**Student Guide to Manipal**

Mini Project Report -Database Lab (DSE 2241)

Department of Data Science & Computer Applications



B. Tech Data Science

4th Semester – Batch: A1 - Group: A13

Submitted By

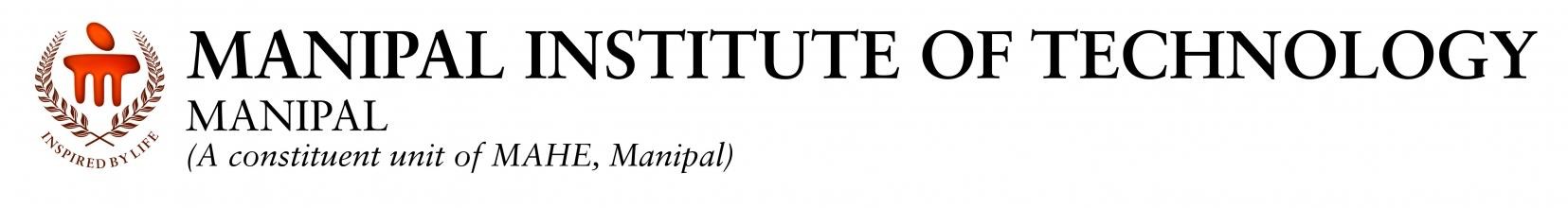
|  |  |
| --- | --- |
| Riddhika Rungta | 230968068 |
| Sthuthi V. Soans | 230968047 |
| Sohan Sanil | 230968074 |
| Rohit Vinod | 230968045 |

**Mentored By**

Mr. Shrinidhi B Mrs. Archana H

Assistant Professor-Senior Assistant Professor-Senior

DSCA, MIT DSCA, MIT



Date: 13 April 2025

**CERTIFICATE**

This is to certify that the Riddhika Rungta (230968068), Sthuthi V. Soans (230968047), Sohan Sanil (230968074) and Rohit Vinod (230968045), have successfully executed a mini project titled “Student Guide to Manipal” rightly bringing fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2241), thereby resulting in the culmination of this project.

**Mr. Shrinidhi B Mrs. Archana H Assistant Professor-Senior Assistant Professor-Senior DSCA, MIT DSCA, MIT**

**ABSTRACT**

Students moving to new places for higher education often struggle to navigate their surroundings and access essential services. In today’s fast-paced world, quick and efficient access to such services is crucial. To address this challenge, the Student Guide to Manipal database is designed to provide students with comprehensive information about Manipal. The main objective of the project is to help students familiarize themselves with their new environment, enabling them to find all kinds of services and their locations in no time.

The database consists of multiple tables pertaining to various services provided in and around Manipal like health care, restaurants, postal services, police stations, laundry facilities, car rentals, tailoring, grocery stores, picnic spots, fashion outlets, theatres etc. Each table includes key details such as location/address, phone number, ratings, online delivery options, etc. The tool used to create the project is SQL Plus.

These are some of the basic necessities of a student in new surroundings. Thus, by implementing this database, we can create a student friendly environment which ensures that all their needs are one tap away, thus making their transition to Manipal smoother and more comfortable. Using SQL Plus, the database so created results in efficient data storage and easy retrieval of information. Security is a high priority in the present world, which is kept in mind while making the database, so that students can safely utilize the data without concerns.

In conclusion, the project is aimed at providing seamless access to important amenities for the students who are starting a new journey of education in a new place. The efficient database allows us to carry out business analysis so as to improve our project and expand services according to the new trends.

**Contents**

[1. Introduction 4](#_Toc66463)

[2. Synopsis 6](#_Toc66464)

[2.1 Proposed System 6](#_Toc66465)

[2.2 Objectives 7](#_Toc66466)

[3. Functional Requirements 8](#_Toc66467)

[4. Detailed Design 13](#_Toc66468)

[4.2 ER Diagram 13](#_Toc66469)

[4.3 Schema Diagram 13](#_Toc66470)

[4.3 Data Dictionary 15](#_Toc66471)

[4.4 Relational Model Implementation 19](#_Toc66472)

[5. Implementation 21](#_Toc66473)

[5.1 Queries 21](#_Toc66474)

[5.2 Triggers 22](#_Toc66475)

[5.3 Stored Procedures 23](#_Toc66476)

[5.4 Stored Functions 32](#_Toc66477)

[6. Result 35](#_Toc66478)

[7. Conclusion and Future Work 42](#_Toc66479)

**Chapter 1**

# Introduction

Relocating to a new city for higher education presents several challenges for students, especially when it comes to accessing essential daily services. For newcomers at Manipal Institute of Technology (MIT), navigating the local ecosystem—which includes everything from finding reliable food outlets and medical facilities to locating laundry services or grocery stores—can often be confusing and time-consuming. In a world where information access and service efficiency are vital, there is an increasing need for centralized platforms that guide students to the right resources in a timely and reliable manner. This project stems from that very need: to build a unified digital system that acts as a comprehensive service guide for students beginning their journey in Manipal.

Currently, students depend on fragmented sources such as social media recommendations, word-of-mouth suggestions, or scattered web searches to find everyday services. This lack of structure often leads to inefficiency, misinformation, and missed opportunities to engage with trusted local providers. Moreover, there is no existing institutional portal or system in place that helps students organize their service needs or offer reviews that could benefit others. These gaps in accessibility and transparency clearly highlight the demand for a system that is both reliable and student-centric.

The proposed solution is a structured backend database system titled *Student Guide to Manipal*. This database stores details of critical services such as hostels, restaurants, gyms, medical stores, transport providers, salons, and more. Through a well-designed schema comprising entities like Students, Services, Service Providers, Bookings, and Reviews, the system facilitates core functionalities like registration, service discovery, booking, and feedback. In addition to helping students, the system also empowers local service providers to register and maintain their information, thus enhancing mutual accessibility.

The advantages of implementing such a system are multifold. It simplifies students’ day-to-day lives, reduces dependence on unverified information sources, and improves efficiency in accessing local resources. The project also ensures data consistency, scalability, and ease of retrieval. Once integrated with a frontend interface in the future, this database system could serve as a one-stop solution for MIT students to fulfill their everyday needs—securely, quickly, and with confidence. Furthermore, the use of Oracle SQL and PL/SQL allows for precise control over data integrity, performance, and transactional operations, ensuring a solid technical foundation for potential expansion.

**Chapter 2**

# Synopsis

## 2.1 Proposed System

The Student Guide to Manipal database system is designed to provide an organized and centralized digital backend that enables students at Manipal Institute of Technology to easily access information about essential services available in and around the campus. This includes services such as hostels, restaurants, gyms, medical stores, grocery outlets, salons, transport, and more.

The problem addressed by this project is the lack of a unified system for new and existing students to explore, compare, and avail services quickly in a new environment. Currently, students rely on scattered platforms such as social media groups, personal references, or offline exploration, which are often inefficient and unreliable. The proposed system allows both students and service providers to register into the platform, maintain their profiles, and interact through features such as bookings, reviews, and service discovery.

The project builds a fully functional backend using Oracle SQL and PL/SQL. The database schema includes multiple interrelated entities like Students, Services, Service\_Details, Avails, Writes\_Reviews, Hostels, and Emergency\_Contacts. It also includes procedures for registration, login validation, booking, feedback submission, service filtering, and automatic ID generation. This backend system lays the foundation for a future full-stack application that can serve as an official student portal for service accessibility and campus support.

## 2.2 Objectives

The main objectives of the project are:

* To create a comprehensive and structured backend database system for student services in Manipal.
* To allow new students to explore nearby services in categories such as food, health, transport, fitness, and essentials.
* To enable seamless student registration, login validation, and emergency contact mapping.
* To allow service providers to register their services and manage offerings including price, availability, and contact details.
* To allow students to avail services, view distance-based recommendations, and post verified reviews.
* To auto-generate student IDs, login credentials, service detail IDs, and emergency contact IDs dynamically using PL/SQL logic.
* To reduce manual effort in accessing student-relevant services and improve accessibility through structured information.
* To ensure secure and normalized data storage using Oracle SQL and PL/SQL, ensuring scalability and data integrity for future integration.

# Chapter 3: Functional Requirements

This backend database project serves as a student-centric portal designed to simplify access to essential services in and around Manipal Institute of Technology. It supports core functionalities like student/service provider registration, login validation, service booking, review submission, and booking status updates. The system is implemented entirely in Oracle SQL and PL/SQL and simulates end-toend flow of interaction between users and services using modular database logic.

### 3.1 User Registering/Login Module

This module allows students and service providers to register securely, log in using system-generated credentials, and reset passwords in case they forget them. It supports functionalities:

* New user registration

* Login

* Forgot password

#### 3.1.1 New User Registration

A new student or service provider must provide their details. Unique IDs (Login\_ID, Student\_ID, Detail\_ID, and Contact\_ID) are auto-generated based on record count. Passwords must be secure and validated.

##### Table 3.1 Registration

|  |  |
| --- | --- |
| INPUT | Role(Student/Service), Name, Email, Phone, Password, Additional profile fields |
| Processing | * Validate password strength * Auto-generate Login ID * Generate Student ID / Detail ID / |
|  | Contact ID  ● Insert into Login, Students, Service\_Details,  Emergency\_Contacts as required |
| OUTPUT | Success message with generated IDs / Prompt to re-enter incorrect values |

#### 3.1.2 Login

Users (students or service providers) can log in with their generated credentials. **Table 3.2 Login**

|  |  |
| --- | --- |
| INPUT | Login\_ID, Password |
| Processing | * Match Login\_ID and Password in Login table * Trigger trg\_validate\_login can also be used to simulate login attempt |
| OUTPUT | Login successful with role displayed / Error if credentials are incorrect |

#### 3.1.3 Forgot Password

If users forget their passwords, they can reset it by verifying their registered phone number.

##### Table 3.3 Forgot Password

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INPUT | Login\_ID, Password | Phone | Number, | New |
| Processing | * Validate user-phone match * Validate new password length * Update password in Login table | | | |
| OUTPUT | Password changed successfully / Error message on mismatch | | | |

### 3.2 Service Booking Module

This module enables students to view and book services based on service type, proximity, and ratings.

#### Table 3.4 Service Booking

|  |  |
| --- | --- |
| INPUT | Student\_ID, Selected Detail\_ID |
| Processing | * Use proc\_show\_service\_options to list nearby services * Avail\_ID is auto-generated |
| OUTPUT | Booking recorded in Avails table with  “Pending” status |

### 3.3 Booking Status Update Module

Allows service providers or system to update the status of a booking from "Pending" to either "Confirmed" or "Cancelled".

#### Table 3.5 Update Booking Status

|  |  |
| --- | --- |
| INPUT | Avail\_ID, New Status |
| Processing | ● Validate that current status is "Pending" |
|  | ● Update the record using proc\_update\_avail\_status |
| OUTPUT | Success message confirming update |

### 3.4 Review and Feedback Module

Enables students to post reviews after availing a service. The system can also compute and update average ratings automatically.

#### Table 3.6 Review Submission

|  |  |
| --- | --- |
| INPUT | Student\_ID, Detail\_ID, Rating, Comment |
| Processing | * Insert record into   Writes\_Reviews   * Optionally call proc\_update\_rating using func\_get\_avg\_rating to update service average rating |
| OUTPUT | Review saved / Service rating updated |

### 3.5 Reporting and Analytics Module

This module includes useful queries for gaining insights about student engagement and service performance. These queries assist in identifying popular services, student activity, and cost trends, thereby helping improve service offerings.

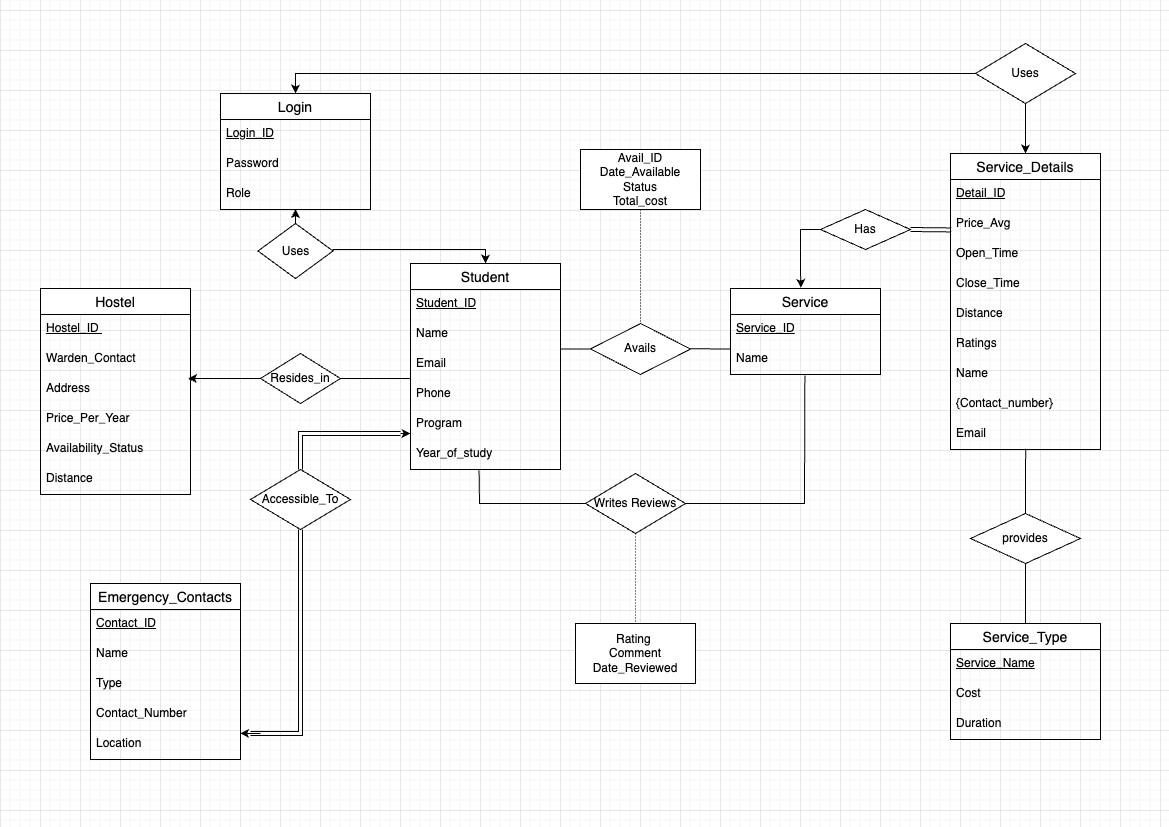
**Table 3.7 Reporting Queries**

|  |  |
| --- | --- |
| **Functionality** | **Description** |
| Top 5 highest rated services | Query to retrieve top-rated service providers using AVG() |
| Login IDs that have never used the system | Identifies users who have never booked or reviewed |
| Services with rating greater than 8 | Shows high-quality services based on user reviews |
| Average cost of services by service type | Provides cost trend analysis per category |
| Number of services availed by each student | Shows student engagement and usage frequency |
| Reviews posted for a particular service provider | Lists feedback and ratings for a selected Detail\_ID |

**Chapter 4**

# Detailed Design

## 4.1 ER Diagram



**Figure 4.1 ER Diagram**

## 4.2 Schema Diagram

**Student**( Student\_ID, Name, Email, Phone, Program, Year\_of\_Study, Login\_ID, Hostel\_No, Contact\_ID)

**Login**( Login\_ID, Password, Role)

**Hostel**( Hostel\_No, Warden\_Contact, Address, Price\_per\_Year, Availabiliy\_Status, Distance)

**Emergency\_Contacts**( Contact\_ID, Name, Type, Contact\_Number, Location)

**Service**( Service\_ID, Name)

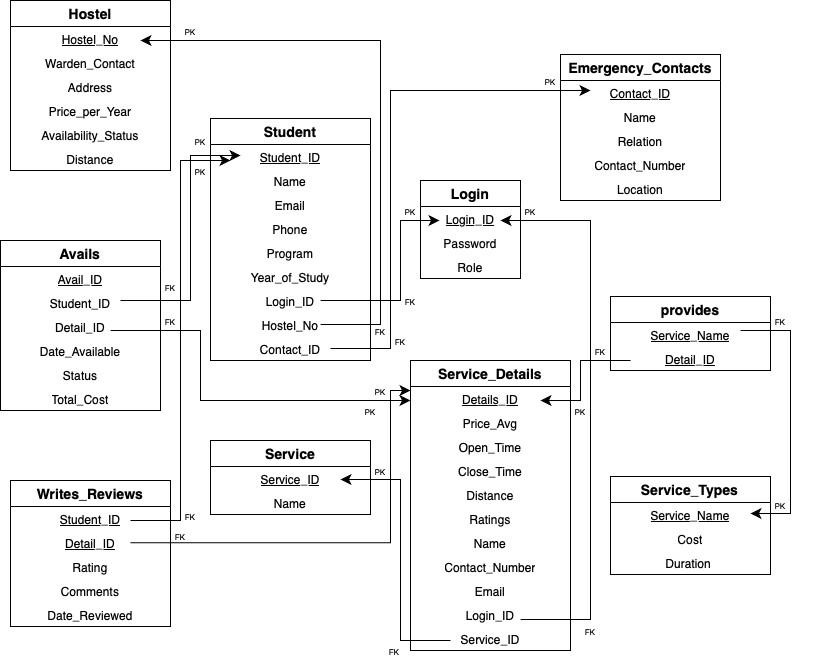
**Service\_Details**( Detail\_ID, Price\_Avg, Open\_Time, Close\_Time, Distance, Ratings, Name, Contact\_Number, Email, Login\_ID, Service\_ID)

**Service\_Types**( Service\_Name, Cost, Duration)

**Provides**( Service\_Name, Detail\_ID)

**Avails**( Avail\_ID, Student\_ID, Service\_ID, Date\_Available, Status, Total\_Cost)

**Writes\_Reviews**( Student\_ID, Service\_ID, Rating, Comments, Date\_Reviewed)



**Figure 4.2 Schema Diagram**

## 4.3 Data Dictionary

#### LOGIN

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraints | Constraint\_name |
| Login\_ID | varchar2(9) | Primary Key |  |
| Password | varchar2(15) | unique | unq\_pass |
| Role | varchar2(10) | valid-Student,Admin,Service | const\_role |

#### EMERGENCY\_CONTACTS

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraints | Constraint\_name |
| Contact\_ID | varchar2(5) | Primary Key, starts with 'C' |  |
| Name | varchar2(20) | Unique |  |
| Relation | varchar2(20) |  |  |
| Contact\_Number | number(10) | valid- ‘\_\_\_\_\_\_\_\_\_\_’ | const\_contact |
| Location | varchar2(50) |  |  |

#### SERVICE

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraint | Constraint\_name |
| Service\_ID | varchar2(6) | Primary Key, valid'S\_\_\_\_\_' | const\_service\_ID |
| Name | varchar2(30) | Unique |  |

#### SERVICE\_DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraints | Constraint\_Name |
| Detail\_ID | varchar2(6) | Primary Key, valid- 'D\_\_\_\_' | const\_detail\_ID |
| Price\_avg | number(5) | >0 | const\_price\_avg |
| Open\_Time | Date |  |  |
| Close\_Time | Date |  |  |
| Distance | number(5,2) |  |  |
| Ratings | number(2) | >0 and <=10 | Rating\_lim |
| Name | varchar2(30) |  |  |
| Contact\_num | number(10) | unique,'\_\_\_\_\_\_\_\_\_\_' | unique\_no |
| Email | varchar2(30) | contains'%@%.com' | email\_const |
| Login\_ID |  | references Login |  |
| Service\_ID |  | references Service |  |

##### PROVIDES

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraints | Constraint\_name |
| Service\_name |  | References Service\_Types |  |
| Detail\_ID |  | References Service\_Details |  |

##### SERVICE\_TYPES

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraints | Constraint\_name |
| Service\_Name | varchar2(20) | Primary Key |  |
| Cost | number(5) |  |  |
| Duration | number(5) |  |  |

##### HOSTEL

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraint | Constraint\_name |
| Hostel\_No | varchar2(6) | Primary Key, Valid- 'B\_\_' | const\_hostel\_no |
| Warden\_contact | number(10) | Unique, ‘\_\_\_\_\_\_\_\_\_\_’ |  |
| Address | varchar2(100) |  |  |
| Price\_per\_Year | number(10,2) | >0 |  |
| Availability\_Status | varchar2(3) | Valid-Yes,No |  |
| Distance | number(5,2) | >0 |  |

##### STUDENT

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type | Constraints | Constraint\_name |
| Student\_ID | varchar2(6) | Primary Key, Valid- 'M\_\_\_\_\_' | const\_stud\_ID |
| Name | varchar2(40) |  |  |
| Email | varchar2(40) | contains '@manipal.edu' | const\_stud\_email |
| Phone | number(10) | unique, '\_\_\_\_\_\_\_\_\_\_' |  |
| Program | varchar2(6) | valid-BTECH, MTECH, MBA, BBA, BCA, MCA,  MMBS | const\_prog |
| Year\_of\_Study | number(4) |  |  |
| Login\_ID |  | references Login |  |
| Hostel\_No |  | references Hostel |  |
| Contact\_ID |  | References Emergency  Contacts |  |

##### AVAILS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Constraint | | Constraint\_name |
| Avail\_ID | varchar2(6) | Primary Key, Starts with 'A' | |  |
| Student\_ID |  | references Student | |  |
| Detail\_ID |  | references Service Details | |  |
| Data\_Avails | Date |  | |  |
| Status | varchar2(20) | valid-Confirmed, Cancelled | Pending, | const\_status |
| Total cost | number(7) | >0 |  |  |

##### WRITES\_REVIEWS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Constraint |  | Constraint\_name |
| Student\_ID |  | Primary Key,  Student | references |  |
| Detail\_ID |  | Primary Key,  Service Details | references |  |
| Rating | number(2) | >0 and <=10 |  |  |
| Comments | varchar2(50) |  |  |  |
| Date\_Reviewed | Date |  |  |  |

## 4.4 Relational Model Implementation

**CREATE TABLE LOGIN** (LOGIN\_ID VARCHAR2(9) PRIMARY KEY, PASSWORD VARCHAR2(15) CONSTRAINT UNQ\_PASS UNIQUE, ROLE VARCHAR2(10) CONSTRAINT CONST\_ROLE CHECK(ROLE IN ('STUDENT','ADMIN','SERVICE')));

**CREATE TABLE EMERGENCY\_CONTACTS**(CONTACT\_ID VARCHAR2(5) PRIMARY KEY CHECK(CONTACT\_ID LIKE 'C%'), NAME VARCHAR2(20) UNIQUE, RELATION VARCHAR2(20) , CONTACT\_NUMBER NUMBER(10) CHECK(CONTACT\_NUMBER LIKE '\_\_\_\_\_\_\_\_\_\_'), LOCATION VARCHAR2(50));

**CREATE TABLE SERVICE**(SERVICE\_ID VARCHAR2(6) CONSTRAINT ONST\_SERVICE\_ID CHECK(SERVICE\_ID LIKE 'S\_\_\_\_\_') PRIMARY KEY, NAME VARCHAR2(30) UNIQUE);

**CREATE TABLE SERVICE\_TYPES**( SERVICE\_NAME VARCHAR2(20) PRIMARY KEY, COST NUMBER(5), DURATION NUMBER(5));

**CREATE TABLE SERVICE\_DETAILS**(DETAIL\_ID VARCHAR2(6) CONSTRAINT CONST\_DETAIL\_ID CHECK(DETAIL\_ID LIKE 'D\_\_\_\_\_') PRIMARY KEY, PRICE\_AVG NUMBER(5) CONSTRAINT CONST\_PRICEAVG CHECK(PRICE\_AVG >0), OPEN\_TIME DATE, CLOSE\_TIME DATE, DISTANCE NUMBER(5,2), RATINGS NUMBER(2) CONSTRAINT RATING\_LIM CHECK(RATINGS BETWEEN 0 AND 10), NAME VARCHAR2(30), CONTACT\_NUM NUMBER(10) CONSTRAINT UNIQUE\_NUM UNIQUE CHECK(CONTACT\_NUM LIKE '\_\_\_\_\_\_\_\_\_\_'), EMAIL VARCHAR2(30) CONSTRAINT EMAIL\_CONST

CHECK(EMAIL LIKE '%@%.COM'), LOGIN\_ID VARCHAR2(9) REFERENCES LOGIN, SERVICE\_ID VARCHAR2(6) REFERENCES SERVICE);

**CREATE TABLE HOSTEL**( HOSTEL\_ID VARCHAR2(6) PRIMARY KEY CONSTRAINT CONST\_HOSTEL\_NO CHECK(HOSTEL\_ID LIKE 'B\_\_\_\_\_'), WARDEN\_CONTACT NUMBER(10) UNIQUE CHECK(WARDEN\_CONTACT LIKE '\_\_\_\_\_\_\_\_\_\_'), ADDRESS VARCHAR2(100), PRICE\_PER\_YEAR NUMBER(10,2) CHECK(PRICE\_PER\_YEAR>0), AVAILABILITY\_STATUS VARCHAR2(3) CHECK(AVAILABILITY\_STATUS IN ('YES','NO')), DISTANCE NUMBER(5,2) CHECK(DISTANCE>0));

**CREATE TABLE PROVIDES**( SERVICE\_NAME VARCHAR2(20) REFERENCES SERVICE\_TYPES, DETAIL\_ID VARCHAR2(6) REFERENCES SERVICE\_DETAILS);

**CREATE TABLE STUDENTS**( STUDENT\_ID VARCHAR2(6) PRIMARY KEY CONSTRAINT CONST\_STUD\_ID CHECK(STUDENT\_ID LIKE 'M\_\_\_\_\_'), NAME VARCHAR2(40), EMAIL VARCHAR2(40) CONSTRAINT CONST\_STUD\_EMAIL CHECK(EMAIL LIKE '%@MANIPAL.EDU'), PHONE NUMBER(10) UNIQUE CHECK(PHONE LIKE '\_\_\_\_\_\_\_\_\_\_'), PROGRAM VARCHAR2(6) CONSTRAINT CONST\_PROG CHECK(PROGRAM IN ('BTECH','MTECH','MBA','BBA','MBBS','BCA','MCA')), YEAR\_OF\_STUDY NUMBER(4), LOGIN\_ID VARCHAR2(9) REFERENCES LOGIN, HOSTEL\_NO VARCHAR2(6) REFERENCES HOSTEL, CONTACT\_ID VARCHAR2(5) REFERENCES

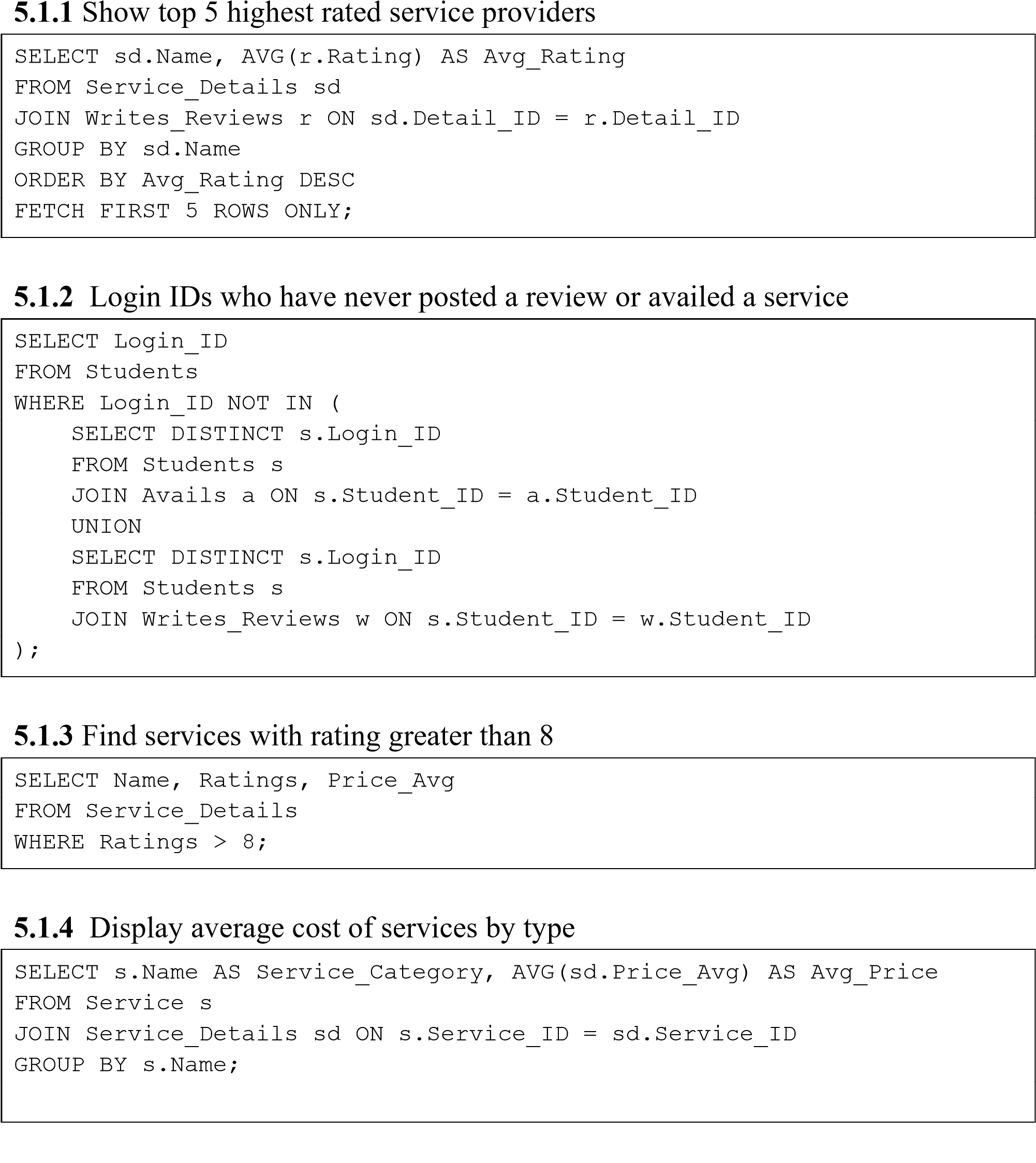
EMERGENCY\_CONTACTS);

**CREATE TABLE AVAILS** ( AVAIL\_ID VARCHAR2(6) PRIMARY KEY, STUDENT\_ID VARCHAR2(6) REFERENCES STUDENTS, DETAIL\_ID VARCHAR2(6) REFERENCES SERVICE\_DETAILS, DATE\_AVAILS DATE, STATUS VARCHAR2(20) CHECK(STATUS IN ('CONFIRMED','PENDING','CANCELLED')), TOTAL\_COST NUMBER(7) CHECK(TOTAL\_COST > 0));

**CREATE TABLE WRITES\_REVIEWS**( STUDENT\_ID VARCHAR2(6) REFERENCES STUDENTS, DETAIL\_ID VARCHAR2(6) REFERENCES SERVICE\_DETAILS, RATING NUMBER(2) CHECK(RATING>0 AND RATING<=10), COMMENTS VARCHAR2(50), DATE\_REVIEWED DATE, PRIMARY KEY(STUDENT\_ID, DETAIL\_ID));

# 5. Implementation

## 5.1 Queries



**5.1.5** Count of services availed by each student

|  |
| --- |
| SELECT s.Student\_ID, s.Name, COUNT(\*) AS Services\_Availed |
| FROM Students s  JOIN Avails a ON s.Student\_ID = a.Student\_ID  GROUP BY s.Student\_ID, s.Name; |

**5.1.6**

Show reviews for a particular service provider

SELECT wr.Student\_ID, wr.Rating, wr.Comments

FROM Writes\_Reviews wr

WHERE wr.Detail\_ID = 'D00005';

## 5.2 Triggers

**5.2.1**  trg\_validate\_login (for login check)

Validates login credentials and throws error if password doesn't match. Only valid during a login attempt. Normally done in app layer, but can simulate in DB.

CREATE OR REPLACE TRIGGER trg\_validate\_login

BEFORE INSERT ON Login

FOR EACH ROW

DECLARE

v\_password VARCHAR2(50);

BEGIN

SELECT Password INTO v\_password FROM Login WHERE Login\_ID =

:

NEW.Login\_ID;

IF v\_password != :NEW.Password THEN

RAISE\_APPLICATION\_ERROR(

-

20002

, 'Invalid password.');

END IF;

END;

/

|  |
| --- |
| SET SERVEROUTPUT ON; |
| DECLARE  v\_login\_id VARCHAR2(10) := '&login\_id'; v\_password VARCHAR2(50) := '&password'; v\_stored\_pass VARCHAR2(50); v\_role VARCHAR2(10); BEGIN |
| SELECT Password, Role INTO v\_stored\_pass, v\_role |
| FROM Login |
| WHERE Login\_ID = v\_login\_id; |
| IF v\_password = v\_stored\_pass THEN |
| DBMS\_OUTPUT.PUT\_LINE('Login successful! Role: ' || v\_role); |
| ELSE  DBMS\_OUTPUT.PUT\_LINE('Login failed: Incorrect password.'); END IF; |
| EXCEPTION |
| WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Login failed: User does not exist.'); |
| END; |
| / |

## 5.3 Stored Procedures

**5.3.1** proc\_submit\_review

Inserts a review for a service into the Writes\_Reviews table along with the current date.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_submit\_review ( |
| p\_stud\_id IN VARCHAR2, p\_detail\_id IN VARCHAR2, p\_rating IN NUMBER, p\_comment IN VARCHAR2  ) AS  BEGIN  INSERT INTO Writes\_Reviews (Student\_ID, Detail\_ID, Rating, Comments, |
| Date\_Reviewed) |
| VALUES (p\_stud\_id, p\_detail\_id, p\_rating, p\_comment, SYSDATE); END; |
| / |

**Procedure call:**

|  |
| --- |
| SET SERVEROUTPUT ON; |
| DECLARE  v\_student\_id VARCHAR2(10) := '&student\_id'; v\_detail\_id VARCHAR2(10) := '&detail\_id'; v\_rating NUMBER := &rating;  v\_comment VARCHAR2(200) := '&comment'; BEGIN  proc\_submit\_review(v\_student\_id, v\_detail\_id, v\_rating, v\_comment); END; |

/

**5.3.2** proc\_register\_new\_student

Registers a new student along with emergency contact and login details, generating IDs automatically.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_register\_new\_student ( |
| p\_stud\_name IN VARCHAR2, |
| p\_stud\_email IN VARCHAR2, p\_stud\_phone IN NUMBER, p\_stud\_program IN VARCHAR2, |
| p\_stud\_year IN NUMBER, p\_stud\_hostel IN VARCHAR2, p\_contact\_name IN VARCHAR2, p\_contact\_relation IN VARCHAR2, p\_contact\_number IN NUMBER, p\_contact\_address IN VARCHAR2, p\_password IN VARCHAR2  ) IS  v\_login\_id VARCHAR2(10); v\_student\_id VARCHAR2(6); v\_contact\_id VARCHAR2(5); v\_count NUMBER;  BEGIN  SELECT COUNT(\*) + 1 INTO v\_count FROM Login; v\_login\_id := 'L' || LPAD(v\_count, 8, '0'); INSERT INTO Login (Login\_ID, Password, Role)  VALUES (v\_login\_id, p\_password, 'STUDENT'); |
|  |
| SELECT COUNT(\*) + 1 INTO v\_count FROM Emergency\_Contacts;  v\_contact\_id := 'C' || LPAD(v\_count, 3, '0');  INSERT INTO Emergency\_Contacts (Contact\_ID, Name, Relation, Contact\_Number, Location) |
| VALUES (v\_contact\_id, p\_contact\_name, p\_contact\_relation, |
| p\_contact\_number, p\_contact\_address);      SELECT COUNT(\*) + 1 INTO v\_count FROM Students; v\_student\_id := 'M' || LPAD(v\_count, 5, '0');  INSERT INTO Students (  Student\_ID, Name, Email, Phone, Program, Year\_of\_Study, Login\_ID, |
| Hostel\_No, Contact\_ID |
| )  VALUES ( v\_student\_id, p\_stud\_name, p\_stud\_email, p\_stud\_phone, p\_stud\_program,  p\_stud\_year, v\_login\_id, p\_stud\_hostel, v\_contact\_id ); |

DBMS\_OUTPUT.PUT\_LINE('Student registered successfully!');

DBMS\_OUTPUT.PUT\_LINE('Student\_ID: ' || v\_student\_id);

DBMS\_OUTPUT.PUT\_LINE('Login\_ID: ' || v\_login\_id);

DBMS\_OUTPUT.PUT\_LINE('Contact\_ID: ' || v\_contact\_id);

END;

/

**Procedure call:**

SET SERVEROUTPUT ON;

DECLARE

v\_stud\_name VARCHAR2(50) := '&stud\_name'; v\_stud\_email VARCHAR2(50) := '&stud\_email'; v\_stud\_phone NUMBER := &stud\_phone; v\_stud\_program VARCHAR2(10) := '&stud\_program'; v\_stud\_year NUMBER := &stud\_year; v\_stud\_hostel VARCHAR2(6) := '&stud\_hostel'; v\_contact\_name VARCHAR2(50) := '&contact\_name'; v\_contact\_relation VARCHAR2(20) := '&contact\_relation'; v\_contact\_number NUMBER := &contact\_number; v\_contact\_address VARCHAR2(100):= '&contact\_address'; v\_password VARCHAR2(50) := '&password'; BEGIN

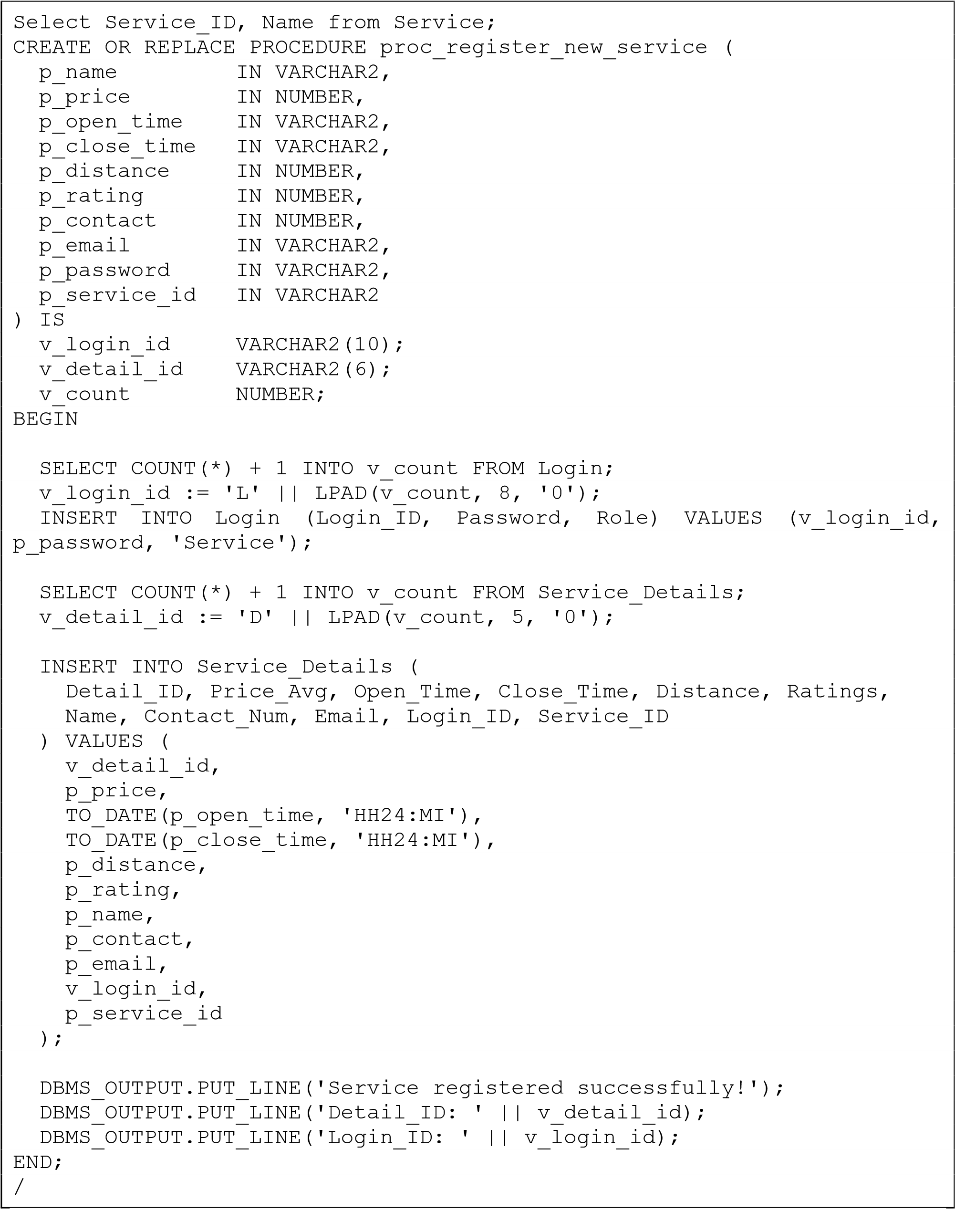
proc\_register\_new\_student( p\_stud\_name => v\_stud\_name, p\_stud\_email => v\_stud\_email, p\_stud\_phone => v\_stud\_phone, p\_stud\_program => v\_stud\_program, p\_stud\_year => v\_stud\_year, p\_stud\_hostel => v\_stud\_hostel, p\_contact\_name => v\_contact\_name, p\_contact\_relation => v\_contact\_relation, p\_contact\_number => v\_contact\_number, p\_contact\_address => v\_contact\_address, p\_password => v\_password

);

END; /

##### 5.3.3 proc\_register\_new\_service

Registers a new service provider and their service details while generating Login\_ID and Detail\_ID and inserts into Login and Service\_Details.



**Procedure call:**

|  |
| --- |
| SET SERVEROUTPUT ON; |
| DECLARE |
| v\_name VARCHAR2(50) := '&name'; |
| v\_price NUMBER := &price; v\_open\_time VARCHAR2(10) := '&open\_time'; |
| v\_close\_time VARCHAR2(10) := '&close\_time'; |
| v\_distance NUMBER := &distance; v\_rating NUMBER := &rating; |
| v\_contact NUMBER := &contact; |
| v\_email VARCHAR2(50) := '&email'; |
| v\_password VARCHAR2(50) := '&password'; |
| v\_service\_id VARCHAR2(6) := '&service\_id'; BEGIN  proc\_register\_new\_service( p\_name => v\_name, p\_price => v\_price, p\_open\_time => v\_open\_time, p\_close\_time => v\_close\_time, p\_distance => v\_distance, |
| p\_rating => v\_rating, |
| p\_contact => v\_contact, p\_email => v\_email, p\_password => v\_password, p\_service\_id => v\_service\_id ); |
| END; |
| / |

**5.3.4** proc\_show\_services\_by\_type

Displays service providers matching a specific service type, showing price and rating.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_show\_services\_by\_type ( |
| p\_service\_name IN VARCHAR2 ) AS |
| BEGIN |
| FOR rec IN (  SELECT sd.Name, sd.Price\_Avg, sd.Ratings  FROM Provides p  JOIN Service\_Details sd ON p.Detail\_ID = sd.Detail\_ID  WHERE p.Service\_Name = p\_service\_name |

)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Name: ' || rec.Name || ', Price: ' ||

rec.Price\_Avg || ', Rating: ' || rec.Ratings);

END LOOP;

END;

/

**Procedure call:**

SET SERVEROUTPUT ON;

DECLARE

v\_service\_name VARCHAR2(50) := '&service\_name';

BEGIN

proc\_show\_services\_by\_type(v\_service\_name);

END;

/

##### 5.3.5 proc\_show\_service\_options

Shows nearby service options sorted by distance from the student’s hostel for a given service type.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_show\_service\_options ( |
| p\_student\_id IN VARCHAR2, p\_service\_name IN VARCHAR2, p\_service\_type IN VARCHAR2  ) AS  v\_hostel\_no VARCHAR2(6); v\_student\_dist NUMBER; CURSOR c\_service\_options IS  SELECT sd.Detail\_ID, sd.Price\_Avg, sd.Distance, sd.Ratings, |
| sd.Open\_Time, sd.Close\_Time |
| FROM Service\_Details sd  JOIN Provides p ON sd.Detail\_ID = p.Detail\_ID  WHERE p.Service\_Name = p\_service\_type  ORDER BY ABS(sd.Distance - v\_student\_dist);  BEGIN  SELECT Hostel\_No INTO v\_hostel\_no FROM Students WHERE Student\_ID = |
| p\_student\_id; |
| SELECT Distance INTO v\_student\_dist FROM Hostel WHERE Hostel\_No = v\_hostel\_no;  DBMS\_OUTPUT.PUT\_LINE('Available Service Details for ' ||  p\_service\_name || ' (' || p\_service\_type || '):');  FOR rec IN c\_service\_options LOOP |

DBMS\_OUTPUT.PUT\_LINE('Detail ID: ' || rec.Detail\_ID || ', Price: ' || rec.Price\_Avg ||

', Distance: ' || rec.Distance || ' km,

Rating: ' || rec.Ratings ||

', Open: ' || TO\_CHAR(rec.Open\_Time,

'HH24:MI') || ', Close: ' || TO\_CHAR(rec.Close\_Time, 'HH24:MI'));

END LOOP;

END; /

**Procedure call:**

SET SERVEROUTPUT ON;

DECLARE

v\_student\_id VARCHAR2(10) := '&student\_id';

v\_service\_name VARCHAR2(50) := '&service\_name';

v\_service\_type VARCHAR2(50) := '&service\_type';

BEGIN

v\_service\_name,

proc\_show\_service\_options(v\_student\_id,

v\_service\_type);

END;

/

**5.3.6** proc\_book\_selected\_service

Books a selected service for a student, generates an Avail\_ID, and inserts a pending booking.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_book\_selected\_service ( |
| p\_student\_id IN VARCHAR2, |
| p\_detail\_id IN VARCHAR2 ) AS |
| v\_cost NUMBER; v\_count NUMBER;  v\_avail\_id VARCHAR2(10);  BEGIN    SELECT Price\_Avg INTO v\_cost FROM Service\_Details WHERE Detail\_ID = p\_detail\_id;      SELECT COUNT(\*) INTO v\_count FROM Avails;    v\_avail\_id := 'A' || LPAD(v\_count + 1, 5, '0'); |

INSERT INTO Avails (Avail\_ID, Student\_ID, Detail\_ID, Date\_Avails,

Status, Total\_cost)

VALUES (v\_avail\_id, p\_student\_id, p\_detail\_id, SYSDATE, 'Pending',

v\_cost);

DBMS\_OUTPUT.PUT\_LINE('Service booked with Avail\_ID ' || v\_avail\_id

|| ' for Student ID: ' || p\_student\_id || ' and Detail ID: ' ||

p\_detail\_id);

END;

/

**Procedure call:**

SET SERVEROUTPUT ON;

DECLARE

v\_student\_id VARCHAR2(10) := '&student\_id';

v\_detail\_id VARCHAR2(10) := '&detail\_id';

BEGIN

proc\_book\_selected\_service(v\_student\_id, v\_detail\_id);

END;

/

**5.3.7** proc\_update\_avail\_status

Updates the status of an existing booking (e.g., from Pending to Confirmed/Cancelled).

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_update\_avail\_status ( |
| p\_avail\_id IN VARCHAR2, p\_new\_status IN VARCHAR2 |
| ) AS |
| BEGIN  UPDATE Avails  SET Status = p\_new\_status  WHERE Avail\_ID = p\_avail\_id AND Status = 'Pending';    DBMS\_OUTPUT.PUT\_LINE('Updated Avail\_ID ' || p\_avail\_id || ' to status:  ' || p\_new\_status);  END;  / |

**Procedure call:**

SET SERVEROUTPUT ON;

DECLARE

v\_avail\_id VARCHAR2(10) := '&avail\_id';

v\_new\_status VARCHAR2(20) := '&new\_status';

BEGIN

proc\_update\_avail\_status(

p\_avail\_id => v\_avail\_id,

p\_new\_status => v\_new\_status

)

;

END;

/

**5.3.8** proc\_forgot\_password

Resets the user’s password after validating phone number and login ID match.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_forgot\_password ( |
| p\_login\_id IN VARCHAR2, p\_phone\_number IN NUMBER, p\_new\_password IN VARCHAR2  ) AS  v\_stored\_phone NUMBER; v\_role VARCHAR2(10);  BEGIN  SELECT s.Phone, l.Role INTO v\_stored\_phone, v\_role  FROM Students s  JOIN Login l ON s.Login\_ID = l.Login\_ID  WHERE l.Login\_ID = p\_login\_id AND l.Role = 'Student'; |
| IF v\_stored\_phone = p\_phone\_number THEN |
| IF LENGTH(p\_new\_password) >= 8 THEN  UPDATE Login  SET Password = p\_new\_password  WHERE Login\_ID = p\_login\_id;    DBMS\_OUTPUT.PUT\_LINE('Password changed successfully for Login ID:  ' || p\_login\_id); ELSE  DBMS\_OUTPUT.PUT\_LINE('New password must be at least 8 characters long.'); END IF;  ELSE  DBMS\_OUTPUT.PUT\_LINE('Phone number does not match the registered phone.'); |

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Login ID not found or role mismatch.');

END;

/

SET SERVEROUTPUT ON;

DECLARE

v\_login\_id VARCHAR2(10) := '&login\_id';

v\_phone\_number NUMBER := &phone\_number;

v\_new\_password VARCHAR2(50) := '&new\_password';

BEGIN

proc\_forgot\_password(

p\_login\_id => v\_login\_id,

p\_phone\_number => v\_phone\_number,

p\_new\_password => v\_new\_password

)

;

END;

/

## 5.4 Stored Functions

**5.4.1** func\_get\_avg\_rating

Calculates the average rating for a given service based on entries in Writes\_Reviews.

CREATE OR REPLACE FUNCTION func\_get\_avg\_rating(p\_detail\_id VARCHAR2)

RETURN NUMBER IS

avg\_rating NUMBER;

BEGIN

SELECT AVG(Rating) INTO avg\_rating FROM Writes\_Reviews WHERE Detail\_ID

=

p\_detail\_id;

RETURN avg\_rating;

END;

/

Calls func\_get\_avg\_rating and updates the Ratings column in the Service\_Details table.

CREATE OR REPLACE PROCEDURE proc\_update\_rating (

p\_detail\_id IN VARCHAR2

AS

)

v\_avg\_rating NUMBER;

BEGIN

v\_avg\_rating := func\_get\_avg\_rating(p\_detail\_id);

UPDATE Service\_Details

SET Ratings = ROUND(v\_avg\_rating)

WHERE Detail\_ID = p\_detail\_id;

DBMS\_OUTPUT.PUT\_LINE('Updated average rating for Detail\_ID ' ||

p\_detail\_id || ' to ' || ROUND(v\_avg\_rating));

END;

/

SET SERVEROUTPUT ON;

DECLARE

v\_detail\_id VARCHAR2(10) := '&detail\_id';

BEGIN

proc\_update\_rating(v\_detail\_id);

END;

/

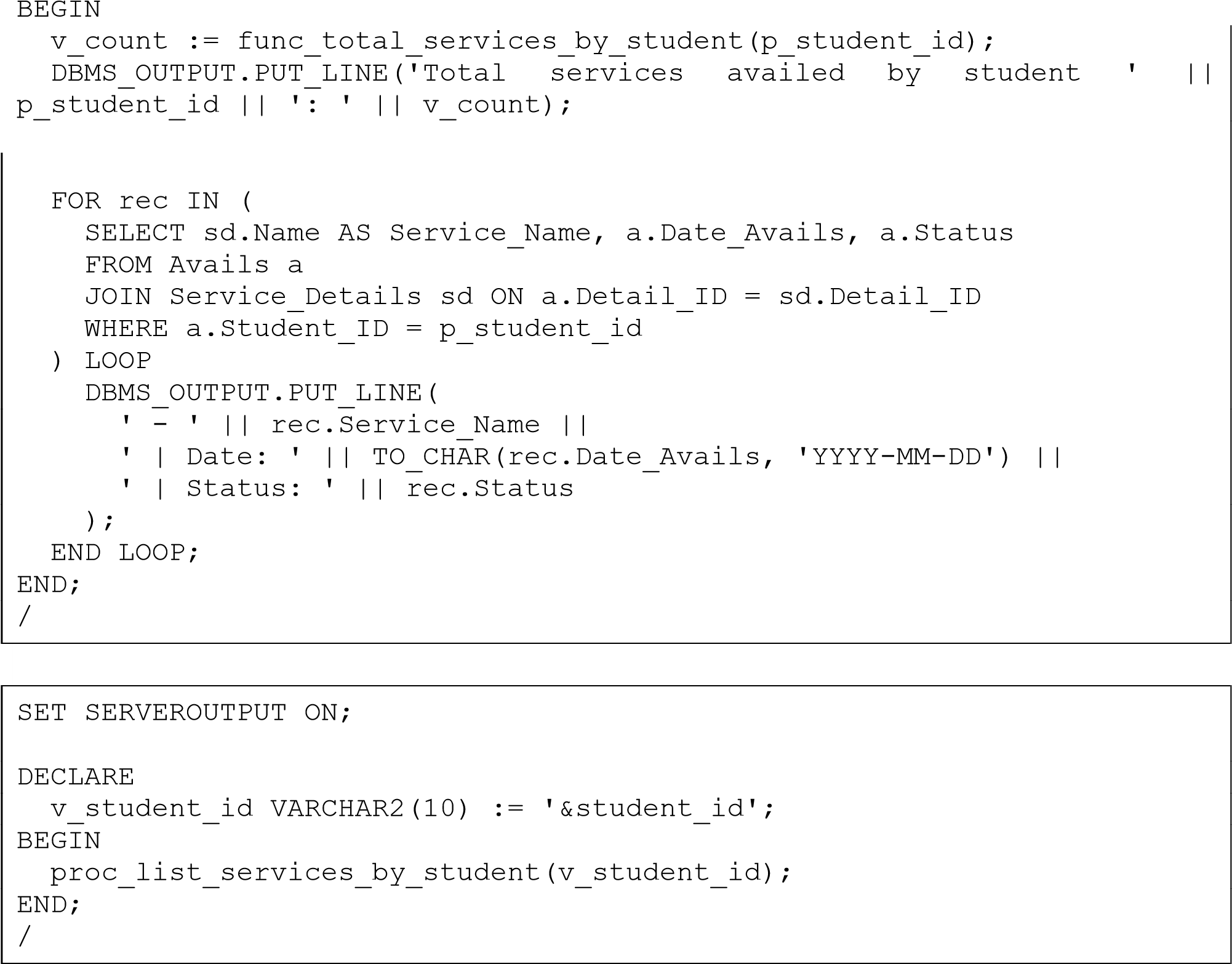
**5.4.2** func\_total\_services\_by\_student

Returns the total number of services availed by a particular student.

|  |
| --- |
| CREATE OR REPLACE FUNCTION func\_total\_services\_by\_student(p\_student\_id VARCHAR2) RETURN NUMBER IS  count\_services NUMBER;  BEGIN  SELECT COUNT(\*) INTO count\_services FROM Avails WHERE Student\_ID =  p\_student\_id;  RETURN count\_services;  END;  / |

Displays the count and list of services availed by a student along with date and booking status.

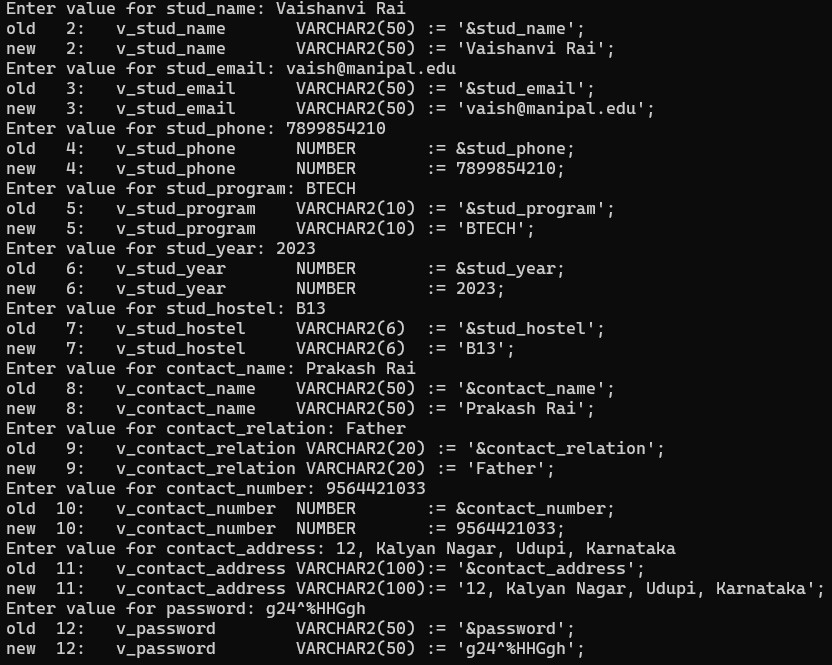
|  |
| --- |
| CREATE OR REPLACE PROCEDURE proc\_list\_services\_by\_student(p\_student\_id |
| VARCHAR2) AS v\_count NUMBER; |



# 6. Result

#### 6.1 Registration of Student (5.3.2)

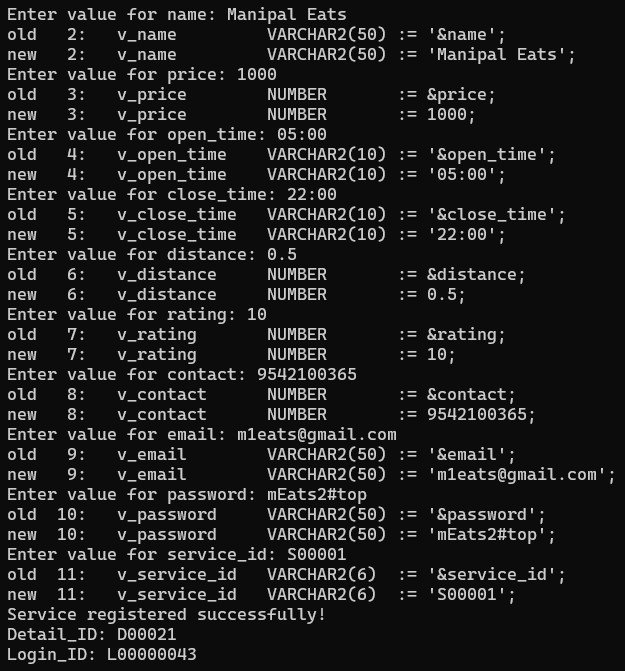
A new student can register into the Student Guide to Manipal by entering the details like Name, Email ID, Phone Number, Program, Year of Admission, Hostel, Emergency Contact Details as well as the Password for future use. The Login ID, Student ID and Contact ID will be auto-generated by the system.



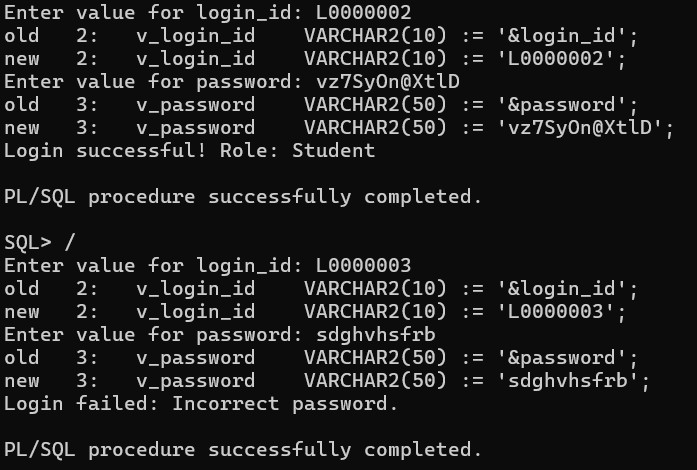


#### 6.2 Registration of Service Provider (5.3.3)

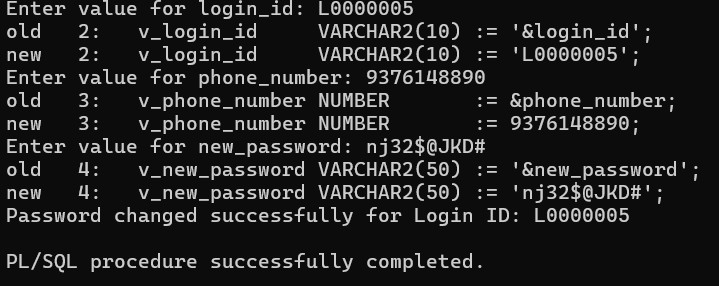
A new service provider can register into the database by entering their details like the Name of the service, Average Price, Open and Close Time, Ratings, Distance from a fixed point (MIT), Contact Details as well as the Password for future use. The System generates the Detail\_ID and Login\_ID for the provider.



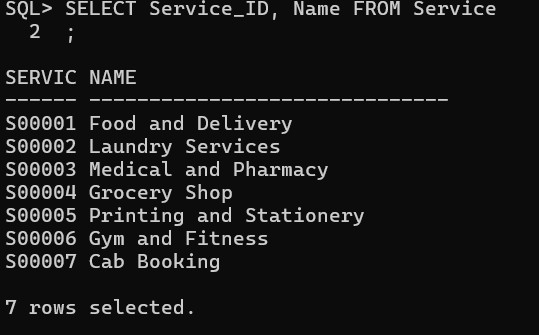
**6.3 Login (**The usual method for logging in to the site**) (5.2.1)**



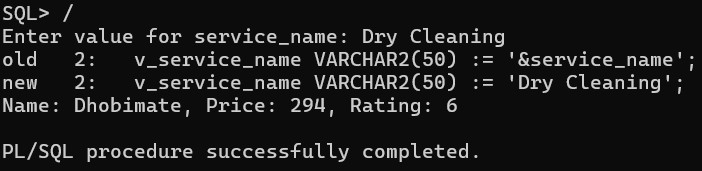
#### 6.4 Change the old password to a new one if user forgets their old password (5.3.8)



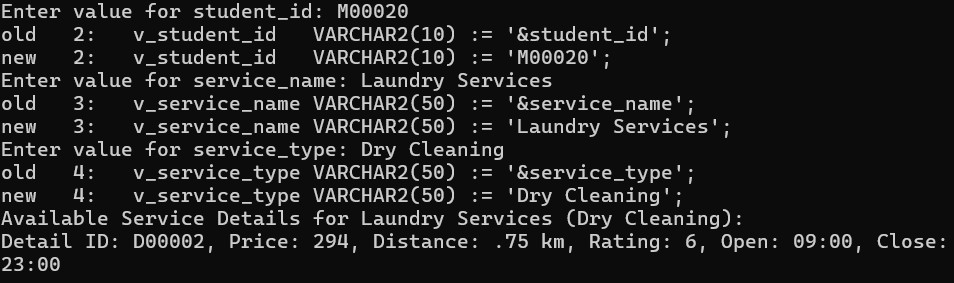
#### 6.5 Displaying the different types of Services Available



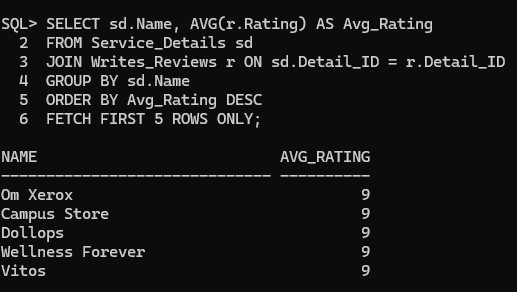
#### 6.6 Searching specific services based on the type of Service (5.3.4)



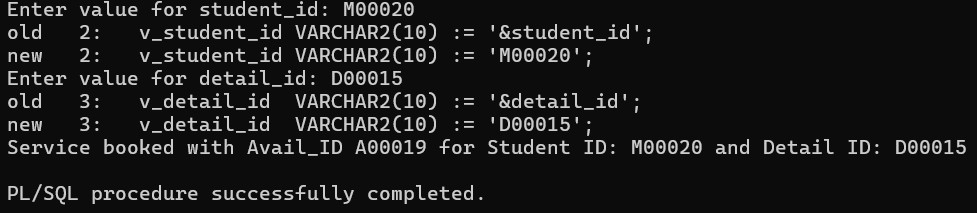
#### 6.7 Searching service sorted by distance from the Student’s Location (MIT) (5.3.5)



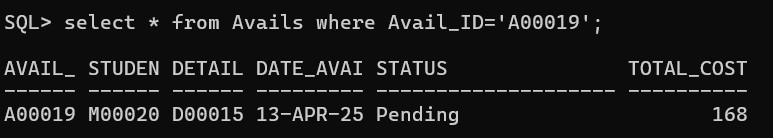
#### 6.8 Finding average ratings of the Services based of Student reviews (5.1.1)



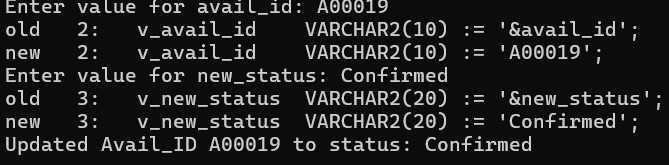
#### 6.9 Booking service by using their Detail ID (5.3.6)



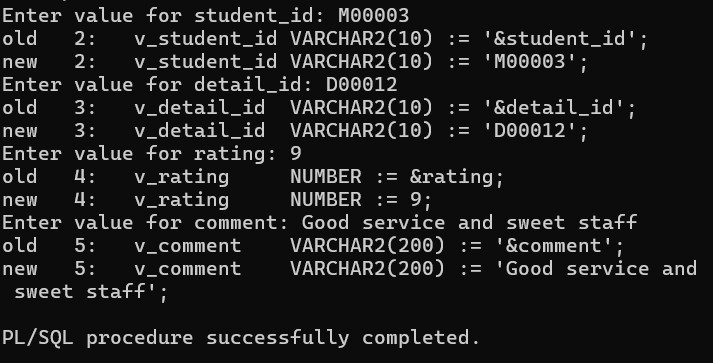
#### 6.10 Checking status of booking made by the student



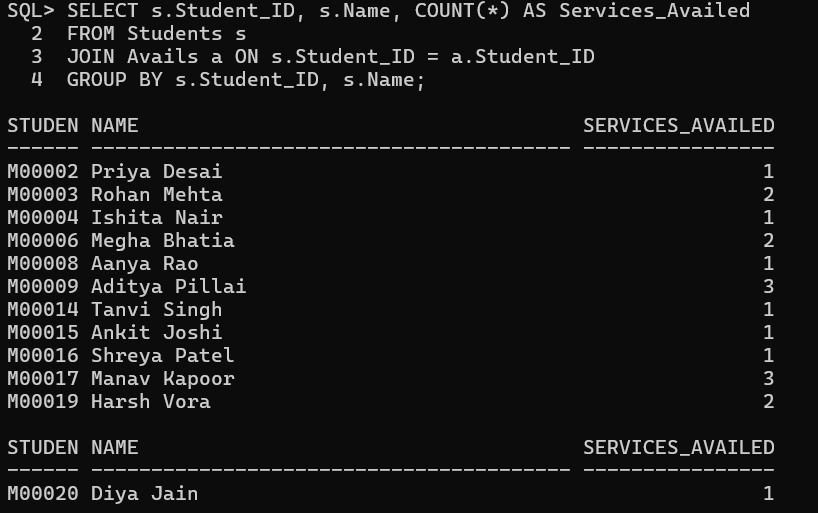
#### 6.11 Confirming the booking (updating booking status) of the student which is carried out by the service provider (5.3.7)



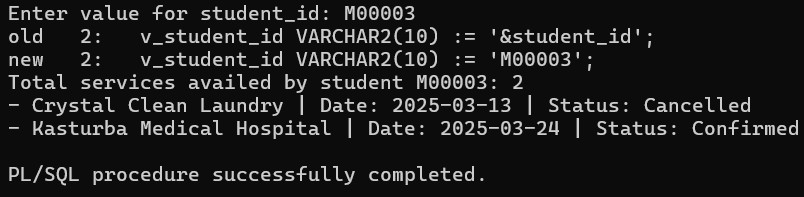
#### 6.12 Writing reviews for the service provided to the student (5.3.1)



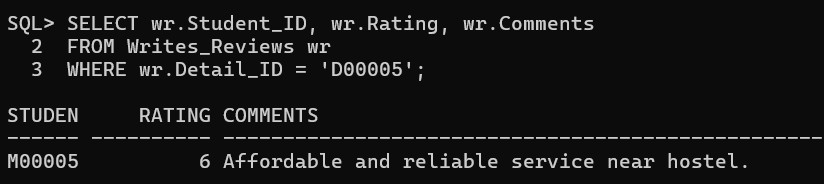
#### 6.13 Checking the number of bookings made by the students (specifically those who have made any bookings) (5.1.5)



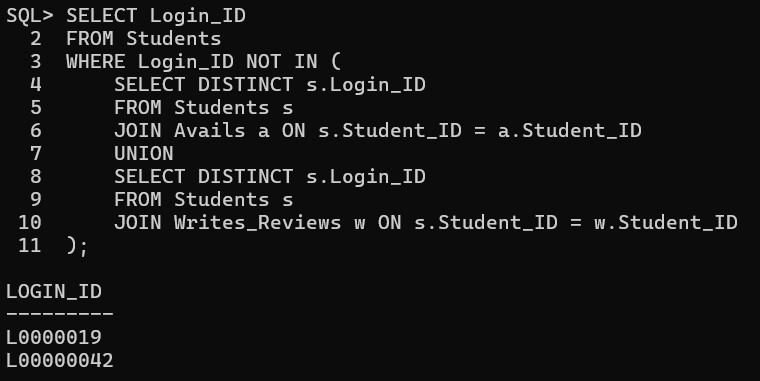
#### 6.14 Checking the number of bookings made by a specific student using their student id (5.4.2)



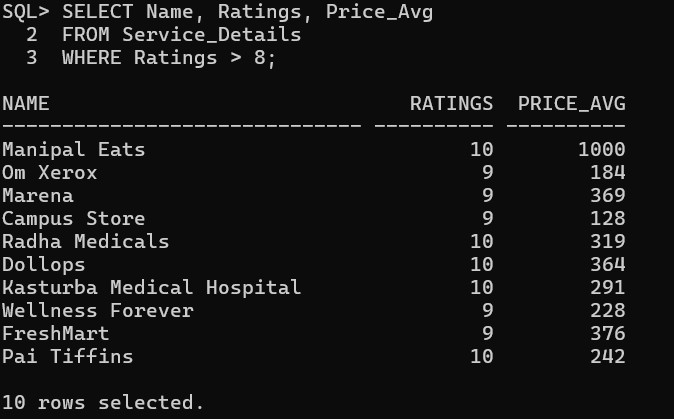
#### 6.15 Show reviews for a particular service provider (5.1.6)



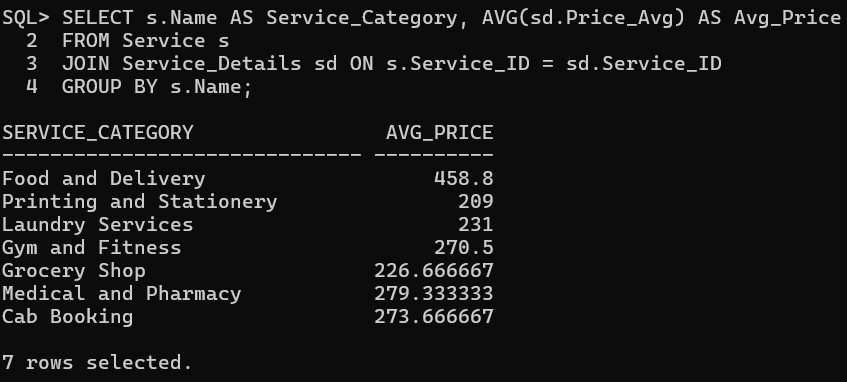
#### 6.16 Lists out all the students who have never submitted a review (5.1.2)



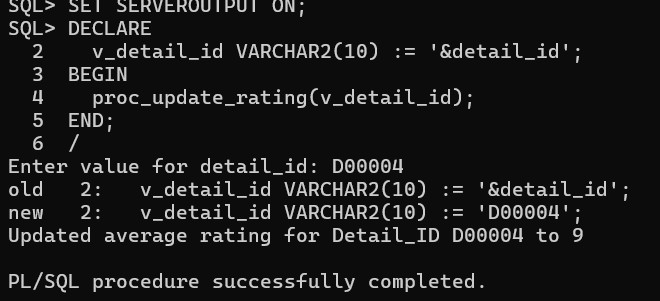
#### 6.17 Displays all the services that have their ratings above 8 (5.1.3)



#### 6.18 Displays the average cost of the different types of services (5.1.4)



#### 6.19 Updates the rating of a service with the average rating given for that service detail (5.4.1)



# 7. Conclusion and Future Work

### 7.1 Conclusion

The Student Guide to Manipal project successfully implements a robust and efficient backend database system that caters to the needs of students relocating to a new environment. It provides structured access to essential services such as food, laundry, gyms, medical stores, grocery outlets, hostels, and more. The system ensures seamless interaction between students and service providers by offering modules for secure registration, login authentication, service booking, review submission, and data analytics.

With features like auto-generation of unique IDs, login verification, distance-based service filtering, and modular procedures for various interactions, the system demonstrates the effectiveness of SQL and PL/SQL in handling real-world service management use cases. The triggers and functions ensure data integrity, while reports and queries aid in analyzing usage trends and service quality. The project achieves its objective of making student transitions smoother and improving service accessibility in and around the Manipal Institute of Technology campus.

### 7.2 Scope for Future Work

While the current system operates as a backend database, it opens up several possibilities for future enhancements:

* A user-friendly frontend web or mobile application can be developed to interact with the database, enabling live service booking and real-time updates.
* A payment gateway module can be integrated for online service payments and order tracking.
* Geolocation features can be added to dynamically calculate distance from the user's current location instead of hostel blocks.
* Admin panel functionalities can be extended to manage user accounts, deactivate outdated service providers, and view feedback reports.
* AI/ML integration could be used for personalized service recommendations based on past booking behavior.

By expanding the current structure and incorporating these features, the Student Guide to Manipal system can evolve into a complete digital assistant for students and campus service providers alike.