**CAPSTONE PROJECT REPORT**

**BATCH-05**

**TRIP PLANNER APPLICATION**

**YOKHALAKSHMI K**

**INTERN-DIGITAL DEPARTMENT**

**INDIUM SOFTWARE**

**ABSTRACT**

The application's database contains information on customers, country, Places, and Trip data. The database is designed using SQL to ensure efficient storage and retrieval of data. The application uses SQL queries to retrieve data based on user input and preferences, providing a personalized travel experience.The Trip Planner application allows users to create custom journey by selecting their travel dates, destinations, and activities. The application uses SQL queries to retrieve data on available customers, country, Places, and Trip data allowing users to make informed decisions about their travel plans.The application's booking system is integrated with the database, allowing users to book transportation and activities directly from the application. The database stores information on user bookings, allowing users to manage their travel plans effectively.This application uses SQL to generate reports on user data, including travel history, preferences, and booking patterns. The reports provide valuable insights into user behavior, allowing the application to make personalized recommendations and improve the overall user experience.

**LITERATURE**

**Trip Advisor:** Trip Advisor is a popular travel planning platform that allows users to explore destinations, read reviews, book accommodations and activities, and plan their trips. TripAdvisor uses a mix of user-generated content and editorial content to provide recommendations to users.

**Expedia:** Expedia is an online travel agency that allows users to book flights, hotels, rental cars, and activities. Expedia also offers a trip planning tool that allows users to create custom itineraries based on their preferences.

**Kayak:** Kayak is a travel search engine that allows users to compare prices on flights, hotels, rental cars, and activities. Kayak also offers a trip planning tool that allows users to create custom itineraries based on their preferences.

**Road trippers:** Road trippers is a trip planning platform that is designed for road trips. The platform allows users to create custom itineraries, discover new destinations, and book accommodations and activities.

**Google Trips:** Google Trips is a trip planning app that allows users to create custom itineraries, explore destinations, and book accommodations and activities. The app also provides personalized recommendations based on the user's search history and preferences.

**AGILE METHODOLOGY**

Agile methodology is a popular approach to software development that emphasizes collaboration, flexibility, and iterative development. When it comes to building a trip planner application in SQL, the agile methodology can be used to effectively manage the development process. Here are a few ways that agile methodology can be applied to the development of a trip planner application in SQL:

*User stories:*

User stories are short, simple descriptions of a feature or functionality from the perspective of the user. In the context of a trip planner application, user stories can be used to define the various features and functionalities that users need in order to plan and manage their trips. These user stories can then be used as the basis for development sprints, where developers work on specific features or functionalities in a focused and iterative manner.

*Scrum framework:*

The Scrum framework is a popular approach to agile development that emphasizes frequent communication, collaboration, and iterative development. The Scrum framework involves breaking the development process down into a series of sprints, where developers work on specific features or functionalities in a focused and iterative manner. Regular meetings, such as daily stand-ups and sprint retrospectives, help ensure that the development team stays on track and is able to adapt to changing requirements.

*Continuous integration and testing:*

Continuous integration and testing is a key component of agile methodology, and can be used to ensure that the trip planner application is functioning properly at all times. Continuous integration involves regularly integrating code changes into the application, while continuous testing involves regularly testing the application to identify and address any issues that arise.

*Collaborative approach:*

One of the key principles of agile methodology is collaboration, and this can be applied to the development of a trip planner application in SQL. Developers, designers, and stakeholders can work together throughout the development process to ensure that the application meets the needs of the users and is delivered on time and within budget.

**TOOLS USED**

1. **Tables:**

Tables are used to store and organize data. A table is a collection of related data that is organized into rows and columns. Each row represents a single record or instance of the data, while each column represents a specific attribute or property of the data.It consists of different components.

1. Columns: Columns are used to define the attributes or properties of the data that is being stored. Each column has a name, data type, and optional constraints that can be used to define the format and content of the data.

ii . Rows: Rows represent the individual records or instances of the data that is being stored. Each row contains a value for each column in the table.

iii.Primary key: The primary key is a column or set of columns that uniquely identifies each row in the table. The primary key is used to ensure that there are no duplicate records in the table and to facilitate searching and sorting of the data.

iv.Foreign key: A foreign key is a column or set of columns that links one table to another. The foreign key is used to establish a relationship between two tables, and is typically used to enforce referential integrity.

1. **Joins:**

Joins are used to combine data from two or more tables in a relational database based on a related column between them. Joins are powerful operations that allow you to retrieve data from multiple tables and combine it into a single result set.

**Types of Joins:**

1. **Inner Join**: An inner join returns only the rows that have matching data in both tables being joined. It combines rows from two or more tables based on the specified join condition, and only includes rows where the join condition is satisfied in both tables.
2. **Left Join** : A left join returns all the rows from the left table (table1), and the matched rows from the right table (table2). If there is no match in the right table, NULL values are returned.
3. **Right Join** **:** A right join returns all the rows from the right table (table2), and the matched rows from the left table (table1). If there is no match in the left table, NULL values are returned.
4. **Full Join :** A full join returns all the rows from both the left table (table1) and the right table (table2), and NULL values are returned for non-matching rows in both tables.
5. **Cross Join:** A cross join returns the Cartesian product of the two tables being joined, which means it returns all possible combinations of rows from the two tables.
6. **VIEWS:**

A view is a virtual table that is defined by a query and is stored in the database with a name. A view does not contain any data itself, but it is a saved query that produces a result set when it is executed. Views are used to simplify complex queries, provide a layer of abstraction over the underlying data, and encapsulate business logic for data retrieval and manipulation.

**IMPLEMENTATION**

**A.Establish a schema to support the implementation.**

The schema consists of four tables: Customer\_Registered,Country,Places,Trip\_Data,Offers.

1. Customer\_Registered Table:This table contains information about the each customer such as customer\_id,name,age,phone and registration date.

CREATE TABLE Customer\_Registered (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

age int not null,

phone VARCHAR(15) NOT NULL,

registration\_date DATE NOT NULL

);

insert into Customer\_Registered values

(1, 'John Smith',23, '555-1234', '2022-01-01'),

(2, 'Jane Doe', 24, '555-5678', '2022-01-02'),

(3, 'Bob Johnson',24, '555-9012','2022-01-03'),

(4, 'Samantha Williams', 25, '555-3456', '2022-01-04'),

(5, 'Mike Brown', 43, '555-7890', '2022-01-05'),

(6, 'Lisa Davis', 58, '555-2345', '2022-01-06'),

(7, 'David Lee',34, '555-6789', '2022-01-07'),

(8, 'Karen Johnson', 50, '555-0123', '2022-01-08'),

(9, 'Tom Smith',46, '555-4567', '2022-01-09'),

(10, 'Emily Jones', 21, '555-8901', '2022-01-10'),

(11, 'Chris Wilson', 20,'555-2345', '2022-01-11'),

(12, 'Sarah Brown',27, '555-6789', '2022-01-12'),

(13, 'Paul Kim', 43, '555-0123', '2022-01-13'),

(14, 'Laura Lee',32, '555-4567', '2022-01-14'),

(15, 'Mike Johnson',24, '555-8901', '2022-01-15');

1. Country Table: This table consists of Information abount the Country details such as country id and name of the country.

CREATE TABLE Country (

country\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL);

insert into country values

(1,'India'),

(2,'Vatican City'),

(3,'Saudi Arabia'),

(4, 'Malasiya'),

(5, 'United States'),

(6, 'United Kingdom'),

(7, 'Australia'),

(8, 'Canada'),

(9, 'China'),

(10, 'Japan'),

(11, 'France'),

(12, 'Germany'),

(13, 'Italy'),

(14, 'Spain'),

(15, 'South Africa');

1. Place Table:The place table consists of information about the trip place such as place\_id,name,city and type of the place.

CREATE TABLE Placess (

place\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

city VARCHAR(50) NOT NULL,

country\_id INT NOT NULL,

type varchar(23),

FOREIGN KEY (country\_id) REFERENCES Country(country\_id)

);

insert into Placess values

(1,'Auroville', 'Pondicherry', 1, 'within city'),

(2,'Bharatpur Bird Sanctuary', 'Bharatpur', 1, 'outside city'),

(3,'Kurukshetra', 'Kurukshetra', 2, 'outside city'),

(4,'Kasi', 'Varanasi', 1, 'outside city'),

(5,'Ayodhya', 'Ayodhya', 1, 'outside city'),

(6,'Puri', 'Puri', 1, 'outside city'),

(7,'Peters Basilica', 'Vatican City', 2, 'outside city '),

(8,'Mecca Masjid', 'Mecca', 3, 'outside city'),

(9,'Auroville', 'Pondicherry', 1, 'within city'),

(10,'Badarika Ashram', 'Badarika', 1, 'outside city'),

(11,'Kurukshetra', 'Kurukshetra', 2, 'outside city'),

(12,'Kashi Vishwanath Temple', 'Varanasi', 1, 'outside city'),

(13,'Ayodhya', 'Ayodhya', 1, 'outside city'),

(14,'Puri', 'Puri', 1, 'outside city'),

(15,'Peters Basilica', 'Vatican City', 2, 'outside city');

1. Trip\_Data Table:This table consists of Trip\_id,start\_date,end\_date ,custer\_id and place\_id.

CREATE TABLE Tripsplan(

trip\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT NOT NULL,

place\_id INT NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

booking\_count INT NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES Customer\_Registered(customer\_id),

FOREIGN KEY (place\_id) REFERENCES Placess(place\_id)

);

insert into Tripsplan values

(1,1, 1, '2022-01-10', '2022-01-15', 100),

(2,11, 2, '2022-02-20', '2022-02-22', 500),

(3,1, 7, '2022-03-10', '2022-03-12', 1500),

(4,4, 4, '2022-04-01', '2022-04-05', 2000),

(5,3, 5, '2022-05-05', '2022-05-10', 2500),

(6,3,8, '2022-06-01', '2022-06-05', 300),

(7,6, 3, '2022-01-10', '2022-01-15', 1000),

(8,9, 11, '2022-03-20', '2022-03-22', 500),

(9,1, 10, '2022-03-10', '2022-03-12', 150),

(10,2, 7, '2022-04-01', '2022-04-05', 2000),

(11,3, 5, '2022-05-05', '2022-05-10', 2500),

(12,3, 8, '2022-06-01', '2022-06-05', 3000),

(13,7, 7, '2022-01-10', '2022-01-15', 100),

(14,8, 2, '2022-02-20', '2022-02-22', 500),

(15,2, 8, '2022-03-10', '2022-03-12', 1500);

1. Offers Table:This offer table consists of Offer\_id,name,Start\_date,end\_date.

CREATE TABLE Offer (

offer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL

);

INSERT INTO Offer (offer\_id,name, start\_date, end\_date)

VALUES (1,'New Year Offer', '2022-01-01', '2022-01-31'),

(2,'Valentines Day Offer', '2022-02-01', '2022-02-14'),

(3,'Holi Offer', '2022-03-01', '2022-03-29'),

(4,'Christmas', '2022-12-01', '2022-12-25'),

(5,'Big billion day', '2022-02-01', '2022-02-14'),

(6,'Diwali', '2022-03-01', '2022-03-29'),

(7,'pongal', '2022-01-01', '2022-01-31'),

(8,'Womens day', '2022-02-01', '2022-02-14'),

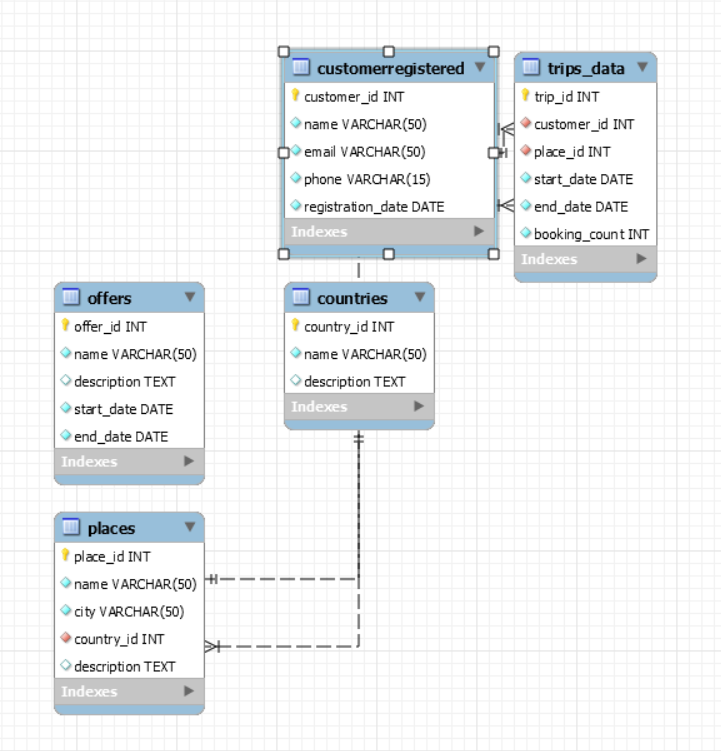
(9,'Holi Offer', '2022-03-01', '2022-03-29'),

(10,'New Year Offer', '2022-01-01', '2022-01-31'),

(11,'Valentines Day Offer', '2022-02-01', '2022-02-14'),

(12,'Holi Offer', '2022-03-01', '2022-03-29');

**B. Use draw.io(or any other suitable tool) to show the logical flow of the data.**



1. **Prepare the Tables like Trips\_Data, Customer\_Registered, Various Offers etc., with suitable constraints to store the records.**

Constraints are rules that are defined on a table to enforce the integrity and consistency of data. Constraints are used to ensure that data in a table follows certain rules or conditions, and they can be used to prevent invalid data from being inserted, updated, or deleted from the table.

There are several types of constraints that can be used in SQL, including:

**Primary Key Constraint:** A primary key constraint is used to uniquely identify each row in a table. It ensures that the values in the specified column(s) are unique and not null, and it is used to uniquely identify rows in the table. Only one primary key constraint can be defined per table.

**Foreign Key Constraint:** A foreign key constraint is used to establish a relationship between two tables based on the values of one or more columns. It ensures that the values in the foreign key column(s) in the referencing table match the values in the primary key column(s) of the referenced table. Foreign key constraints are used to enforce referential integrity between related tables.

**Unique Constraint:** A unique constraint is used to ensure that the values in the specified column(s) are unique. It prevents duplicate values from being inserted into the column(s), but unlike primary key constraints, it allows null values.

**Check Constraint:** A check constraint is used to define a condition that must be met for a row to be inserted, updated, or deleted from a table. It can be used to enforce business rules or other conditions on the data.

Not Null Constraint: A not null constraint is used to ensure that a column does not contain null values. It prevents null values from being inserted into the column.

1. **Customer\_Registered Table**:In this table Customer\_id field has a primary key and not null constraint and other fields are set to not null constraint.

CREATE TABLE Customer\_Registered (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

age int not null,

phone VARCHAR(15) NOT NULL,

registration\_date DATE NOT NULL

);

1. **Country Table**:In this table country\_id field has a primary key and set to be not null constraint and other fields are set to be not null constraint.

CREATE TABLE Country (

country\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL);

1. Places Table:In this table place\_id has a primary key constraint, country\_id has a foreign key constraint and not null and the other fields are set to be not null.

CREATE TABLE Placess (

place\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

city VARCHAR(50) NOT NULL,

country\_id INT NOT NULL,

type varchar(23),

FOREIGN KEY (country\_id) REFERENCES Country(country\_id)

);

1. Trip\_Data Table:In this table Trip\_id has a primary key constraint,customer\_id has a foreign key constraint,place\_id has a foreign key constraint and other fields are set to be not null.

CREATE TABLE Tripsplan(

trip\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT NOT NULL,

place\_id INT NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

booking\_count INT NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES Customer\_Registered(customer\_id),

FOREIGN KEY (place\_id) REFERENCES Placess(place\_id)

);

1. Offers Table:In this table Offer\_id has a primary key constraint and other fields are set to be not null constraint.

CREATE TABLE Offer (

offer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL

);

1. **Populate the data and display the most frequently Opted Place within the City Pondicherry.**

SELECT p.name, COUNT(\*) AS bookings

FROM TripDatas t

INNER JOIN Places p ON t.place\_id = p.place\_id

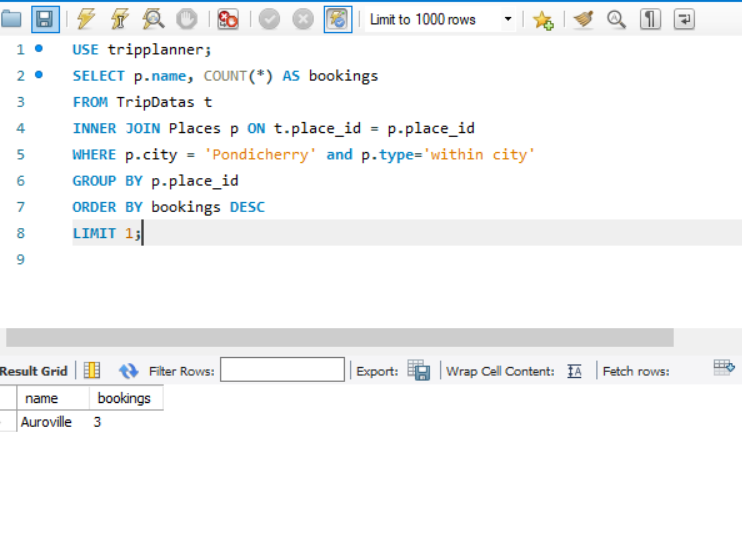
WHERE p.city = 'Pondicherry' and p.type='within city'

GROUP BY p.place\_id

ORDER BY bookings DESC

LIMIT 1;

*Output:*

**

*Explanation:*

SELECT p.name, COUNT(\*) AS bookings: This is the select statement that specifies the columns to be retrieved from the query result. It selects the "name" column from the "Places" table and also calculates the count of rows (i.e., bookings) using the "COUNT(\*)" function and aliases it as "bookings".

FROM Trips\_Data t: This specifies the table "Trips\_Data" to be used in the query and aliases it as "t".

INNER JOIN Places p ON t.place\_id = p.place\_id: This performs an inner join between the "Trips\_Data" table (aliased as "t") and the "Places" table (aliased as "p") using the common column "place\_id" as the joining condition.

WHERE p.city = 'Pondicherry': This specifies a condition for filtering the rows based on the value of the "city" column in the "Places" table. It selects only the rows where the "city" is equal to 'Pondicherry'.

GROUP BY p.place\_id: This groups the result by the "place\_id" column from the "Places" table. This is done to aggregate the bookings count for each unique place.

ORDER BY bookings DESC: This orders the result by the "bookings" count in descending order, so that the place with the highest number of bookings will be listed first.

LIMIT 1: This limits the result to only one row, which will be the place with the highest number of bookings in Pondicherry. The "LIMIT" clause is used to restrict the number of rows returned by the query. In this case, it is set to 1 to retrieve only the top result. The result will be the name of the place with the highest number of bookings in Pondicherry.

1. **Display the Number of Customers who Registered for “Badarika Ashram in the Month of March 2022”**

use tripplanner;

SELECT COUNT(\*) AS registrations

FROM Tripss t

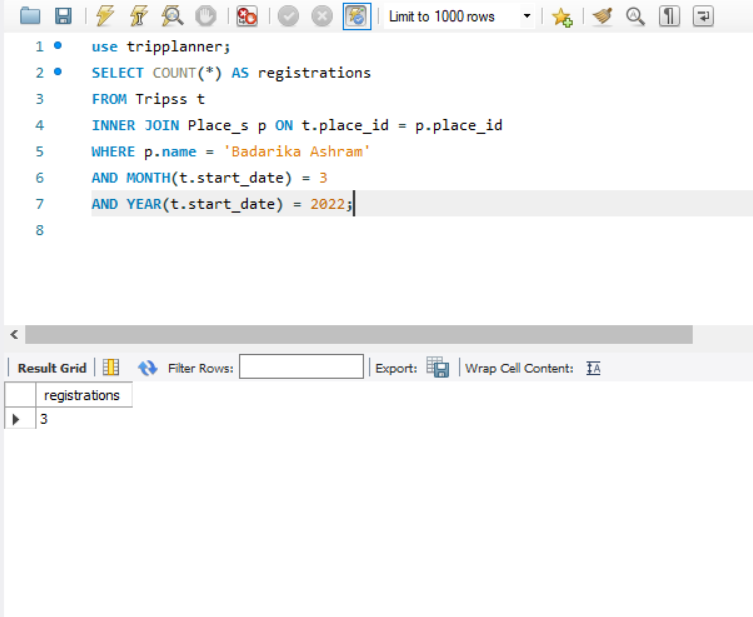
INNER JOIN Place\_s p ON t.place\_id = p.place\_id

WHERE p.name = 'Badarika Ashram'

AND MONTH(t.start\_date) = 3

AND YEAR(t.start\_date) = 2022;

Output:



*Explanation:*

This query will retrieve the count of registrations from the Trips\_Data table in the tripplanner database, where the place name is 'Badarika Ashram', the start date month is March (3), and the start date year is 2022. The INNER JOIN clause is used to join the Trips\_Data table with the Places table on the place\_id column, and the COUNT(\*) function is used to count the number of rows that match the given conditions in the Trips\_Data table. The MONTH() and YEAR() functions are used to extract the month and year from the start\_date column in the Trips\_Data table, respectively.

1. **Display the Bookings count is >1000 for the place Kurukshetra.**

use tripplanner;

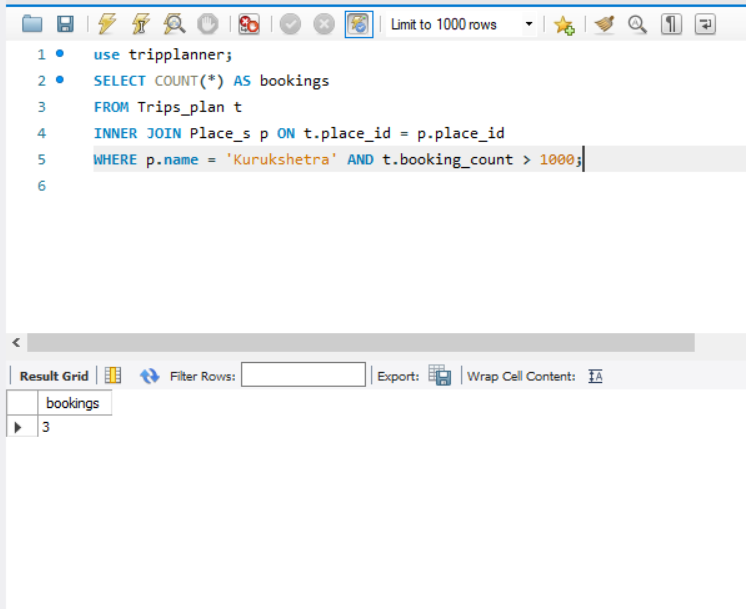
SELECT COUNT(\*) AS bookings

FROM Trips\_plan t

INNER JOIN Place\_s p ON t.place\_id = p.place\_id

WHERE p.name = 'Kurukshetra' AND t.booking\_count > 1000;

Output:



*Explanation:*

1. USE tripplanner;: This is a MySQL command to switch to the tripplanner database.
2. SELECT COUNT(\*) AS bookings: This is the SELECT statement to count the number of records in the result set, and it aliases the result column as 'bookings'.
3. FROM Trips\_plan t: This specifies the table 'Trips\_plan' and aliases it as 't'.
4. INNER JOIN Place\_s p ON t.place\_id = p.place\_id: This is an inner join between the 'Trips\_plan' and 'Place\_s' tables on the 'place\_id' column.
5. WHERE p.name = 'Kurukshetra' AND t.booking\_count > 1000;: This specifies the conditions to filter the results. It checks if the 'name' column in the 'Place\_s' table is equal to 'Kurukshetra' and the 'booking\_count' column in the 'Trips\_plan' table is greater than 1000. Only the records satisfying these conditions will be counted.
6. **List out most Opted Destinations in India.**

USE tripplanner;

SELECT COUNT(\*) AS inquiries

FROM Country c

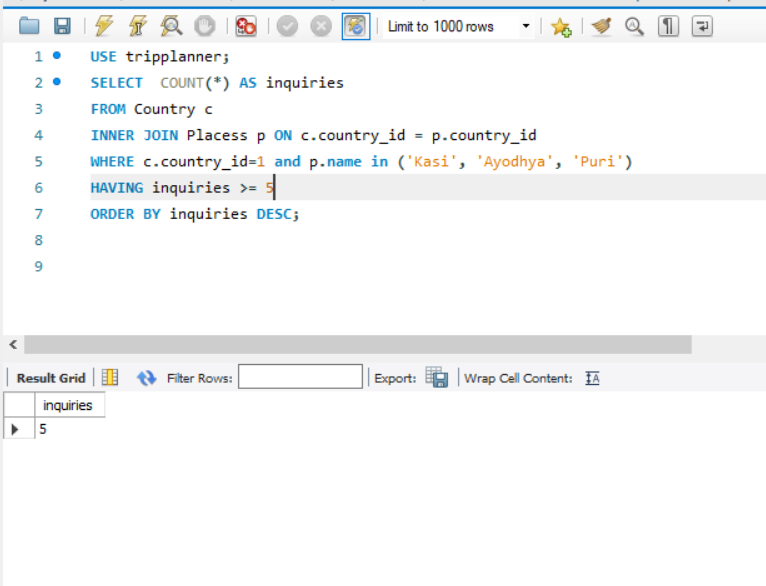
INNER JOIN Placess p ON c.country\_id = p.country\_id

WHERE c.country\_id=1 and p.name in ('Kasi', 'Ayodhya', 'Puri')

HAVING inquiries >= 5

ORDER BY inquiries DESC;

Output:



*Explanation:*

This SQL query is selecting the count of inquiries made for three specific places ('Kasi', 'Ayodhya', 'Puri') in the country with a country\_id of 1, which is assumed to be India. The INNER JOIN statement is joining the Country and Places tables based on their country\_id keys. The WHERE clause is filtering the results to only include data related to India and the specified places. The HAVING clause is filtering the results to only include places with 5 or more inquiries. Finally, the ORDER BY clause is sorting the results in descending order by the number of inquiries.

1. **List out the total registrations received to provide the service for outside of India such as (Peter's Basilica in Vatican City, Mecca Masjid)**

use tripplanner;

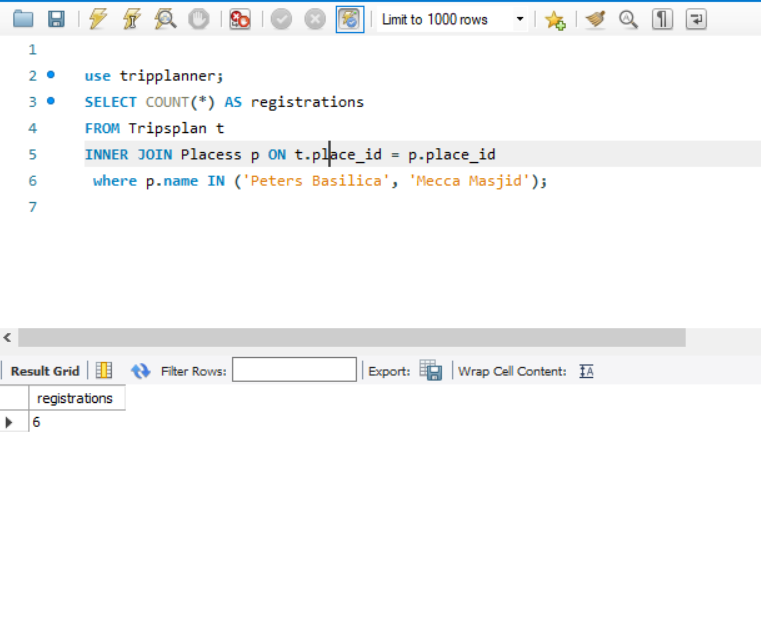
SELECT COUNT(\*) AS registrations

FROM Tripsplan t

INNER JOIN Placess p ON t.place\_id = p.place\_id

where p.name IN ('Peters Basilica', 'Mecca Masjid');

Output:



*Explanation:*

It joins the 'Tripsplan' table with the 'Placess' table on the 'place\_id' column, and then filters the results to only include rows where the 'name' column in the 'Placess' table matches either 'Peters Basilica' or 'Mecca Masjid'.

The 'COUNT(\*)' function is then used to count the number of rows that meet this criteria and returns the result as 'registrations'.

1. **Comment your observations to improve the customer experience.**

***Simplify the user interface:*** A simple and intuitive user interface can help customers navigate the application or website more easily, reducing frustration and improving the overall experience. Use clear and concise language, avoid clutter, and design the interface to be visually appealing and easy to use.

***Provide clear and helpful instructions:*** Provide clear instructions and guidance to help customers achieve their goals. Use step-by-step guides, tooltips, and other forms of instructional support to guide users through the application or website and make the experience more efficient.

***Ensure fast loading and response times:*** Slow loading times or unresponsive pages can be frustrating for customers, especially those with slow internet connections. Optimize the application or website to ensure fast loading and response times, and consider using caching or other performance optimization techniques to improve the user experience.

1. **Take any existing Service Providers such as Club Mahindra Holidays, Goibibo etc., and observe the process of data base management and list out some of the important and possible implementations for our Project.**

***Data warehousing:*** Implement a data warehouse to consolidate data from different sources and provide a single source of truth for reporting and analysis. This can include data from customer registrations, trip bookings, feedback surveys, and other sources.

***Customer relationship management (CRM):*** Implement a CRM system to manage customer interactions and data. This can include customer profiles, trip history, preferences, feedback, and other information. A CRM system can help personalize the customer experience and improve customer satisfaction.

***Data analytics:*** Implement data analytics tools to gain insights into customer behavior, trends, and preferences. This can include tools for data visualization, predictive analytics, and machine learning. By analyzing data, businesses can optimize their offerings and improve the customer experience.

***Cloud-based infrastructure:*** Implement a cloud-based infrastructure to improve scalability, reliability, and accessibility. This can include cloud-based databases, data warehousing, and analytics tools. Cloud-based infrastructure can help reduce IT costs and improve performance.

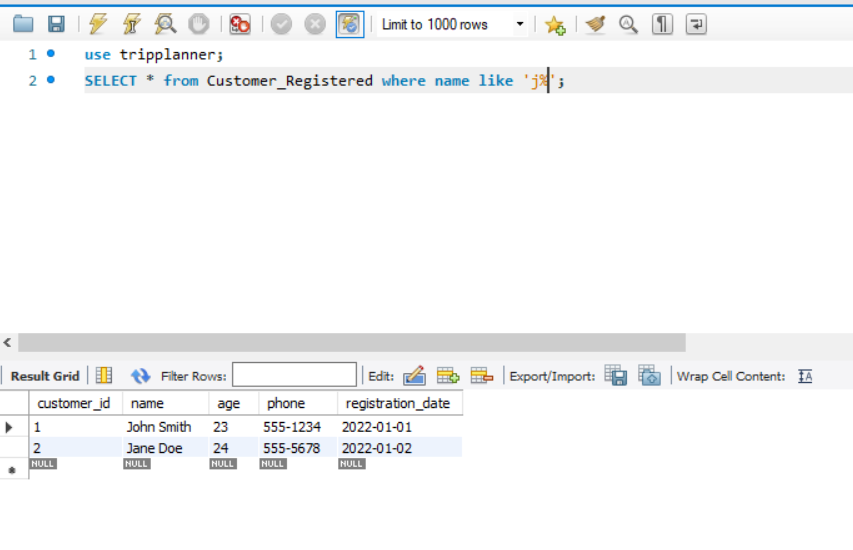
**TEST CASES**

1. **Display customers whose name starts with ‘j’.**

use tripplanner;

SELECT \* from Customer\_Registered where name like ‘j%’;

Output:



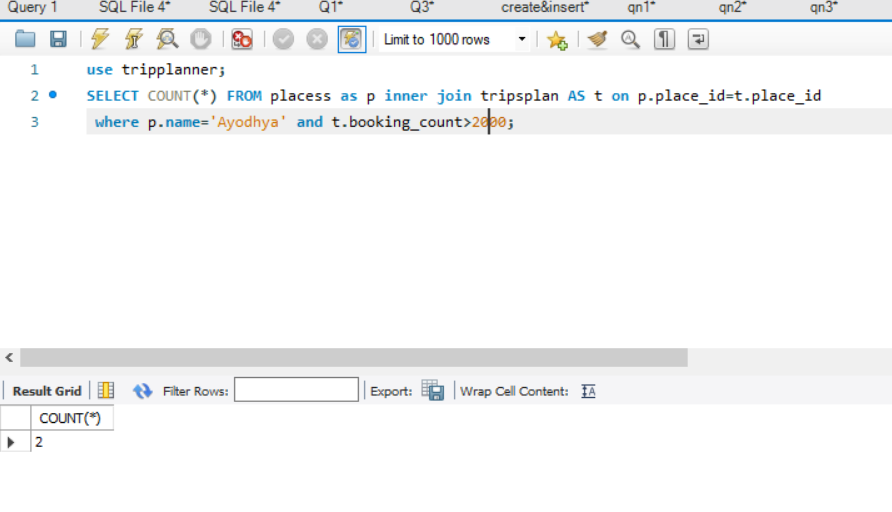
1. **Display the Bookings count is >2000 for the place Ayodhya.**

use tripplanner;

SELECT COUNT(\*) FROM placess as p inner join tripsplan AS t on p.place\_id=t.place\_id

where p.name='Ayodhya' and t.booking\_count>2000;

Output:



1. **Display count the no.of bookings in the city Kurukshetra.**

use tripplanner;

SELECT p.name, COUNT(\*) AS bookings

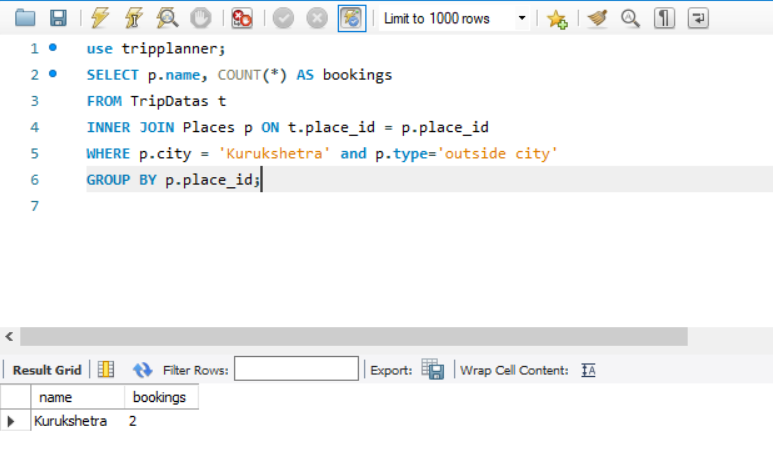
FROM TripDatas t

INNER JOIN Places p ON t.place\_id = p.place\_id

WHERE p.city = 'Kurukshetra' and p.type='outside city'

GROUP BY p.place\_id;

Output:



1. **Display the no.of registration in the city 'Mecca Masjid' and in the month of 6 and year 2022.**

use tripplanner;

SELECT COUNT(\*) AS registrations

FROM Tripsplan t

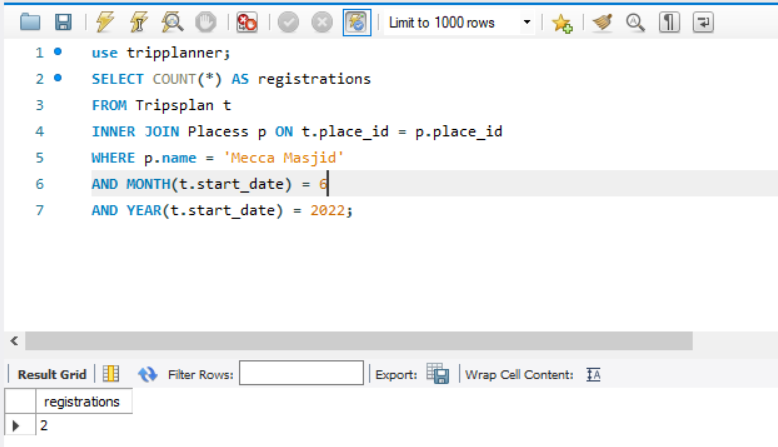
INNER JOIN Placess p ON t.place\_id = p.place\_id

WHERE p.name = 'Mecca Masjid'

AND MONTH(t.start\_date) = 6

AND YEAR(t.start\_date) = 2022;

Output:



1. **List out most Opted Destinations not in India like ('Peters Basilica','Mecca Masjid').**

USE tripplanner;

SELECT \*,COUNT(\*) AS inquiries

FROM Country c

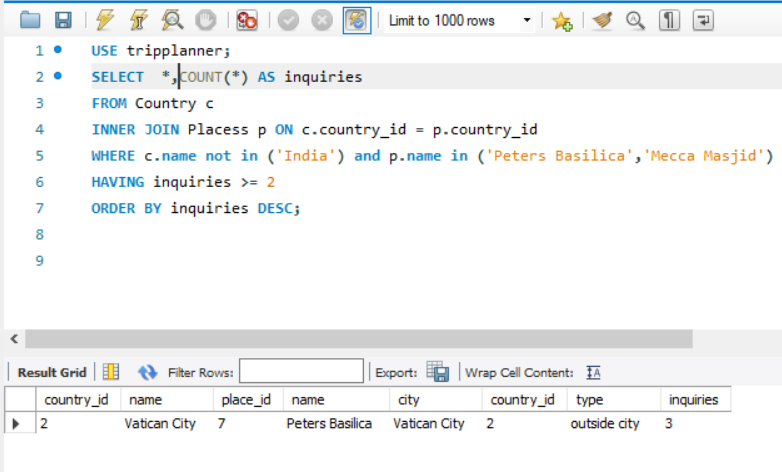
INNER JOIN Placess p ON c.country\_id = p.country\_id

WHERE c.name not in ('India') and p.name in ('Peters Basilica','Mecca Masjid')

HAVING inquiries >= 2

ORDER BY inquiries DESC;

Output:

****

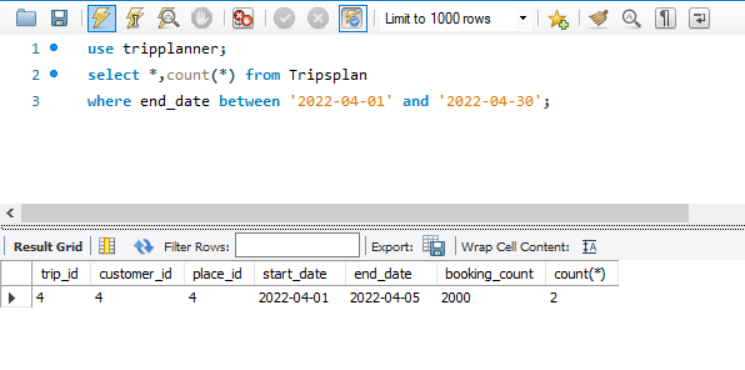
1. **Count the No.of plans end in the date Between '2022-04-01' and '2022-04-30'.**

use tripplanner;

select \*,count(\*) from Tripsplan

where end\_date between '2022-04-01' and '2022-04-30';

Output:



1. **Display the city which are in the country of ‘India’.**

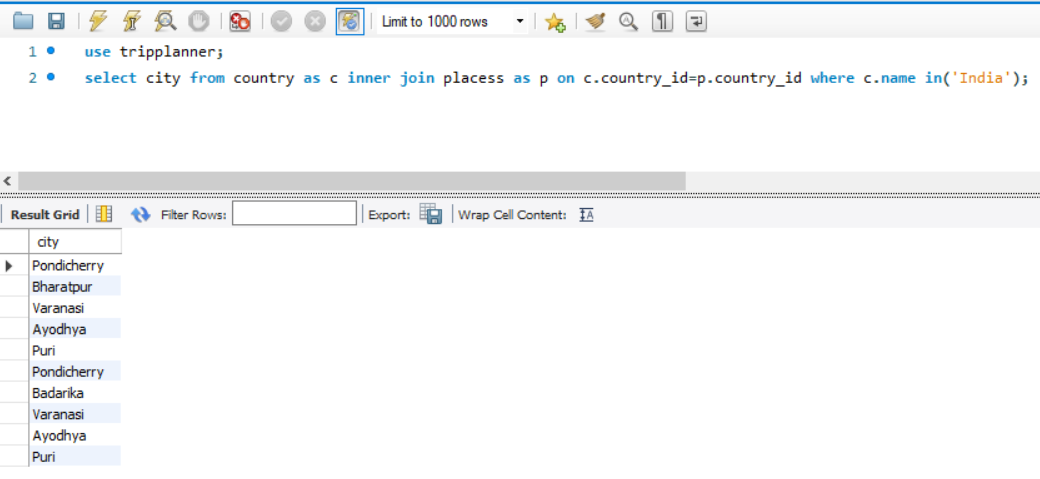
use tripplanner;

select city from

country as c inner join placess as p on c.country\_id=p.country\_id

where c.name in('India');

Output:



**CONCLUSION**

A good Trip Planner application in SQL should have a well-designed database schema, a user-friendly interface, robust security measures, and the ability to efficiently manage and manipulate data to provide accurate and relevant information to users. The application should be able to handle a large volume of data while maintaining good performance. The application should have a user-friendly interface that allows users to search and filter options based on their preferences, view and compare prices, and make reservations or bookings. The application should also have appropriate security measures in place to protect user data. The success of a Trip Planner application in SQL can be determined by how effectively it can manage and manipulate data to provide users with accurate and relevant information. The application should be able to generate reports, provide real-time updates, and offer personalized recommendations based on the user's search history and preferences.

**FUTURE SCOPE OF THE PROJECT**

The future scope of a Trip Planner application in SQL could involve incorporating advanced analytics and data visualization tools to help users make informed decisions about their travel plans. The application could use predictive analytics to forecast travel trends, anticipate user needs, and optimize pricing strategies. Another possible future scope of a Trip Planner application in SQL could be to integrate blockchain technology to provide secure and transparent payment processing, as well as to facilitate smart contracts between travel service providers

and users. Finally, the application could also leverage the Internet of Things (IoT) to provide users with real-time updates and alerts about their travel plans. For example, the application could use IoT devices to track the user's location, provide updates on flight or train delays, and offer personalized recommendations based on local weather conditions.