booking

ADD CONSTRAINT fk_4 FOREIGN KEY (transportation_id) REFERENCES Transportation(transportation_id);
```

The above code adds a foreign key constraint to the "transportation_id" column in the "Flight_Booking" table, referencing the "transportation_id" column in the "Transportation" table. This ensures that the "transportation_id" value entered in the "Flight_Booking" table must already exist in the "Transportation" table, enforcing referential integrity between the two tables.

```
CREATE TABLE Package (

package_id int(10) PRIMARY KEY,

dest_id int(10),

package_name varchar(255),

number_of_people int(10),

package_type varchar(64),

number_of_days int(10),
package_cost decimal (10,2));
```

The above code creates a table named "Package" with columns "package_id", "dest_id", "package_name", "number_of_people", "package_type", "number_of_days", and "package_cost". "package_id" is the primary key for this table and has data type int(10). "dest_id" is a foreign key referencing the "dest_id" column in the "Destination" table, which specifies the destination of the package. It has data type int(10). "package_name" is a varchar(255) column that specifies the name of the package. "number_of_people" is an integer column that specifies the number of people the package is intended for. "package_type" is a varchar(64) column that specifies the type of package. "number_of_days" is an integer column that specifies the number of days the package will last. "package_cost" is a decimal column that specifies the cost of the package.

```
ALTER TABLE Package

ADD CONSTRAINT fk_5 FOREIGN KEY (dest_id) REFERENCES Destination(dest_id);
```

The above code is adding a foreign key constraint named "fk_5" to the existing "Package" table. The constraint states that the "dest_id" column in the "Package" table references the "dest_id" column in the "Destination" table, ensuring that only valid destination IDs can be entered into the "dest_id" column in the "Package" table.

```
CREATE TABLE Booking (

booking_id int(10) PRIMARY KEY,

package_id int(10),

customer_id int(10),

accommodation_id int(10),

transportation_id int(10),

booking_date date);
```

The above code creates a table named "Booking" with five columns. "booking_id" is an integer column with a maximum value of 10 digits, which is the primary key for this table. "package_id" is an integer column with a maximum value of 10 digits, which represents the foreign key of the Package table. "customer_id" is an integer column with a maximum value of 10 digits, which represents the foreign key of the Customer table. "accommodation_id" is an integer column with a maximum value of 10 digits, which represents the foreign key of the Hotel_Accommodation table. "transportation_id" is an integer column with a maximum value of 10 digits, which represents the foreign key of the Transportation table. "booking_date" is a date column representing the date of the booking.

ALTER TABLE Booking

ADD CONSTRAINT fk_6 FOREIGN KEY (package_id) REFERENCES Package(package_id);

ALTER TABLE Booking

ADD CONSTRAINT fk_7 FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);

ALTER TABLE Booking

ADD CONSTRAINT fk_8 FOREIGN KEY (accommodation_id) REFERENCES Hotel_Accommodation(accommodation_id);

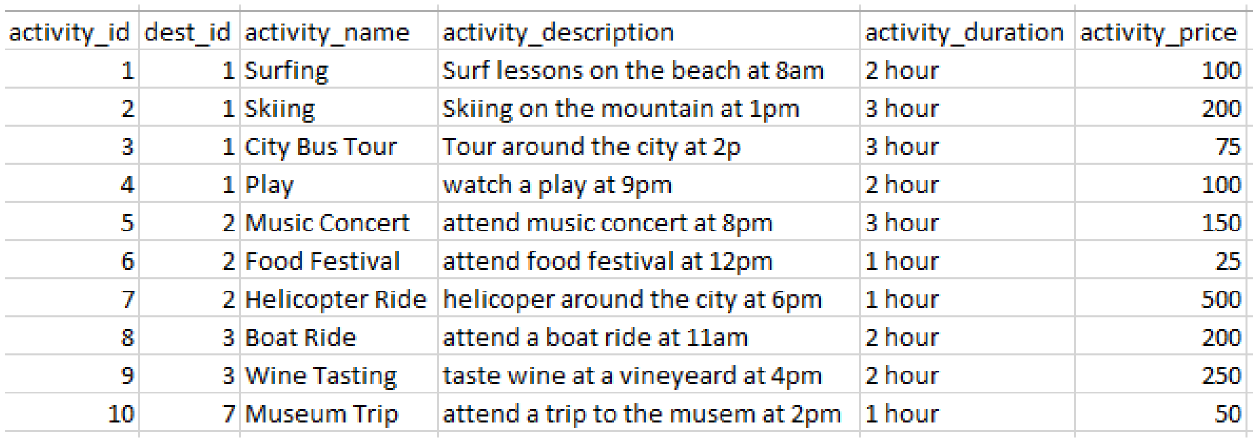
ALTER TABLE Booking

ADD CONSTRAINT fk_9 FOREIGN KEY (transportation_id) REFERENCES Transportation(transportation_id);

The above SQL commands add foreign key constraints to the Booking table. The first command adds a foreign key constraint named fk_6 that references the package_id column in the Package table. The second command adds a foreign key constraint named fk_7 that references the customer_id column in the Customer table. The third command adds a foreign key constraint named fk_8 that references the accommodation_id column in the Hotel_Accommodation table. The fourth command adds a foreign key constraint named fk_9 that references the transportation_id column in the Transportation table.

Data Entry

Example CSV File for Activity Table:



activity_id	dest_id	activity_name	activity_description	activity_duration	activity_price
1	1	Surfing	Surf lessons on the beach at 8am	2 hour	100
2	1	Skiing	Skiing on the mountain at 1pm	3 hour	200
3	1	City Bus Tour	Tour around the city at 2p	3 hour	75
4	1	Play	watch a play at 9pm	2 hour	100
5	2	Music Concert	attend music concert at 8pm	3 hour	150
6	2	Food Festival	attend food festival at 12pm	1 hour	25
7	2	Helicopter Ride	helicoper around the city at 6pm	1 hour	500
8	3	Boat Ride	attend a boat ride at 11am	2 hour	200
9	3	Wine Tasting	taste wine at a vineyeard at 4pm	2 hour	250
10	7	Museum Trip	attend a trip to the musem at 2pm	1 hour	50

In total, the Monarch Tourism Company has 11 tables in their database. Above is an example CSV file for data entry into the Activity table. Similarly, 10 similar CSV files were used to upload 10 records into each table.

<u>SQL Queries</u>

1) **Which customers have booked packages with a package cost equal to or greater than $5000, and what are their customer IDs, first names, and package costs?**

Query :

```
SELECT c.customer_id,

c.first_name,p.package_cost

FROM Customer c

JOIN Booking b on c.customer_id  =b.customer_id  # joining customer table with booking table where the customer id matches in both tables

 JOIN Package p on p.package_id =b.package_id  #joining package table with booking table on package id

WHERE p.package_cost >=5000; # filtering where package cost is greater than or equal to 5000
```

Query 1 Explanation:

Purpose: The purpose of this query is to retrieve information about customers, their first names, and the cost of packages they have booked, from a database. The query filters the results based on the condition that the cost of the package must be equal to or greater than $5000.

Benefit: The query helps to identify customers who have booked expensive packages with a cost of $5000 or more. This information can be useful for various purposes, such as identifying high-value customers, analyzing revenue generated from premium packages, and targeting marketing or promotional campaigns towards customers who are willing to spend more on packages.

Use in Business Metrics: The query can provide valuable insights and contribute to several business metrics, including:

Revenue: By identifying customers who have booked expensive packages, the query can help track the revenue generated from premium packages, which can be an important metric to measure the financial performance of the business.

Customer Segmentation: The query can be used to segment customers based on their spending behavior, specifically those who are willing to spend more on packages. This information can help businesses tailor their marketing strategies to target different customer segments with specific offers and promotions.

Customer Lifetime Value (CLTV): By identifying high-value customers who book expensive packages, the query can contribute to calculating the CLTV, which is a metric used to estimate the potential value a customer can bring to the business over their entire relationship with the company.

Package Performance Analysis: The query can be used to analyze the performance of different packages based on their cost. Businesses can evaluate the popularity and profitability of various packages to make data-driven decisions on pricing, promotion, and inventory management.

In summary, the query provides insights into customer behavior and package performance, which can be used to make informed decisions and optimize business strategies.

2) **What are the details of customers, bookings, packages, and destinations for bookings that were made for flights to New York City, including customer information, booking details, package information, and destination information?**

Query:

```
SELECT *

FROM Customer c
```

JOIN Booking b on c.customer_id = b.customer_id

JOIN Package p on p.package_id = b.package_id

JOIN Destination d on d.dest_id =p.dest_id

WHERE b.transportation_id in (

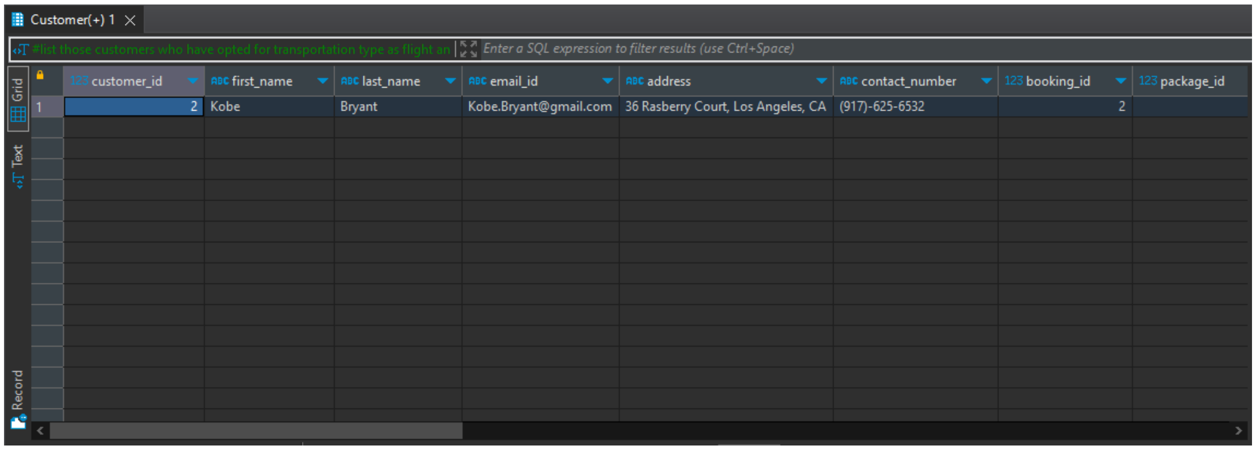
SELECT t.transportation_id

FROM Transportation t

WHERE t.transportation_type like '%Flight%') *# filtering where transportation type is flights*

AND d.dest_name like "New York City"; *#filtering where dest_name is "New York City"*

Output


Customer(+) 1

#list those customers who have opted for transportation type as flight an

	customer_id	first_name	last_name	email_id	address	contact_number	booking_id	package_id
1	2	Kobe	Bryant	Kobe.Bryant@gmail.com	36 Rasberry Court, Los Angeles, CA	(917)-625-6532	2	

Query 2 Explanation:

Purpose: The purpose of this query is to retrieve information from a database related to customers, bookings, packages, destinations, and transportation. The query specifically focuses on customers who have booked packages for New York City and have chosen transportation options that include "Flight" in their transportation type.

Benefit: The query can provide insights into customers who have booked packages for New York City and have chosen flights as their transportation option. This information can be valuable for various purposes, including:

Transportation Analysis: The query allows businesses to analyze the performance of transportation options, specifically flights, for packages to New York City. This can help businesses evaluate the popularity and effectiveness of flights as a transportation choice for this destination and make data-driven decisions on transportation offerings and partnerships.

Customer Behavior Analysis: The query provides information on customers who have booked packages for New York City, which can help analyze customer behavior and preferences. This information can be used to understand customer preferences for transportation options and destinations, tailor marketing strategies, and personalize offers and promotions to target customers who prefer flights as their transportation choice.

Destination Performance Analysis: The query can contribute to analyzing the performance of New York City as a destination in terms of customer bookings. Businesses can evaluate the demand for packages to New York City and the effectiveness of marketing efforts for this destination, and use the insights to optimize destination-specific strategies and offerings.

Booking and Revenue Analysis: The query provides data on bookings that meet the specified criteria, which can be used to analyze booking trends, revenue generated from packages to New York City with flights, and other booking-related metrics. This information can help businesses track the performance of bookings associated with flights and New York City, and make data-driven decisions on pricing, inventory management, and revenue optimization.

In summary, the query provides insights into customer behavior, destination performance, and transportation analysis, which can be used to make informed decisions and optimize business strategies related to bookings, revenue, and customer satisfaction.

3) **What is the total sum of package costs for packages that were booked with booking dates falling within the month of January 2023?**

   Query:

SELECT sum(p.package_cost) *#aggregate function to sum up the package cost*

FROM Package p

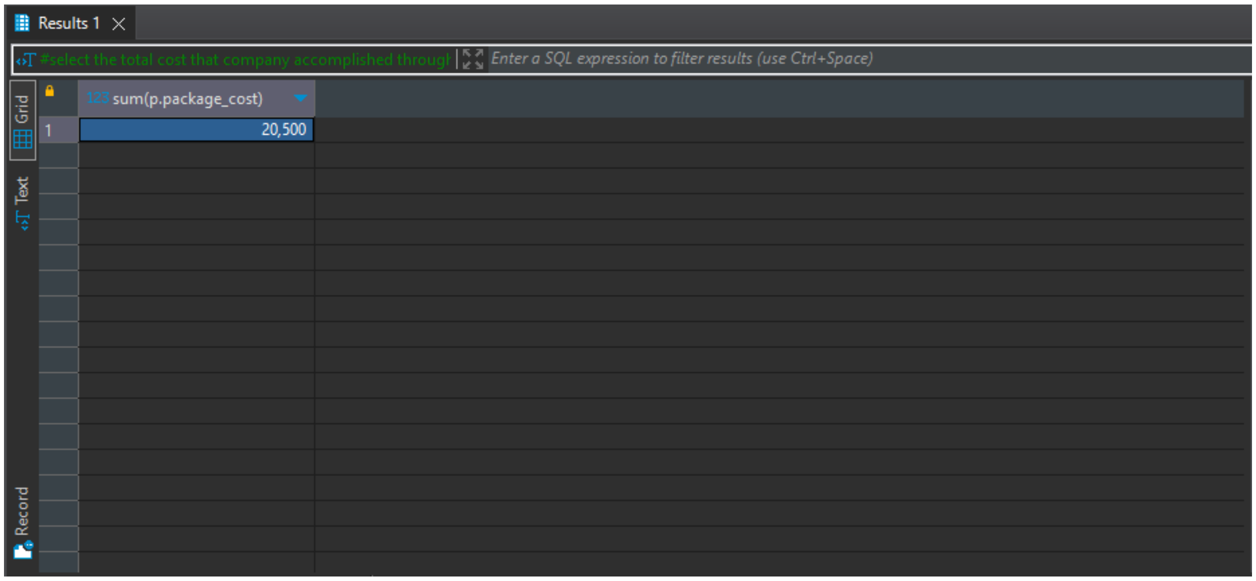
WHERE p.package_id in (

SELECT b.package_id

FROM Booking b

WHERE booking_date like '2023-01-%'); *# filtering the booking date in the year 2023 and month january and % implies any date in january*.

Output



Results 1

#select the total cost that company accomplished througt

Enter a SQL expression to filter results (use Ctrl+Space)

```
   sum(p.package_cost)
1               20,500
```

Query 3 Explanation:

Purpose: The purpose of this query is to retrieve the total cost of packages that were booked in the month of January 2023 from a database. The query uses a subquery to first retrieve the package IDs of bookings that were made in January 2023, and then calculates the sum of package costs associated with those package IDs.

Benefit: The query can provide valuable insights into the total revenue generated from bookings made in the month of January 2023. This information can be beneficial for various purposes, including:

Revenue Analysis: The query helps businesses to analyze the revenue generated from bookings made in a specific time period, in this case, the month of January 2023. This can help businesses evaluate the performance of bookings during a specific time period and make data-driven decisions on pricing, promotions, and revenue optimization.

Booking Analysis: The query allows businesses to analyze the performance of bookings made in a specific month, which can help in identifying booking trends, patterns, and customer preferences.

This information can be used to optimize booking strategies, identify potential issues, and plan for future bookings.

Performance Evaluation: The query can contribute to evaluating the performance of packages offered by the business during a specific time period. By calculating the total cost of packages booked in January 2023, businesses can assess the popularity and profitability of their packages and make informed decisions on package offerings, pricing, and inventory management.

Budgeting and Forecasting: The query provides information on the total cost of packages booked in January 2023, which can be used for budgeting and forecasting purposes. Businesses can use this information to estimate revenues, plan expenses, and make financial projections for future months or quarters.

In summary, the query provides insights into revenue, booking trends, package performance, and financial planning, which can be used to make informed decisions and optimize business strategies related to bookings, revenue, and performance evaluation.

4) **What are the details of bookings made by customers with customer IDs from 1to 10 including the customer's first name, booking ID, package name, destination name, booking date, package cost, and package category based on the package cost thresholds (Silver, Gold, Platinum)?**

Query 4:

```
SELECT

c.first_name ,

b.booking_id,

p.package_name,

 d.dest_name,

b.booking_date,

p.package_cost ,

CASE

WHEN p.package_cost < 2000  THEN 'Silver' # using case to categorize the package cost
```

WHEN p.package_cost < 4000 THEN 'Gold'

WHEN p.package_cost <= 6000 THEN 'Platinum'

END AS package_category

FROM Customer c

JOIN Booking b ON c.customer_id = b.customer_id

JOIN Package p ON b.package_id = p.package_id

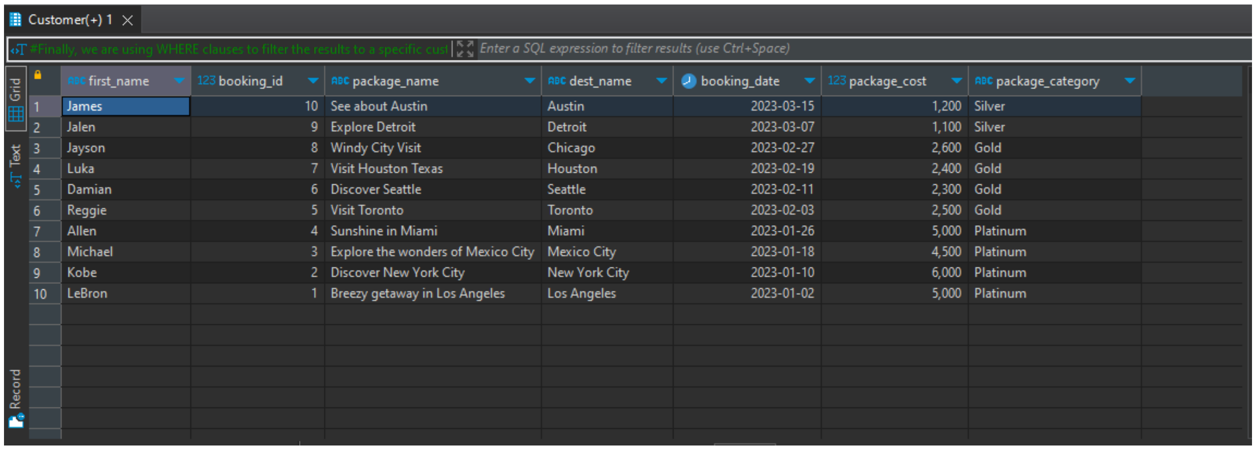
JOIN Destination d ON p.dest_id = d.dest_id

WHERE c.customer_id in (1,2,3,4,5,6,7,8,9,10) *#filtering all the customers from 1 to 10*

ORDER BY b.booking_date DESC; *# ordering by booking date in descending order*

Query Additional Comments: In this query, we are joining multiple tables together using their foreign key relationships. We are selecting fields from the customer, booking, package, and destination tables, including the customer name, booking ID, package name, destination name, booking date, and booking state. We are also using a CASE statement to determine the booking price based on the booking state. If the booking state is 'confirmed', we use the confirmed price from the booking table. If the booking state is 'pending', we calculate the booking price as the base price of the package multiplied by the number of travelers in the booking. If the booking state is 'cancelled', we set the booking price to 0. Finally, we are using WHERE clauses to filter the results to a specific customer and bookings on or after a certain date. We are also sorting the results by booking date in descending order

Output:



Customer(+) 1

#Finally, we are using WHERE clauses to filter the results to a specific cust | Enter a SQL expression to filter results (use Ctrl+Space)

	first_name	booking_id	package_name	dest_name	booking_date	package_cost	package_category
1	James	10	See about Austin	Austin	2023-03-15	1,200	Silver
2	Jalen	9	Explore Detroit	Detroit	2023-03-07	1,100	Silver
3	Jayson	8	Windy City Visit	Chicago	2023-02-27	2,600	Gold
4	Luka	7	Visit Houston Texas	Houston	2023-02-19	2,400	Gold
5	Damian	6	Discover Seattle	Seattle	2023-02-11	2,300	Gold
6	Reggie	5	Visit Toronto	Toronto	2023-02-03	2,500	Gold
7	Allen	4	Sunshine in Miami	Miami	2023-01-26	5,000	Platinum
8	Michael	3	Explore the wonders of Mexico City	Mexico City	2023-01-18	4,500	Platinum
9	Kobe	2	Discover New York City	New York City	2023-01-10	6,000	Platinum
10	LeBron	1	Breezy getaway in Los Angeles	Los Angeles	2023-01-02	5,000	Platinum

Query 4 Explanation:

Purpose: The purpose of this query is to retrieve information related to customer bookings, package details, and destination information from a database. The query specifically focuses on customers with customer IDs 1 to 10 and retrieves data such as customer first name, booking ID, package name, destination name, booking date, package cost, and package category based on package cost ranges.

Benefit: The query can provide valuable insights into customer bookings, package details, and destination information, and can be beneficial for various purposes, including:

Customer Analysis: The query allows businesses to analyze the booking data of specific customers (with customer IDs 1 to 10) and retrieve their first names, booking dates, package details, and destination information. This can help businesses understand customer preferences, booking patterns, and customer-specific trends, and tailor marketing strategies, promotions, and offers accordingly.

Package Analysis: The query provides information on package details such as package name, package cost, and package category based on the package cost ranges defined in the CASE statement. This can help businesses analyze the performance of different packages based on their categories (Silver, Gold, Platinum) and make data-driven decisions on pricing, package offerings, and inventory management.

Destination Analysis: The query retrieves destination information such as destination name from the Destination table, which can help businesses analyze the popularity and demand for different destinations among the selected customers (with customer IDs 1 to 10). This information can be used to optimize destination-specific strategies, promotions, and offerings.

Booking Analysis: The query provides booking details such as booking ID and booking date, and orders the results by booking date in descending order. This can help businesses analyze booking trends, patterns, and recent bookings among the selected customers, and identify potential issues or opportunities for improvement.

Performance Evaluation: The query calculates the package category based on the package cost ranges defined in the CASE statement, which can help businesses evaluate the performance of packages in terms of their cost categories (Silver, Gold, Platinum). This information can be used to assess the effectiveness of pricing strategies, package offerings, and revenue generation from different package categories.

In summary, the query provides insights into customer behavior, package performance, destination popularity, and booking trends, which can be used to make informed decisions and optimize business strategies related to customer bookings, revenue, and performance evaluation.

5) **Which all customers have booked packages with costs higher than the average package cost across all packages in the Package table, and what are their customer IDs and first names?"**

Query 5:

SELECT c.customer_id,

c.first_name,

c.first_name

FROM Package p2

JOIN Booking b on p2.package_id =b.package_id

JOIN Customer c on c.customer_id =b.booking_id

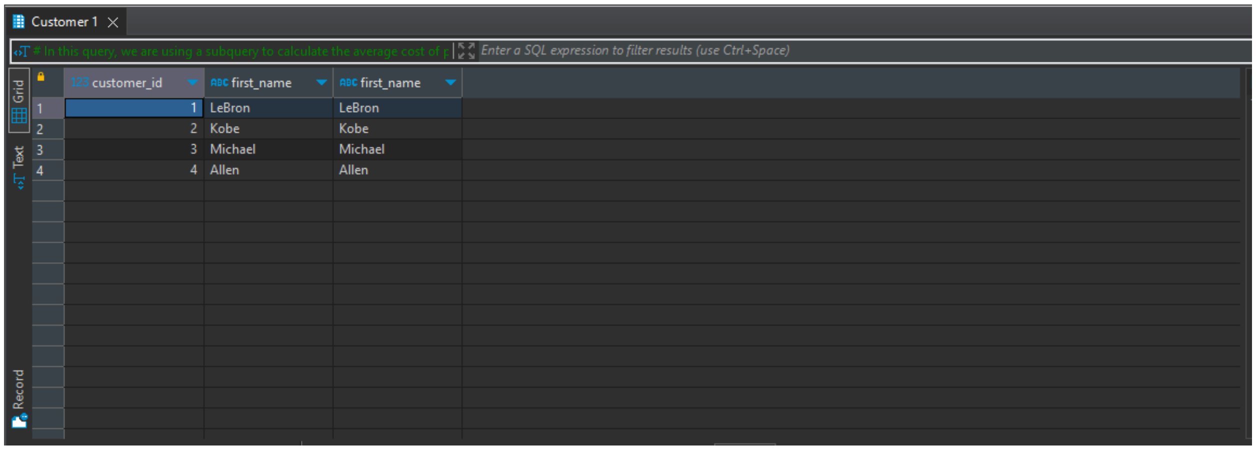
WHERE package_cost >= ANY (

SELECT AVG(package_cost) *#using avg aggregate function to calculate the avg package cost*

FROM Package p);

Query Additional Comments: This SQL query retrieves the customer ID, first name, and first name (duplicated) from the Customer table, where the package cost in the Package table is greater than or equal to any value in the average package cost calculated from the Package table. The query performs JOIN operations to connect the Package, Booking, and Customer tables based on their respective foreign keys.

Output



In this query, we are using a subquery to calculate the average cost of p

Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	first_name	first_name
1	1	LeBron	LeBron
2	2	Kobe	Kobe
3	3	Michael	Michael
4	4	Allen	Allen

Query 5 Explanation:

Purpose: The purpose of this query is to retrieve customer IDs and first names from the Customer table for customers who have booked packages with a cost greater than or equal to the average package cost across all packages in the Package table.

Benefit: The query can provide valuable insights into customers who have booked packages with higher costs compared to the average package cost, and can be beneficial for various purposes, including:

High-Value Customer Analysis: The query identifies customers who have booked packages with costs higher than the average package cost, indicating that they may be high-value customers for the business. By retrieving customer IDs and first names of these customers, businesses can prioritize their engagement and retention efforts towards these high-value customers, offering personalized services, promotions, and loyalty programs to enhance customer satisfaction and loyalty.

Pricing Strategy Analysis: The query compares the package cost of each booking with the average package cost across all packages in the Package table. This can provide insights into how the business's pricing strategy is performing in relation to the average market pricing. If a significant number of bookings have package costs higher than the average, it may indicate that the business's pricing strategy is effective in generating higher revenue or attracting premium customers. On the other hand, if the bookings have package costs lower than the average, it may indicate that the business needs to reevaluate its pricing strategy to remain competitive in the market.

Benchmarking Analysis: The query calculates the average package cost across all packages in the Package table and uses it as a benchmark to compare individual bookings. This can help businesses assess the performance of individual bookings in terms of their pricing compared to the market average. By identifying bookings with package costs significantly higher or lower than the average, businesses can take corrective actions to optimize their pricing and revenue generation strategies.

In summary, the query provides insights into customers who have booked packages with costs higher than the average package cost, which can be used for high-value customer analysis, pricing strategy analysis, and benchmarking analysis. This information can help businesses make data-driven decisions to optimize pricing strategies, customer engagement efforts, and revenue generation.