**HOTEL PACKAGE TRACKING SYSTEM**

**INFO 605**

**DATABASE MANAGEMENT SYSTEMS**

**SESSION: FALL 2023/2024**

**GROUP MEMBERS:**

**ABEBE, ESTIFANOS**

**EZIEFULE, CHINONSO**

**MARTINSON, NANA AFUA**

**PIRMOHAMMED, MAHIR**

**Table of Contents**

[1. Introduction 4](#_Toc153266921)

[1.1 Context and Importance of the System 5](#_Toc153266922)

[1.2 Overall Goals of the System 6](#_Toc153266923)

[1.3 Scope of the Project 6](#_Toc153266924)

[1.3.1 IN-Scope: 7](#_Toc153266925)

[1.3.2 OUT-Scope: 7](#_Toc153266926)

[1.4 Related Systems and Open-Source Tools 8](#_Toc153266927)

[2. Requirement Specification 10](#_Toc153266928)

[2.1. Data Requirements 10](#_Toc153266929)

[2.1.1. Users 10](#_Toc153266930)

[2.1.2. Guests 10](#_Toc153266931)

[2.1.3. Staff 10](#_Toc153266932)

[2.1.4. Supplier 10](#_Toc153266933)

[2.1.5. Package 10](#_Toc153266934)

[2.1.6. Robot 10](#_Toc153266935)

[2.1.7. Building 11](#_Toc153266936)

[2.1.8. Charging Station 11](#_Toc153266937)

[2.1.9. Elevator 11](#_Toc153266938)

[2.1.10. Delivery 11](#_Toc153266939)

[2.1.11. Supplier 11](#_Toc153266940)

[2.1.12. Room 11](#_Toc153266941)

[2.1.13. Notification 11](#_Toc153266942)

[2.2. Business Rules and Logic 11](#_Toc153266943)

[2.2.1. Data Entry 12](#_Toc153266944)

[2.2.2. Relationship between Entities 12](#_Toc153266945)

[2.2.3. Delivery Process 12](#_Toc153266946)

[2.2.4. Robot Selection 13](#_Toc153266947)

[2.3. Sample Output 13](#_Toc153266948)

[2.4. Other Assumptions 14](#_Toc153266949)

[3. Conceptual Design 15](#_Toc153266950)

[3.1 Conceptual Model 15](#_Toc153266951)

[3.2 Explanations on ERD 16](#_Toc153266952)

[4. Relational Schema 17](#_Toc153266953)

[4.1 Relational Schema: 17](#_Toc153266954)

[4.1.1 Logical ERD 19](#_Toc153266955)

[4.2 Referential Integrity Diagram 20](#_Toc153266956)

[5. Data Dictionary 21](#_Toc153266957)

[6. Database Implementation 28](#_Toc153266958)

[6.1 CREATE TABLE Commands 28](#_Toc153266959)

[6.2 INSERT INTO Commands 32](#_Toc153266960)

[6.3 Data from SELECT \* FROM your\_table\_name; commands: 42](#_Toc153266961)

[7. Data queries 49](#_Toc153266962)

[7.1 Queries by Estifanos Abebe 49](#_Toc153266963)

[7.2 Queries by Chinonso Eziefule 51](#_Toc153266964)

[7.3 Queries by Nana Afua Martinson 54](#_Toc153266965)

[7.4 Queries by Mahir Pirmohammed 56](#_Toc153266966)

[8. Data Manipulation 58](#_Toc153266967)

[8.1 DML by Estifanos Abebe 58](#_Toc153266968)

[8.2 DML by Chinonso Eziefule 62](#_Toc153266969)

[8.3 DML by Nana Afua Martinson 66](#_Toc153266970)

[8.4 DML by Mahir Pirmohammed 69](#_Toc153266971)

[8. Summary 78](#_Toc153266972)

[9.1 Summary by Estifanos Abebe 78](#_Toc153266973)

[9.2 Summary by Chinonso Eziefule 78](#_Toc153266974)

[9.3 Summary by Nana Afua Martinson 79](#_Toc153266975)

[9.4 Summary by Mahir Pirmohammed 79](#_Toc153266976)

# 1. Introduction

The hospitality business is going through a digital transition, with hotels looking for new methods to improve the guest experience and operational efficiency. Hotels face an increasing issue in controlling the influx of parcels for their residents in this age of internet shopping and doorstep deliveries (Bosscher, 2019). The Package Tracking System for Hotel Residents offers a sophisticated solution to this widespread problem by utilizing the power of current database management and automation.

From the moment of check in to the time they receive their items, hotel guests require a seamless experience. A missing or delayed parcel might ruin an otherwise excellent stay. The Package Tracking System attempts to eliminate such problems by streamlining the entire process, guaranteeing that products are precisely received, tracked, and delivered to hotel residents.

This project not only addresses the logistical challenges of package management, but it also raises the bar for hotel service. Hotels can provide personalized service and timely deliveries by maintaining an accurate database of guest profiles, package records, and inventory management. The system takes things a step further by introducing robotic delivery fleets, which not only add a futuristic touch but also contribute to faster and more reliable package deliveries.

Guest communication is also at the forefront of this innovation. A dedicated chat system allows customers and hotel staff to connect in real time, ensuring that guest requests and inquiries are handled quickly promptly, which not only addresses communication gaps, but also leads to faster and more dependable package deliveries.

Furthermore, the project assumes a hotel environment where a robust wireless network is in place and committed staff are available to manage operations effectively.

In the pages that follow, this proposal will delve into the details of the Package Tracking System, outlining its goals, the context in which it operates, the scope of the project, and the related systems and open-source tools that will drive its implementation. It represents a comprehensive strategy to resolve frequent problems while keeping up with the changing landscape of the hotel business. The Package Tracking System for Hotel Residents is about more than simply effectively managing parcels; it's about upgrading the guest experience and setting new service standards in the hospitality industry.

## 1.1 Context and Importance of the System

In the dynamic landscape of the hospitality industry, where excellence in service is the goal, the Package Tracking System for Hotel Residents emerges as a pivotal innovation. This system addresses the challenge posed by the surge in online shopping, which has significantly increased the volume of packages delivered to hotels. Managing these packages efficiently is essential to ensure a seamless guest experience.

**Why the System is Essential:**

1. **Efficient Package handling:** The significance of this system rests in its ability to streamline the handling of incoming goods, resulting in a smooth procedure. Automated tracking, correct allocation, and on-time delivery are critical for avoiding errors and delays that can disrupt a guest's stay.
2. **Guest Satisfaction:** The core of the hospitality sector is guest satisfaction. The system aims to improve guest experiences by ensuring that items are delivered quickly and securely. A satisfied guest is more likely to return to the hotel and suggest it to others.
3. **Operational Efficiency**: In addition to guest satisfaction, the system prioritizes operational efficiency. It optimizes resource allocation and reduces the manual workload of hotel staff by maintaining a precise database of guest profiles, package records, and inventories.
4. **Personalized Service**: The system adheres to the hotel industry's personalized service ideal. During package deliveries, it seeks to provide visitors with both efficient automation and the pleasant, individualized touch of human engagement.
5. **Real-time Communication:** The use of a chat system allows for real-time communication between visitors and hotel staff, allowing for fast responses to passenger inquiries and requests. This immediate connection enhances the guest's sensation of being cared for.

**Assumptions:**

The system operates within specific assumptions:

* **Hotel Environment**: The project is set in a hotel with modern amenities and infrastructure, including a dependable Wi-Fi network for effective communication.
* **Robotic Delivery Fleet:** The system assumes that robotic technology may be employed for package delivery, but it does not dismiss the importance of human staff involvement if preferred.
* **Data Accuracy:** The project is dedicated to data accuracy, ensuring that visitor profiles, package records, and inventory management are always up to date and error-free.
* **Staff Training:** Hotel staff are assumed to receive adequate training and orientation to effectively utilize the system and deliver packages seamlessly.

## 1.2 Overall Goals of the System

The Package Tracking System for Hotel Residents will be designed with a set of overarching objectives, driven by the vision of enhancing the guest experience and improving the efficiency of package management within the hotel environment. The following objectives highlight the system's significance and purpose:

1. **Efficient Package Management:** The primary goal of the system is to automate and simplify the entire process of receiving, tracking, and delivering items for hotel tenants. This efficiency lowers package handling errors, delays, and guarantees that guests receive their packages on time and in a secure manner.
2. **Guest Satisfaction:** Central to the system's goals is the enhancement of guest satisfaction. Timely and accurate package deliveries contribute significantly to the overall guest experience. The system hopes to impress customers and earn their loyalty by making package monitoring and delivery as simple as possible.
3. **Operational Efficiency:** The system aims for improved operational efficiency by maintaining a centralized and precise database of guest profiles, package records, and inventory management. This, in turn, reduces the manual workload of hotel staff, reducing errors and ensuring seamless package allocation.
4. **Personalized Service:** Personalized service is a hallmark of the hospitality industry, and the system seeks to complement automation with human interaction. While automation plays a role in delivery, the human touch remains pivotal as staff oversee the process and interact with guests, ensuring a seamless and personalized experience.
5. **Real-time Communication:** The introduction of a chat system ensures real-time and direct communication between guests and hotel staff. This is to enhance guest-staff interactions, and to facilitate immediate responses to guest requests and inquiries. The goal is to make guests feel valued, attended to, and at ease during their stay.
6. **Security and Accuracy:** The system places a premium on package security and accuracy. It guarantees that packages are allocated accurately and that there is a detailed record of each package's path from arrival to delivery by maintaining a meticulous database of guest profiles and package records within the hotel environment.
7. **Assumption Alignment:** The system's aims are aligned with specific assumptions, such as the existence of a dependable wireless network, a robotic delivery fleet, data accuracy, and personnel training. These objectives are intended to align with these assumptions in order to build a seamless package tracking procedure that combines technology with human interaction to deliver optimal results.

## 1.3 Scope of the Project

The scope of the Package Tracking System for Hotel Residents project is defined by the specific components and functionalities that will be included in the system, as well as those that will be excluded, ensuring clarity and alignment with the project's objectives.

### 1.3.1 IN-Scope:

1. **Guest Profiles:** The system will encompass the management of guest information, including names, contact details, room numbers, floor numbers, check-in/out dates, and preferences. This data is essential for personalized service and efficient package allocation.
2. **Package Records:** The system will manage package information, such as sender/recipient details, package type, size, weight, delivery status, and timestamps. This includes tracking the packages from the moment they arrive at the hotel until they are successfully delivered to the intended recipient.
3. **Human Staff Delivery:** The package delivery process will involve hotel staff who will be responsible for delivering packages directly to guests' rooms. This includes staff coordination and communication with guests during the delivery process.
4. **Chat System:** The project will include a chat system that facilitates real-time communication between guests and hotel staff. This feature allows guests to make requests, ask questions, and interact with hotel personnel, enhancing their overall experience.
5. **Real-time Communication:** The system will offer real-time communication, allowing guests to receive timely updates on the status of their packages and enabling them to communicate directly with staff when needed.
6. **Robotic Fleet Details**: Data from the robotic fleet utilized for package delivery is included in the project. This includes information such as robot specifications, serial numbers, energy usage, and battery condition, while the primary focus remains on the database system.
7. **Security Measures:** Security measures for package handling, such as verifying package recipients and ensuring data accuracy, will be included to maintain the integrity of the process.

### 1.3.2 OUT-Scope:

1. **External Financial Transactions:** The project does not involve guest billing and payment information. Financial transactions will be out of scope and managed by the hotel's financial system.
2. **Unrelated Hotel Employee Records:** Records of hotel employees not directly involved in the package tracking system, such as HR or payroll data, will not be part of the system.
3. **Detailed Inventory Procurement:** The system will not handle the procurement and restocking processes for hotel inventory. It focuses on managing the availability of items.
4. **External Reservation Systems:** While the system tracks guest check-in and check-out dates, it will not handle full hotel reservations or bookings, which are typically managed through separate reservation systems.
5. **Package Tracking Beyond Hotel Premises:** The system's tracking is limited to packages within the hotel premises and does not extend to monitoring packages in transit or after they leave the hotel.

## 1.4 Related Systems and Open-Source Tools

The success of the Package Tracking System for Hotel Residents relies on its integration with related systems and the utilization of open-source tools to enhance efficiency and functionality. Below are the related systems and open-source tools that will complement the project:

**Related Systems:**

1. **Hotel Management System:** The Package Tracking System will integrate with the hotel's existing Hotel Management System (HMS). This integration ensures that guest profiles, including room numbers and check-in/check-out dates, are synchronized between the two systems, providing up-to-date information for package allocation.
2. **Wireless Network Infrastructure:** A robust and reliable wireless network infrastructure is fundamental for real-time communication and data transfer. The project assumes seamless integration with the hotel's existing network infrastructure to ensure continuous connectivity.
3. **Robotic Delivery Fleet Management System:** While the project will focus on database administration and tracking systems, it will also interact with a Robotic Delivery Fleet Administration System. This system will offer real-time information on the position and availability of package-delivery robots. Package allocation and delivery coordination are flawless as a result of the connection.
4. **Elevator Control System:** The system will interact with the Elevator Control System to ensure efficient movement of robots within the hotel, allowing them to access different floors and deliver packages in a timely manner.
5. **Inventory Management System:** The system will connect with the hotel's Inventory Management System to maintain accurate records of available items for guests, such as toiletries, towels, and snacks. This integration ensures that the inventory is always up to date.

**Open-Source Tools:**

1. **Database Management System (Oracle SQL):** The project will leverage open-source database management systems to store, manage, and retrieve guest profiles, package records, and inventory information efficiently. These tools provide robust data management capabilities.
2. **Real-time Messaging Platforms (Google Meet/Hangout):** Open-source messaging platforms will be utilized to facilitate real-time communication between different system components, ensuring that guests receive timely package status updates and enabling instant chat functionality, this is in consideration that most clients make use of Gmail.
3. **Web Development Frameworks (React, Django and/or Python):** Open-source web development frameworks will be used to create user interfaces, enabling guests to interact with the system and hotel staff to access and manage data efficiently.
4. **Data Visualization Tools (e.g., Apache Superset):** Data visualization tools may be utilized to create informative dashboards and reports for hotel staff to monitor package tracking, inventory levels, and other relevant data in real-time.
5. **Open-Source Chat Libraries (Socket.IO, SignalR):** For implementing the chat system, open-source chat libraries will be integrated to enable direct communication between guests and hotel staff, ensuring a seamless and real-time communication experience.
6. **Wireless Network Tools (e.g., Open-source Wi-Fi management tools):** Open-source wireless network management tools may be utilized to ensure the hotel's network is robust, reliable, and capable of supporting the wireless communication necessary for the system's operation.
7. **Security and Authentication Tools (Open Source OWASP ZAP):** To enhance security, open-source security and authentication will be employed to safeguard the database systema, implement user authentication, and protect against potential threats and vulnerabilities.

These related systems and open-source tools are essential components of the Package Tracking System, ensuring data accuracy, real-time communication, and seamless integration with existing hotel systems. By incorporating these resources, the system can effectively streamline package management, enhance the guest experience, and elevate operational efficiency.

# 2. Requirement Specification

This section presents a comprehensive overview of the data management requirements. It specifies the data entities and elements that need to be defined to develop a database management system for the hotel package tracking system. Furthermore, it presents the types of reports and queries that need to be generated, as well as the technology-level business rules that the database management system must adhere to.

## 2.1. Data Requirements

To build a database that meets the needs of a hotel package management system, the data requirements must be clearly defined. This ensures simplified and logical data modelling, data consistency, efficient data retrieval, system scalability, and data security. This section defines the different data entities that need to be modelled and their respective attributes.

### 2.1.1. Users

The user entity will be used to store users’ account creation time, first name, last name, address, phone number, the last login time into the system, and their role in the system (guest, staff, and admin). The data stored in this entity will be used for access control and ensures the right users have the appropriate permission in the system on the application level.

### 2.1.2. Guests

The guest entity represents the individuals who are staying at the hotel. It is used to store details such as the guest id, address, emergency contact (name, phone number), check-in date, check-out date, and id of the room assigned. Since guests are at the center of the hotel’s operation, managing guest information helps with package delivery coordination and notification management, ensuring a positive guest experience.

### 2.1.3. Staff

The staff entity represents hotel employees who may interact with the package tracking system. It will consist of attributes such as staff id, user id (referenced from user entity), and working shift.

### 2.1.4. Supplier

The supplier entity contains information about the organization or individuals who send the packages. The data points that need to be saved in this entity include the supplier’s id, name (individual’s or organization’s name), address, phone number, and email.

### 2.1.5. Package

The package entity records information about the packages that are sent to the guests from the suppliers. Attributes that need to be saved in this entity include package id, package type, package weight, package volume category, indicator for fragile items, delivery id (referenced from the delivery entity), delivery status, sender id (referenced from the supplier entity), and receiver id (referenced from the guest entity).

### 2.1.6. Robot

The robot entity represents the mobile robots that will be used for package delivery. It will be used to save the information of the robot such as robot id, model, manufacturer, weight, maximum weight capacity, maximum volume capacity, battery status, and current location.

### 2.1.7. Building

The hotel facility may consist of different buildings. Therefore, data such as building id, building name, and location need to be saved in the database.

### 2.1.8. Charging Station

This entity is used to represent the stations the robots use to charge their batteries. The robots could automatically go to the station and dock themselves to charge their batteries. The attributes that will be saved in this entity include station id, power rating, id of current robot docked to the station, and its operational status.

### 2.1.9. Elevator

The elevator entity is used to represent the elevator the robots would use to move between floors. It will be used to save information such as elevator id, elevator’s maximum weight capacity, the size type of the robots that could ride the elevator, and current total weight of the elevator.

### 2.1.10. Delivery

The delivery entity is used to represent information about the delivery process. The attributes that will be saved in this entity include delivery id, delivery type (either staff or robot), robot id (referenced from robot entity), staff id (referenced from staff entity), current location of the package, guest id (referenced from the guest entity), the expected arrival time of the package, the actual package arrival time, time at which the guest received the package, guest’s feedback and rating, and the delivery status.

### 2.1.11. Supplier

The supplier entity is responsible for saving information related to the suppliers that ship the packages to the hotel’s guest. Information such as the supplied name, email address, phone number and address will be saved in this entity.

### 2.1.12. Room

The room entity is used to model the rooms the guests stay in. It will be used to store data such as room id, room number, area of room, number of beds, maximum occupancy, and the room’s corresponding floor.

### 2.1.13. Notification

The notification entity will be used to manage push notifications that are sent to users regarding package arrival, updates, and other relevant information. Data fields that will be saved in this entity include notification id, time of notification generation, notification type (alert or status), user to which the notification would be delivered to (referenced from user entity), content of notification, notification status (read or unread), and delivery id (referenced from delivery entity).

## 2.2. Business Rules and Logic

The hotel package tracking system incorporates various business rules and logic to ensure the hotels offer a convenient and satisfactory service to their customers. In this section, a comprehensive list of the business rules and logic that must be taken care of when designing the hotel package tracking system is presented.

### 2.2.1. Data Entry

Guests are automatically registered in the system when they book the hotel online. Information such as their names, addresses, contact details are collected in an online form. This information could also be collected automatically from if they book the hotel in person, the data is manually entered into the system by the hotel staff at reception. Data for hotel staff and admin is assumed to be collected from the hotel’s Human Resource Management System (HRMS).

Static data about the mobile robots is manually entered into the system by hotel staff whenever new robots are added to the fleet. Dynamic data from the robots such as their location and battery status will be transmitted from the robots to the database using protocols such as MQ Telemetry Transport (MQTT) or through web REST API. How the raw data from the robots is transferred and stored in the database before it is saved as an attribute in different tables is beyond the scope of this project.

Data for suppliers is automatically fetched from third party systems such as ecommerce and logistic applications. If the senders do not use these technologies, the corresponding data for suppliers is manually entered by the hotel staff. Static data related to buildings, rooms, robot charging stations, and elevators are also manually entered by the hotel staff.

### 2.2.2. Relationship between Entities

The following is a list of constraints on the relationships that exist between the entities.

* There is at least one elevator in each building.
* There is at least one charging station in each building.
* Each building will have at least one robot for the delivery operation. A robot could either be located in an elevator, charging station, or somewhere else in the building.
* All users in the system are either guests or staff. A staff member works a single shift every day of the week.
* There is at least one room that guests could reserve in each building.
* Each guest will have at least one emergency contact registered in the system.
* Each guest will be assigned to one and only one room.
* Each package is delivered by a single supplier to a single guest. On the other hand, each supplier could ship multiple packages.
* A single staff member will be assigned for each package that arrives at the hotel. A staff member could be responsible for multiple packages.
* Each package is tracked by a unique delivery id.
* For every delivery, there could be multiple notifications that could be sent to the guest.
* If a guest is to be deleted from the system, data related to their emergency contact, packages, delivery, and notifications will have to be deleted.

### 2.2.3. Delivery Process

The package is first shipped to the hotel once an order is made by the guests using an external system. Once the package arrives at the hotel, it is moved to the storage room to be inspected and registered in the system by a hotel staff. The hotel staff then logs the package into the database system and initiates a delivery process. Once the delivery is initialized, guests will get a push notification telling them that their package is on the way.

After inspection, if the package is found to be too heavy or needs careful handling, it is directly delivered by the hotel staff to the guests. Otherwise, the package will be delivered by a mobile robot with a self-driving capability. The robot selection is automatically done by the system based on the size and weight of the package and the availability of a suitable robot. Once an appropriate robot is selected by the system, the robot is instructed to move to the storage room to pick up the package.

After picking up the package, the robot moves to a predetermined elevator and moves to the floor where the guest’s room is located at. The robot exits the elevator and moves to the guest's room. Once it arrives at the door of the guest’s room, the guest will get a push notification telling them that their package has arrived.

After the guest receives their package, they get an option to leave a review about the delivery. This will ensure the staff know of any complaints the guests may have about the delivery. Once the robot completes the delivery, it will make its status available unless it must go to a nearby charging station to charge its battery.

### 2.2.4. Robot Selection

A robot is selected for a given delivery based on the following criteria.

1. The robot should have enough battery capacity to complete the delivery.
2. The robot should have enough space to hold a package. It should also have enough capacity to sustain the weight of the package.
3. A robot that will have to travel the least distance is selected for the delivery. For example, if there are two available robots on the 3rd and 5th floors and the package is to be delivered to a room on the 10th floor, the robot on the 3rd floor will be chosen for the delivery assuming that the storage room is located on the 1st floor.

## 2.3. Sample Output

The database is expected to be used to give answers to queries the system needs to know for the delivery system to work efficiently. Below are some example queries that could be answered using the database:

1. A package weighs 5lb and volume is classified as medium size. Which robots can deliver this package?
2. The package mentioned in (a) is to be delivered to the 10th floor, which robots travel the least distance when making the delivery?
3. Among the robots that are chosen in (b), which ones have sufficient battery capacity to complete the delivery?
4. What is the nearest empty charging station that a given robot could use to charge its batteries?
5. How many robots are there on a given floor?

The database could also serve as a valuable resource for generating internal analytic reports, which, in turn, could point out potentially inefficient practices and help improve the overall business process. Some questions that could be answered based on the data collected in the database are listed below.

1. What was the average number of customers who used the hotel delivery system each month?
2. What was the average rating of the delivery service for each corresponding month?
3. How many robots were used to complete delivery operations during each month?
4. What percentage of the guests who booked into the hotel used the hotel’s delivery system for each month?

## 2.4. Other Assumptions

Assumptions that were made when specifying the data requirements of the hotel package delivery system include the following:

* The hotel package management system seamlessly integrates with third-party systems, such as ERP (Enterprise Resource Planning) and logistic systems, enabling the automated retrieval of data concerning hotel staff members involved in package deliveries and details regarding the package senders.
* User permissions and roles will be clearly defined and controlled at the application level. This ensures that each category of users operates within predefined parameters, thus safeguarding data and system integrity.
* The robots deployed for package delivery are assumed to be equipped with advanced sensors, enabling them to gain awareness of their surroundings. These sensors will allow for precise location determination and obstacle avoidance. Additionally, the robots will possess path-planning capabilities to efficiently reach desired destinations.
* Each robot is assumed to manage its own separate system and database, localized on the robot itself. The robots can communicate among themselves and with a central server with access to the database designed in this project. This setup is designed to support the seamless execution of their software, ensuring operational efficiency and independence of each robotic entity. The design of the local database for each robot is beyond the scope of this project.

# 3. Conceptual Design

## 3.1 Conceptual Model

ERD Assumptions:

* Users can be either a Guest or a Staff member, not both.
* Guests can have many Emergency Contacts.
* A Guest is assigned to only one room.
* A room can be assigned to no guest (vacant) or assigned to a guest (occupied)
* Package is received by one and only one Guest.
* Package is handled by one and only one Staff member.
* Guests can receive multiple packages.
* Staff members are responsible for many packages at any given time.
* A package may have only one supplier.
* Each package is tracked by one delivery.
* A delivery creates many notifications to one or more Users.
* A notification is created for one and only one Delivery.
* A package is delivered by either a robot or Staff.
* A robot can deliver many packages at a time.
* The building is made up of multiple Rooms.
* The building contains multiple elevators.
* The building holds multiple charging stations.
* A robot may charge at any unoccupied charging station (cardinality 1).
* A robot may need to wait to charge at a Charging station (participation 0).
* Many robots can ride in an Elevator but should not exceed elevator weight capacity.

## 3.2 Explanations on ERD

The ERD is based on section 2. Refer back to this section for more details on the ERD’s entities, relationships, attributes, and cardinalities.

A computer screen shot of a computer flowchart

Description automatically generated

# 4. Relational Schema

## 4.1 Relational Schema:

Building(buildingID, buildingName)

Room(buildingID\*, roomID, roomNumber, area, beds, maxOccupancy, floor)  
 Foreign key buildingID REFERENCES Building(buildingID);

Elevator(buildingID\*, elevatorID, weightCapacity, maxRobots, sizeType, totalWeight)  
 Foreign key buildingID REFERENCES Building(buildingID);

ChargingStation(buildingID\*, stationID, powerRating, operationalStatus)  
 Foreign key buildingID REFERENCES Building(buildingID);

Users(userID, firstName, lastName, email, phone, street, city, state, country, zip, creationTime, lastLoggedIn, role);

Guests(userID\*, guestID, checkInTimestamp, checkOutTimestamp, buildingID\*, roomID\*)  
 Foreign key userID REFERENCES User(userID);  
 Foreign key (buildingID, roomID) REFERENCES Room(buildingID, roomID);

EmergencyContact(userID\*, guestID\*, contactID, firstName, lastName, email, phone)  
 Foreign key (userID, guestID) REFERENCES Guest(userID, guestID);

Staffs(userID\*, staffID, shift)  
 Foreign key userID REFERENCES User(userID);

Supplier(supplierID, name, email, phone, street, city, state, country, zip)

Robot(robotID, r\_model, manufacturer, weight, weightCapacity, volumeType, batteryStatus, floorNum, buildingBuildingID\*, elevatorBuildingID\*, elevatorID\*, stationBuildingID\*, stationID\*)

Foreign key buildingBuildingID REFERENCES Buildin(buildingID);

Foreign key elevatorBuildingID REFERENCES Building(buildingID);  
 Foreign key elevatorID REFERENCES Elevator(elevatorID);

Foreign key stationBuildingID REFERENCES Building(buildingID);  
 Foreign key stationID REFERENCES Station(stationID);

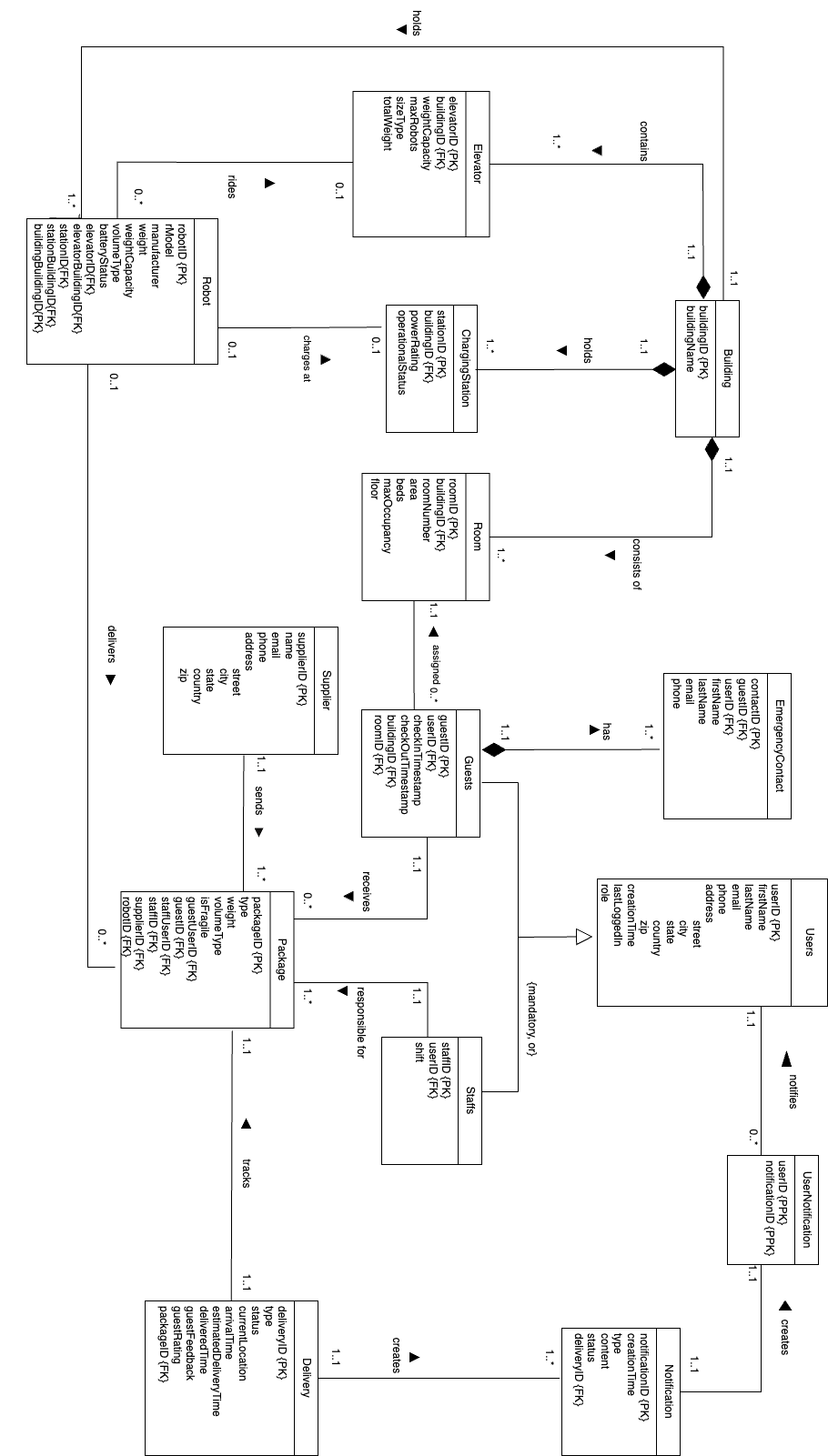
Package(packageID, type, weight, volumeType, isFragile, guestUserID\*, guestID\*, staffUserID\*, staffID\*, supplierID\*, robotID\*)  
 Foreign key (guestUserID, guestID) REFERENCES Guest(userID, guestID);  
 Foreign key (staffUserID, staffID) REFERENCES User(userID, staffID);  
 Foreign key supplierID REFERENCES Supplier(supplierID);  
 Foreign key robotID REFERENCES Robot(robotID);

Delivery(deliveryID, type, status, currentLocation, arrivalTime, estimatedDeliveryTime, deliveredTime, guestFeedback, guestRating, packageID\*)  
 Foreign key packageID REFERENCES Package(packageID);

Notification(notificationID, creationTime, type, content, status, deliveryID\*)  
 Foreign key deliveryID REFERENCES Delivery(deliveryID);

UserNotifications(userID, notificationID)  
 Foreign key userID REFERENCES User(userID);  
 Foreign key notificationID REFERENCES Notification(notificationID);

### 4.1.1 Logical ERD



## 4.2 Referential Integrity Diagram

A black and white text

Description automatically generated

# 5. Data Dictionary

Building

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| buildingID | NOT NULL | char(3) | Building ID | Yes | No |
| buildingName | NOT NULL | varchar2(30) | Name of building | Yes | No |

Room

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| buildingID | NOT NULL | char(3) | Building ID | Yes | Yes |
| roomID | NOT NULL | char(4) | Room ID | Yes | No |
| roomNumber | NOT NULL | number(3) | Room Number | No | No |
| area | NOT NULL | number(4) | Room area | No | No |
| beds | NOT NULL | number(2) | Number of Beds | No | No |
| maxOccupancy | NOT NULL | number(2) | Maximum occupancy | No | No |
| floor | NOT NULL | number(2) | Floor number | No | No |

Elevator

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| buildingID | NOT NULL | char(3) | Building ID | Yes | Yes |
| elevatorID | NOT NULL | varchar2(3) | Elevator ID | Yes | No |
| weightCapacity | NOT NULL | number(8, 2) | Maximum weight | No | No |
| maxRobots | NOT NULL | number(2) | Maximum robots the elevator can hold | No | No |
| sizeType | NOT NULL | char(1) | “S”, “M”, “L” | No | No |
| totalWeight | NOT NULL | number(8,2) | Current weight | No | No |

Charging Station

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| buildingID | NOT NULL | char(3) | Building ID | Yes | Yes |
| stationID | NOT NULL | varchar2(3) | Building ID | Yes | No |
| powerRating | NOT NULL | number(8, 2) | Power rating in watts | No | No |
| operationalStatus | NOT NULL | char(1) | “O”, “R” for Operational and Under-Repair respectively | No | No |

Users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| userID | NOT NULL | varchar2(10) | User ID | Yes | No |
| firstName | NOT NULL | varchar2(75) | User’s first name | No | No |
| lastName | NOT NULL | varchar2(75) | User’s last name | No | No |
| email | NOT NULL | varchar2(40) | User’s email address | No | No |
| phone | NOT NULL | char(12) | User’s phone number | No | No |
| street | NOT NULL | varchar2(40) | Street | No | No |
| city | NOT NULL | varchar2(20) | City | No | No |
| state | NOT NULL | char(2) | State | No | No |
| country | NOT NULL | varchar2(12) | Country | No | No |
| zip | NOT NULL | char(10) | Zip | No | No |
| creationTime | NOT NULL | date | Time of creation | No | No |
| role | NOT NULL | char(5) | "Admin”, “Staff”, “Guest” | No | No |
| lastLoggedIn | NOT NULL | date | Last log in time | No | No |

Guests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| userID | NOT NULL | varchar2(10) | User ID | Yes | Yes |
| guestID | NOT NULL | varchar2(10) | Guest ID | Yes | No |
| checkinTimestamp | NULL | date | Check-in time and date | No | No |
| checkoutTimestamp | NULL | date | Check-in time and date | No | No |
| buildingID | NULL | char(3) | Foreign Key Room(buildingID, roomID) | No | Yes |
| roomID | NULL | char(4) | Foreign Key Room(buildingID, roomID) | No | No |

EmergencyContact

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| userID | NOT NULL | varchar2(10) | User ID | Yes | Yes |
| guestID | NOT NULL | varchar2(10) | Guest ID | Yes | Yes |
| contactID | NOT NULL | varchar2(12) | Contact ID | Yes | No |
| firstName | NOT NULL | varchar2(75) | Emergency contact first name | No | No |
| lastName | NOT NULL | varchar2(75) | Emergency contact’s last name | No | No |
| email | NULL | varchar2(40) | Emergency contact ‘s email | No | No |
| phone | NOT NULL | char(12) | Emergency contact ‘s phone | No | No |

Staffs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| userID | NOT NULL | varchar2(10) | User ID | Yes | Yes |
| staffID | NOT NULL | varchar2(10) | Staff ID | Yes | No |
| shift | NULL | varchar2(20) | “morning”, “afternoon, “evening” | No | No |

Supplier

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| supplierID | NOT NULL | varchar2(8) | Supplier ID | Yes | No |
| name | NOT NULL | varchar2(75) | Supplier name | No | No |
| email | NULL | varchar2(40) | Supplier email | No | No |
| phone | NOT NULL | char(12) | Supplier phone | No | No |
| street | NULL | varchar2(40) | Street | No | No |
| city | NULL | varchar2(20) | City | No | No |
| state | NULL | char(2) | State | No | No |
| country | NULL | varchar2(12) | Country | No | No |
| zip | NULL | char(10) | Zip | No | No |

Robot

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| robotID | NOT NULL | varchar2(5) | Robot ID | Yes | No |
| rModel | NOT NULL | varchar2(15) | Robot model | No | No |
| manufacturer | NOT NULL | varchar2(20) | Robot manufacturer | No | No |
| weight | NOT NULL | number(12,2) | Robot weight | No | No |
| weightCapacity | NOT NULL | number(12,2) | Maximum weight capacity | No | No |
| volumeType | NOT NULL | char(1) | “S”, “M”, “L” | No | No |
| batteryStatus | NOT NULL | number(3) | Battery status between 0 to 100 | No | No |
| buildingBuildingID | NOT NULL | char(3) | Building ID | No | No |
| floorNum | NULL | number(2) | Floor number | No | No |
| elevatorBuildingID | NULL | char(3) | Building ID | No | No |
| elevatorID | NULL | varchar2(3) | Elevator ID | No | Yes |
| stationBuildingID | NULL | char(3) | Foreign Key ChargingStation(buildingID, stationID) | No | Yes |
| stationID | NULL | varchar2(3) | Foreign Key ChargingStation(buildingID, stationID | No | Yes |

Package

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| packageID | NOT NULL | varchar2(13) | Package ID | Yes | No |
| type | NOT NULL | varchar2(15) | Type of package e.g. “Clothes”, “Food”, “Electronics” | No | No |
| weight | NOT NULL | number(12,2) | Package weight in kilograms | No | No |
| volumeType | NOT NULL | char(1) | “S”, “M”, “L” | No | No |
| isFragile | NOT NULL | char(1) | “Y”, “N” | No | No |
| guestUserID | NOT NULL | varchar2(10) | Refers to userID from Guest – Composite Foreign Key from Guest(userID, guestID) | No | Yes |
| guestID | NOT NULL | varchar2(10) | Refers to guestID from Guest – Composite Foreign Key from Guest(userID, guestID) | No | Yes |
| staffUserID | NOT NULL | varchar2(10) | Refers to userID from Staff – Composite Foreign Key from Users(userID, staffID) | No | Yes |
| staffID | NOT NULL | varchar2(10) | Users(userID, staffID) Staff ID – Composite Foreign Key from Users(userID, staffID) | No | Yes |
| supplierID | NOT NULL | varchar2(8) | Supplier ID | No | Yes |
| robotID | NULL | varchar2(5) | Robot ID | No | Yes |

Delivery

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| deliveryID | NOT NULL | varchar2(13) | Primary Key | Yes | No |
| type | NOT NULL | varchar2(10) | Subject responsible for making delivery e.g. “Staff”, “Robot” | No | No |
| status | NOT NULL | varchar2(20) | “Delivered”, “In progress”, “Not delivered” | No | No |
| currentLocation | NOT NULL | varchar2(20) | Current location of package | No | No |
| arrivalTime | NOT NULL | date | Arrival date and time of package at the hotel | No | No |
| estimatedDeliveryTime | NOT NULL | date | Estimated date and time of delivery | No | No |
| deliveredTime | NULL | date | Date and time package was delivered | No | No |
| guestFeedback | NULL | varchar2(50) | Guest feedback | No | No |
| guestRating | NULL | Number(1) | Guest rating e.g. 1, 2, 3, 4, 5 | No | No |
| packageID | NOT NULL | varchar2(13) | Package ID | No | Yes |

Notification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| notificationID | NOT NULL | varchar2(15) | Notification ID | Yes | No |
| creationTime | NOT NULL | date | Time notification is created | No | No |
| type | NOT NULL | varchar2(10) | Notification type e.g. “alert”, “status” | No | No |
| content | NOT NULL | varchar2(50) | Notification content | No | No |
| status | NOT NULL | varchar2(20) | Notification status e.g. “read”, “unread” | No | No |
| deliveryID | NOT NULL | varchar2(12) | Delivery ID | No | Yes |

UserNotification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Column | NULL? | Type | Comments | PK | FK |
| userID | NOT NULL | varchar2(10) | Partial Primary Key –User ID | Yes | Yes |
| notificationID | NOT NULL | varchar2(15) | Partial Primary Key – User ID | Yes | Yes |

# 6. Database Implementation

## 6.1 CREATE TABLE Commands

|  |
| --- |
| CREATE TABLE building (  buildingID char(3) not null,  buildingName varchar2(30) not null,  constraint buildingPK primary key (buildingID)  ); |

|  |
| --- |
| CREATE TABLE room (  buildingID char(3) not null,  roomID char(4) not null,  roomNumber number(3) not null,  area number(4) not null,  beds number(2) not null,  maxOccupancy number(2) not null,  floor number(2) not null,  constraint roomPK primary key (buildingID, roomID),  constraint roomFKbuilding foreign key (buildingID) references building(buildingID)  ); |

|  |
| --- |
| CREATE TABLE elevator (  buildingID char(3) not null,  elevatorID varchar2(3) not null,  weightCapacity number(8, 2) not null,  maxRobots number(2) not null,  sizeType char(1) check (sizeType in ('S','M', 'L' )),  totalWeight number(8, 2) not null,  constraint elevatorPK primary key (buildingID, elevatorID),  constraint elevatorFKbuilding foreign key (buildingID) references building(buildingID)  ); |

|  |
| --- |
| CREATE TABLE chargingStation (  buildingID char(3) not null,  stationID varchar2(3) not null,  powerRating number(8, 2) not null,  operationalStatus char(1) check (operationalStatus in ('O', 'R')),  constraint chargingStationPK primary key (buildingID, stationId),  constraint chargingStationFKbuilding foreign key (buildingID) references building(buildingID)  ); |

|  |
| --- |
| CREATE TABLE users (  userID varchar2(10) not null,  firstName varchar2(75) not null,  lastName varchar2(75) not null,  email varchar2(40) not null,  phone char(12) not null,  street varchar2(40) not null,  city varchar2(20) not null,  state char(2) not null,  country varchar2(12) not null,  zip char(10) not null,  creationTime date not null,  lastLoggedIn date not null,  role char(5) check (role in ('Admin', 'Staff', 'Guest')),  constraint userPK primary key (userID)  ); |

|  |
| --- |
| CREATE TABLE guests (  userID varchar2(10) not null,  guestID varchar2(10) not null,  checkinTimestamp date null,  checkoutTimestamp date null,  buildingID char(3) null,  roomID char(4) null,  constraint guestPK primary key (userID, guestID),  constraint guestFKuser foreign key (userID) references users(userID),  constraint guestFKroom foreign key (buildingID, roomID) references room(buildingID, roomID)  ); |

|  |
| --- |
| CREATE TABLE emergencyContact (  userID varchar2(10) not null,  guestID varchar2(10) not null,  contactID varchar2(12) not null,  firstName varchar2(75) not null,  lastName varchar2(75) not null,  email varchar2(40) null,  phone char(12) not null,  constraint emergencycontactPK primary key (userID, guestID, contactID),  constraint emergencycontactFKguest foreign key (userID, guestID) references guests(userID, guestID) ON DELETE CASCADE  ); |

|  |
| --- |
| CREATE TABLE staffs (      userID varchar2(10) not null,      staffID varchar2(10) not null,      shift varchar2(20) null,      constraint staffPK primary key (userID, staffID),      constraint staffFKuser foreign key (userID) references users(userID) ON DELETE CASCADE  ); |

|  |
| --- |
| CREATE TABLE supplier (  supplierID varchar2(8) not null,  name varchar2(75) not null,  email varchar2(40) null,  phone char(12) not null,  street varchar2(40) null,  city varchar2(20) null,  state char(2) null,  country varchar2(12) null,  zip char(10) null,  constraint supplierPK primary key (supplierID)  ); |

|  |
| --- |
| CREATE TABLE robots (  robotID varchar2(5) not null,  rModel varchar2(15) not null,  manufacturer varchar2(20) not null,  weight number(12, 2) not null,  weightCapacity number(12, 2) not null,  volumeType char(1) check (volumeType in ('S', 'M', 'L')),  batteryStatus number(3) not null,  buildingBuildingID char(3) not null,  floorNum number(2) null,  elevatorBuildingID char(3) null,  elevatorID varchar2(3) null,  stationBuildingID char(3) null,  stationID varchar2(3) null,  constraint robotsPK primary key (robotID),  constraint robotsFKelevator foreign key (elevatorBuildingID, elevatorID) references elevator(buildingID, elevatorID),  constraint robotsFKchargingStation foreign key (stationBuildingID, stationID) references chargingStation(buildingID, stationID),  constraint robotsFKbuilding foreign key (buildingBuildingID) references building(buildingID)  ); |

|  |
| --- |
| CREATE TABLE package (  packageID varchar2(13) not null,  type varchar2(15) not null,  weight number(12, 2) not null,  volumeType char(1) check (volumeType in ('S', 'M', 'L')),  isFragile char(1) check ( isFragile in ('Y', 'N')),  guestUserID varchar2(10) not null,  guestID varchar2(10) not null,  staffUserID varchar2(10) not null,  staffID varchar2(10) not null,  supplierID varchar2(8) not null,  robotID varchar2(5) null,  constraint packagePK primary key (packageID),  constraint packageFKguest foreign key (guestUserID, guestID) references guests(userID, guestID) ON DELETE CASCADE,  constraint packageFKstaff foreign key (staffUserID, staffID) references staffs(userID, staffID),  constraint packageFKsupplier foreign key (supplierID) references supplier(supplierID),  constraint packageFKrobot foreign key (robotID) references robots(robotID)  ); |

|  |
| --- |
| CREATE TABLE delivery (  deliveryID varchar2(13) not null,  type varchar2(10) check (type in ('Staff', 'Robot')),  status varchar2(20) check (status in ('Delivered', 'In progress', 'Not delivered')),  currentLocation varchar2(20) not null,  arrivalTime date not null,  estimatedDeliverytime date not null,  deliveredTime date null,  guestFeedback varchar2(50) null,  guestRating number(1) null,  packageID varchar2(13) not null,  constraint deliveryPK primary key (deliveryID),  constraint deliveryFKpackage foreign key (packageID) references package(packageID) ON DELETE CASCADE  ); |
|  |

|  |
| --- |
|  |
| CREATE TABLE notification (  notificationID varchar2(15) not null,  creationTime date not null,  type varchar2(10) not null check (type in ('alert', 'status')),  content varchar2(50) not null,  status varchar2(20) not null,  deliveryID varchar2(12) not null,  constraint notificationPK primary key (notificationID),  constraint notificationFKdelivery foreign key (deliveryID) references delivery(deliveryID) ON DELETE CASCADE  ); |

|  |
| --- |
| CREATE TABLE usernotification (  userID varchar2(10) not null,  notificationID varchar2(15) not null,  constraint usernotificationPK primary key (userID, notificationID),  constraint usernotificationFKuser foreign key (userID) references users(userID) ON DELETE CASCADE,  constraint usernotificationFKnotification foreign key (notificationID) references notification(notificationID)  ); |

## 6.2 INSERT INTO Commands

Building

|  |
| --- |
| -- Building  INSERT INTO building VALUES ('B01', 'Main Building');  INSERT INTO building VALUES ('B02', 'Annex Building');  INSERT INTO building VALUES ('B03', 'East Wing');  INSERT INTO building VALUES ('B04', 'West Wing'); |

Room

|  |
| --- |
| -- Room  -- Building 1  INSERT INTO room VALUES ('B01', 'R101', 101, 200, 3, 2, 1);  INSERT INTO room VALUES ('B01', 'R201', 201, 200, 4, 5, 2);  INSERT INTO room VALUES ('B01', 'R202', 202, 100, 2, 4, 2);  INSERT INTO room VALUES ('B01', 'R203', 203, 200, 2, 1, 2);  INSERT INTO room VALUES ('B01', 'R301', 301, 150, 1, 2, 3);  INSERT INTO room VALUES ('B01', 'R302', 302, 200, 1, 1, 3);  INSERT INTO room VALUES ('B01', 'R401', 401, 150, 3, 5, 4);  INSERT INTO room VALUES ('B01', 'R402', 402, 200, 4, 4, 4); |
| -- Building 2  INSERT INTO room VALUES ('B02', 'R101', 101, 150, 4, 6, 1);  INSERT INTO room VALUES ('B02', 'R201', 201, 300, 3, 3, 2);  INSERT INTO room VALUES ('B02', 'R401', 401, 150, 2, 1, 4);  INSERT INTO room VALUES ('B02', 'R402', 402, 300, 1, 1, 4);  -- Building 3  INSERT INTO room VALUES ('B03', 'R101', 101, 100, 1, 3, 1);  INSERT INTO room VALUES ('B03', 'R102', 102, 100, 2, 2, 1);  INSERT INTO room VALUES ('B03', 'R103', 103, 100, 1, 1, 1);  INSERT INTO room VALUES ('B03', 'R301', 301, 200, 1, 1, 3);  INSERT INTO room VALUES ('B03', 'R302', 302, 100, 1, 2, 3);  INSERT INTO room VALUES ('B03', 'R303', 303, 200, 1, 2, 3);  INSERT INTO room VALUES ('B03', 'R401', 401, 150, 1, 1, 4);  INSERT INTO room VALUES ('B03', 'R501', 501, 200, 3, 4, 5);  INSERT INTO room VALUES ('B03', 'R502', 502, 200, 2, 1, 5);  INSERT INTO room VALUES ('B03', 'R503', 503, 200, 1, 1, 5);  INSERT INTO room VALUES ('B03', 'R701', 701, 200, 3, 4, 7);  INSERT INTO room VALUES ('B03', 'R702', 702, 200, 3, 2, 7);  INSERT INTO room VALUES ('B03', 'R801', 801, 150, 3, 3, 8);  -- Building 4  INSERT INTO room VALUES ('B04', 'R101', 101, 300, 3, 5, 1);  INSERT INTO room VALUES ('B04', 'R102', 102, 200, 4, 4, 1);  INSERT INTO room VALUES ('B04', 'R103', 103, 200, 2, 3, 1);  INSERT INTO room VALUES ('B04', 'R201', 201, 300, 2, 1, 2);  INSERT INTO room VALUES ('B04', 'R401', 401, 200, 2, 4, 4);  INSERT INTO room VALUES ('B04', 'R402', 402, 100, 2, 4, 4);  INSERT INTO room VALUES ('B04', 'R403', 403, 300, 1, 2, 4);  INSERT INTO room VALUES ('B04', 'R501', 501, 150, 2, 1, 5);  INSERT INTO room VALUES ('B04', 'R502', 502, 300, 1, 2, 5); |

Elevator

|  |
| --- |
| -- Elevator  -- Building 1  INSERT INTO elevator VALUES ('B01', 'E01', 300, 4, 'S', 100);  INSERT INTO elevator VALUES ('B01', 'E02', 950, 8, 'M', 200);  INSERT INTO elevator VALUES ('B01', 'E03', 1550, 12, 'L', 300);  INSERT INTO elevator VALUES ('B01', 'E04', 1000, 10, 'L', 150);  -- Building 2  INSERT INTO elevator VALUES ('B02', 'E01', 350, 4, 'S', 0);  INSERT INTO elevator VALUES ('B02', 'E02', 850, 8, 'M', 100);  INSERT INTO elevator VALUES ('B02', 'E03', 1250, 11, 'L', 50);  INSERT INTO elevator VALUES ('B02', 'E04', 1300, 11, 'L', 120); |
| -- Building 3  INSERT INTO elevator VALUES ('B03', 'E01', 300, 3, 'S', 0);  INSERT INTO elevator VALUES ('B03', 'E02', 750, 8, 'M', 400);  INSERT INTO elevator VALUES ('B03', 'E03', 1150, 12, 'L', 300);  INSERT INTO elevator VALUES ('B03', 'E04', 1200, 13, 'L', 250);  -- Building 4  INSERT INTO elevator VALUES ('B04', 'E01', 300, 3, 'S', 150);  INSERT INTO elevator VALUES ('B04', 'E02', 950, 8, 'M', 100);  INSERT INTO elevator VALUES ('B04', 'E03', 1450, 13, 'L', 150);  INSERT INTO elevator VALUES ('B04', 'E04', 1100, 12, 'L', 230); |

Charging Station

|  |
| --- |
| -- Charging Station  -- Building 1  INSERT INTO chargingStation VALUES ('B01', 'S01', 100, 'O');  INSERT INTO chargingStation VALUES ('B01', 'S02', 150, 'R');  INSERT INTO chargingStation VALUES ('B01', 'S03', 200, 'O');  INSERT INTO chargingStation VALUES ('B01', 'S04', 250, 'O');  -- Building 2  INSERT INTO chargingStation VALUES ('B02', 'S01', 100, 'O');  INSERT INTO chargingStation VALUES ('B02', 'S02', 150, 'O');  INSERT INTO chargingStation VALUES ('B02', 'S03', 200, 'O');  INSERT INTO chargingStation VALUES ('B02', 'S04', 250, 'O');  -- Building 3  INSERT INTO chargingStation VALUES ('B03', 'S01', 100, 'O');  INSERT INTO chargingStation VALUES ('B03', 'S02', 150, 'R');  INSERT INTO chargingStation VALUES ('B03', 'S03', 200, 'O');  INSERT INTO chargingStation VALUES ('B03', 'S04', 250, 'O');  -- Building 4  INSERT INTO chargingStation VALUES ('B04', 'S01', 100, 'O');  INSERT INTO chargingStation VALUES ('B04', 'S02', 150, 'O');  INSERT INTO chargingStation VALUES ('B04', 'S03', 200, 'O');  INSERT INTO chargingStation VALUES ('B04', 'S04', 250, 'O'); |

Users

|  |
| --- |
| -- Users  INSERT INTO users VALUES ('U001', 'Jane', 'Smith', 'jane.smith@example.com', '019-555-5678', '456 Oak St', 'Townsville', 'NY', 'USA', '67890', TO\_DATE('2022-12-05', 'YYYY-MM-DD'), TO\_DATE('2022-12-24', 'YYYY-MM-DD'), 'Guest');  INSERT INTO users VALUES ('U002', 'John', 'Doe', 'john.doe@example.com', '123-555-1234', '123 Main St', 'Cityville', 'CA', 'USA', '12345', TO\_DATE('2023-01-08', 'YYYY-MM-DD'), TO\_DATE('2023-01-22', 'YYYY-MM-DD'), 'Guest');  INSERT INTO users VALUES ('U003', 'Emily', 'Johnson', 'emily.johnson@example.com', '231-555-1234', '456 Oak St', 'Cityville', 'CA', 'USA', '98765', TO\_DATE('2012-05-15', 'YYYY-MM-DD'), TO\_DATE('2012-05-15', 'YYYY-MM-DD'), 'Guest'); |
| INSERT INTO users VALUES ('U004', 'Sophia', 'Williams', 'sophia.williams@example.com', '555-987-6543', '789 Maple Ave', 'Suburbia', 'TX', 'USA', '54321', TO\_DATE('2023-11-28', 'YYYY-MM-DD'), TO\_DATE('2023-11-28', 'YYYY-MM-DD'), 'Guest');  INSERT INTO users VALUES ('U009', 'Bob', 'Johnson', 'bob.johnson@example.com', '903-555-9876', '789 Pine St', 'Villagetown', 'TX', 'USA', '54321', TO\_DATE('2023-09-20', 'YYYY-MM-DD'), TO\_DATE('2023-11-25', 'YYYY-MM-DD'), 'Guest');  INSERT INTO users VALUES ('U005', 'George', 'Clark', 'george.clark@example.com', '015-555-1111', '234 Redwood St', 'Hometown', 'WA', 'USA', '87654', TO\_DATE('2022-10-19', 'YYYY-MM-DD'), TO\_DATE('2023-11-28', 'YYYY-MM-DD'), 'Staff');  INSERT INTO users VALUES ('U006', 'Fiona', 'Roberts', 'fiona.roberts@example.com', '340-555-4444', '876 Elm St', 'Ruraltown', 'KS', 'USA', '34567', TO\_DATE('2021-06-25', 'YYYY-MM-DD'), TO\_DATE('2023-11-28', 'YYYY-MM-DD'), 'Staff');  INSERT INTO users VALUES ('U007', 'David', 'Hill', 'david.hill@example.com', '591-555-7777', '567 Pine St', 'Outskirts', 'NE', 'USA', '76543', TO\_DATE('2020-04-09', 'YYYY-MM-DD'), TO\_DATE('2023-11-15', 'YYYY-MM-DD'), 'Admin');  INSERT INTO users VALUES ('U008', 'Grace', 'Miller', 'grace.miller@example.com', '945-555-9999', '123 Cedar St', 'Countryside', 'GA', 'USA', '98765', TO\_DATE('2019-01-10', 'YYYY-MM-DD'), TO\_DATE('2023-11-20', 'YYYY-MM-DD'), 'Admin'); |

Guests

|  |
| --- |
| -- Guests  INSERT INTO guests VALUES ('U001', 'G001', TO\_DATE('2022-12-10 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2022-12-16 11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'B01', 'R201');  INSERT INTO guests VALUES ('U002', 'G002', TO\_DATE('2023-01-15 12:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-01-20 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'B02', 'R401');  INSERT INTO guests VALUES ('U003', 'G003', TO\_DATE('2012-05-09 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2012-05-14 14:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'B03', 'R702');  INSERT INTO guests VALUES ('U004', 'G004', null, null, null, null);  INSERT INTO guests VALUES ('U009', 'G005', TO\_DATE('2023-11-10 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-11-30 19:18:00', 'YYYY-MM-DD HH24:MI:SS'), 'B04', 'R502'); |

EmergencyContact

|  |
| --- |
| -- EmergencyContact  INSERT INTO emergencyContact VALUES ('U001', 'G001', 'C001', 'Sarah', 'Johnson', 'sarah.johnson@example.com', '123-555-2222');  INSERT INTO emergencyContact VALUES ('U001', 'G001', 'C002', 'Michael', 'Smith', 'michael.smith@example.com', '123-555-3333');  INSERT INTO emergencyContact VALUES ('U002', 'G002', 'C003', 'Olivia', 'Brown', 'olivia.brown@example.com', '123-555-4444');  INSERT INTO emergencyContact VALUES ('U003', 'G003', 'C004', 'Daniel', 'Williams', 'daniel.williams@example.com', '123-555-5555');  INSERT INTO emergencyContact VALUES ('U004', 'G004', 'C005', 'Emma', 'Davis', 'emma.davis@example.com', '123-555-6666');  INSERT INTO emergencyContact VALUES ('U004', 'G004', 'C006', 'Liam', 'Anderson', 'liam.anderson@example.com', '123-555-7777');  INSERT INTO emergencyContact VALUES ('U009', 'G005', 'C007', 'James', 'Bond', 'james.bond@myemail.com', '903-555-8888'); |

Staff

|  |
| --- |
| -- Staff  INSERT INTO staffs VALUES ('U005', 'S001', 'morning');  INSERT INTO staffs VALUES ('U006', 'S002', 'afternoon');  INSERT INTO staffs VALUES ('U007', 'S003', 'afternoon');  INSERT INTO staffs VALUES ('U008', 'S004', 'evening'); |

Supplier

|  |
| --- |
| -- Supplier  INSERT INTO supplier VALUES ('SUP001', 'ABC Electronics', 'contactus@abc.com', '123-555-1111', '7936 Dunbar Ave.', 'Ellenwood', 'GA', 'USA', '30294');  INSERT INTO supplier VALUES ('SUP002', 'XYZ Logistics', 'help@xyzlogistics.com', '123-666-7088', '456 Supply Ave', 'Supplytown', 'TX', 'USA', '73412');  INSERT INTO supplier VALUES ('SUP003', 'Fashion Emporium', 'info@fashemporium.com', '345-790-2333', '259 Kirkland St.', 'San Lorenzo', 'CA', 'USA', '94580');  INSERT INTO supplier VALUES ('SUP004', 'Home Essentials Co.', 'questions@homeessentials.com', '364-215-9852', '8746 8th Street', 'Port Orange', 'FL', 'USA', '32127');  INSERT INTO supplier VALUES ('SUP005', 'Gadget Haven', 'info@gadgethaven.com', '901-241-5893', '867 Smith Ave.', 'Roy', 'UT', 'USA', '84067');  INSERT INTO supplier VALUES ('SUP006', 'Amazing Prints Inc.', 'info@amazingprints.com', '412-809-2357', '649 Buttonwood St', 'Brainerd', 'MN', 'USA', '56401');  INSERT INTO supplier VALUES ('SUP007', 'Food Restaurant', 'info@foodrestaurant.com', '312-360-1268', '9625 NW. Division Road', 'Winter Garden', 'FL', 'USA', '34787'); |

Robot

|  |
| --- |
| -- Robot  -- Building 1  INSERT INTO robots VALUES ('RB001', 'Model A', 'TechRobotics', 20, 5, 'S', 90, 'B01', 3, null, null, null, null);  INSERT INTO robots VALUES ('RB002', 'Model W', 'TechRobotics', 50, 20, 'M', 60, 'B01', 10, null, null, null, null);  INSERT INTO robots VALUES ('RB003', 'Model X', 'TechRobotics', 100, 50, 'L', 75, 'B01', 0, null, null, null, null);    -- Building 2  INSERT INTO robots VALUES ('RB004', 'Model A', 'TechRobotics', 20, 5, 'S', 75, 'B02', null, 'B02', 'E01', null, null);  INSERT INTO robots VALUES ('RB005', 'Model W', 'TechRobotics', 50, 20, 'M', 100, 'B02', 2, null, null, null, null);  INSERT INTO robots VALUES ('RB006', 'Model X', 'TechRobotics', 100, 50, 'L', 80, 'B02', 4, null, null, null, null);    -- Building 3  INSERT INTO robots VALUES ('RB007', 'Model A', 'TechRobotics', 20, 5, 'S', 100, 'B03', 0, null, null, null, null);  INSERT INTO robots VALUES ('RB008', 'Model W', 'TechRobotics', 50,20, 'M', 75, 'B03', 0, null, null, null, null);  INSERT INTO robots VALUES ('RB009', 'Model X', 'TechRobotics', 100, 50, 'L', 90, 'B03', null, null, null, 'B03', 'S01'); |
| -- Building 4  INSERT INTO robots VALUES ('RB010', 'Model A', 'TechRobotics', 20, 5, 'S', 100, 'B04', 1, null, null, null, null);  INSERT INTO robots VALUES ('RB011', 'Model W', 'TechRobotics', 50, 20, 'M', 80, 'B04', 5, null, null, null, null);  INSERT INTO robots VALUES ('RB012', 'Model X', 'TechRobotics', 100, 50, 'L', 20, 'B04', null, null, null, 'B04', 'S02'); |

Package

|  |
| --- |
| -- Package  INSERT INTO package VALUES ('P001', 'Clothes', 0.8, 'S', 'N', 'U001', 'G001', 'U005', 'S001', 'SUP003', 'RB001');  INSERT INTO package VALUES ('P002', 'Food', 2.25, 'S', 'N', 'U001', 'G001', 'U005', 'S001', 'SUP007', null);  INSERT INTO package VALUES ('P003', 'Electronics', 20.91, 'L', 'N', 'U002', 'G002', 'U005', 'S001', 'SUP001', 'RB006');  INSERT INTO package VALUES ('P004', 'Items', 15.45, 'M', 'Y', 'U002', 'G002', 'U005', 'S001', 'SUP004', null);  INSERT INTO package VALUES ('P005', 'Items', 10.45, 'M', 'N', 'U003', 'G003', 'U006', 'S002', 'SUP004', 'RB008');  INSERT INTO package VALUES ('P006', 'Electronics', 20.51, 'L', 'Y', 'U003', 'G003', 'U006', 'S002', 'SUP001', 'RB009');  INSERT INTO package VALUES ('P007', 'Clothes', 5.8, 'S', 'N', 'U009', 'G005', 'U007', 'S003', 'SUP003', 'RB010');  INSERT INTO package VALUES ('P008', 'Food', 1.2, 'S', 'N', 'U009', 'G005', 'U007', 'S003', 'SUP004', 'RB011'); |

Delivery

|  |
| --- |
| -- Delivery  INSERT INTO delivery VALUES ('D001', 'Robot', 'Delivered', 'Delivered', TO\_DATE('2022-12-13 15:05:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2022-12-14 12:23:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2022-12-13 20:06:00', 'YYYY-MM-DD HH24:MI:SS'), 'Delivery was perfect!', 5, 'P001');  INSERT INTO delivery VALUES ('D002', 'Staff', 'Delivered', 'Delivered', TO\_DATE('2022-12-11 14:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2022-12-11 15:15:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2022-12-13 3:00:00', 'YYYY-MM-DD HH24:MI:SS'), null, 3, 'P002'); |
| INSERT INTO delivery VALUES ('D003', 'Robot', 'Delivered', 'Delivered', TO\_DATE('2023-01-17 18:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-01-16 17:55:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-01-17 20:30:00', 'YYYY-MM-DD HH24:MI:SS'), null, 4, 'P003');  INSERT INTO delivery VALUES ('D004', 'Staff', 'Delivered', 'Delivered', TO\_DATE('2023-01-19 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-03-20 10:05:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-01-19 15:00:00', 'YYYY-MM-DD HH24:MI:SS'), null, 3, 'P004');  INSERT INTO delivery VALUES ('D005', 'Robot', 'Delivered', 'Delivered', TO\_DATE('2012-05-10 18:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2012-05-10 17:55:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2012-05-10 20:30:00', 'YYYY-MM-DD HH24:MI:SS'), null, 5, 'P005');  INSERT INTO delivery VALUES ('D006', 'Robot', 'Delivered', 'Delivered', TO\_DATE('2012-05-14 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2012-05-13 10:05:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2012-05-14 10:30:00', 'YYYY-MM-DD HH24:MI:SS'), null, 3, 'P006');  INSERT INTO delivery VALUES ('D007', 'Robot', 'Delivered', 'Delivered', TO\_DATE('2023-11-15 18:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-11-16 17:55:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-11-15 20:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'Excellent service', 4, 'P007');  INSERT INTO delivery VALUES ('D008', 'Robot', 'In progress', 'Elevator', TO\_DATE('2023-11-29 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO\_DATE('2023-11-28 10:05:00', 'YYYY-MM-DD HH24:MI:SS'), null, null, null, 'P008'); |

Notification

|  |
| --- |
| -- notification  -- user 1  INSERT INTO notification VALUES ('N001', TO\_DATE('2022-12-09 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your package is awaiting confirmation', 'unread', 'D001');  INSERT INTO notification VALUES ('N002', TO\_DATE('2022-12-10 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your delivery is delayed due to weather conditions', 'unread', 'D001');  INSERT INTO notification VALUES ('N003', TO\_DATE('2022-12-10 12:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'unread', 'D001');  INSERT INTO notification VALUES ('N004', TO\_DATE('2022-12-10 22:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'unread', 'D001');  INSERT INTO notification VALUES ('N005', TO\_DATE('2022-12-14 12:10:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'read', 'D001');    INSERT INTO notification VALUES ('N006', TO\_DATE('2022-12-10 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your delivery is delayed due to weather conditions', 'unread', 'D002');  INSERT INTO notification VALUES ('N007', TO\_DATE('2022-12-11 12:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'read', 'D002');  INSERT INTO notification VALUES ('N008', TO\_DATE('2022-12-12 09:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'unread', 'D002');  INSERT INTO notification VALUES ('N009', TO\_DATE('2022-12-19 03:05:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'read', 'D002'); |
| -- user 2  INSERT INTO notification VALUES ('N010', TO\_DATE('2023-01-13 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your package is awaiting confirmation', 'unread', 'D003');  INSERT INTO notification VALUES ('N011', TO\_DATE('2023-01-15 10:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your delivery is delayed due to weather conditions', 'unread', 'D003');  INSERT INTO notification VALUES ('N012', TO\_DATE('2023-01-17 12:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'red', 'D003');  INSERT INTO notification VALUES ('N013', TO\_DATE('2023-01-17 18:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'unread', 'D003');  INSERT INTO notification VALUES ('N014', TO\_DATE('2023-01-17 21:10:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'read', 'D003');  INSERT INTO notification VALUES ('N015', TO\_DATE('2023-01-18 01:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'read', 'D004');  INSERT INTO notification VALUES ('N016', TO\_DATE('2023-01-19 12:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'read', 'D004');  INSERT INTO notification VALUES ('N017', TO\_DATE('2023-01-19 15:01:10', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'unread', 'D004');  -- user 3  INSERT INTO notification VALUES ('N018', TO\_DATE('2012-05-09 01:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'read', 'D005');  INSERT INTO notification VALUES ('N019', TO\_DATE('2012-05-10 19:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'unread', 'D005');  INSERT INTO notification VALUES ('N020', TO\_DATE('2012-05-19 20:35:10', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'read', 'D005');  INSERT INTO notification VALUES ('N021', TO\_DATE('2012-05-12 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your package is awaiting confirmation', 'unread', 'D006');  INSERT INTO notification VALUES ('N022', TO\_DATE('2012-05-13 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your delivery is delayed due to weather conditions', 'unread', 'D006');  INSERT INTO notification VALUES ('N023', TO\_DATE('2012-05-14 01:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'unread', 'D006');  INSERT INTO notification VALUES ('N024', TO\_DATE('2012-05-14 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'read', 'D006');  INSERT INTO notification VALUES ('N025', TO\_DATE('2012-05-14 10:35:10', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'read', 'D006');  -- user 9  INSERT INTO notification VALUES ('N026', TO\_DATE('2023-11-14 08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your package is awaiting confirmation', 'unread', 'D007');  INSERT INTO notification VALUES ('N027', TO\_DATE('2023-11-14 10:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your delivery is delayed due to weather conditions', 'unread', 'D007');  INSERT INTO notification VALUES ('N028', TO\_DATE('2023-11-15 01:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'read', 'D007');  INSERT INTO notification VALUES ('N029', TO\_DATE('2023-11-15 18:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is in the storage room', 'unread', 'D007');  INSERT INTO notification VALUES ('N030', TO\_DATE('2023-11-15 20:31:10', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package has been delivered to your room', 'read', 'D007');  INSERT INTO notification VALUES ('N031', TO\_DATE('2023-11-27 10:25:00', 'YYYY-MM-DD HH24:MI:SS'), 'alert', 'Your delivery is delayed due to weather conditions', 'read', 'D008');  INSERT INTO notification VALUES ('N032', TO\_DATE('2023-11-28 01:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'status', 'Your package is en route to the hotel', 'read', 'D008'); |

UserNotification

|  |
| --- |
| -- UserNotification  -- user 1  INSERT INTO usernotification VALUES ('U001', 'N001');  INSERT INTO usernotification VALUES ('U001', 'N002');  INSERT INTO usernotification VALUES ('U001', 'N003');  INSERT INTO usernotification VALUES ('U001', 'N004');  INSERT INTO usernotification VALUES ('U001', 'N005');  INSERT INTO usernotification VALUES ('U001', 'N006');  INSERT INTO usernotification VALUES ('U001', 'N007');  INSERT INTO usernotification VALUES ('U001', 'N008');  INSERT INTO usernotification VALUES ('U001', 'N009');  -- user 2  INSERT INTO usernotification VALUES ('U002', 'N010');  INSERT INTO usernotification VALUES ('U002', 'N011');  INSERT INTO usernotification VALUES ('U002', 'N012');  INSERT INTO usernotification VALUES ('U002', 'N013');  INSERT INTO usernotification VALUES ('U002', 'N014');  INSERT INTO usernotification VALUES ('U002', 'N015');  INSERT INTO usernotification VALUES ('U002', 'N016');  INSERT INTO usernotification VALUES ('U002', 'N017');  -- user 3  INSERT INTO usernotification VALUES ('U003', 'N018');  INSERT INTO usernotification VALUES ('U003', 'N019');  INSERT INTO usernotification VALUES ('U003', 'N020');  INSERT INTO usernotification VALUES ('U003', 'N021');  INSERT INTO usernotification VALUES ('U003', 'N022');  INSERT INTO usernotification VALUES ('U003', 'N023');  INSERT INTO usernotification VALUES ('U003', 'N024');  INSERT INTO usernotification VALUES ('U003', 'N025');  -- user 9  INSERT INTO usernotification VALUES ('U009', 'N026');  INSERT INTO usernotification VALUES ('U009', 'N027');  INSERT INTO usernotification VALUES ('U009', 'N028');  INSERT INTO usernotification VALUES ('U009', 'N029');  INSERT INTO usernotification VALUES ('U009', 'N030');  INSERT INTO usernotification VALUES ('U009', 'N031');  INSERT INTO usernotification VALUES ('U009', 'N032'); |

## 6.3 Data from SELECT \* FROM your\_table\_name; commands:

|  |
| --- |
| SELECT \* FROM BUILDING; |

A white paper with black text

Description automatically generated

|  |
| --- |
| SELECT \* FROM ROOM; |

A white sheet with numbers and letters

Description automatically generated with medium confidence

**A table with numbers and text

Description automatically generated**

|  |
| --- |
| SELECT \* FROM ELEVATOR; |

**A table with numbers and letters

Description automatically generated**

|  |
| --- |
| SELECT \* FROM CHARGINGSTATION; |

A table with numbers and letters

Description automatically generated

|  |
| --- |
| SELECT \* FROM USERS; |

A screenshot of a computer

Description automatically generated

* Left half of USERS table:

A screenshot of a computer

Description automatically generated

* Right half of USERS table:

A table with numbers and a date

Description automatically generated

|  |
| --- |
| SELECT \* FROM GUESTS; |

**A white table with black text and numbers

Description automatically generated**

|  |
| --- |
| SELECT \* FROM EMERGENCYCONTACT; |

A table of names

Description automatically generated

|  |
| --- |
| SELECT \* FROM STAFFS; |

A white paper with black text

Description automatically generated

|  |
| --- |
| SELECT \* FROM SUPPLIER; |

A screenshot of a computer

Description automatically generated

|  |
| --- |
| SELECT \* FROM ROBOTS; |

**A screenshot of a computer

Description automatically generated**

* Left half of the ROBOTS table:

A table with text on it

Description automatically generated

* Right half of the ROBOTS table:

A white sheet with black text

Description automatically generated

|  |
| --- |
| SELECT \* FROM PACKAGE; |

A white table with black and orange text

Description automatically generated

|  |
| --- |
| SELECT \* FROM DELIVERY; |

A screenshot of a computer

Description automatically generated

|  |
| --- |
| SELECT \* FROM NOTIFICATION; |

A screenshot of a computer

Description automatically generated

A white box with black text

Description automatically generated

|  |
| --- |
| SELECT \* FROM USERNOTIFICATION; |

A screenshot of a computer

Description automatically generated **A screenshot of a computer

Description automatically generated**

# 7. Data queries

## 7.1 Queries by Estifanos Abebe

1. For each staff member that assists with the delivery process, find the minimum, maximum, and average time taken for the package to reach the guests’ room once it arrives at the hotel. Order the output in increasing order of the average time it takes for the package to be received by the guests. This could be used to identify the most efficient staff member.

|  |
| --- |
| SELECT  s.staffID as staff\_id,  u.firstName AS first\_name,  u.lastName AS last\_name,  COUNT(d.deliveryId) as num\_deliveries,  ROUND(MIN(d.deliveredTime - d.arrivalTime)\*24, 2) AS min\_delivery\_time\_hrs,  ROUND(MAX(d.deliveredTime - d.arrivalTime)\*24, 2) AS max\_delivery\_time\_hrs,  ROUND(AVG(d.deliveredTime - d.arrivalTime)\*24, 2) AS avg\_delivery\_time\_hrs  FROM staffs s  LEFT JOIN users u ON s.userId = u.userID  LEFT JOIN package p ON s.staffID = p.staffID AND s.userId = p.staffUserID  LEFT JOIN delivery d ON p.packageID = d.packageID  WHERE d.status = 'Delivered'  GROUP BY s.staffID, u.firstName, u.lastName  ORDER BY AVG(d.deliveredTime - d.arrivalTime); |



1. A new guest has booked room number “R401” on building “West Wing”. A new package that weighs 4.5lb has arrived at the hotel for this guest. Up on inspection, the package’s volume was determined to be small. List all the robots that are available for delivery. Sort them in an increasing order of the distance the robots would have to travel to complete the delivery.

|  |
| --- |
| SELECT  rb.robotId AS robot\_id,  rb.rModel AS robot\_model,  rb.weightCapacity AS weight\_capacity,  rb.volumeType AS volume\_type,  rb.batteryStatus AS battery\_status,  rb.buildingBuildingID AS building\_id,  rb.floorNum AS floor\_num,  b.BUILDINGNAME AS building\_name  FROM robots rb, building b  WHERE  rb.buildingBuildingID = b.buildingID  AND b.buildingName = 'West Wing'  AND rb.batteryStatus > 25 -- battery status constraint  AND rb.volumeType IN ('S', 'M', 'L') -- volume constraint  AND rb.weightCapacity > 4.5 -- weight constraint  AND rb.stationID IS NULL -- robot must not be docked to charging Station  AND rb.robotID NOT IN (  SELECT rb.robotId  FROM delivery d  JOIN package p ON d.packageID = p.packageID  JOIN robots rb ON rb.robotID = p.robotID  JOIN building b ON rb.buildingBuildingID = b.buildingID  WHERE  b.buildingName = 'West Wing'  AND d.type = 'Robot'  AND d.status = 'In progress'  ) -- robots that are currently in the process of delivering a package  ORDER BY rb.floorNum ASC; -- robot closer to the 1st floor is listed first |



* Since only one robot satisfies the constraints, there is only one row in the output.
* The following additional assumptions were made in the query:
  + A robot needs to have at least 25% of battery status to be eligible for a delivery.
  + A robot that is docked to a charging station is not assumed to be available for a delivery.

1. For each building in the hotel, find out the proportion of the package deliveries made by the staff personnel and robots in percents. Order the output in ascending order of the building name and total deliveries made.

|  |
| --- |
| SELECT  b.buildingID as bld\_id,  b.buildingName AS bld\_name,  COUNT(\*) AS total\_deliveries,  (COUNT(CASE WHEN d.type = 'Staff' THEN 1 END) / COUNT(\*)) \* 100 AS prop\_staff\_deliveries,  (COUNT(CASE WHEN d.type = 'Robot' THEN 1 END) / COUNT(\*)) \* 100 AS prop\_robot\_deliveries  FROM building b  LEFT JOIN room r ON b.buildingID = r.buildingID  LEFT JOIN guests g ON r.roomID = g.roomID AND r.buildingID = g.buildingID  LEFT JOIN package p ON g.guestID = p.guestID AND p.guestUserID = g.USERID  LEFT JOIN delivery d ON d.packageID = p.packageID  WHERE d.status = 'Delivered';  GROUP BY b.buildingID, b.buildingName  ORDER BY b.buildingName, COUNT(\*) |

A white sheet with black text

Description automatically generated

* The values for the columns PROP\_STAFF\_DELIVERIES and PROP\_ROBOT\_DELIVERIES are the percentage of the deliveries made by the staff personnel and robots respectively.

## 7.2 Queries by Chinonso Eziefule

1. Obtain a detailed statistics for each guest, including their guest IDs, user IDs, and information about the suppliers from whom they received packages. For each supplier, display the supplier ID, supplier name, total number of packages received, and the cumulative weight of all packages received by the guest. Ensure that the results are grouped by guest, user, and supplier, and sort the output in ascending order based on the guest ID.

|  |
| --- |
| SELECT  g.guestID,  g.userID,  s.supplierID,  s.name AS supplierName,  COUNT(p.packageID) AS totalPackagesReceived,  SUM(p.weight) AS totalWeightReceived  FROM  guests g  JOIN  package p ON g.userID = p.guestUserID AND g.guestID = p.guestID  JOIN  supplier s ON p.supplierID = s.supplierID  GROUP BY  g.guestID, g.userID, s.supplierID, s.name  ORDER BY  g.guestID ASC; |

A screenshot of a computer program

Description automatically generated

1. Retrieve detailed information about packages delivered by robots. Include package details such as package ID, type, weight, volume type, and fragility status. Additionally, include guest information, specifically the guest's user ID and guest ID. Include robot details, such as robot ID and robot model. Retrieve delivery details, including delivery ID, delivery type, status, current location, arrival time, estimated delivery time, delivered time, guest feedback, and guest rating. Ensure that the delivery type is specified as 'Robot' and the status is 'Delivered'. Organize the results in ascending order based on the estimated delivery time.

|  |
| --- |
| SELECT p.packageid, p.type, p.weight, p.volumetype, p.isfragile, g.guestId, g.userid, r.robotid, r.rmodel, d.deliveryid, d.type, d.status, d.currentlocation, d.arrivaltime, d.estimateddeliverytime, d.deliveredtime, d.guestfeedback, d.guestrating from package p  JOIN delivery d ON d.packageid = p.packageid  JOIN guests g ON g.guestid = p.guestid  JOIN users u ON u.userid = g.userid  JOIN supplier s On s.supplierid = p.supplierid  JOIN robots r ON r.robotID = p.robotid  WHERE d.type = 'Robot' AND d.status = 'Delivered'  ORDER BY d.estimateddeliverytime asc; |

A white background with black and white text

Description automatically generated

* Left half of the result:

A close-up of a computer code

Description automatically generated

* Right half of the result:

A calendar with date and numbers

Description automatically generated

1. Retrieve the details of the guest(s) who checked into the hotel during the last month, including their full names, room information (building ID, room number), check-in and check-out timestamps, and the total number of packages they received. Additionally, display only the guest(s) who received packages during their stay.

|  |
| --- |
| SELECT u.firstName || ' ' || u.lastName AS guestName, r.buildingID, r.roomNumber, g.checkinTimestamp, g.checkoutTimestamp, COUNT(p.packageID) AS totalPackagesReceived  FROM users u  JOIN guests g ON u.userID = g.userID  JOIN room r ON g.buildingID = r.buildingID  AND g.roomID = r.roomID  LEFT JOIN package p ON g.userID = p.guestUserID  AND g.guestID = p.guestID  WHERE g.checkinTimestamp >= ADD\_MONTHS(TRUNC(SYSDATE, 'MONTH'), -1)  AND g.checkinTimestamp < TRUNC(SYSDATE, 'MONTH')  GROUP BY  u.userID, u.firstName, u.lastName, r.buildingID,  r.roomNumber, g.checkinTimestamp, g.checkoutTimestamp  HAVING COUNT(p.packageID) > 0; |

A screenshot of a computer code

Description automatically generated

## 7.3 Queries by Nana Afua Martinson

1. Retrieve name, and phone details of emergency contact of a guest with first name = John, Last Name = Doe, Guest ID = ‘GOO2’ who after checking out of hotel, left his package behind. Attempts to reach him are unsuccessful.

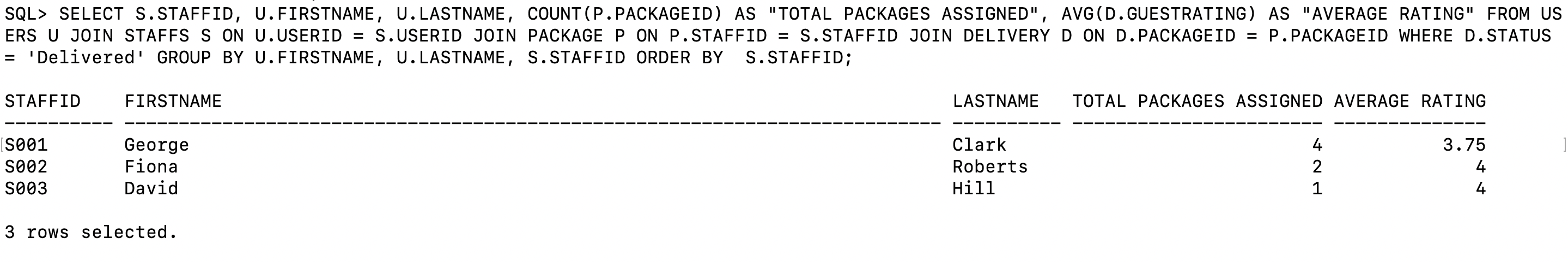
|  |
| --- |
| SELECT  E.FIRSTNAME, E.LASTNAME, E.PHONE  FROM EMERGENCYCONTACT E  JOIN GUESTS G ON G.GUESTID = E.GUESTID  JOIN USERS U ON G.USERID = U.USERID  WHERE U.FIRSTNAME = 'John' AND U.LASTNAME ='Doe' AND G.GUESTID = 'G002'; |

A close-up of a phone number

Description automatically generated

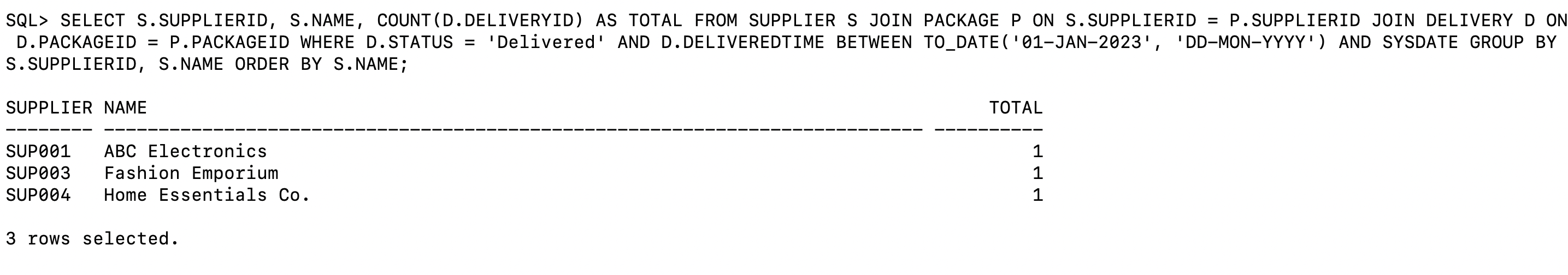
1. Retrieve staff ID, staff first and last name, total number of packages each staff has been assigned to and average rating of deliveries for his packages. Arrange in ascending order of staff ID

|  |
| --- |
| SELECT  S.STAFFID, U.FIRSTNAME, U.LASTNAME,  COUNT(P.PACKAGEID) AS "TOTAL PACKAGES ASSIGNED",  AVG(D.GUESTRATING) AS "AVERAGE RATING"  FROM USERS U JOIN STAFFS S ON U.USERID = S.USERID  JOIN PACKAGE P ON P.STAFFID = S.STAFFID  JOIN DELIVERY D ON D.PACKAGEID = P.PACKAGEID  WHERE D.STATUS = 'Delivered' GROUP BY U.FIRSTNAME, U.LASTNAME, S.STAFFID  ORDER BY S.STAFFID; |



1. Retrieve total number of packages each supplier has had delivered successfully to guests in the hotel from the beginning of this year (2023) to date. Order in increasing alphabetical order of supplier name.

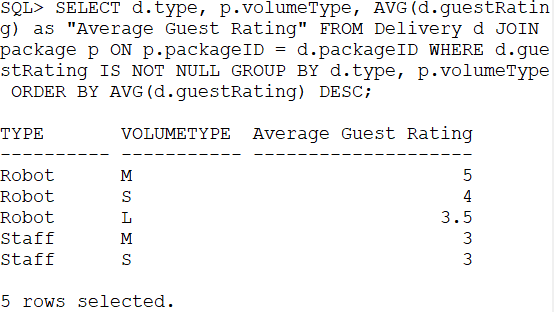
|  |
| --- |
| SELECT S.SUPPLIERID, S.NAME, COUNT(D.DELIVERYID) AS TOTAL FROM SUPPLIER S JOIN PACKAGE P ON S.SUPPLIERID = P.SUPPLIERID JOIN DELIVERY D ON D.PACKAGEID = P.PACKAGEID WHERE D.STATUS = 'Delivered' AND D.DELIVEREDTIME BETWEEN TO\_DATE('01-JAN-2023', 'DD-MON-YYYY') AND SYSDATE GROUP BY S.SUPPLIERID, S.NAME ORDER BY S.NAME; |



## 7.4 Queries by Mahir Pirmohammed

1. Finding the average guest rating for each delivery type to see if we need to be better in some delivery methods

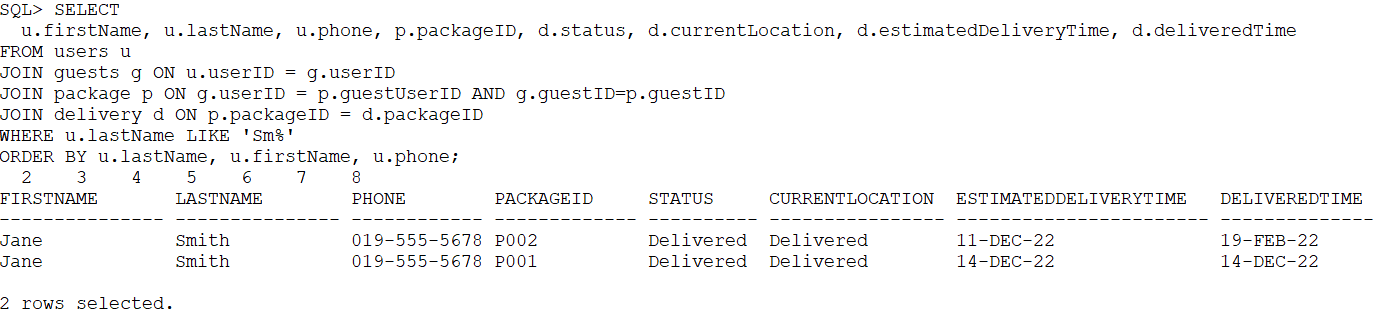
|  |
| --- |
| SELECT d.type, p.volumeType, AVG(d.guestRating) as "Average Guest Rating" FROM Delivery d JOIN package p ON p.packageID = d.packageID WHERE d.guestRating IS NOT NULL GROUP BY d.type, p.volumeType ORDER BY AVG(d.guestRating) DESC; |



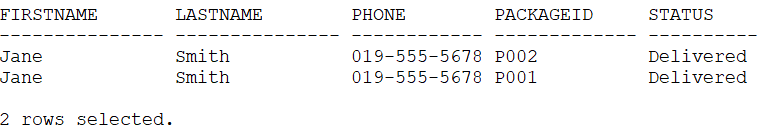
5 rows selected

1. Finding a package and its status for an individual who came to the front desk, but you can only understand their last name starts with Sm.

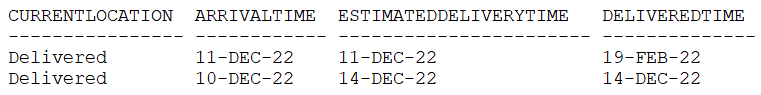
|  |
| --- |
| SELECT  u.firstName, u.lastName, u.phone, p.packageID, d.status, d.currentLocation, d.arrivalTime, d.estimatedDeliveryTime, d.deliveredTime  FROM users u  JOIN guests g ON u.userID = g.userID  JOIN package p ON g.userID = p.guestUserID AND g.guestID=p.guestID  JOIN delivery d ON p.packageID = d.packageID  WHERE u.lastName LIKE 'Sm%'  ORDER BY u.lastName, u.firstName, u.phone; |



• Left half of the result:



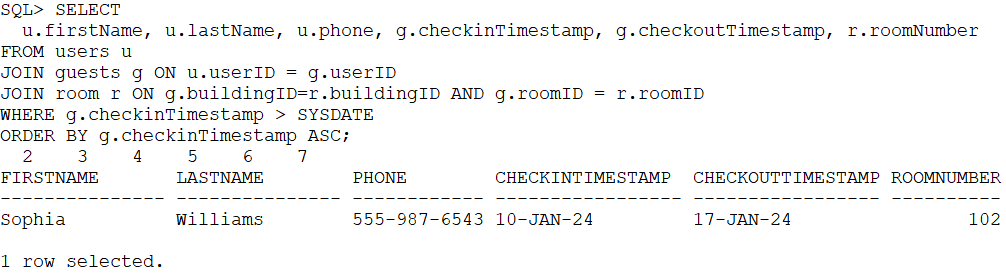
• Right half of the result:



2 rows selected

1. What are the next guests to checkin to the hotel? This will probably be used by a software system to see the next guests that will be arriving at the hotel to checkin from the Database.

|  |
| --- |
| SELECT  u.firstName, u.lastName, u.phone, g.checkinTimestamp, g.checkoutTimestamp, r.roomNumber  FROM users u  JOIN guests g ON u.userID = g.userID  JOIN room r ON g.buildingID=r.buildingID AND g.roomID = r.roomID  WHERE g.checkinTimestamp > SYSDATE  ORDER BY g.checkinTimestamp ASC; |



For our data we only have 1 row selected that will be checking in to our hotel because of the low number of inserts and the one update statement.

Note: I performed this query after doing the update command in 9.4 II. but before I rolled back in 9.4 III.

# 8. Data Manipulation

## 8.1 DML by Estifanos Abebe

1. The hotel finished installing a new elevator in building B01. Insert the elevator’s data into the database.
   1. Data before the INSERT command

|  |
| --- |
| SELECT \* FROM elevator; |

A table with numbers and letters

Description automatically generated

* 1. INSERT command

|  |
| --- |
| INSERT INTO elevator VALUES ('B01', 'E05', 1100, 10, 'L', 350); |

* 1. Data after the INSERT command

|  |
| --- |
| SELECT \* FROM elevator; |

A table with numbers and letters

Description automatically generated

* 1. ROLLBACK

|  |
| --- |
| ROLLBACK;  SELECT \* FROM elevator; |

A table with numbers and letters

Description automatically generated

1. It was discovered that charging station S03 in building B01 is malfunctioning. Update its operational status.
   1. Data before the UPDATE command

|  |
| --- |
| SELECT \* FROM ChargingStation; |

A screenshot of a graph

Description automatically generated

* 1. UPDATE command

|  |
| --- |
| UPDATE ChargingStation  SET operationalStatus = 'R'  WHERE buildingID = 'B01' AND stationID = 'S03'; |

* 1. Data after the UPDATE command

|  |
| --- |
| SELECT \* FROM ChargingStation; |

A screenshot of a graph

Description automatically generated

* 1. ROLLBACK

|  |
| --- |
| ROLLBACK;  SELECT \* FROM ChargingStation; |

A table with numbers and letters

Description automatically generated

1. Guest G001 deletes their emergency contact C001 from their emergency contact list.
   1. Data before the DELETE command

|  |
| --- |
| SELECT \* FROM EmergencyContact; |

A screenshot of a computer

Description automatically generated

* 1. DELETE command

|  |
| --- |
| DELETE FROM EmergencyContact  WHERE guestID = 'G001' AND userID = 'U001' and CONTACTID = 'C001'; |

* 1. Data after the DELETE command

|  |
| --- |
| SELECT \* FROM EmergencyContact; |

A screenshot of a computer

Description automatically generated

* + One row (first row in the previous table) is deleted.
  1. ROLLBACK

|  |
| --- |
| ROLLBACK;  SELECT \* FROM EmergencyContact; |

A table of names

Description automatically generated

## 8.2 DML by Chinonso Eziefule

1. Insert new supplier details into supplier table

Data before Insert:

A close-up of a website

Description automatically generated

7 Rows.

DML STATEMENT:

|  |
| --- |
| INSERT INTO supplier  VALUES ('SUP011', 'Goldman Sachs', 'info@goldmansachs.com', '143-597-3494', '3675 OldMen Ave.', 'Darby', 'PA', 'USA', '19042'); |

OUTPUT:

A white sheet with black text

Description automatically generated with medium confidence

8 Rows

After ROLLBACK;

A close up of a list

Description automatically generated

1. Update existing supplier phone and email

Data before Insert:

A close up of a document

Description automatically generated

DML STATEMENT:

|  |
| --- |
| UPDATE supplier  SET EMAIL = 'info@newupdate.com', phone = '246-802-4682' where supplierID = 'SUP001'; |

Output:

A close up of a document

Description automatically generated

After ROLLBACK:

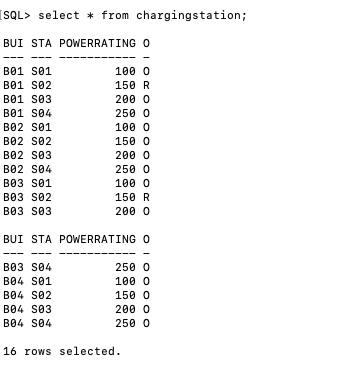
A close up of a document

Description automatically generated

## 8.3 DML by Nana Afua Martinson

1. Insert new charging station into the chargingstation table.

Data before insert command:

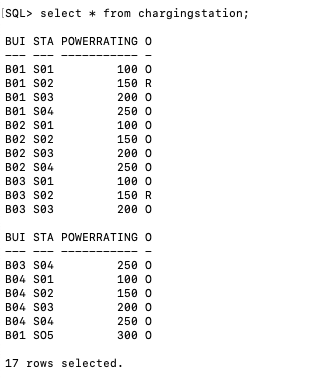


DML STATEMENT:

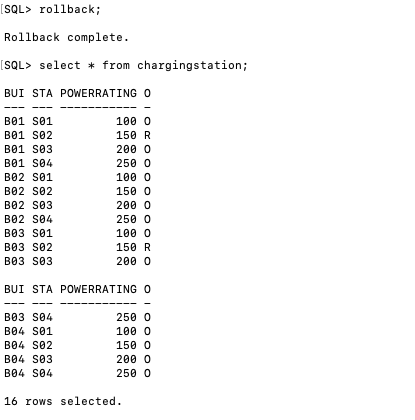
|  |
| --- |
| INSERT INTO CHARGINGSTATION VALUES('B01','SO5', 300, 'O'); |



Data after Insert Command:

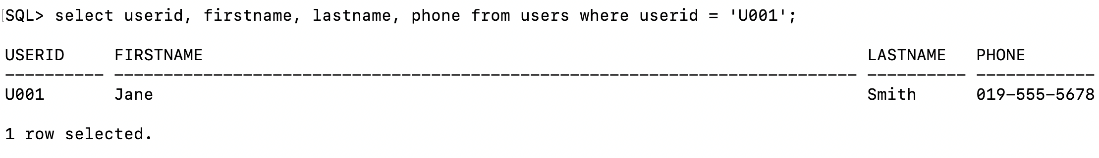


Data after Rollback Command:



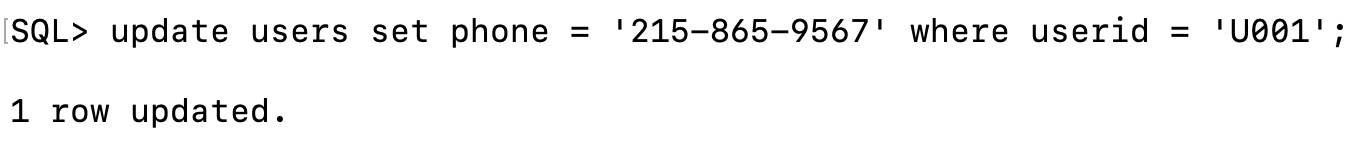
1. Update phone number and email for User with UserID = ‘U001’

Data before Update command:

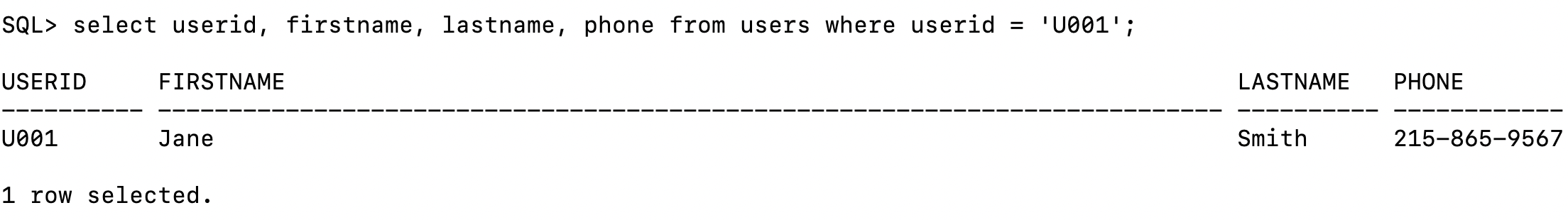


DML STATEMENT:

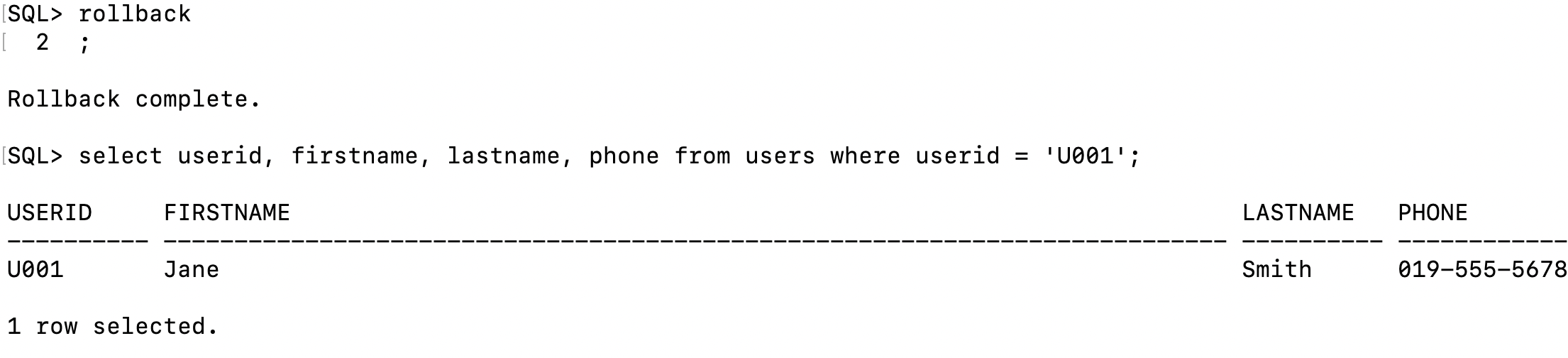
|  |
| --- |
| update users set phone = '215-865-9567' where userid = 'U001'; |



Data after update command:

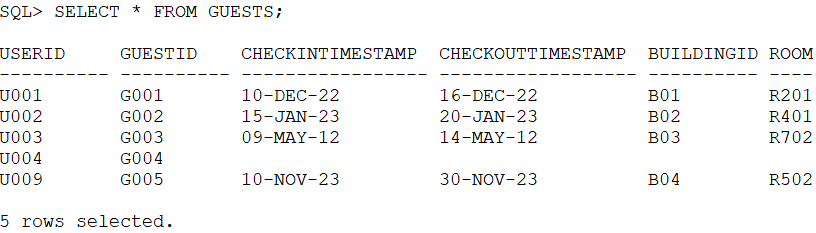


Data after Rollback Command:



## 8.4 DML by Mahir Pirmohammed

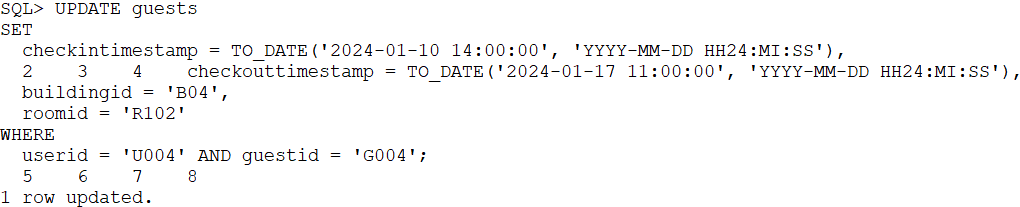
1. Data before the UPDATE command



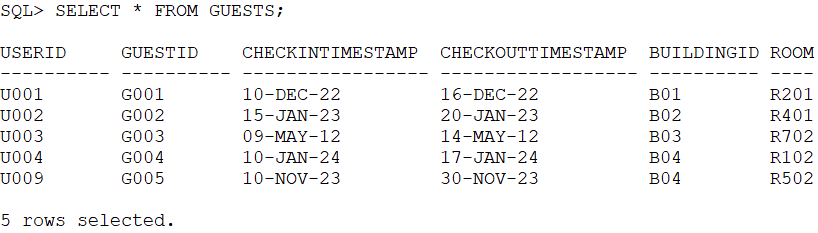
1. UPDATE command

After a user signs up for an account at our hotel website and is a guest/customer. They will then book a room and check-in/check-out times for their stay.

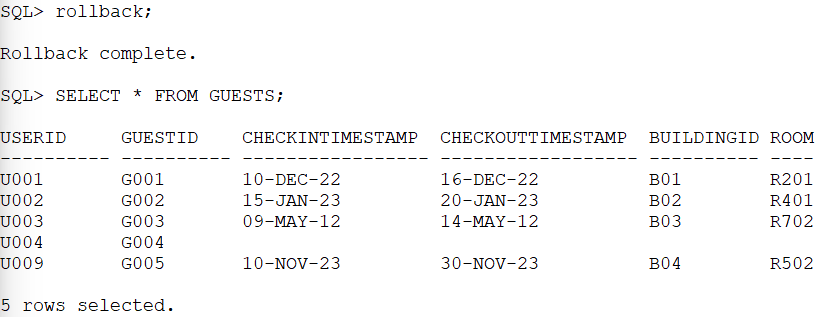
|  |
| --- |
| UPDATE guests  SET  checkintimestamp = TO\_DATE('2024-01-10 14:00:00', 'YYYY-MM-DD HH24:MI:SS'),  checkouttimestamp = TO\_DATE('2024-01-17 11:00:00', 'YYYY-MM-DD HH24:MI:SS'),  buildingid = 'B04',  roomid = 'R102'  WHERE  userid = 'U004' AND guestid = 'G004'; |



1. Data after UPDATE command



1. ROLLBACK



1. Data before the DELETE command  
   1. Users table data

A screenshot of a computer

Description automatically generated

* 1. Guests table data

**A white table with black text and numbers

Description automatically generated**

* 1. EmergencyContact table data

A screenshot of a computer

Description automatically generated

* 1. Package table data

A table with numbers and letters

Description automatically generated

* 1. Delivery table data

A white sheet with black text

Description automatically generated

* 1. Notification table data

A screenshot of a white table

Description automatically generated A white sheet with black text

Description automatically generated

* 1. UserNotification table data

A screenshot of a computer

Description automatically generated **A screenshot of a computer

Description automatically generated**

1. DELETE Command

Delete old users that have not logged in for over 5 years. This will help keep the system maintained since hotel guests usually go to one hotel for one vacation and never visit the same hotel/vacation location again.

Per our business logic when a user that is a guest is deleted all data for that user will be deleted from the following tables: Guests, EmergencyContact, Package, Delivery, Notification, and UserNotification

Delete statement:

|  |
| --- |
| DELETE FROM users WHERE lastloggedin < SYSDATE – 1825; |



1. Data after the DELETE command:
   1. Users table data

A screenshot of a computer

Description automatically generated

* 1. Guests table data

A white paper with black text and numbers

Description automatically generated

* 1. EmergencyContact table data

A screenshot of a computer

Description automatically generated

* 1. Package table data

A table with numbers and letters

Description automatically generated

* 1. Delivery table data

A white table with black text

Description automatically generated

* 1. Notification table data

A white rectangular object with black text

Description automatically generated with medium confidence

* 1. UserNotification table data

A screenshot of a computer

Description automatically generated

1. ROLLBACK
   1. Users table data

A screenshot of a computer

Description automatically generated

* 1. Guests table data

A white table with black text and numbers

Description automatically generated

* 1. EmergencyContact table data

A screenshot of a computer

Description automatically generated

* 1. Package table data

A table with numbers and letters

Description automatically generated

* 1. Delivery table data

A white sheet with black text

Description automatically generated

* 1. Notification table data

A screenshot of a white table

Description automatically generated A white sheet with black text

Description automatically generated

* 1. UserNotification table data

A screenshot of a computer

Description automatically generated **A screenshot of a computer

Description automatically generated**

# Summary

## 9.1 Summary by Estifanos Abebe

This project offered me valuable experience in conceptualizing and implementing a relational database system. It helped me practically learn the essential steps of database design from requirement analysis to actual system implementation. Our team focused on developing a tailored database system for a hotel package tracking system that leverages mobile robots for deliveries. We narrowed our domain to address post-delivery business challenges within the hotel. In the modeling phase, we considered key entities in the delivery process, including staff members, guests, robots, and building infrastructure. The iterative design approach underscored the meticulous considerations necessary for a robust database system. This project not only deepened my understanding of database design and enhanced my teamwork skills, but also highlighted the importance of defining project scope when designing a database system. The implemented database system is expected to optimize efficiency and customer satisfaction in hotel package management. Given more time, potential enhancements, such as incorporating financial transactions and hotel inventory deliveries, could further improve the system's capabilities.

## 9.2 Summary by Chinonso Eziefule

In this project, I took the lead in initiating key components of our documentation, including crafting a comprehensive introduction that set the stage for our database management system. I articulated the context and emphasized the importance of our system within the hospitality industry.

One of my critical contributions was the validation of our Entity-Relationship Diagram (ERD), Relational Schema, Logical ERD, and the subsequent design of the Referential Integrity Diagram. These steps were pivotal in ensuring the robustness of our database foundation.

Collaboration was a cornerstone of our success. We actively engaged with each other to implement agreed-upon ideas, fostering an environment that valued everyone's input and taught me more about teamwork.

Ensuring the accuracy of our overall report was another area where I played a crucial role. The attention to detail we all exhibited, laid the groundwork for our well-designed and functional database system.

Applying the SQL skills I learned in class, helped us to insert the data of all tables, while I carried out implementations in my Local PC (I installed Oracle to my Windows PC), contributing to the overall development of the project. Additionally, I provided valuable queries and Data Manipulation Language (DML) statements as required. Through this journey, I've not only gained practical experience in database design but also honed essential teamwork skills. This project has been an inspiring exploration into the real-world applications of database management, particularly in the context of optimizing efficiency and customer satisfaction within a hotel package management system and I am confident with the knowledge gained.

## 9.3 Summary by Nana Afua Martinson

This project provided me with a valuable opportunity to collaborate with a team in the development of a Database for Tracking Package Deliveries within a hotel setting. Our aim was to create a system that could potentially become a standard upgrade in the hospitality industry. Throughout this project, I gained hands-on experience in implementing SQL and understanding its connection to Relational Diagrams.

Our team focused on creating a comprehensive database that covers entities within our control, including users, robots, packages, notifications, and different components of a hotel building. The scope of our current project is limited, focusing on these key entities. However, we envision the potential for future expansions to accommodate additional entities as the needs of the hotel evolve.

This experience not only enhanced my technical skills but also provided insights into the practical applications of database management in a real-world context.

## 9.4 Summary by Mahir Pirmohammed

This project was great to learn all parts of a Database Management System, from the design to hands-on creation in Oracle. I learned through this project how to effectively separate attributes into appropriate tables and then relate them together through relationships and SQL queries. The domain for this futuristic problem of robot delivery is slowly becoming a reality and since everything is now being delivered to your door, the hotel business should easily be able to accommodate this. For our scope we opted to keep it restricted to what occurs to the package when it enters the hotel and until the guests remain at the hotel, so we did not involve tracking the packages from an external source like FedEx and UPS show for a package. Our implementation shows a good subset of what each table will contain and how the tables are related in a manner where it is easily able to query important things in a few joins.