**Project Report**

**Course Name:** Emerging Non-Traditional Database System

**Course Code:** COMP8390

**Project Report Topic:** Hotel Information System Data Warehouse

**Student Name 1:** Asmita Prabhakar

**Student ID:** 110158616

**Student Name 2:** Atul Kumar

**Student ID:** 110143931

**Submitted to:** Dr. Christie Ezeife

**Due Date:** 15th April 2024

# Table of Contents

# Abstract

The hotel industry relies heavily on data to optimize operations, enhance guest experiences, and drive business growth. In this context, the Hotel Management Data Warehouse System (HMDWS) serves as a comprehensive solution for managing and analyzing hotel-related data. HMDWS integrates data from various sources within the hotel ecosystem, including reservation systems, guest information, room type systems, and financial databases, into a centralized data warehouse. This centralized repository allows hotel managers and stakeholders to access and analyze a wide range of data, including guest demographics, booking patterns, revenue trends, and operational performance metrics. By leveraging advanced analytics and reporting capabilities, HMDWS enables hoteliers to make data-driven decisions, identify opportunities for improvement, and optimize resource allocation. Additionally, HMDWS supports predictive modelling and forecasting to anticipate future demand, optimize pricing strategies, and enhance overall revenue management. Overall, HMDWS empowers hotels to harness the power of data to drive operational efficiency, enhance guest satisfaction, and achieve strategic business objectives in the dynamic and competitive hospitality industry.

# Acknowledgement

We would like to extend our sincere gratitude to Dr. C.I. Ezeife for providing the idea for this project and providing invaluable guidance throughout its development. We also express our appreciation for the insightful feedback and support offered, which significantly contributed to the refinement and success of this project.

# Introduction

**Database:** A database is a structured collection of data organized and stored electronically in a computer system. It is designed to efficiently manage, retrieve, and manipulate large volumes of data. Databases are used to store information in a structured format, making it easy to search, query, and update as needed. They consist of tables, which contain rows and columns to represent specific types of data, such as customer information, product details, or financial records. Databases are essential components of modern computing systems and are used in various applications, including websites, enterprise systems, mobile apps, and more.

**Data Warehouse:** A data warehouse is a centralized repository that stores large volumes of data collected from multiple sources within an organization. It is designed to support decision-making processes by providing a consolidated view of historical and current data. Data warehouses are typically optimized for complex queries and analysis rather than transactional processing. They are essential for business intelligence, data analysis, and reporting purposes, providing decision-makers with insights to support strategic planning, performance monitoring, and informed decision-making.

**ETL:** It stands for Extraction, Transformation and Loading. It consists of first extracting the data from the source databases and then transforming the extracted values into the form, so it becomes consistent in a data warehouse and finally ends with loading this transformed data into the data warehouse. For example, on our website, as soon as a user enters the data into source databases, the PHP script fetches the recently added data and pushes it to the warehouse.

This project represents a fusion of database and data warehouse concepts, aimed at enhancing hotel management operations. The Hotel Management Data Warehouse encompasses several modules, including guest information, reservations, room details, and room information. These

modules are supported by source databases created using MySQL, containing comprehensive data from two distinct hotel branches. Unlike the source databases, the data warehouse is designed to address complex queries and consolidate information from all modules across both branches. The graphical user interface (GUI), developed using PHP, adds a visually appealing and user-friendly layer to the project. This report serves as a comprehensive overview of the tools utilized and their contributions to the project's success. Additionally, it delves into data warehouse modelling techniques and strategies employed to optimize performance and achieve superior outcomes.

# Purpose of application

This project is designed to create a comprehensive system for managing data from two different branches of a hotel. It leverages the power of databases and incorporates a specialized database known as a data warehouse to efficiently store and organize this data. Additionally, the system features a user-friendly interface that allows users to easily view and interact with the stored information. Key functionalities include managing data related to guests, room reservations made, and rooms for both hotel branches.

This system is also able to reduce the storage burden by deleting the old entries from the source databases and keeping them safe in the warehouse forever. This warehouse can make the process quite faster as it would require fewer join operations during querying without needing to bring all the databases from different hotels to one place.

By consolidating data from these branches into a centralized data warehouse, the system simplifies access and analysis processes. This study provides insights into how hotel websites collect, organize, and present information to users, highlighting the importance of effective data management and user interface design in the hospitality industry.

# Implemented Modules

## 1. Guest Module

This module is essential in both source databases, managing guest information for both hotels. It contains personal details about customers like name, phone number, address, etc. Along with that, it also contains an attribute named Amount which would be calculated depending on the check-in and check-out date (from reservation table) and type of room booked (for example, single, double, or deluxe) as a part of the trigger function. Each guest would have a unique GuestID which will also serve as a primary key for this table. Despite having similar structures, differences in attributes and values formatting exist between databases.

## 2. Reservation Module

This module is essential in both source databases, managing reservation information for both hotels. This will contain the information about which guests booked which room, for what dates (check-in and check-out) and with what number of guests. Both GuestID and RoomID will serve as a primary key to identify the unique tuple. The GuestID serves as a foreign key to the Guest table and RoomID serves as a foreign key to the RoomInfo table.

## 3. Room Information Module

This module is essential in both source databases, managing room type (single, double, or deluxe) information for both hotels. The RoomID serves as a primary key and there's just one other tuple which tells the RoomType, this RoomType attribute is further a foreign key to the table RoomDetails to fetch additional details about the room like the cost per night, occupancy,

availability, etc. In the data warehouse, these tables are consolidated, enhancing query capabilities beyond individual databases.

## 4. Room Details Module

In the table, we are managing the room details like cost per night, occupancy, availability, etc. Room Type(Size) serves as a primary key to this table to fetch any unique record. There are three different types of rooms available in both the hotels: with the occupancy of 1, 2, and 4 rooms. This table is very important for calculating the amount since this returns the cost defined for each of the rooms.

## Relationships between the modules

- **Guest- Reservation via books relationship:**

In both source databases, the Guest table is associated with the Reservation table through the GuestID relationship, which includes foreign key constraints. These constraints consist of primary keys or attributes from the Guest and Reservation tables. Using the relationship that exists between these two tables helps us to find which room was booked by which customer and with what check-in and check-out date. It additionally helps to find how many guests are staying in a room. For example, users can’t book a single occupancy room, if they are two together. Either they must choose an additional room or choose a room with occupancy 2 or more. Also, it helps to find, which room is already booked, and which one is available. Since allocating the already occupied room to another person would create a mess, this relationship helps that it would find the room which is not booked or if it was booked the person must be already checked out by the time a new person is booking.

- **Reservation – Room Information via room type relationship**:

In both source databases, the reservation table is associated with the room information table through the "RoomID", which includes foreign key constraints. When the user fills the form, it asks what room type (single, double, etc.) the user wants to book. Depending on the type they chose, it extracts all the RoomIDs of this room type and goes to the reservation table to find the first available room. Once it gets, it takes that RoomID and the system-generated GuestID to add the entry record to the reservation table. As we can see this relationship plays an important role while booking the room on the insertion page of the website.

- **Room Information – Room details via detailed room information relationship:**

In both source databases, the room information table is associated with the room detail table through the "detailed information of room " relationship, which includes foreign key constraints. This relationship is also quite important in terms of finding the maximum occupancy of rooms to make sure people staying in a particular room don't exceed the people limit of that room type. It's also important in terms of fetching the price per night for rooms, as it plays an important role in calculating the amount that should be charged to the customer at the front desk.

# Source Database and Data Warehouse Schema

**Source databse1: Fairmount Resorts schema**

Guest (ID, Name , Email, Phone, Address, Nation, Amount)

Reservation (GuestID, RoomID, Check_in, Check_out, NumberOfGuests)

RoomInformation (RoomID, RoomType)

RoomDetails (RoomType, Occupancy, Availability, PricePerNIght)

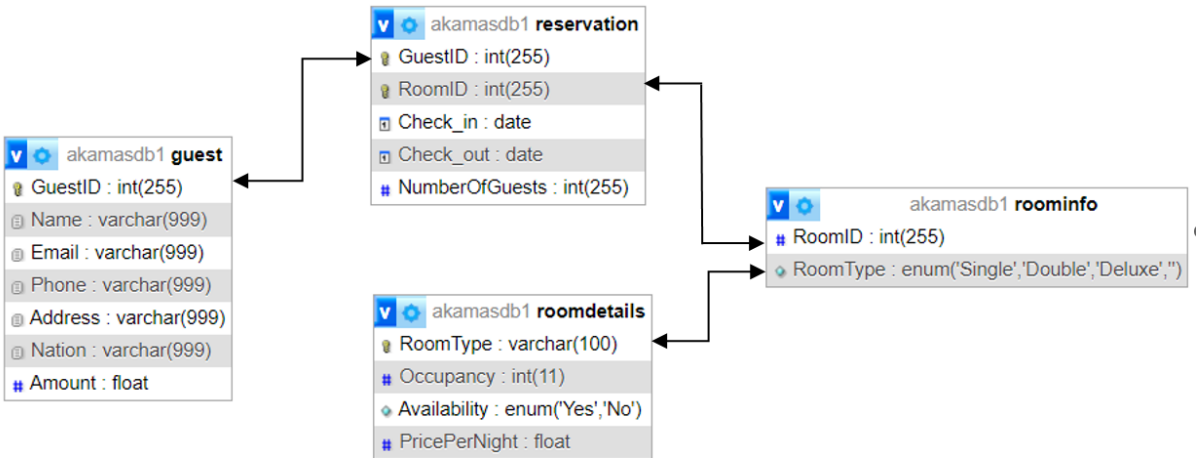


*Figure 1: Source Database 1(Fairmount Resorts)*

## Source database 2: Four Seasons Resorts schema

Guest (GuestID, Name, Email, Phone, Address, Nationality, Amount Paid)

Reservation (GuestID, RoomNo, Check_in, Check_out, NumberOfGuests)

RoomInformation (RoomNo, RoomSize)

RoomDetails (RoomSize, Max.NumberofGuests, Availability, Price)

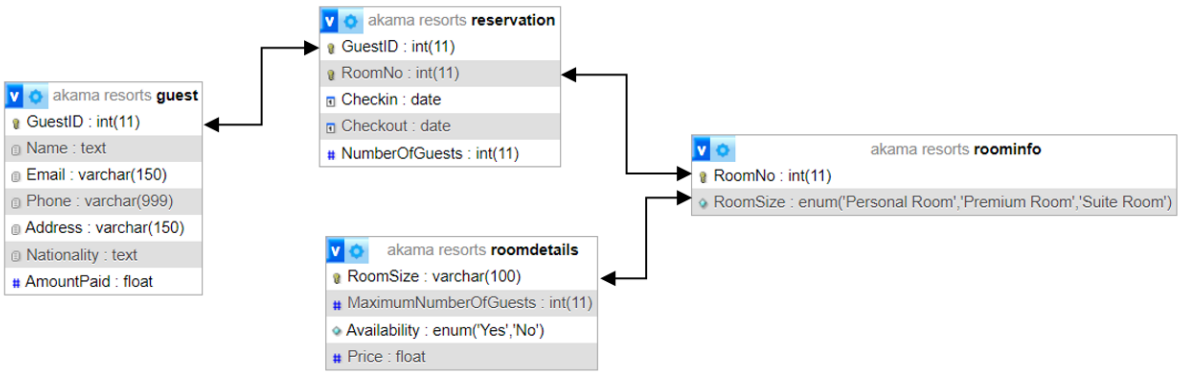


*Figure 2: Source Database 2(Four Seasons Resorts)*

## Data Warehouse

Data Warehouse integrates the information from all the source databases and merges them into one for query purposes. Each of the attributes, except amount, would further have dimension tables which helps to derive the comprehensive details about each of the attributes.

### Fact Table:

Facttable(GuestID, RoomID, BranchID, Date, Total_Amount)

### Dimension Tables:

dim_guest (GuestID, Name , Phone, Email, Address, Nation, Exit Date)

dim_room (RoomID, RoomType, Occupancy, PricePerNight)

dim_branch(BranchID, Name, Location, NumberOfRooms, Contact)

dim_date(Date, Day, Month, Year)

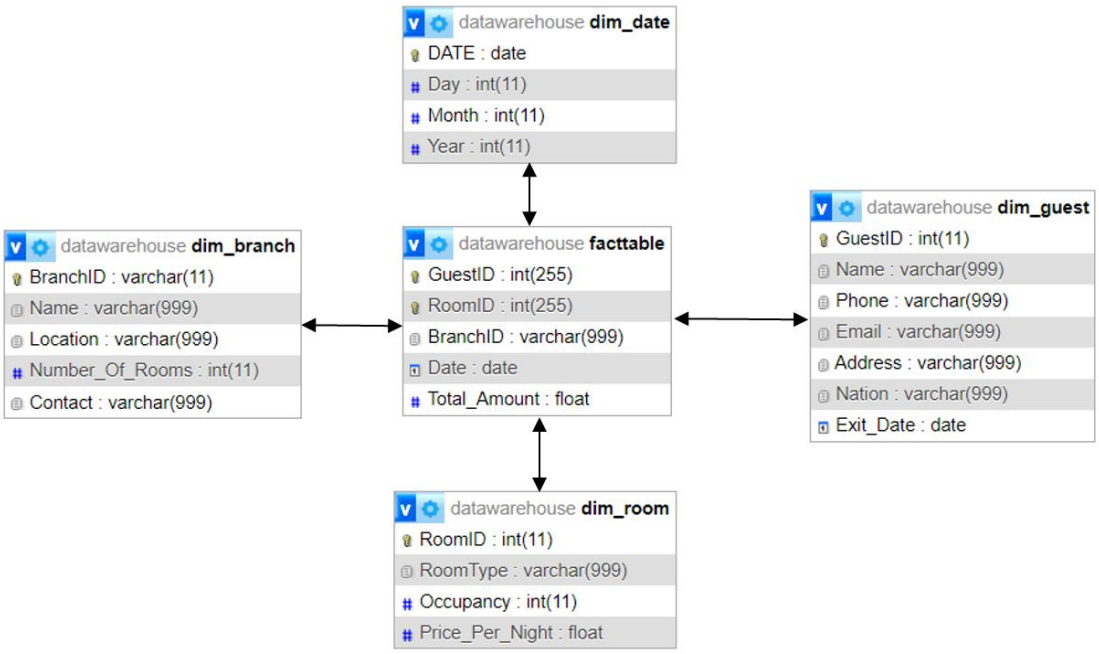


*Figure 3: Fact Table & Dimension Tables*

# Screenshots of Final Work :

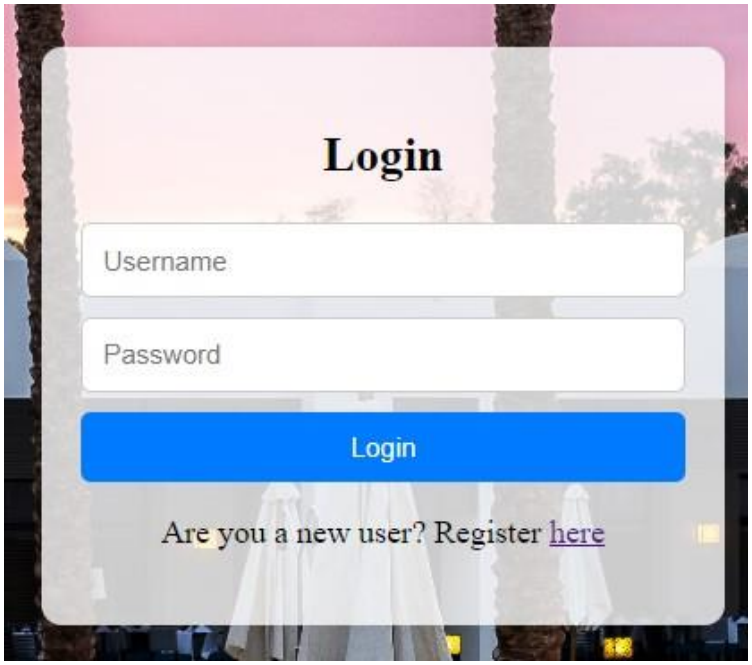


*Figure 4: Login Page*


*Figure 5: Admin User Screen Page where source and warehouse database are accessible*

| Name | RoomID | Phone | Number Of Guests |
|---|---|---|---|
| Mitchell Stark | 3304 | 1-249-563-6868 | 1 |
| Ethan Brown | 3305 | 1-126-482-6473 | 2 |
| Mark Smith | 3303 | 1-437-240-1057 | 2 |

*Figure 6: SDB1 Query: Information about customers who stayed for more than ten days?*

| GuestID | Name | Email |
|---|---|---|
| 104 | Emma Tremblay | emma12@gmail.com |
| 106 | Shane Williamson | swill@gogo.ca |
| 107 | Brady Anderson | brady.anderson@yy.edu |
| 108 | Chris Taylor | ctlr07@gmail.com |

*Figure 7: SDB2 Query: Customers information who checked-in in April?*

| Name | Email | Phone | Address | Nation |
|---|---|---|---|---|
| Ethan Brown | brownethan@gmail.com | 1-126-482-6473 | 789 Oak Lane, Montreal, L9C 3R9, Quebec, Canada | Canada |

*Figure 8: DW Query1: Information about customers who stayed in both the branches?*

| Branch Name | Location | Number of Rooms | Contact | Maximum Revenue($) |
|---|---|---|---|---|
| Fairmont Resorts | 525 Bay Street Toronto, Ontario, Canada, M5G 2L2 | 60 | 1-416-597-9200 | 27405.85 |

*Figure 9: DW Query2: Maximum revenue earned by any branch?*

| Occupancy | Number of Rooms Booked |
|---|---|
| 1 | 9 |
| 2 | 4 |
| 4 | 2 |

*Figure 10: DW Query3: Number of Rooms Booked of each occupancy?*

# Tools and Software:

The Hotel Management System Data Warehouse integrates two databases from different hotels. We utilized XAMPP, which includes Apache, for hosting the websites locally. This system also leverages MySQL for both the source databases and the data warehouse, which integrates them. With regards to the ETL processes, we used PHP to make sure that data is seamlessly loaded into the warehouse upon updates in the source databases. Lastly, to make it user-friendly, HTML and CSS are used to properly provide the skeleton and design of the website, along with the use of PHP which serves front-end and back-end functionalities.

# Conclusion:

This project introduces a Hotel Management Data Warehouse, designed to consolidate and manage data from multiple sources for effective analysis and decision-making. It leverages data from two different branches of a hotel and employs MySQL for the source databases and the data warehouse. The warehouse serves as a centralized repository for both historical and current data, facilitating various queries such as guest information, reservation details, and room occupancy. A user-friendly UI enhances accessibility, allowing users to interact with the warehouse seamlessly. Overall, the project aims to meet the diverse needs of hotel management and can be customized for different hotel branches.

# Future Scope:

As a part of future work, we would like to merge more than two databases and run even more complex queries. This includes the integration of data sources from different departments within the hotel. By integrating them, we aim to gain deeper insights that how profitable the hotel is. Also, we would like to make the design of the website responsive, which means it could shape the website accordingly depending on the device it gets opened on. For example, the navigation bar on the left side may not look good for mobile phones, so it should get adjusted as a drop-down menu when opened on a mobile phone.

# Contributions:

Source Database Creation:

- SDB1- Fairmount Resorts – Atul Kumar
- SDB2- Four Seasons Resorts – Asmita Prabhakar

Data Warehouse Creation:

- Dimension and Fact Tables Creation – Atul Kumar
- Viewing and Indexing – Asmita Prabhakar

ETL Design:

- Extraction – Asmita Prabhakar
- Loading – Atul Kumar

Web Application Design:

- Admin User Page – Atul Kumar
- Login Page and Results of query – Asmita Prabhakar

SQL Queries: Asmita Prabhakar & Atul Kumar

# User Manual:

This section provides detailed information on how to use this system and to understand its features and functionalities of this system. We start by discussing the source database creation for both the hotels and the trigger implementation for them for calculating the amount charged to customers. Then we will see how we are creating the data warehouse. Followed by ETL processes to get the data into the warehouse and the number of views we can get on it. Lastly, we will see how we can install and run this project on any device.

1. **Source Database Creation:**

Here we have some screen snippets for creating the database and tables for one of the source databases. Similarly, a second database can be created using the provided SQL code:

**Guest**

```
CREATE TABLE `guest` (
  `GuestID` int(255) NOT NULL,
  `Name` varchar(999) NOT NULL,
  `Email` varchar(999) NOT NULL,
  `Phone` varchar(999) NOT NULL,
  `Address` varchar(999) NOT NULL,
  `Nation` varchar(999) NOT NULL,
  `Amount` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

**Reservation**

```
CREATE TABLE `reservation` (
  `GuestID` int(255) NOT NULL,
  `RoomID` int(255) NOT NULL,
  `Check_in` date NOT NULL,
  `Check_out` date NOT NULL,
  `NumberOfGuests` int(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

**RoomDetails**

```
CREATE TABLE `roomdetails` (
  `RoomType` varchar(100) NOT NULL,
  `Occupancy` int(11) NOT NULL,
  `Availability` enum('Yes','No') NOT NULL,
  `PricePerNight` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

**RoomInfo**

```
CREATE TABLE `roominfo` (
  `RoomID` int(255) NOT NULL,
  `RoomType` enum('Single','Double','Deluxe','') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Similarly, for the second database, we can use the provided SQL code and populate it with the data.

2. **Trigger:**

We have also implemented two triggers in the system, which automatically update the required things if there's specific event occurs. First, it occurs on the insertion page when a user fills in the details to book the room. As soon as the user fills the form, on the backend, the system fetches the check-in date, check-out date, and type of room booked, it calculates the cost automatically and updates the database.

```
$sql = "UPDATE reservation r
        inner join guest g on g.GuestID = r.GuestID
        inner join roominfo ri on r.RoomID = ri.RoomID
        inner join roomdetails rd on ri.RoomType = rd.RoomType
        set g.amount = datediff(r.check_out, r.check_in)*rd.PricePerNight;";
mysqli_query($conn, $sql);
```

There's also another trigger, that happens whenever we open the source database pages. On each visit, it automatically checks whether there's any entry or not, which is 20 days old. It's because once it gets this old, it will be removed from the source database to reduce the space burden, however, it stays forever in the warehouse.

```php
include("database1.php");
$sql = "DELETE g, r
        FROM guest g
        JOIN reservation r ON g.guestid = r.guestid
        WHERE datediff(now(), r.check_out) > 20;";
$result = mysqli_query($conn, $sql);
$conn->close();
```

### 3. Data Warehouse Creation from Source Database:

Fact table can be created using the following MySQL code:

```sql
CREATE TABLE `facttable` (
  `GuestID` int(255) NOT NULL,
  `RoomID` int(255) NOT NULL,
  `BranchID` varchar(999) NOT NULL,
  `Date` date NOT NULL,
  `Total_Amount` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

This integrates the information from all the source databases and merges them into one for query purposes. Each of the attributes, except amount, would further have dimension tables (which can be created similarly) which help to derive the comprehensive details about each of the attributes, for example name and email address of customers.

### 4. ETL Process:

ETL stands for Extraction, Transformation, and Loading the data into the data warehouse. We utilized PHP for extraction and loading. As soon as we insert something into the source database (means status equals 1), it automatically pushes it to the data warehouse as well.

```php
if ($status == 1) {
    include("dw.php");
    $sql = "INSERT INTO facttable VALUES ('$guest_id', '$room_id', 'Akama_1', '$check_in', '$updatedAmount')";
    mysqli_query($conn, $sql);
    $sql = "INSERT INTO dim_guest VALUES ('$guest_id', '$name', '$phone', '$email', '$address', '$nation', '$check_out')";
    mysqli_query($conn, $sql);

    $sql = "SELECT * FROM dim_room WHERE RoomID = '$room_id'";
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {
    } else {
        if ($room_type = "Single") {
            $sql = "INSERT INTO dim_room VALUES ('$room_id', '$room_type', 1, 240.55)";
            mysqli_query($conn, $sql);
        } elseif ($room_type = "Double") {
            $sql = "INSERT INTO dim_room VALUES ('$room_id', '$room_type', 2, 425)";
            mysqli_query($conn, $sql);
        } elseif ($room_type = "Deluxe") {
            $sql = "INSERT INTO dim_room VALUES ('$room_id', '$room_type', 4, 800)";
            mysqli_query($conn, $sql);
        }
    }

    $day = date('d', strtotime($check_in));
    $month = date('m', strtotime($check_in));
    $year = date('Y', strtotime($check_in));

    $sql = "SELECT * FROM dim_date d WHERE d.DATE = '$check_in'";
    $result = mysqli_query($conn, $sql);

    if (mysqli_num_rows($result) > 0) {
    } else {
        $sql = "INSERT INTO dim_date VALUES ('$check_in', $day, $month, $year)";
        mysqli_query($conn, $sql);
    }
    file_put_contents('sdb1GuestID.txt', $guest_id);
    echo "<script>alert('Entry Successfull :)');</script>";
} else {
    echo "<script>alert('Please fill all the required fields.');</script>";
}
```

## 5. Data Warehouse View:

There are 4 dimensions on which queries can be performed in a data warehouse. These are GuestID, RoomID, BranchID, and Date, along with aggregate column Amount. We can get $2^4 = 16$ views for analyzing the trends and patterns across multiple hotels for achieving business goals. This warehouse can make the process quite faster as it would require fewer join operations during querying without needing to bring all the databases from different hotels to one place.

## 6. Installation Process:

To run this website on the device, the following steps must be followed:

- Make sure you have XAMPP software installed already on your computer to host the website locally. Once downloaded, initiate the Apache and SQL service.
- Open the localhost/phpMyAdmin/index.php and run the code, by going to SQL tab, present in FinalSQLScript.sql to create source databases and warehouse.
- Download the file named Project and extract it to the xampp/htdocs/
- Once done, open the localhost/project/loginpage.php to run the hotel management system and access all the information.

# References:


1) https://cezeife.myweb.cs.uwindsor.ca/courses/60-539/notes/539notes.pdf
2) PHP and MySQL Tutorials: PHP tutorial for beginners - YouTube
3) HTML and CSS Tutorials: W3Schools Online Web Tutorials
4) Data Warehouse Creation: https://www.geeksforgeeks.org/building-a-data-warehouse-in-dbms/