AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH-(AIUB)

Department of Computer Science & Engineering

Course: ADVANCE DATABASE MANAGEMENT SYSTEM

SUMMER 2024-2025

Section : A

Final Term Project

Project Report On : MALL MANAGEMENT SYSTEM

Supervised By

Faculty: JUENA AHMED NOSHIN

Submitted By

Project Member:

NAME	ID	Contribution Percentage
Sumaiya Tasnim	23-50014-1	**25%**

Submission Deadline : 16-09-2025

TABLE OF CONTENTS

“Mall Management System”

1.Introduction

The ***Mall Management System*** is a database-driven application designed to streamline the daily operations of a shopping mall. From managing shops and shopkeepers to handling customer data, inventory, billing, and maintenance records, the system provides a centralized and structured way to organize all essential information. It helps reduce manual errors, saves time, and ensures that operations run smoothly and efficiently.Malls typically generate and handle large amounts of data every day. Without a proper system in place, this can quickly become overwhelming. The Mall Management System addresses this challenge by offering an easy-to-use interface backed by a well-designed database that ensures secure data storage, quick access, and consistent updates.This project highlights the practical use of sound database design principles and demonstrates how technology can simplify complex real-world processes, improve coordination, and support better management decisions.

2.Project Proposal

Objectives

- To create a system that helps manage mall activities in an organized way.
- To reduce manual work and avoid mistakes in storing and updating data.
- To keep all shop, customer, and inventory details in one place for easy access.
- To make billing and reporting faster and more accurate.
- To support mall staff in managing shops, customers, and stock more smoothly.

Problem Statement

Managing a shopping mall involves handling a lot of information like shop details, customer records, inventory, and billing. Doing all of this manually can take a lot of time and may lead to mistakes or missing data. It also becomes hard to find or update information quickly. Because of this, mall operations can slow down and become less efficient. A proper system is needed to store all data in one place, reduce manual work, and help the mall run more smoothly.

Methodology

We will develop the project through a clear step-by-step approach focused on database design and implementation:

Requirement Analysis: We will study the mall management needs to identify what data should be stored and managed.

Database Design: We will create an Entity-Relationship (ER) diagram to organize data and define relationships between entities such as shops, customers, and inventory.

Database Implementation: We will build the database using Oracle 10g, creating tables, keys, and constraints to maintain data integrity.

Query Development: We will write SQL queries, stored procedures, and triggers to handle data operations like inserting, updating, deleting, and generating reports.

Documentation: We will prepare clear documentation detailing the database schema, queries, and usage instructions.

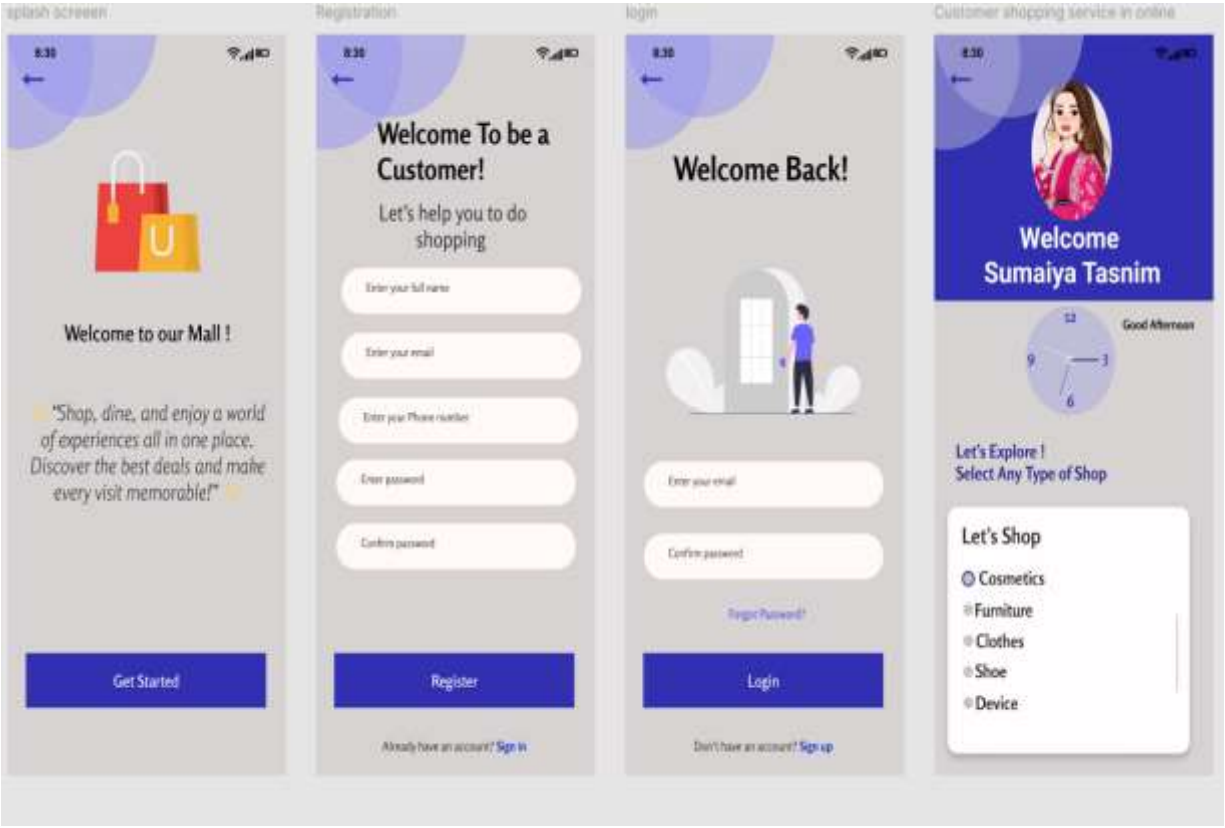
System Features

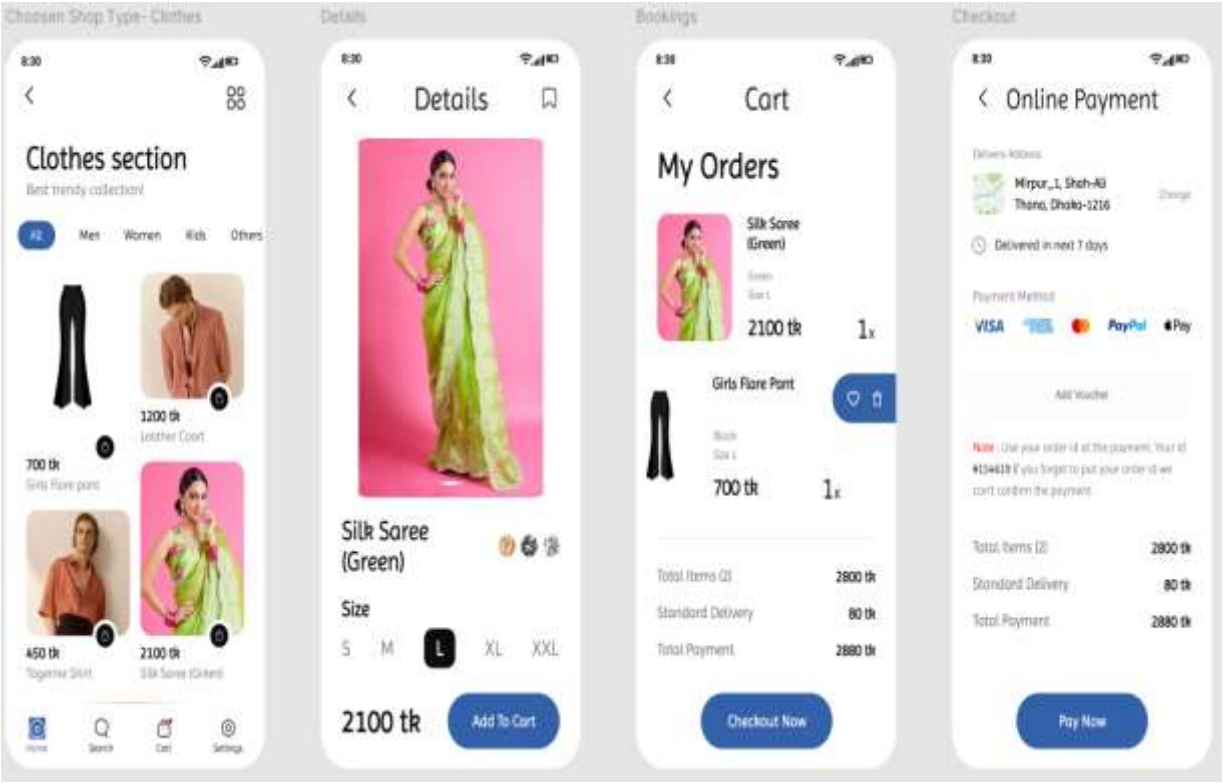
1. Add, update, and delete shop details easily.
2. Manage shopkeeper and employee information.
3. Store and access customer details and purchase history.
4. Keep track of inventory and notify when stock is low.
5. Generate bills and invoices automatically.
6. Create reports on sales, inventory, and maintenance.
7. Handle maintenance requests and track their status.

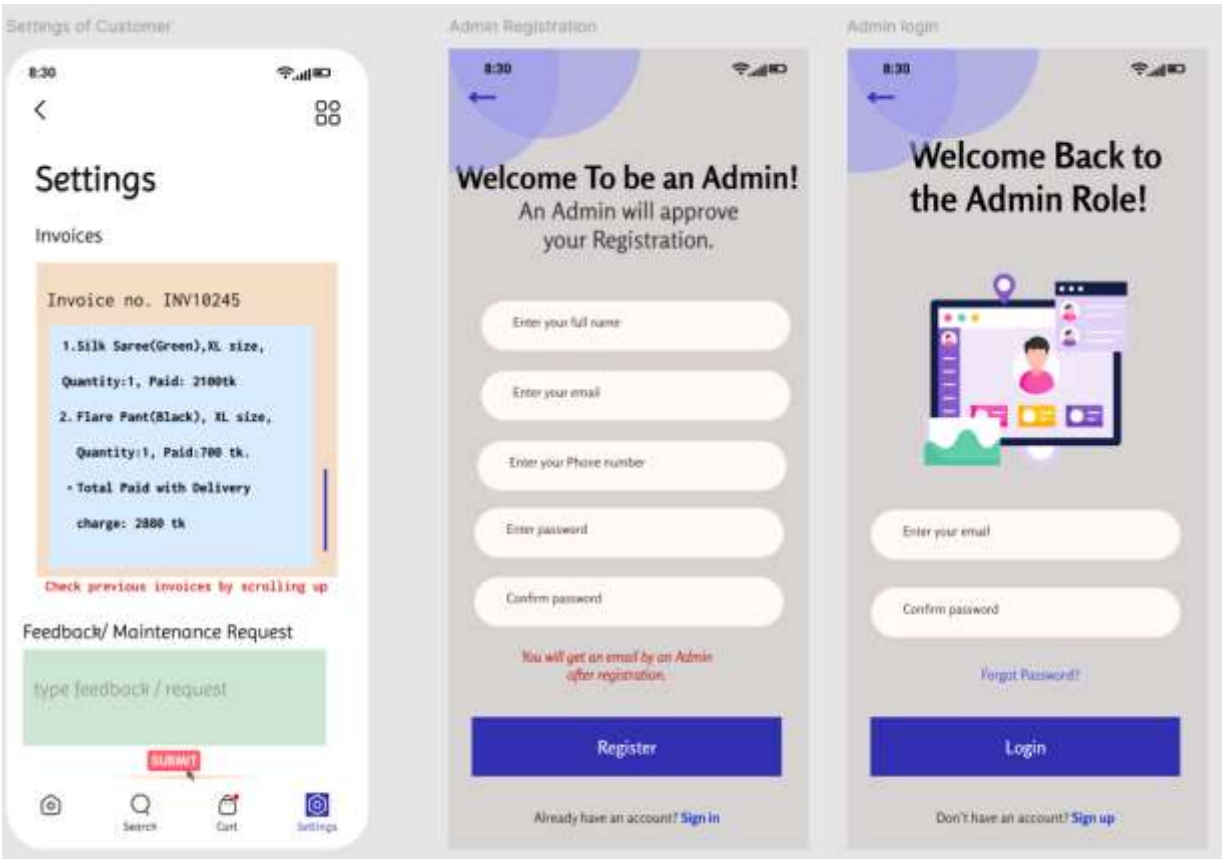
Expected Outcome

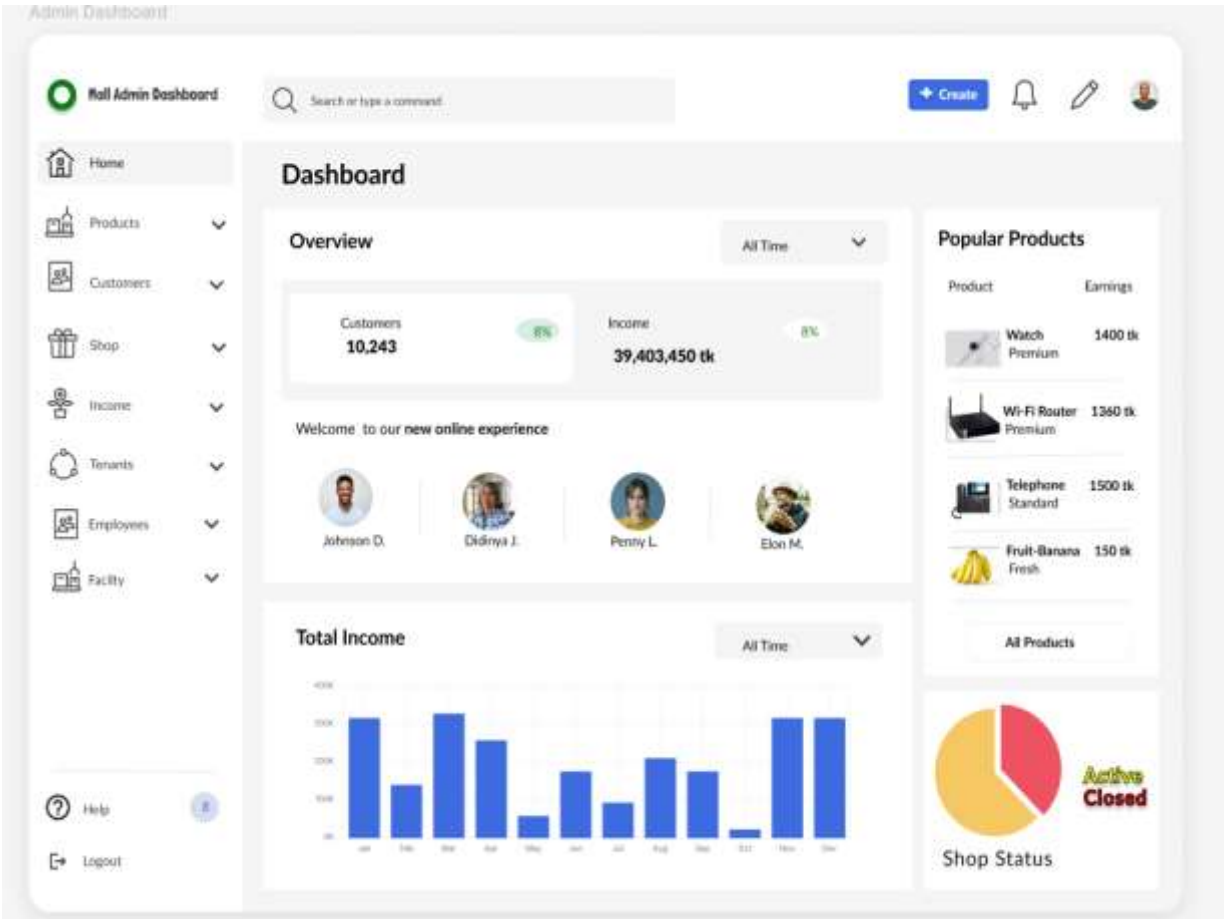
We expect to develop a reliable and efficient database system that centralizes all mall management data in one place. This system will allow us to store, update, and retrieve information quickly and accurately using SQL operations. By automating tasks like billing, inventory tracking, and report generation, we will reduce manual errors and save time. The final outcome will be a well-organized database that supports mall administrators and customers in managing daily operations smoothly and effectively.

3.User Interface Planning






Settings of Customer
8:30
Settings
Invoices
Invoice no. INV10245
1.Silk Saree(Green),XL size,
Quantity:1, Paid: 2100tk
2. Flare Pant(Black), XL size,
Quantity:1, Paid:700 tk.
- Total Paid with Delivery
charge: 2880 tk
Check previous invoices by scrolling up
Feedback/ Maintenance Request
type feedback / request
SUBMIT
Search
Cart
Settings
Admin Registration
8:30
Welcome To be an Admin!
An Admin will approve
your Registration.
Enter your full name
Enter your email
Enter your Phone number
Enter password
Confirm password
You will get an email by an Admin
after registration.
Register
Already have an account? Sign in
Admin login
8:30
Welcome Back to
the Admin Role!
Enter your email
Confirm password
Forgot Password?
Login
Don't have an account? Sign up


Admin Dashboard
Roll Admin Dashboard
Search or type a command
Create
Home
Products
Customers
Shop
Income
Tenants
Employees
Facility
Help
Logout
Dashboard
Overview
All Time
Customers
10,243
8%
Income
39,403,450 tk
8%
Welcome to our new online experience
Johnson D.
Didinya J.
Penny L.
Elon M.
Total Income
All Time
Popular Products
Product
Earnings
Watch
Premium
1400 tk
Wi-Fi Router
Premium
1360 tk
Telephone
Standard
1500 tk
Fruit-Banana
Fresh
150 tk
All Products
Active
Closed
Shop Status

Admin Dashboard

Mall Admin Dashboard

Search or type a command

+ Create

Home

Products

Customers

Shop

Income

Tenants

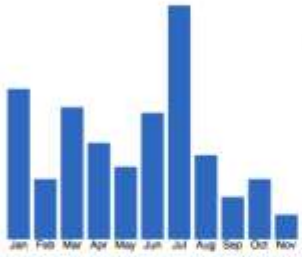
Employees

Facilty

Help

Logout

Products Details Overview



Monthly Sales by total Product

Most sold out Products

Clothing - 70%
Furniture - 50%
Shoe - 20%
Device - 10%

Products-wise-stock levels

1. Iphone -13 pro max
2. Hp Laptop
3. Silk Saree
4. Reading Table
5. Fit Me Foundation
6. MAC Charger
7. Glass Bottle
8. T-Shirt (kids-spiderman)

Total Income Loss

30500tk

Admin Dashboard

Mall Admin Dashboard

Search or type a command

+ Create

Home

Products

Customers

Shop

Income

Tenants

Employees

Facilty

Help

Logout

Manage Customer Details

Customer_ID	Customer_name	Customer_email	Visit_date	Feedback	Customer_phone
A21	Mahmud Hasan	mahmud.hasan@example.com	2025-08-10	Very satisfied	01711220033
G23	Laila Akter	laila.akter@example.com	2025-08-11	Good service	01822334455
T16	Imran Hossain	imran.hossain@example.com	2025-08-12	Average experience	01566778844
B09	Nusrat Jahan	nusrat.jahan@example.com	2025-08-13	Excellent	01944556677
K31	Abdullah Al Mamun	abdullah.mamun@example.com	2025-08-14	Needs improvement	01688990011
R45	Sharmin Sultana	sharmin.sultana@example.com	2025-08-15	Very friendly staff	01799887766
M55	Farid Ahmed	farid.ahmed@example.com	2025-08-16	Quick service	01877665544
Z12	Rukhsana Begum	rukhsana.begum@example.com	2025-08-17	Excellent support	01733445522
P88	Kamrul Hasan	kamrul.hasan@example.com	2025-08-18	Satisfied	01522334455
D07	Anika Chowdhury	anika.chowdhury@example.com	2025-08-19	Could be better	01911223344

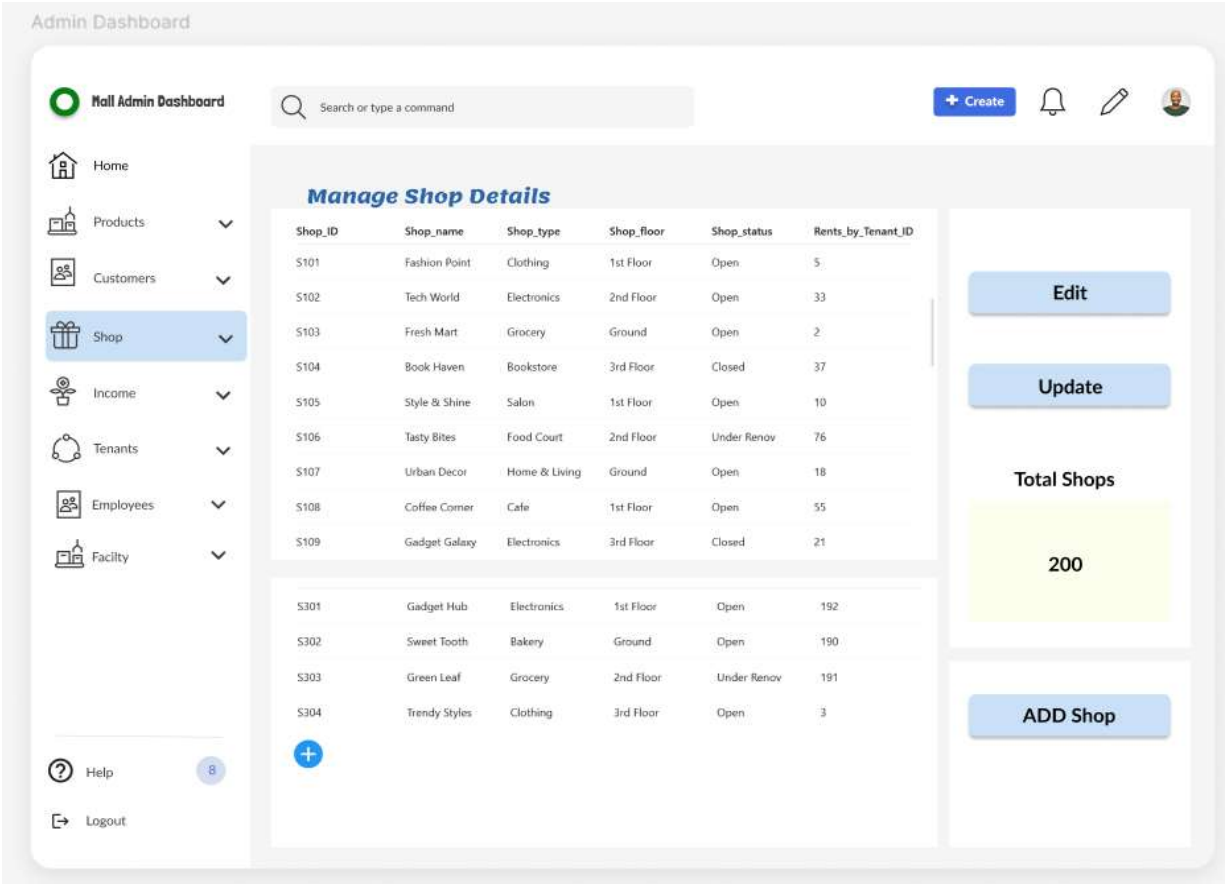
Edit

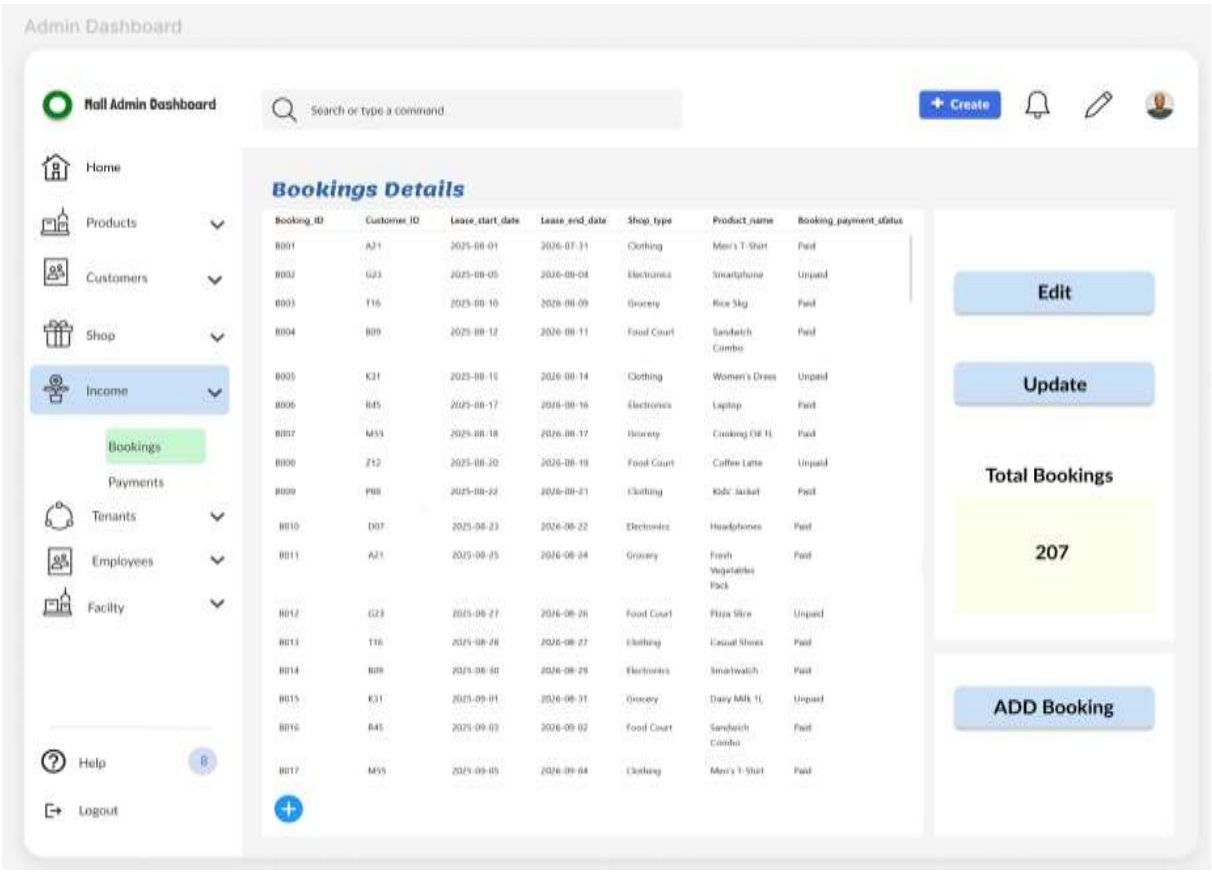
Update

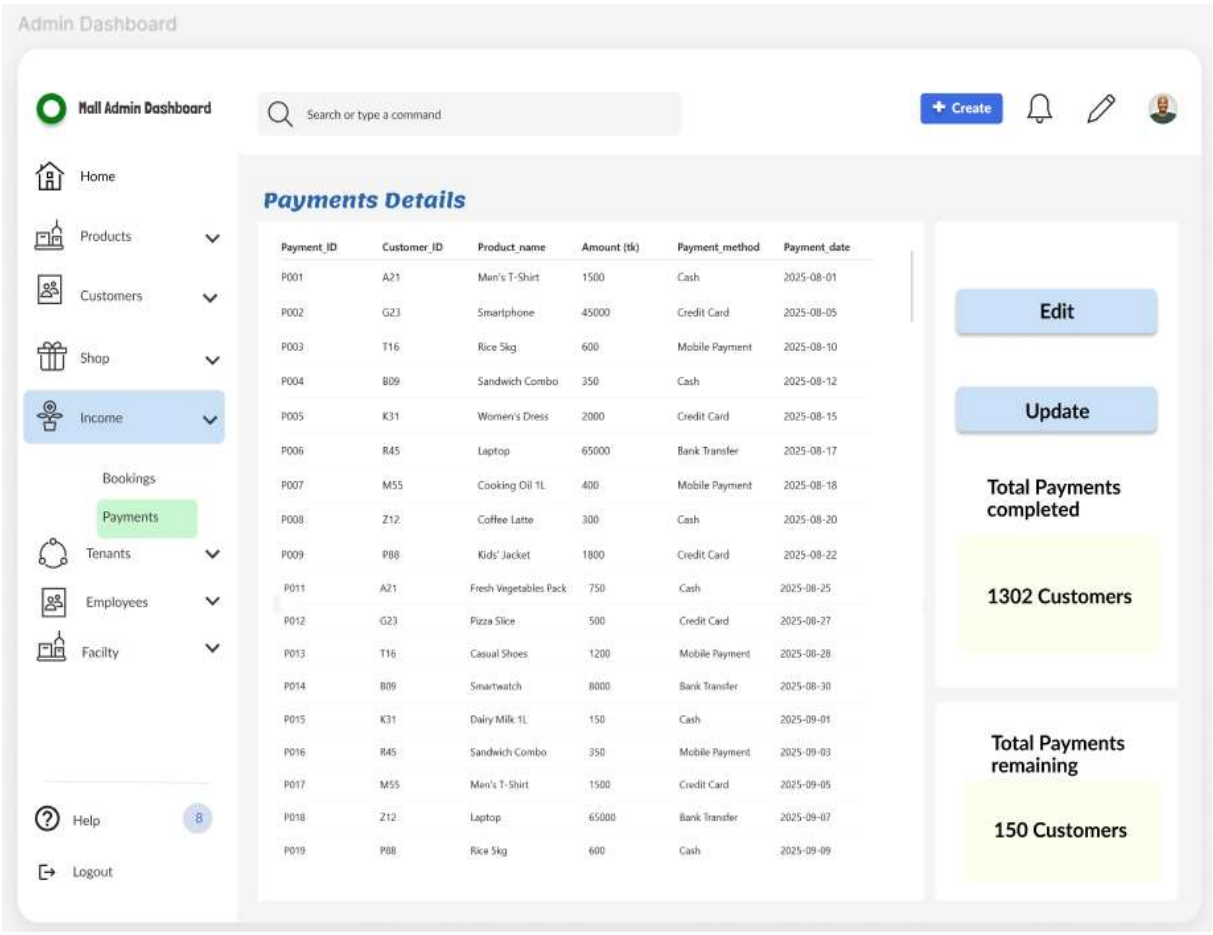
Total Customers

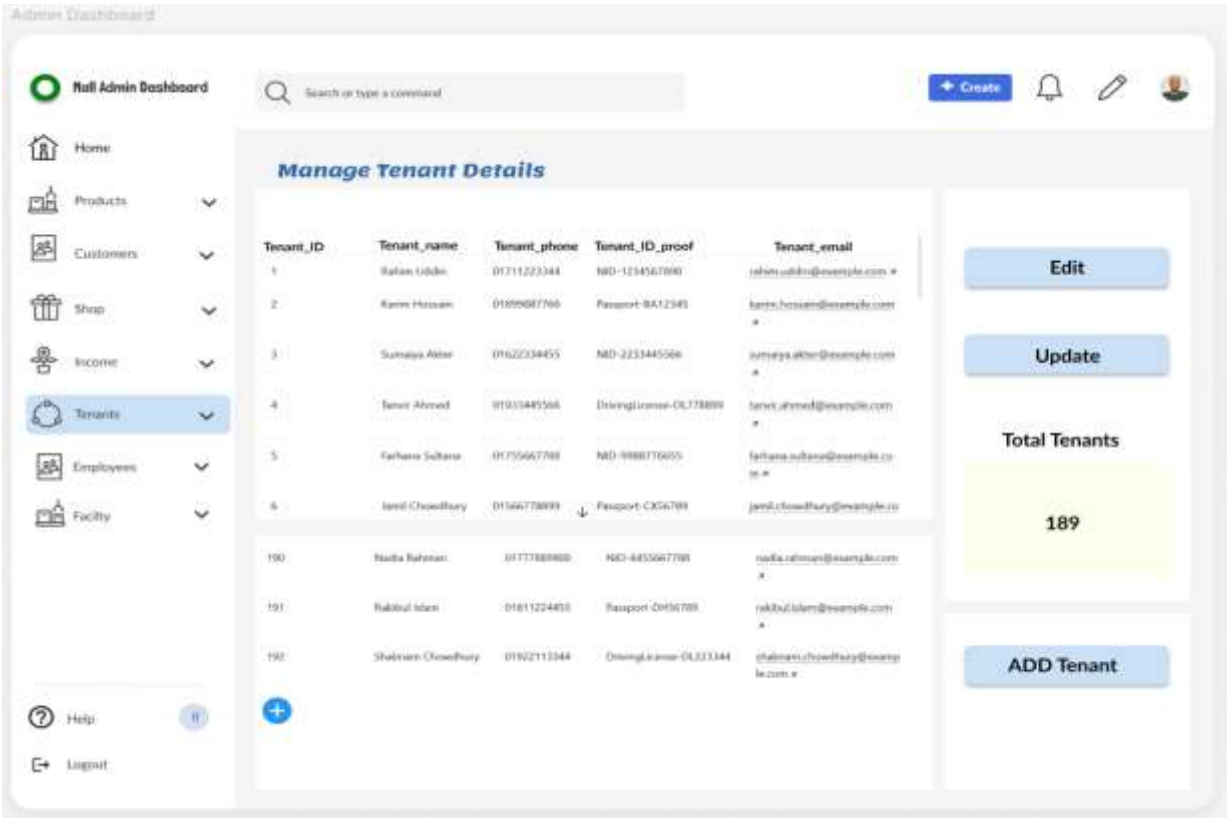
11203

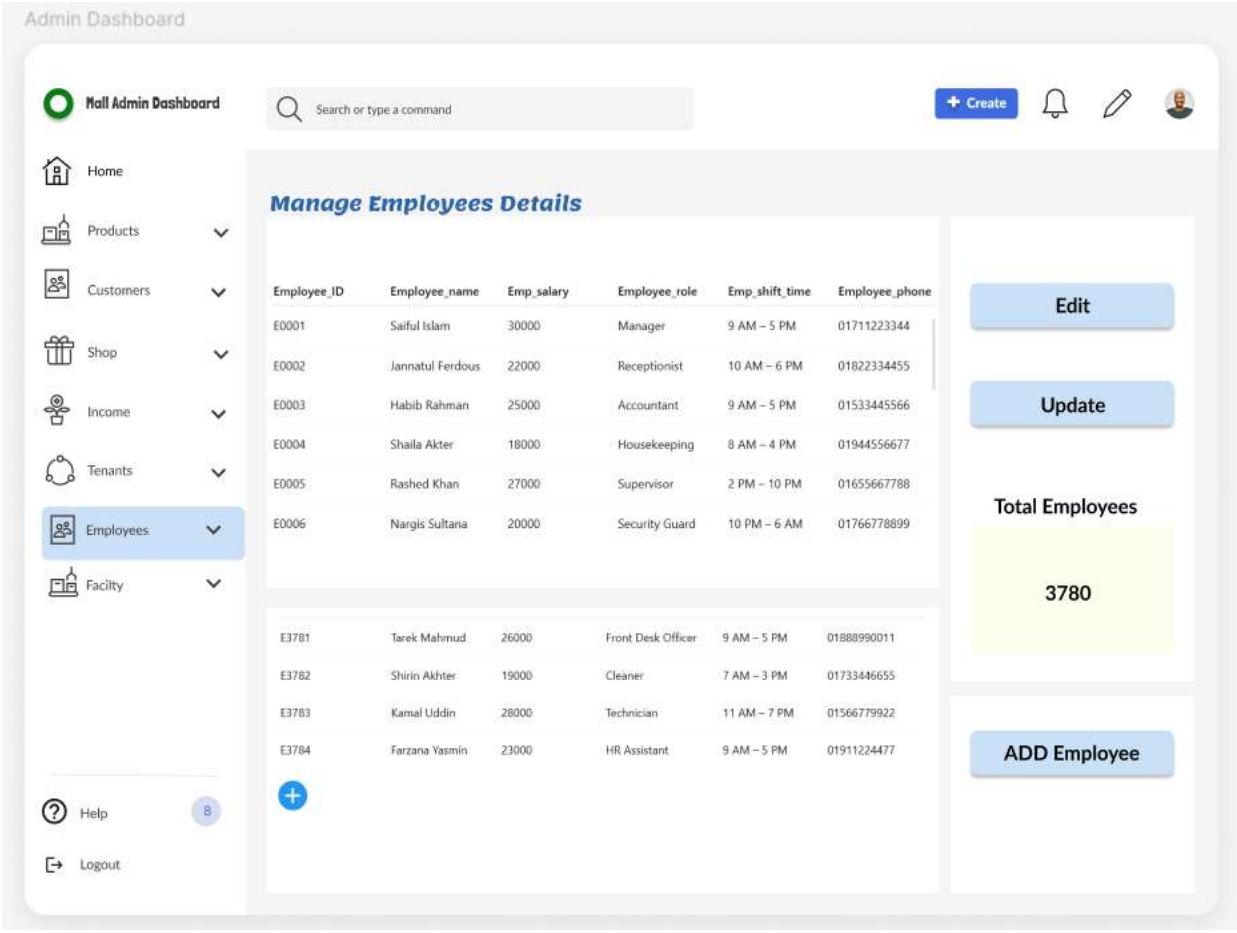
ADD Customer

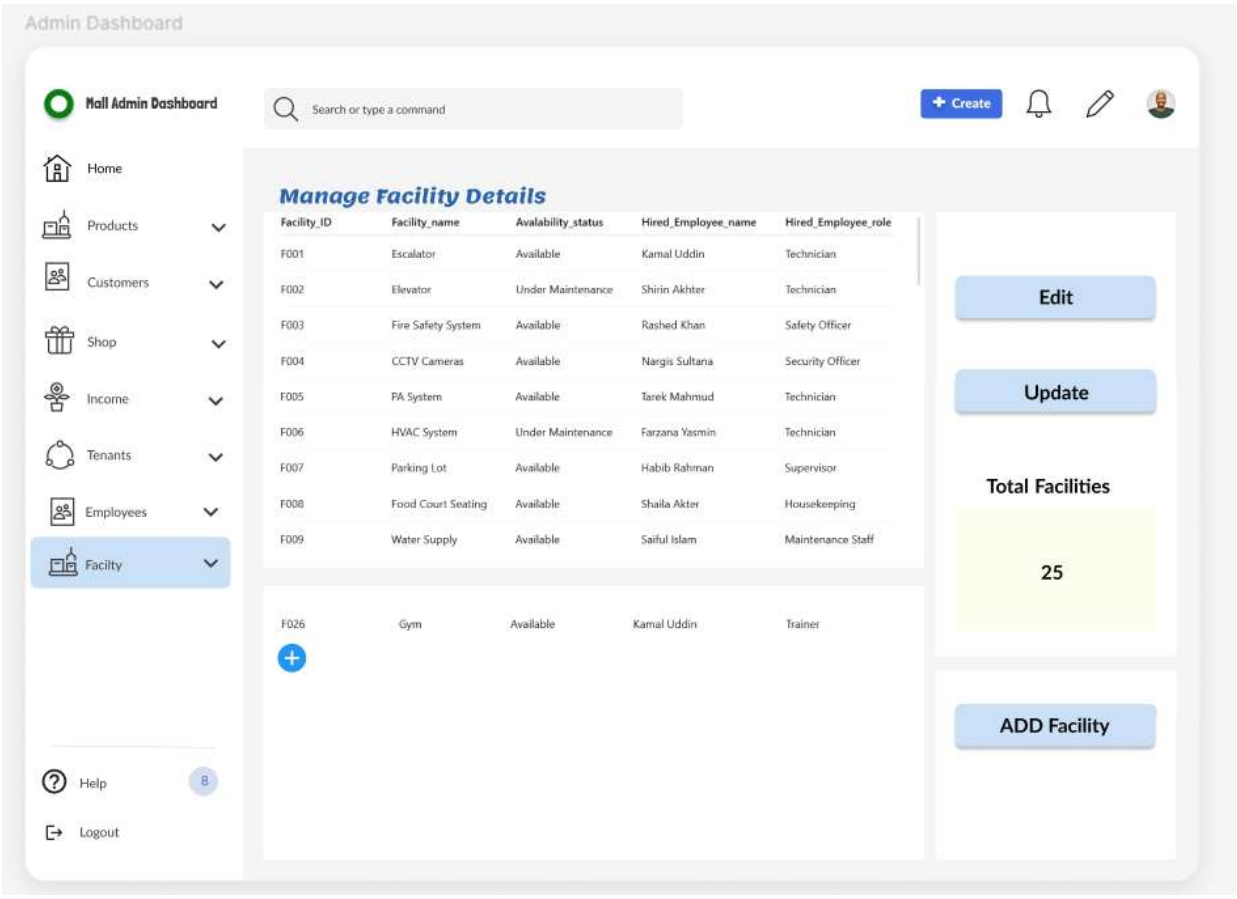

Admin Dashboard
Mall Admin Dashboard
Search or type a command
+ Create
Home
Products
Customers
Shop
Income
Tenants
Employees
Facilty
Help
8
Logout
Manage Shop Details
Shop_ID
Shop_name
Shop_type
Shop_floor
Shop_status
Rents_by_Tenant_ID
S101 Fashion Point Clothing 1st Floor Open 5
S102 Tech World Electronics 2nd Floor Open 33
S103 Fresh Mart Grocery Ground Open 2
S104 Book Haven Bookstore 3rd Floor Closed 37
S105 Style & Shine Salon 1st Floor Open 10
S106 Tasty Bites Food Court 2nd Floor Under Renov 76
S107 Urban Decor Home & Living Ground Open 18
S108 Coffee Corner Cafe 1st Floor Open 55
S109 Gadget Galaxy Electronics 3rd Floor Closed 21
S301 Gadget Hub Electronics 1st Floor Open 192
S302 Sweet Tooth Bakery Ground Open 190
S303 Green Leaf Grocery 2nd Floor Under Renov 191
S304 Trendy Styles Clothing 3rd Floor Open 3
Edit
Update
Total Shops
200
ADD Shop

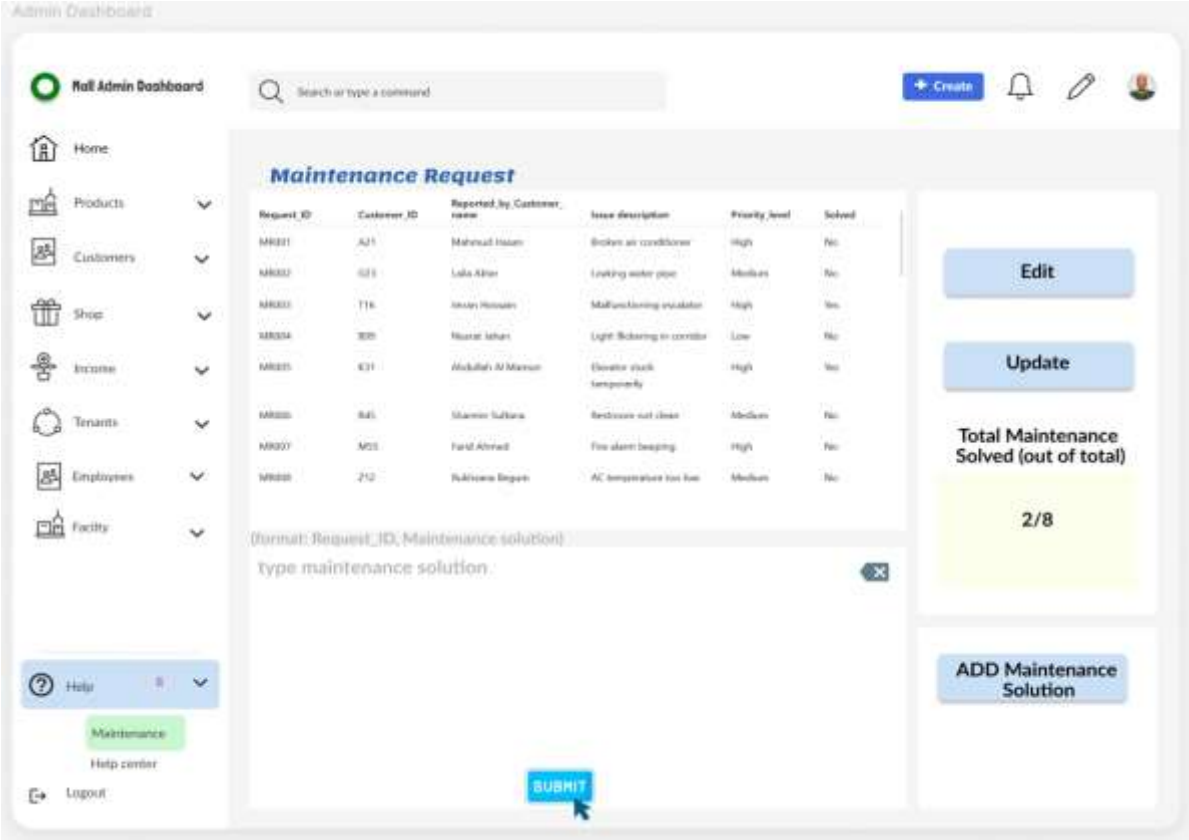

Admin Dashboard
Mall Admin Dashboard
Search or type a command
+ Create
Home
Products
Customers
Shop
Income
Bookings
Payments
Tenants
Employees
Facilty
Help
8
Logout
Bookings Details
Booking_ID
Customer_ID
Lease_start_date
Lease_end_date
Shop_type
Product_name
Booking_payment_status
Edit
Update
Total Bookings
207
ADD Booking


Admin Dashboard
Mall Admin Dashboard
Search or type a command
Create
Home
Products
Customers
Shop
Income
Bookings
Payments
Tenants
Employees
Facilty
Help
Logout
Payments Details
Payment_ID Customer_ID Product_name Amount (tk) Payment_method Payment_date
P001 A21 Men's T-Shirt 1500 Cash 2025-08-01
P002 G23 Smartphone 45000 Credit Card 2025-08-05
P003 T16 Rice 5kg 600 Mobile Payment 2025-08-10
P004 B09 Sandwich Combo 350 Cash 2025-08-12
P005 K31 Women's Dress 2000 Credit Card 2025-08-15
P006 R45 Laptop 65000 Bank Transfer 2025-08-17
P007 M55 Cooking Oil 1L 400 Mobile Payment 2025-08-18
P008 Z12 Coffee Latte 300 Cash 2025-08-20
P009 P88 Kids' Jacket 1800 Credit Card 2025-08-22
P011 A21 Fresh Vegetables Pack 750 Cash 2025-08-25
P012 G23 Pizza Slice 500 Credit Card 2025-08-27
P013 T16 Casual Shoes 1200 Mobile Payment 2025-08-28
P014 B09 Smartwatch 8000 Bank Transfer 2025-08-30
P015 K31 Dairy Milk 1L 150 Cash 2025-09-01
P016 R45 Sandwich Combo 350 Mobile Payment 2025-09-03
P017 M55 Men's T-Shirt 1500 Credit Card 2025-09-05
P018 Z12 Laptop 65000 Bank Transfer 2025-09-07
P019 P88 Rice 5kg 600 Cash 2025-09-09
Edit
Update
Total Payments completed
1302 Customers
Total Payments remaining
150 Customers


Mall Admin Dashboard
Search or type a command
Create
Home
Products
Customers
Shop
Income
Tenants
Employees
Facilty
Help
Logout
Manage Tenant Details
Tenant_ID Tenant_name Tenant_phone Tenant_ID_proof Tenant_email
Edit
Update
Total Tenants
189
ADD Tenant


Admin Dashboard
Mall Admin Dashboard
Search or type a command
+ Create
Home
Products
Customers
Shop
Income
Tenants
Employees
Facilty
Help
8
Logout
Manage Employees Details
Employee_ID
Employee_name
Emp_salary
Employee_role
Emp_shift_time
Employee_phone
E0001 Saiful Islam 30000 Manager 9 AM – 5 PM 01711223344
E0002 Jannatul Ferdous 22000 Receptionist 10 AM – 6 PM 01822334455
E0003 Habib Rahman 25000 Accountant 9 AM – 5 PM 01533445566
E0004 Shaila Akter 18000 Housekeeping 8 AM – 4 PM 01944556677
E0005 Rashed Khan 27000 Supervisor 2 PM – 10 PM 01655667788
E0006 Nargis Sultana 20000 Security Guard 10 PM – 6 AM 01766778899
E3781 Tarek Mahmud 26000 Front Desk Officer 9 AM – 5 PM 01888990011
E3782 Shirin Akhter 19000 Cleaner 7 AM – 3 PM 01733446655
E3783 Kamal Uddin 28000 Technician 11 AM – 7 PM 01566779922
E3784 Farzana Yasmin 23000 HR Assistant 9 AM – 5 PM 01911224477
Edit
Update
Total Employees
3780
ADD Employee


Admin Dashboard
Mall Admin Dashboard
Search or type a command
+ Create
Home
Products
Customers
Shop
Income
Tenants
Employees
Facilty
Help
8
Logout
Manage Facility Details
Facility_ID
Facility_name
Avalability_status
Hired_Employee_name
Hired_Employee_role
F001 Escalator Available Kamal Uddin Technician
F002 Elevator Under Maintenance Shirin Akhter Technician
F003 Fire Safety System Available Rashed Khan Safety Officer
F004 CCTV Cameras Available Nargis Sultana Security Officer
F005 PA System Available Tarek Mahmud Technician
F006 HVAC System Under Maintenance Farzana Yasmin Technician
F007 Parking Lot Available Habib Rahman Supervisor
F008 Food Court Seating Available Shaila Akter Housekeeping
F009 Water Supply Available Saiful Islam Maintenance Staff
F026 Gym Available Kamal Uddin Trainer
Edit
Update
Total Facilities
25
ADD Facility

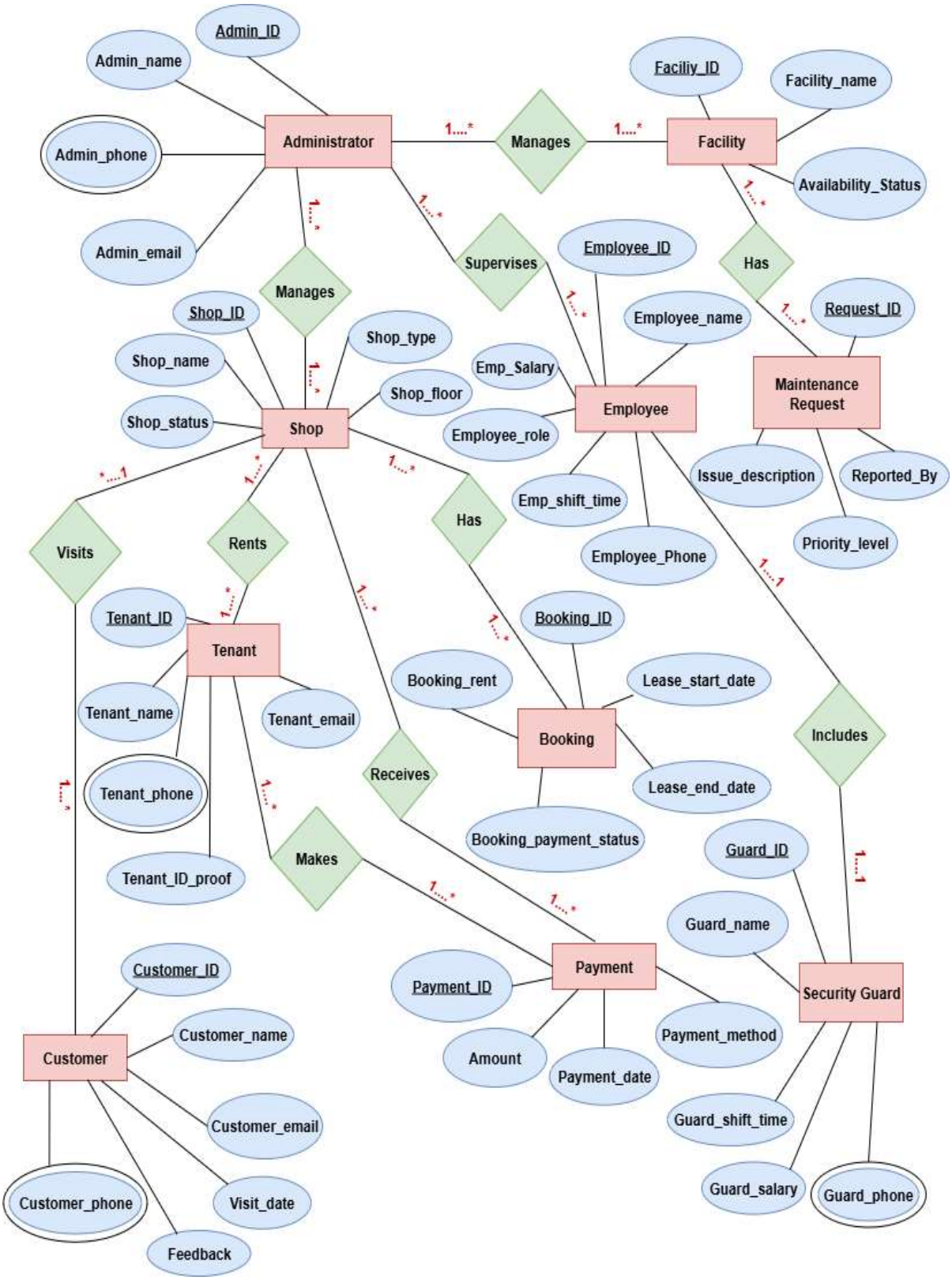


4.Scenario Description

In a Mall Management System, one administrator manages many shops, facilities and supervises employees. Each administrator has a unique identification number, name, phone number, and email. A shop has a unique identification number, name, type, floor, and current status. Each shop is rented by exactly one tenant. One tenant may rent one or more shops. A tenant is defined by a unique tenant ID, name, phone number, email and ID proof.A shop must have a booking record. One shop can have one or more bookings, but a booking is related to one shop only. A booking is defined by a booking ID, rent, lease start and end dates, and payment status. A payment is made by a tenant for a shop. One payment is linked to exactly one tenant and one shop. Payment is identified by a unique payment ID, amount, date and method.A shop can have many customers. Each customer has a unique identification number, name, phone number, email, visit date, and feedback. A customer can have more than one phone number and email.Each employee works under the administrator and may include cleaners, support staff, and other roles. An employee has a unique identification number, name, role, shift time, salary and phone number. The mall provides various facilities like elevators, parking, and restrooms. Each facility has a unique identification number, name, and availability status. Facilities may be associated with one or more maintenance requests. A maintenance request is defined by a request ID, facility ID, issue description, reported by, priority level. Security guards are also part of the employee group, but are maintained separately. Each security guard has a unique identification number, name, shift time, phone number

and salary. The system offers a centralized platform for efficient administration, ensuring that all operations—from shop allocation to facility monitoring—run seamlessly to enhance service quality and overall mall management.

5.ER Diagram



6.Normalization

Manages (Administrator-Shop)

UNF

Manages (Admin_ID, Admin_name, Admin_phone, Admin_email, Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor)

1NF

Admin_phone is a multi valued attribute.

1. Admin_ID, Admin_name, Admin_phone, Admin_email, Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

2NF

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

3NF

There is no transitive dependency. Relation already in 3NF.

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

Table Creation

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, **A_ID**

Manages (Administrator-Facility)

UNF
Manages (Admin_ID, Admin_name, Admin_phone, Admin_email, Facility_ID, Facility_name, Availability_Status)

1NF
Admin_phone is a multivalued attribute.

1. Admin_ID, Admin_name, Admin_phone, Admin_email, Facility_ID, Facility_name, Availability_Status

2NF

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Facility_ID, Facility_name, Availability_Status

3NF
There is no transitive dependency. Relation already in 3NF.

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Facility_ID, Facility_name, Availability_Status

Table Creation

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Facility_ID, Facility_name, Availability_Status, **A_ID**

Supervises (Administrator-Employee)

UNF
Supervises (Admin_ID, Admin_name, Admin_phone, Admin_email, Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time)

1NF
Admin_phone and Employee_phone are multi valued attributes.

1 Admin_ID, Admin_name, Admin_phone, Admin_email

2.Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time

2NF
1. Admin_ID, Admin_name, Admin_phone, Admin_email

2.Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time

3NF
There is no transitive dependency. Relation already in 3NF.

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time

Table Creation

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time, **A_ID**

Has (Facility - Maintenance Request)

UNF
Has (Facility_ID, Facility_name, Availability_Status, Request_ID, Issue_description, Reported_by, Priority_level)

1NF
There is no multi valued attribute. Relation already in 1NF.

1. Facility_ID, Facility_name, Availability_Status
2. Request_ID, Issue_description, Reported_by, Priority_level

2NF

1. Facility_ID, Facility_name, Availability_Status
2. Request_ID, Issue_description, Reported_by, Priority_level

3NF
There is no transitive dependency. Relation already in 3NF.

1. Facility_ID, Facility_name, Availability_Status
2. Request_ID, Issue_description, Reported_by, Priority_level

Table Creation

1. Facility_ID, Facility_name, Availability_Status
2. Request_ID, Issue_description, Reported_by, Priority_level, **F_ID**

Rents (Tenant-Shop)

UNF
Rents (Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof, Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor)

1NF
Tenant_phone is multi valued attribute.

1. Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

2NF

1. Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

3NF
There is no transitive dependency. Relation already in 3NF.

1. Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

Table Creation

1. Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, **T_ID**

Makes (Tenant - Payment)

UNF
Makes (Tenant_ID, Tenant_name, Tenant_phone, Tenant_email, Tenant_ID_proof, Payment_ID, Amount, Payment_date, Payment_method)

1NF
Tenant_phone is a multi valued attribute.

1. Tenant_ID, Tenant_name, Tenant_phone, Tenant_email, Tenant_ID_proof, Payment_ID, Amount, Payment_date, Payment_method

2NF
1. Tenant_ID, Tenant_name, Tenant_phone, Tenant_email, Tenant_ID_proof

2. Payment_ID, Amount, Payment_date, Payment_method

3NF
There is no transitive dependency. Relation already in 3NF.

1. Tenant_ID, Tenant_name, Tenant_phone, Tenant_email, Tenant_ID_proof
2. Payment_ID, Amount, Payment_date, Payment_method

Table Creation

1. Tenant_ID, Tenant_name, Tenant_phone, Tenant_email, Tenant_ID_proof
2. Payment_ID, Amount, Payment_date, Payment_method, **T_ID**

Receives (Shop - Payment)

UNF
Receives (Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, Payment_ID, Amount, Payment_date, Payment_method)

1NF
There is no multivalued attribute. Relation is already in 1NF.

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, Payment_ID, Amount, Payment_date, Payment_method

2NF
1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

2. Payment_ID, Amount, Payment_date, Payment_method

3NF
There is no transitive dependency. Relation already in 3NF.

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

2.Payment_ID, Amount, Payment_date, Payment_method

Table Creation

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor
2. Payment_ID, Amount, Payment_date, Payment_method, **S_ID**

Has (Shop - Booking)

UNF
Has (Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status)

1NF
There is no multivalued attribute. Relation is already in 1NF.

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status

2NF

1.Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

2. Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status

3NF
There is no transitive dependency. Relation already in 3NF.

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor
2. Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status

Table Creation

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor
2. Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status, **S_ID**

Visits (Shop - Customer)

UNF
Visits (Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback)

1NF
Customer_phone is a multi valued attribute.

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback

2NF
1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor

2. Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback

3NF
There is no transitive dependency. Relation already in 3NF.

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor
2. Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback

Table Creation

1. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor
2. Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback, **S_ID**

Includes (Employee – Security Guard)

UNF
Includes (Employee_ID, Employee_name, Emp_salary, Employee_role, Emp_shift_time, Employee_phone, Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary)

1NF
Employee_phone and Guard_phone are multi valued attributes.

1. Employee_ID, Employee_name, Emp_salary, Employee_role, Emp_shift_time, Employee_phone, Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary

2NF

1. Employee_ID, Employee_name, Emp_salary, Employee_role, Emp_shift_time, Employee_phone
2. Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary

3NF
There is no transitive dependency. Relation already in 3NF.

1. Employee_ID, Employee_name, Emp_salary, Employee_role, Emp_shift_time, Employee_phone
2. Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary, Employee_ID

Table Creation

1. Employee_ID, Employee_name, Emp_salary, Employee_role, Emp_shift_time, Employee_phone
2. Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary, **E_ID**

Temporary Tables

1. Admin_ID, Admin_name, Admin_phone, Admin_email
2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, **A_ID**
~~3. Admin_ID, Admin_name, Admin_phone, Admin_email~~
4. Facility_ID, Facility_name, Availability_Status, **A_ID**
~~5. Admin_ID, Admin_name, Admin_phone, Admin_email~~
6. Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time, **A_ID**
~~7. Facility_ID, Facility_name, Availability_Status~~
8. Request_ID, Issue_description, Reported_by, Priority_level, **F_ID**
9. Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof
~~10. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, **T_ID**~~
~~11. Tenant_ID, Tenant_name, Tenant_phone, Tenant_email, Tenant_ID_proof~~
12. Payment_ID, Amount, Payment_date, Payment_method, **T_ID**
~~13. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor~~
~~14. Payment_ID, Amount, Payment_date, Payment_method, **S_ID**~~
~~15. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor~~
16. Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status, **S_ID**
~~17. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor~~
18. Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback, **S_ID**
~~19. Employee_ID, Employee_name, Emp_salary, Employee_role, Emp_shift_time, Employee_phone~~
20. Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary, **E_ID**

Final Tables

1.Admin_ID, Admin_name, Admin_phone, Admin_email

2. Shop_ID, Shop_name, Shop_status, Shop_type, Shop_floor, **A_ID**

3. Facility_ID, Facility_name, Availability_Status, **A_ID**

4. Employee_ID, Employee_name, Emp_salary, Employee_role, Employee_phone, Emp_shift_time, **A_ID**

5. Request_ID, Issue_description, Reported_by, Priority_level, **F_ID**

6. Tenant_ID, Tenant_name, Tenant_email, Tenant_phone, Tenant_ID_proof

7. Payment_ID, Amount, Payment_date, Payment_method, **T_ID**

8. Booking_ID, Booking_rent, Lease_start_date, Lease_end_date, Booking_payment_status, **S_ID**

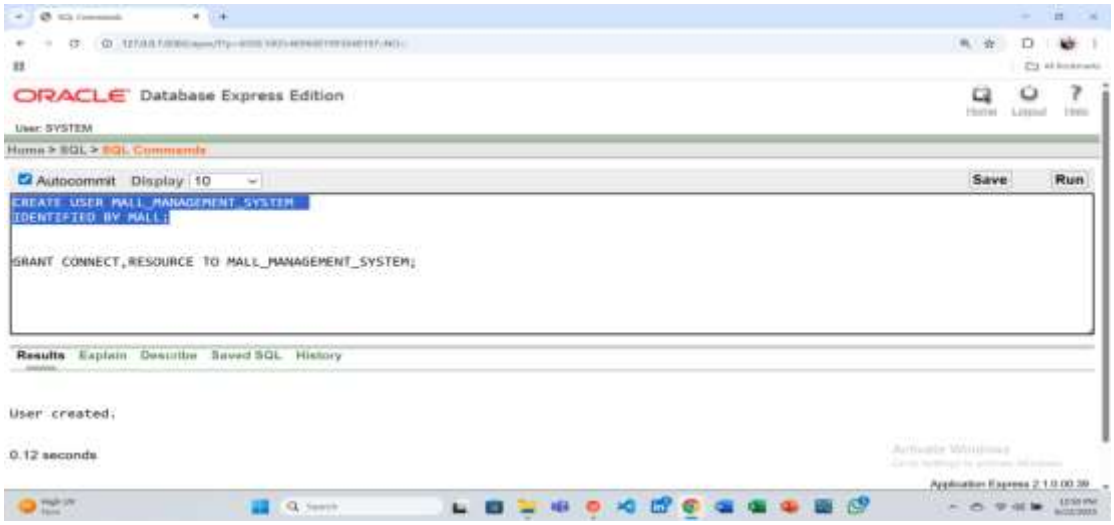
9. Customer_ID, Customer_name, Customer_email, Customer_phone, Visit_date, Feedback, **S_ID**

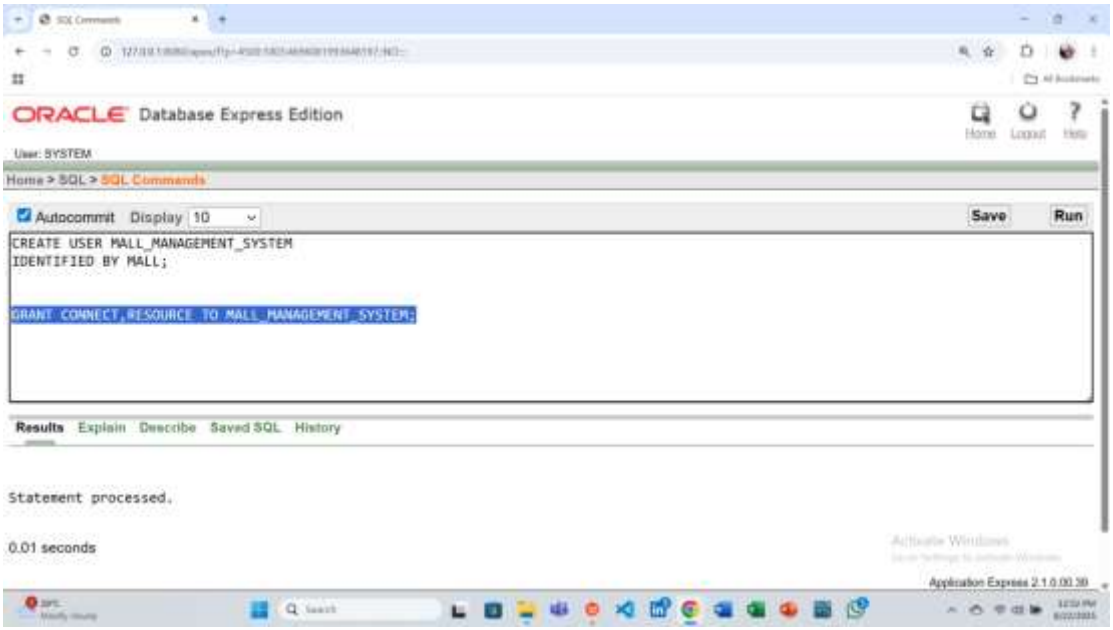
10. Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary, **E_ID**

7.Schema Diagram

8.Table Creation Using SQL

Created A separate Database/Schema for Mall_management_system:





Creating all Final Tables:

-- 1. Admin

CREATE TABLE Admin (

Admin_ID INT PRIMARY KEY,

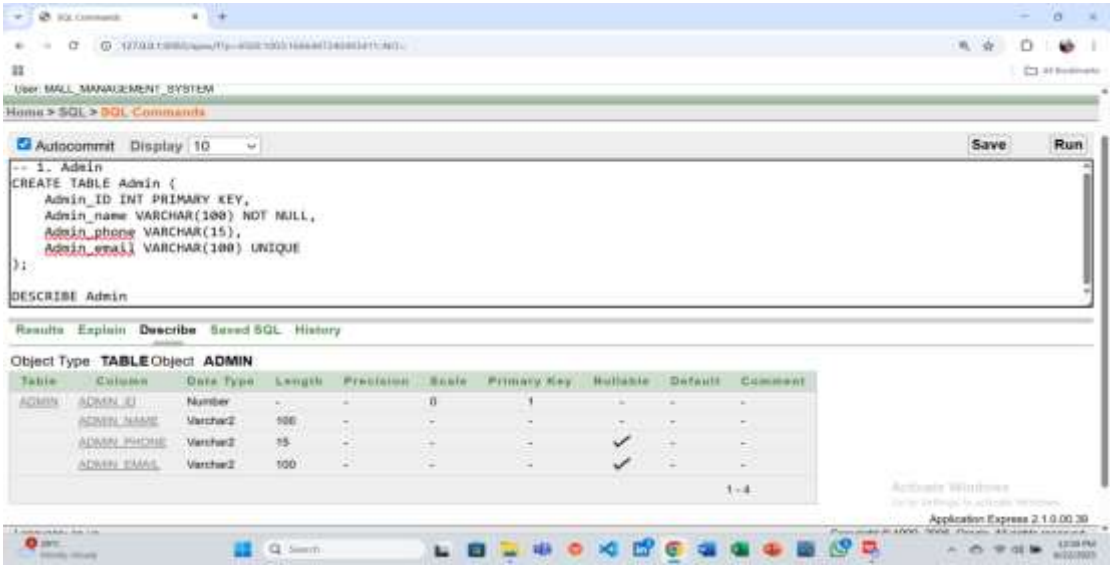
Admin_name VARCHAR(100) NOT NULL,

Admin_phone VARCHAR(15),

Admin_email VARCHAR(100) UNIQUE

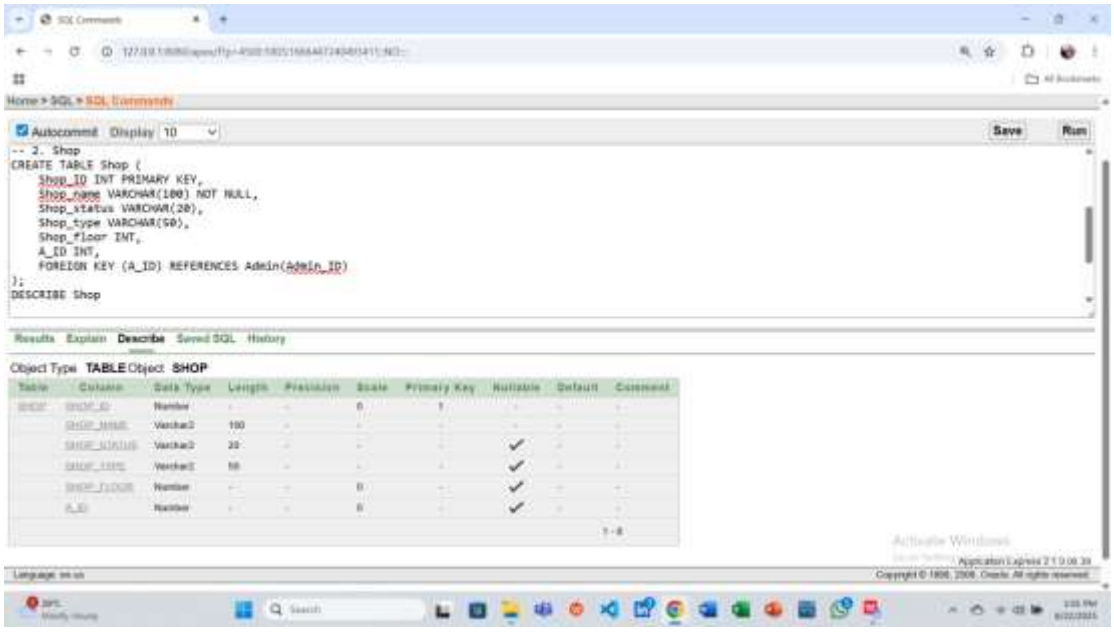
);

DESCRIBE Admin



-- 2. Shop

```
CREATE TABLE Shop (
    Shop_ID INT PRIMARY KEY,
    Shop_name VARCHAR(100) NOT NULL,
    Shop_status VARCHAR(20),
    Shop_type VARCHAR(50),
    Shop_floor INT,
    A_ID INT,
    FOREIGN KEY (A_ID) REFERENCES Admin(Admin_ID)
);
DESCRIBE Shop
```

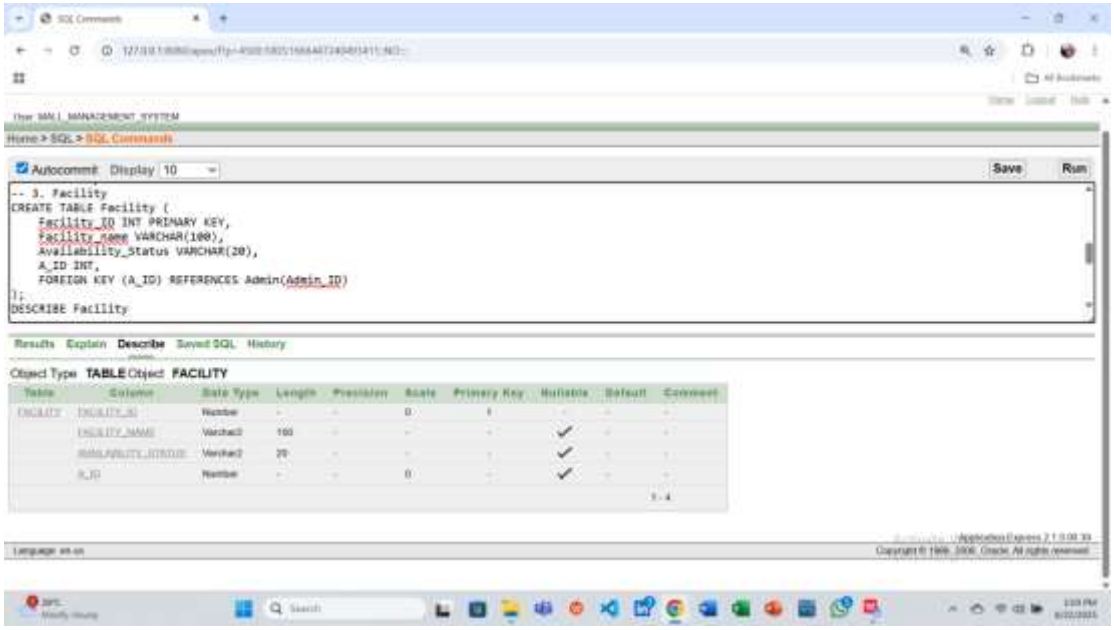


-- 3. Facility

```
CREATE TABLE Facility (
    Facility_ID INT PRIMARY KEY,
    Facility_name VARCHAR(100),
    Availability_Status VARCHAR(20),
    A_ID INT,
    FOREIGN KEY (A_ID) REFERENCES Admin(Admin_ID)
```

);

DESCRIBE Facility



-- 4. Employee

CREATE TABLE Employee (

Employee_ID INT PRIMARY KEY,

Employee_name VARCHAR(100),

Emp_salary DECIMAL(10,2),

Employee_role VARCHAR(50),

Employee_phone VARCHAR(15),

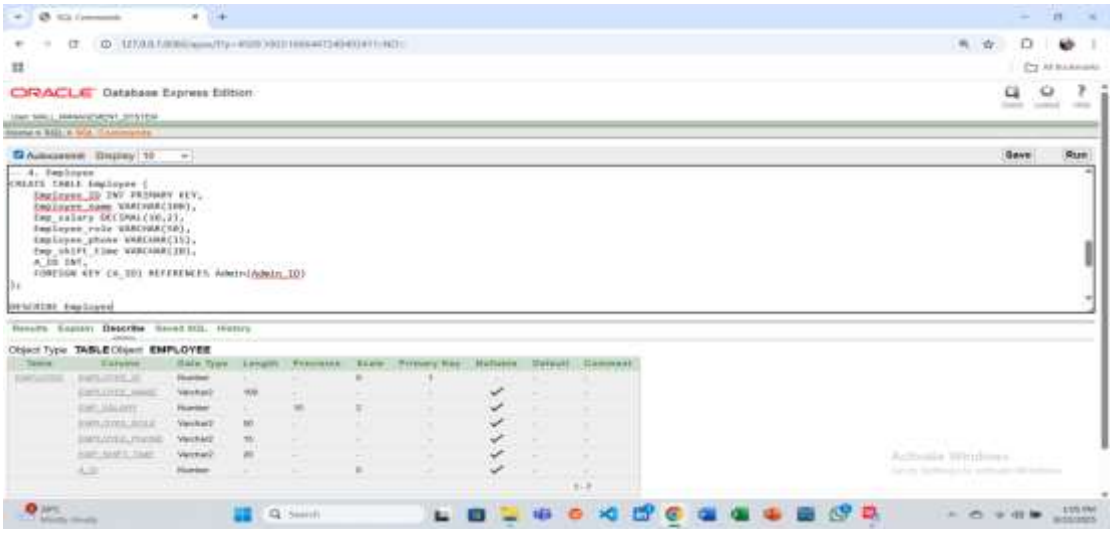
Emp_shift_time VARCHAR(20),

A_ID INT,

FOREIGN KEY (A_ID) REFERENCES Admin(Admin_ID)

);

DESCRIBE Employee



-- 5. Request

CREATE TABLE Request (

Request_ID INT PRIMARY KEY,

Issue_description VARCHAR(255),

Reported_by VARCHAR(100),

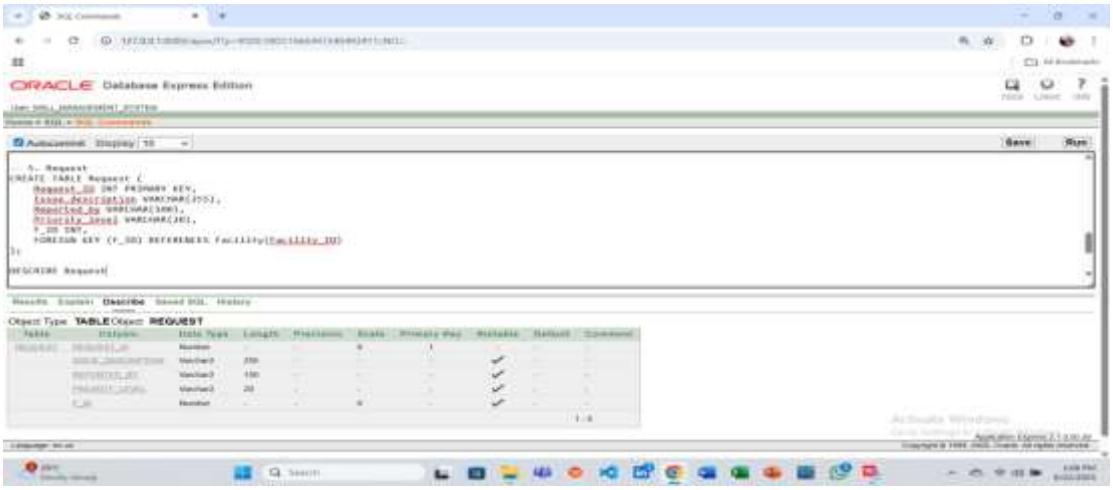
Priority_level VARCHAR(20),

F_ID INT,

FOREIGN KEY (F_ID) REFERENCES Facility(Facility_ID)

);

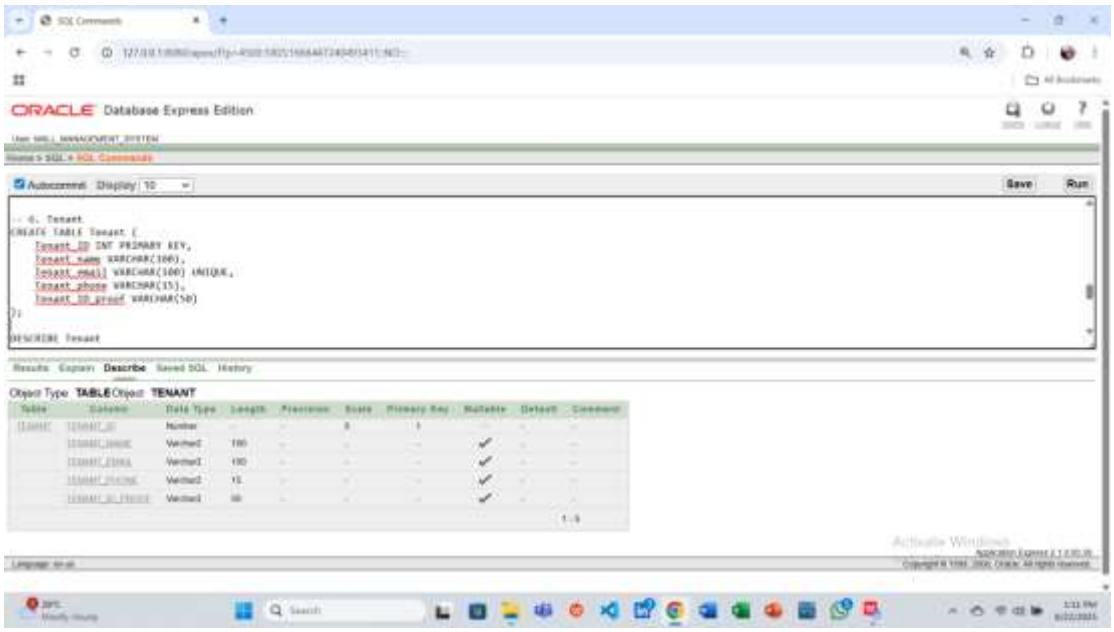
DESCRIBE Request



-- 6. Tenant

```
CREATE TABLE Tenant (
    Tenant_ID INT PRIMARY KEY,
    Tenant_name VARCHAR(100),
    Tenant_email VARCHAR(100) UNIQUE,
    Tenant_phone VARCHAR(15),
    Tenant_ID_proof VARCHAR(50)
);
```

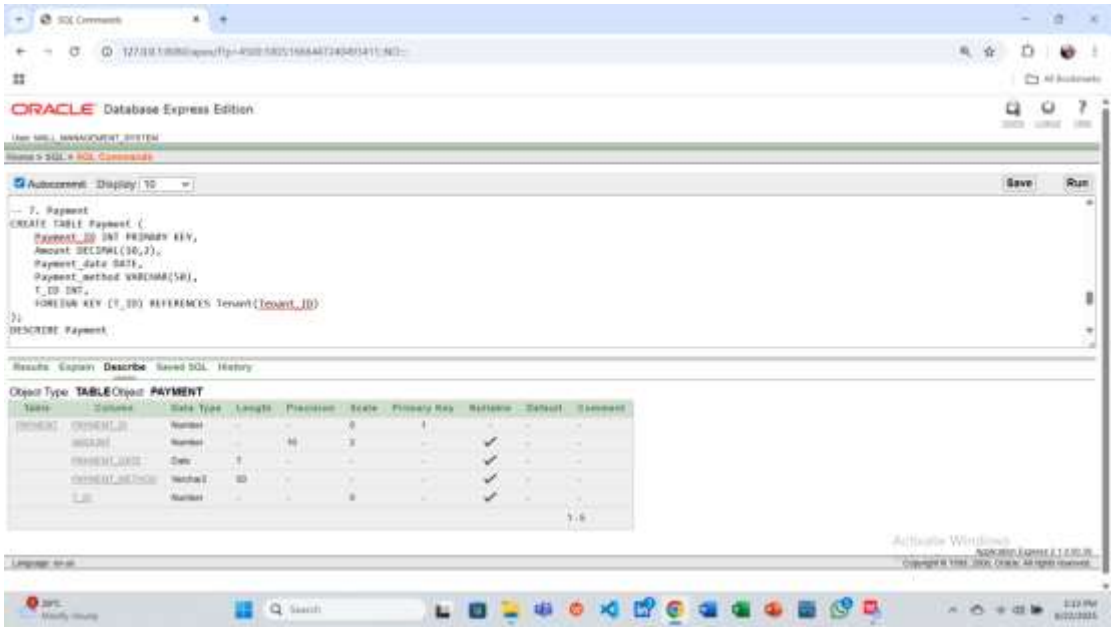
DESCRIBE Tenant



-- 7. Payment

```
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Amount DECIMAL(10,2),
    Payment_date DATE,
    Payment_method VARCHAR(50),
    T_ID INT,
    FOREIGN KEY (T_ID) REFERENCES Tenant(Tenant_ID)
);
```

DESCRIBE Payment



-- 8. Booking

CREATE TABLE Booking (

Booking_ID INT PRIMARY KEY,

Booking_rent DECIMAL(10,2),

Lease_start_date DATE,

Lease_end_date DATE,

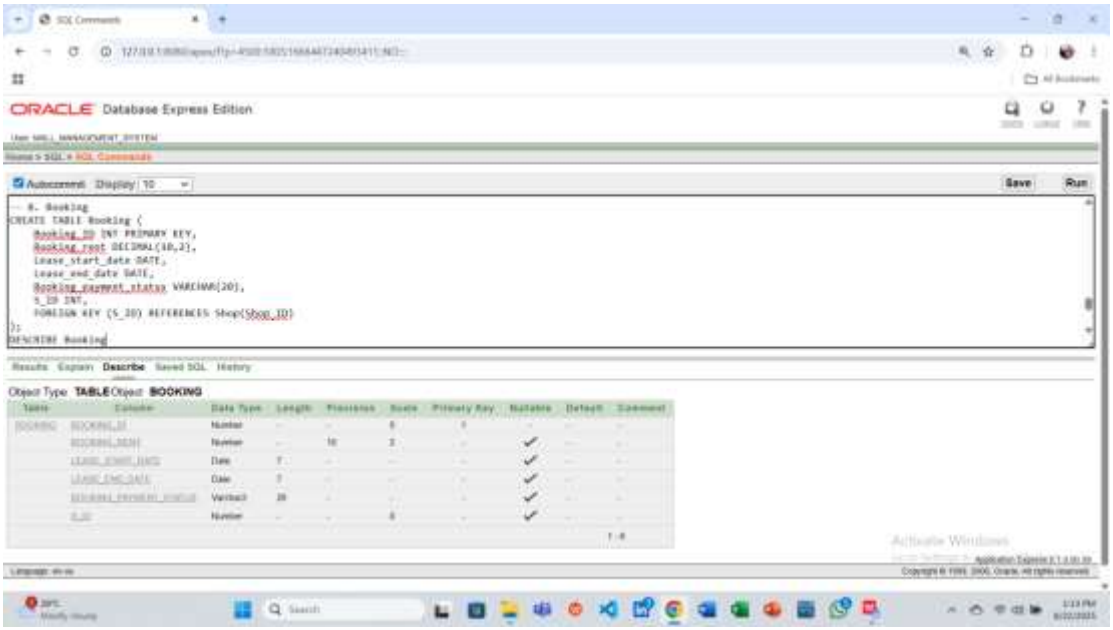
Booking_payment_status VARCHAR(20),

S_ID INT,

FOREIGN KEY (S_ID) REFERENCES Shop(Shop_ID)

);

DESCRIBE Booking



-- 9. Customer

CREATE TABLE Customer (

Customer_ID INT PRIMARY KEY,

Customer_name VARCHAR(100),

Customer_email VARCHAR(100),

Customer_phone VARCHAR(15),

Visit_date DATE,

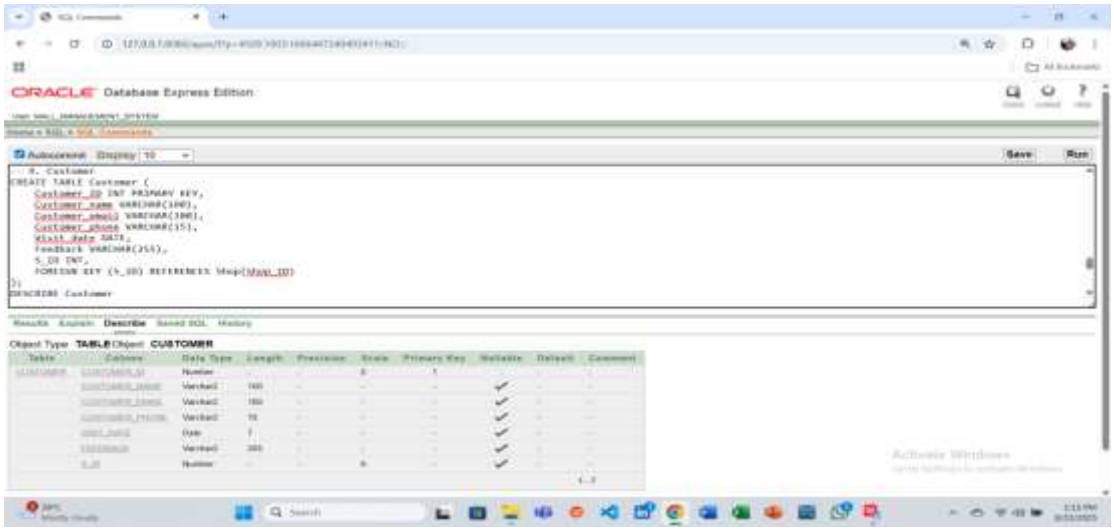
Feedback VARCHAR(255),

S_ID INT,

FOREIGN KEY (S_ID) REFERENCES Shop(Shop_ID)

);

DESCRIBE Customer



-- 10. Guard

CREATE TABLE Guard (

Guard_ID INT PRIMARY KEY,

Guard_name VARCHAR(100),

Guard_phone VARCHAR(15),

Guard_shift_time VARCHAR(20),

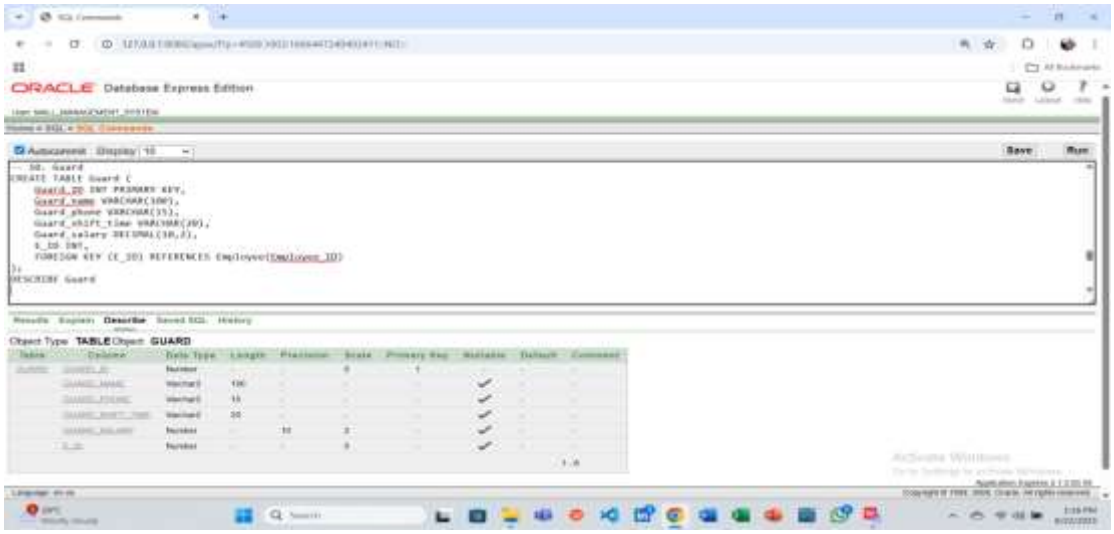
Guard_salary DECIMAL(10,2),

E_ID INT,

FOREIGN KEY (E_ID) REFERENCES Employee(Employee_ID)

);

DESCRIBE Guard



9.Data Insertion

-- Admin

INSERT INTO Admin VALUES (1, 'John Smith', '01710000001', 'john.smith@example.com');

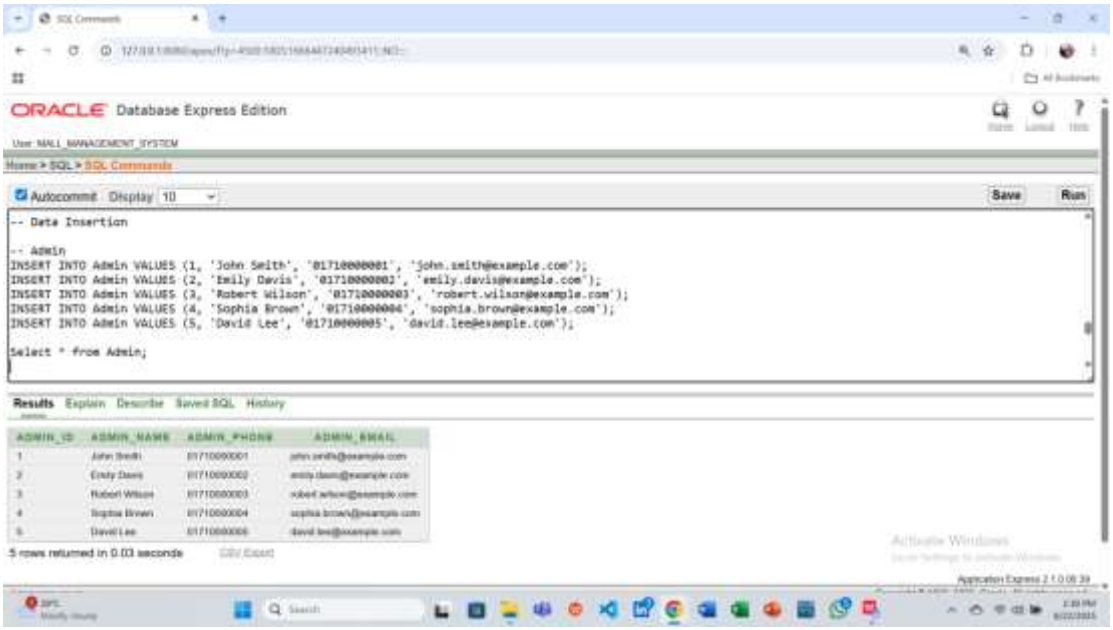
INSERT INTO Admin VALUES (2, 'Emily Davis', '01710000002', 'emily.davis@example.com');

INSERT INTO Admin VALUES (3, 'Robert Wilson', '01710000003', 'robert.wilson@example.com');

INSERT INTO Admin VALUES (4, 'Sophia Brown', '01710000004', 'sophia.brown@example.com');

INSERT INTO Admin VALUES (5, 'David Lee', '01710000005', 'david.lee@example.com');

Select * from Admin;



-- Shop

INSERT INTO Shop VALUES (101, 'Fashion Hub', 'Open', 'Clothing', 1, 1);

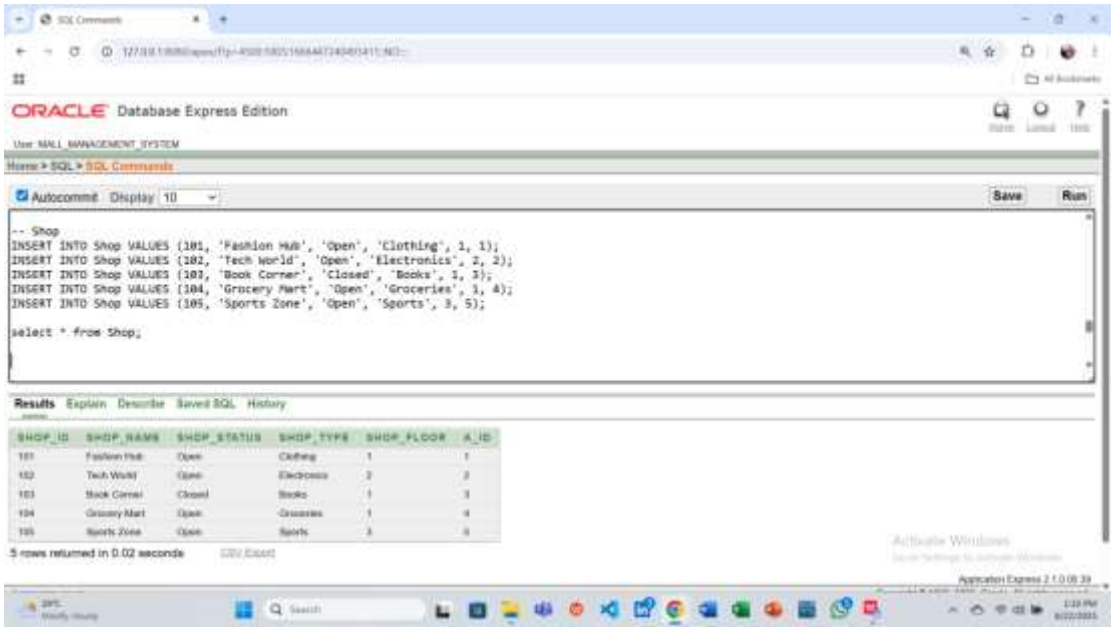
INSERT INTO Shop VALUES (102, 'Tech World', 'Open', 'Electronics', 2, 2);

INSERT INTO Shop VALUES (103, 'Book Corner', 'Closed', 'Books', 1, 3);

INSERT INTO Shop VALUES (104, 'Grocery Mart', 'Open', 'Groceries', 1, 4);

INSERT INTO Shop VALUES (105, 'Sports Zone', 'Open', 'Sports', 3, 5);

select * from Shop;



-- Facility

INSERT INTO Facility VALUES (201, 'Parking Lot', 'Available', 1);

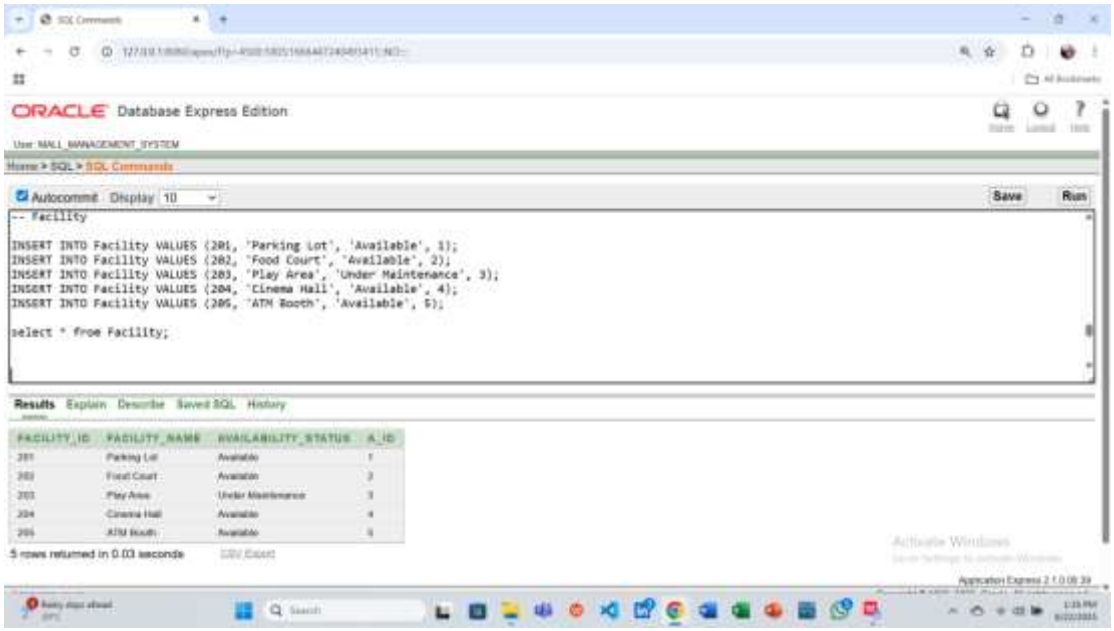
INSERT INTO Facility VALUES (202, 'Food Court', 'Available', 2);

INSERT INTO Facility VALUES (203, 'Play Area', 'Under Maintenance', 3);

INSERT INTO Facility VALUES (204, 'Cinema Hall', 'Available', 4);

INSERT INTO Facility VALUES (205, 'ATM Booth', 'Available', 5);

select * from Facility;



-- Employee

INSERT INTO Employee VALUES (301, 'Alex Johnson', 25000, 'Manager', '01710000011', 'Morning', 1);

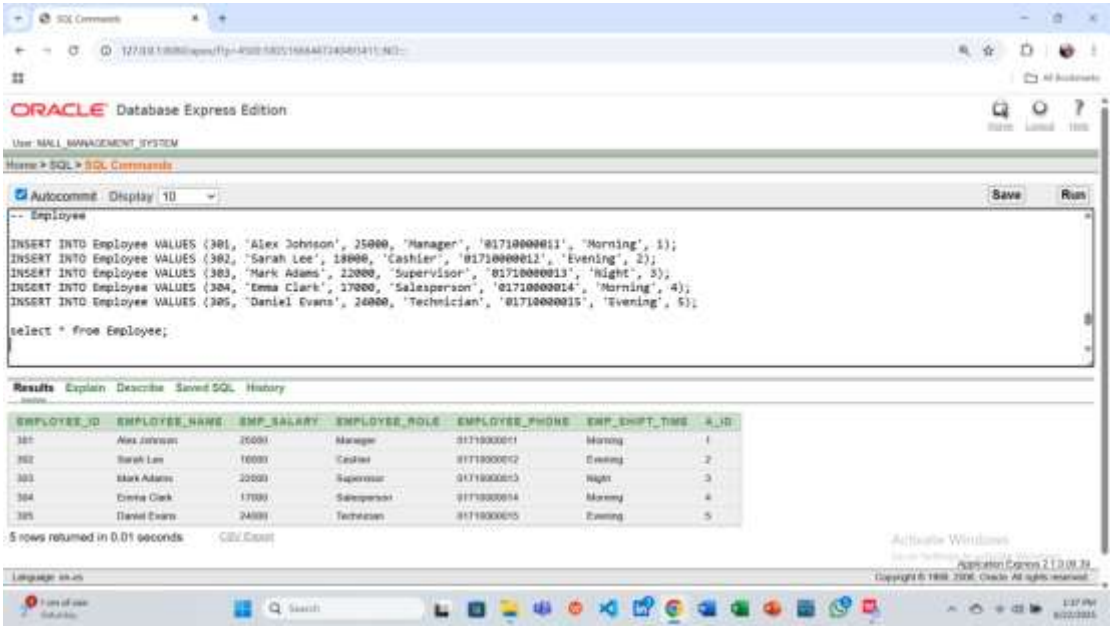
INSERT INTO Employee VALUES (302, 'Sarah Lee', 18000, 'Cashier', '01710000012', 'Evening', 2);

INSERT INTO Employee VALUES (303, 'Mark Adams', 22000, 'Supervisor', '01710000013', 'Night', 3);

INSERT INTO Employee VALUES (304, 'Emma Clark', 17000, 'Salesperson', '01710000014', 'Morning', 4);

INSERT INTO Employee VALUES (305, 'Daniel Evans', 24000, 'Technician', '01710000015', 'Evening', 5);

select * from Employee;



-- Request

INSERT INTO Request VALUES (401, 'Broken AC in Food Court', 'Manager', 'High', 202);

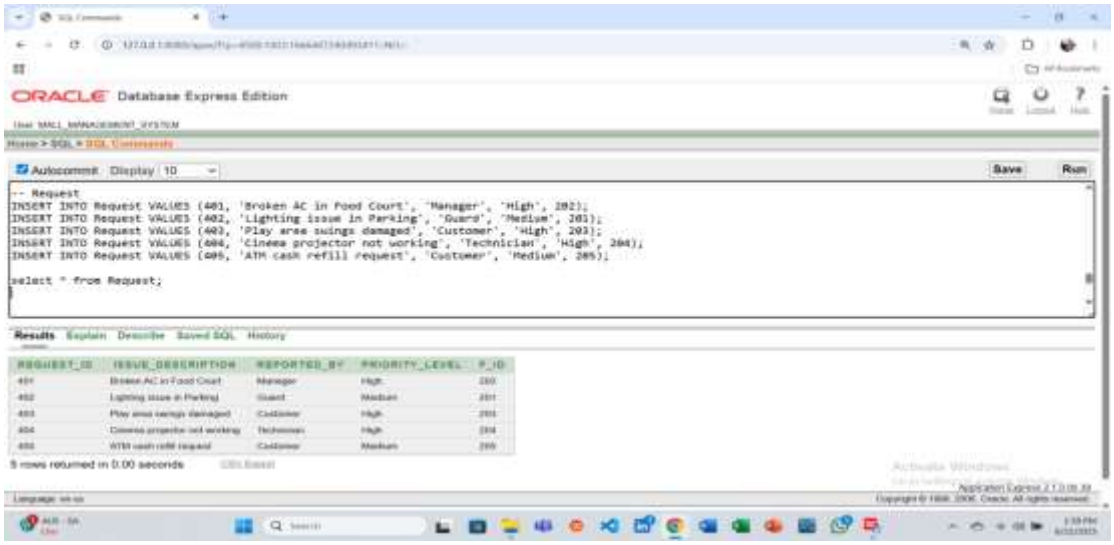
INSERT INTO Request VALUES (402, 'Lighting issue in Parking', 'Guard', 'Medium', 201);

INSERT INTO Request VALUES (403, 'Play area swings damaged', 'Customer', 'High', 203);

INSERT INTO Request VALUES (404, 'Cinema projector not working', 'Technician', 'High', 204);

INSERT INTO Request VALUES (405, 'ATM cash refill request', 'Customer', 'Medium', 205);

select * from Request;



-- Tenant

INSERT INTO Tenant VALUES (501, 'Michael Brown', 'michael.brown@example.com', '01710000021', 'NID123456');

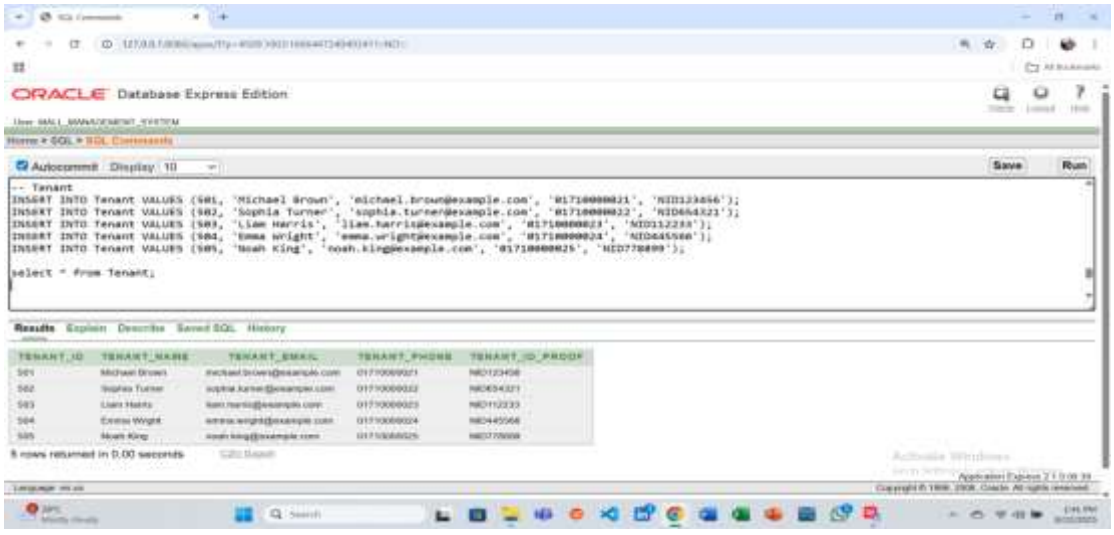
INSERT INTO Tenant VALUES (502, 'Sophia Turner', 'sophia.turner@example.com', '01710000022', 'NID654321');

INSERT INTO Tenant VALUES (503, 'Liam Harris', 'liam.harris@example.com', '01710000023', 'NID112233');

INSERT INTO Tenant VALUES (504, 'Emma Wright', 'emma.wright@example.com', '01710000024', 'NID445566');

INSERT INTO Tenant VALUES (505, 'Noah King', 'noah.king@example.com', '01710000025', 'NID778899');

select * from Tenant;



-- Payment

INSERT INTO Payment VALUES (601, 15000, TO_DATE('2025-08-01','YYYY-MM-DD'), 'Cash', 501);

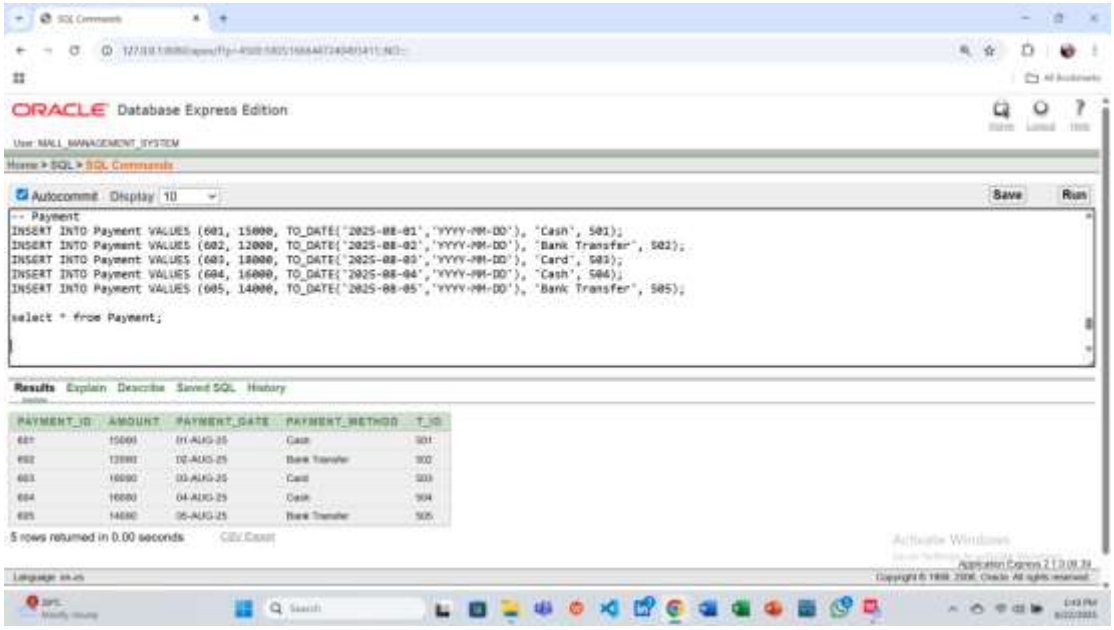
INSERT INTO Payment VALUES (602, 12000, TO_DATE('2025-08-02','YYYY-MM-DD'), 'Bank Transfer', 502);

INSERT INTO Payment VALUES (603, 18000, TO_DATE('2025-08-03','YYYY-MM-DD'), 'Card', 503);

INSERT INTO Payment VALUES (604, 16000, TO_DATE('2025-08-04','YYYY-MM-DD'), 'Cash', 504);

INSERT INTO Payment VALUES (605, 14000, TO_DATE('2025-08-05','YYYY-MM-DD'), 'Bank Transfer', 505);

select * from Payment;



-- Booking

INSERT INTO Booking VALUES

(701, 20000, TO_DATE('2025-08-01','YYYY-MM-DD'), TO_DATE('2026-07-31','YYYY-MM-DD'), 'Paid', 101);

INSERT INTO Booking VALUES

(702, 25000, TO_DATE('2025-08-02','YYYY-MM-DD'), TO_DATE('2026-08-01','YYYY-MM-DD'), 'Unpaid', 102);

INSERT INTO Booking VALUES

(703, 18000, TO_DATE('2025-08-03','YYYY-MM-DD'), TO_DATE('2026-08-02','YYYY-MM-DD'), 'Paid', 103);

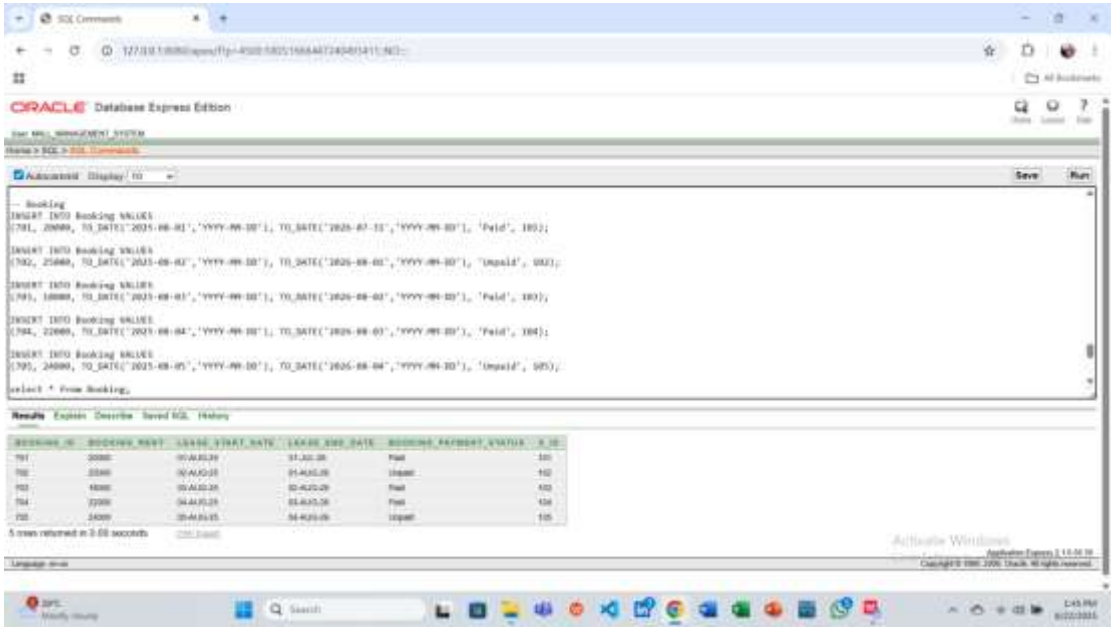
INSERT INTO Booking VALUES

(704, 22000, TO_DATE('2025-08-04','YYYY-MM-DD'), TO_DATE('2026-08-03','YYYY-MM-DD'), 'Paid', 104);

INSERT INTO Booking VALUES

(705, 24000, TO_DATE('2025-08-05','YYYY-MM-DD'), TO_DATE('2026-08-04','YYYY-MM-DD'), 'Unpaid', 105);

select * from Booking;



-- Customer

INSERT INTO Customer VALUES (801, 'Liam Wilson', 'liam.wilson@example.com', '01710000031', TO_DATE('2025-08-01','YYYY-MM-DD'), 'Great service', 101);

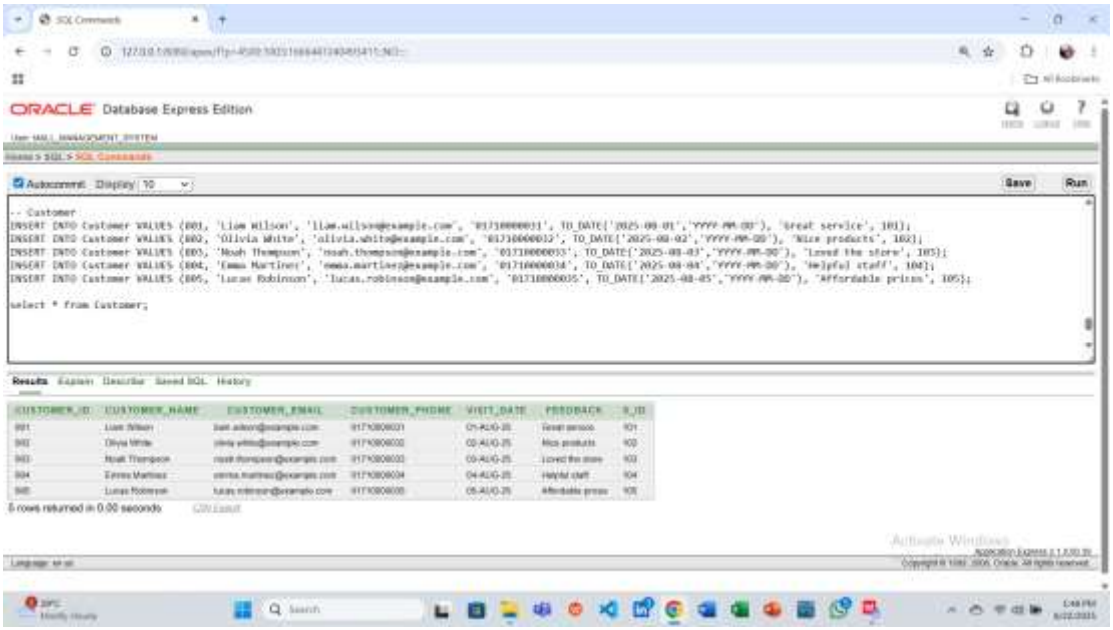
INSERT INTO Customer VALUES (802, 'Olivia White', 'olivia.white@example.com', '01710000032', TO_DATE('2025-08-02','YYYY-MM-DD'), 'Nice products', 102);

INSERT INTO Customer VALUES (803, 'Noah Thompson', 'noah.thompson@example.com', '01710000033', TO_DATE('2025-08-03','YYYY-MM-DD'), 'Loved the store', 103);

INSERT INTO Customer VALUES (804, 'Emma Martinez', 'emma.martinez@example.com', '01710000034', TO_DATE('2025-08-04','YYYY-MM-DD'), 'Helpful staff', 104);

INSERT INTO Customer VALUES (805, 'Lucas Robinson', 'lucas.robinson@example.com', '01710000035', TO_DATE('2025-08-05','YYYY-MM-DD'), 'Affordable prices', 105);

select * from Customer;



-- Guard

INSERT INTO Guard VALUES (901, 'David Miller', '01710000041', 'Night', 15000, 301);

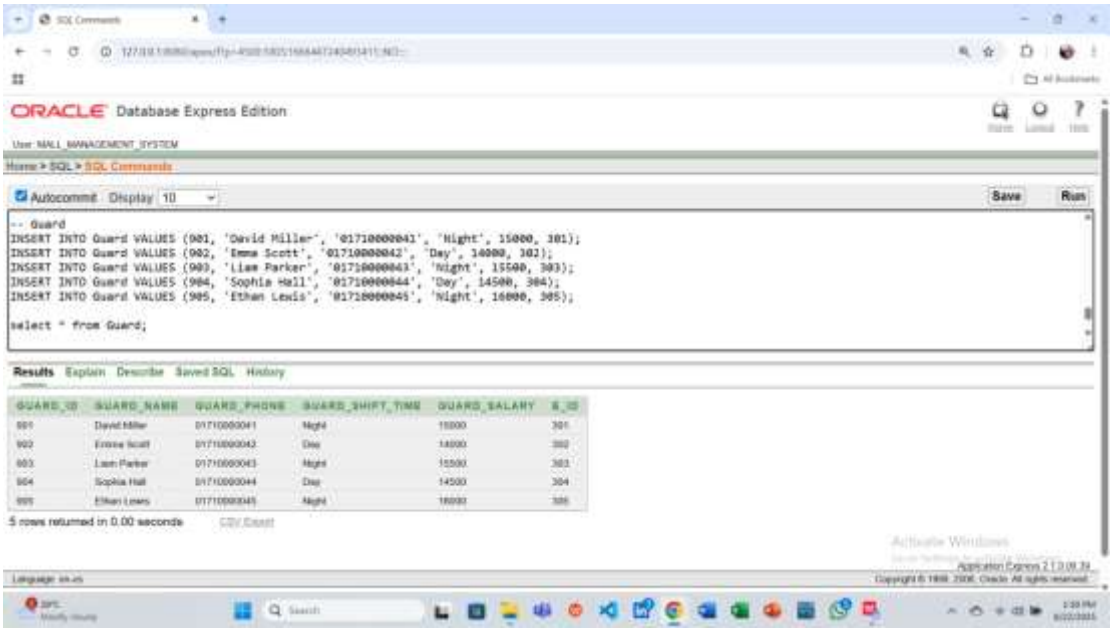
INSERT INTO Guard VALUES (902, 'Emma Scott', '01710000042', 'Day', 14000, 302);

INSERT INTO Guard VALUES (903, 'Liam Parker', '01710000043', 'Night', 15500, 303);

INSERT INTO Guard VALUES (904, 'Sophia Hall', '01710000044', 'Day', 14500, 304);

INSERT INTO Guard VALUES (905, 'Ethan Lewis', '01710000045', 'Night', 16000, 305);

select * from Guard;



10.Query Writing Using PL/SQL

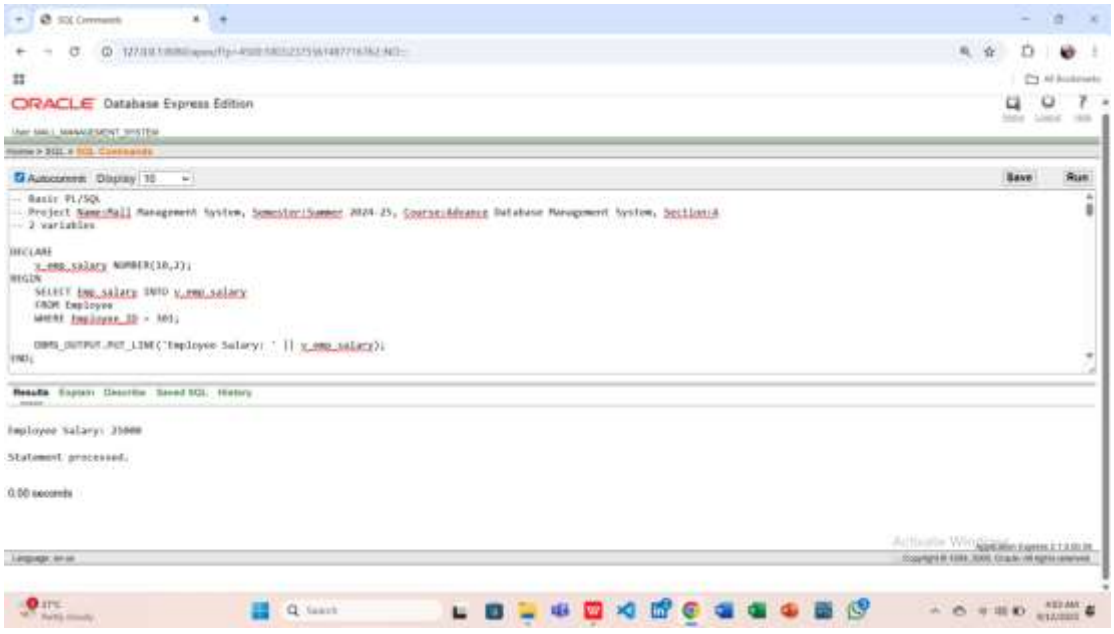
11. Basic PL/SQL

- Using 2 variables

Question 1: Display the salary of Employee_ID = 301 using a variable.

Answer:

```
DECLARE
   v_emp_salary NUMBER(10,2);
BEGIN
   SELECT Emp_salary INTO v_emp_salary
   FROM Employee
   WHERE Employee_ID = 301;

   DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || v_emp_salary);
END;
```

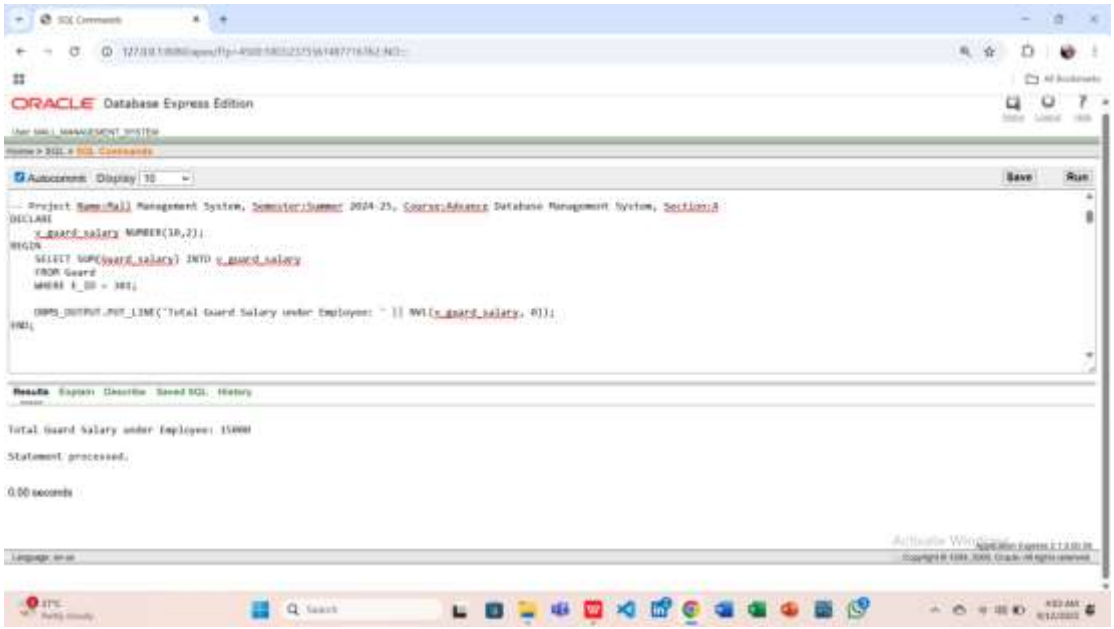


Question 2: Display the total salary of guards reporting to Employee_ID = 301 using a variable.

Answer:

```
DECLARE
```

```
    v_guard_salary NUMBER(10,2);
BEGIN
    SELECT SUM(Guard_salary) INTO v_guard_salary
    FROM Guard
    WHERE E_ID = 301;

    DBMS_OUTPUT.PUT_LINE('Total Guard Salary under Employee: ' || NVL(v_guard_salary, 0));
END;
```



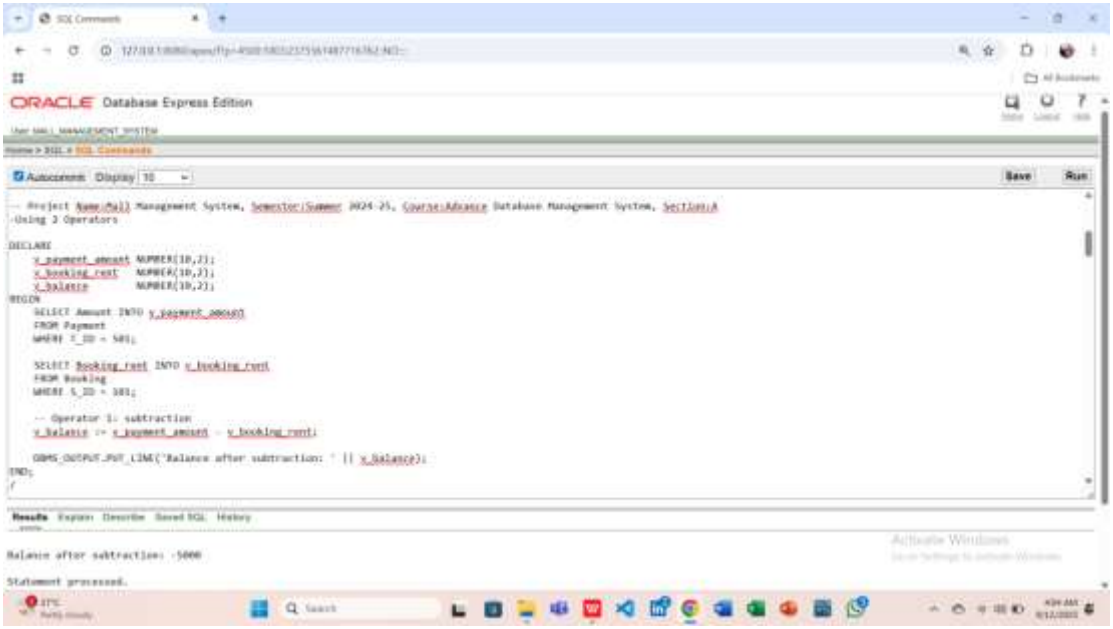
-Using 2 Operators

Question 1: Calculate Balance using Subtraction Operator

Answer:

```
DECLARE
    v_payment_amount NUMBER(10,2);
    v_booking_rent   NUMBER(10,2);
    v_balance        NUMBER(10,2);
BEGIN
    SELECT Amount INTO v_payment_amount
```

```
    FROM Payment
    WHERE T_ID = 501;
    SELECT Booking_rent INTO v_booking_rent
    FROM Booking
    WHERE S_ID = 101;
    -- Operator 1: subtraction
    v_balance := v_payment_amount - v_booking_rent;
    DBMS_OUTPUT.PUT_LINE('Balance after subtraction: ' || v_balance);
END;
/
```

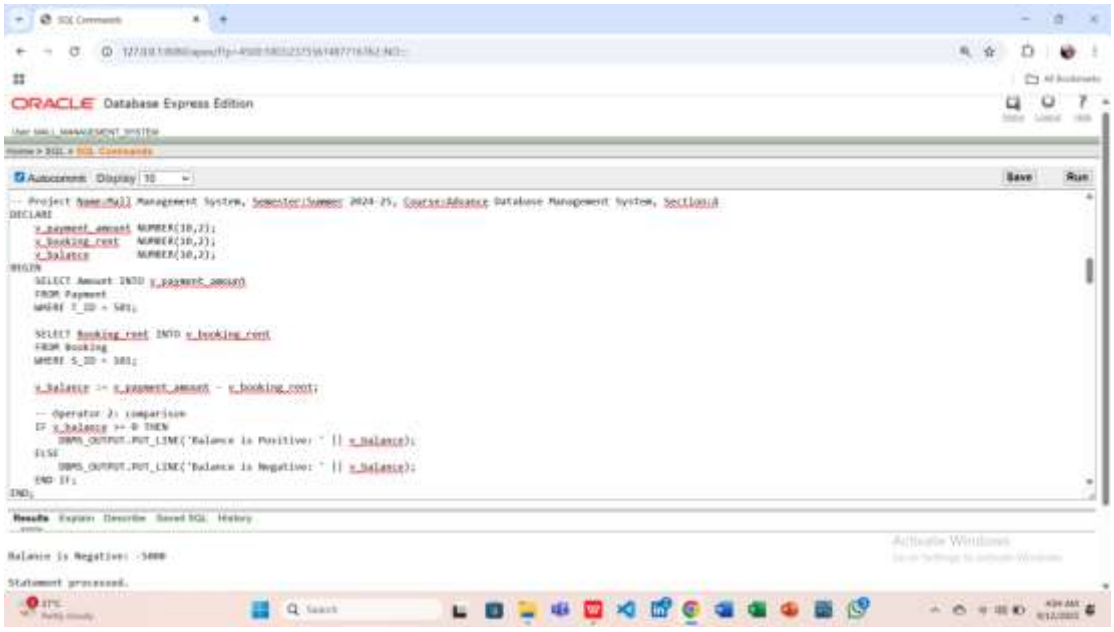


Question 2: Check if Balance is Positive or Negative using Comparison Operator

Answer:

```
DECLARE
    v_payment_amount NUMBER(10,2);
    v_booking_rent   NUMBER(10,2);
    v_balance        NUMBER(10,2);
BEGIN
    SELECT Amount INTO v_payment_amount
```

```
    FROM Payment
    WHERE T_ID = 501;
    SELECT Booking_rent INTO v_booking_rent
    FROM Booking
    WHERE S_ID = 101;
    v_balance := v_payment_amount - v_booking_rent;
    -- Operator 2: comparison
    IF v_balance >= 0 THEN
        DBMS_OUTPUT.PUT_LINE('Balance is Positive: ' || v_balance);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Balance is Negative: ' || v_balance);
    END IF;
END;
/
```

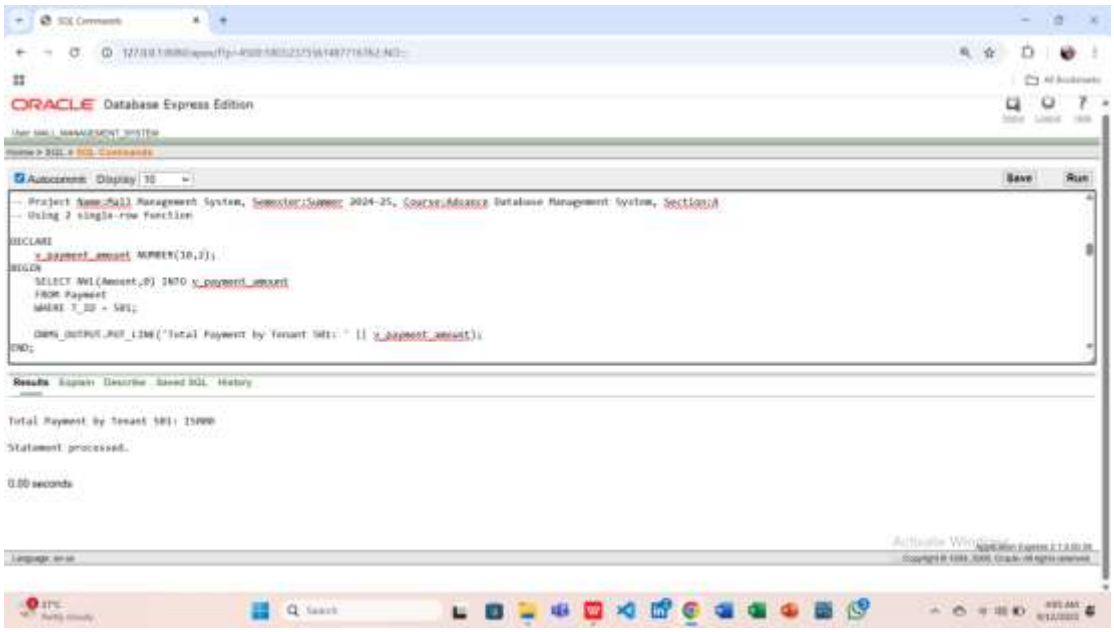


-Using 2 single-row function

Question 1: Display the total payment made by a tenant.

Answer:

```
DECLARE
   v_payment_amount NUMBER(10,2);
BEGIN
   SELECT NVL(Amount,0) INTO v_payment_amount
   FROM Payment
   WHERE T_ID = 501;

   DBMS_OUTPUT.PUT_LINE('Total Payment by Tenant 501: ' || v_payment_amount);
END;
```

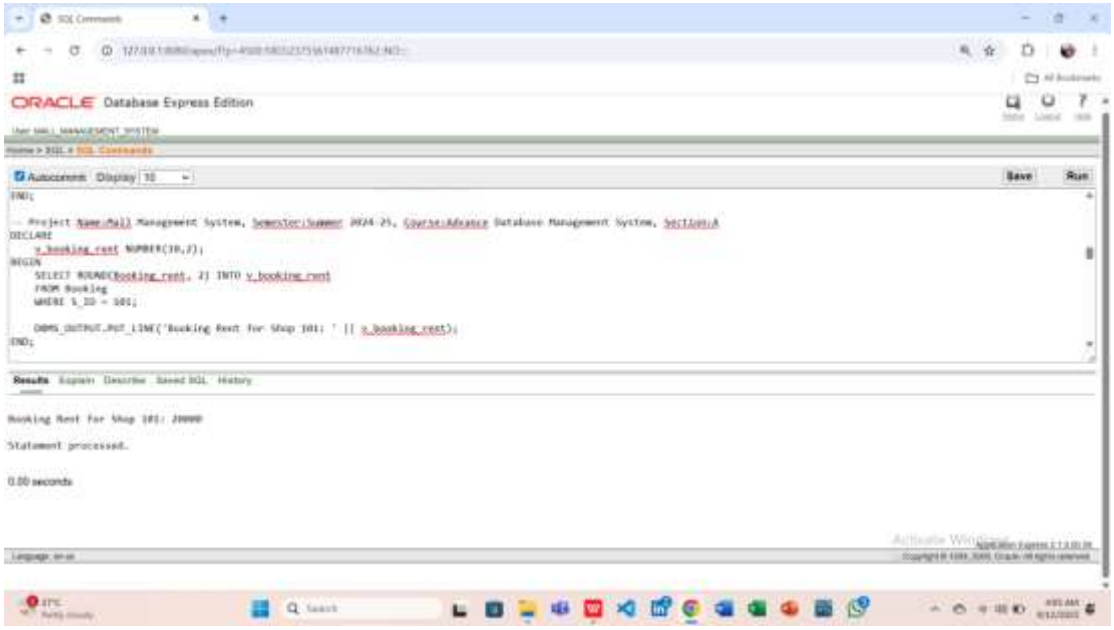


Question 2: Display the booking rent for a shop.

Answer:

```
DECLARE
   v_booking_rent NUMBER(10,2);
BEGIN
   SELECT ROUND(Booking_rent, 2) INTO v_booking_rent
   FROM Booking
   WHERE S_ID = 101;
```

```
    DBMS_OUTPUT.PUT_LINE('Booking Rent for Shop 101: ' || v_booking_rent);
END;
```

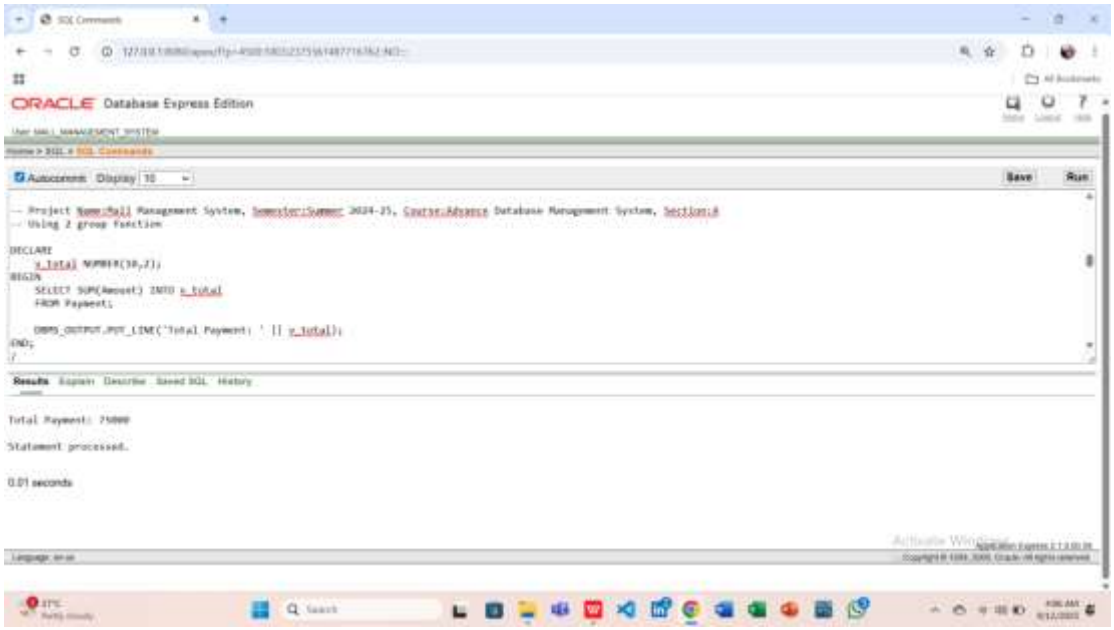


- Using 2 group function

Question 1: Find the Total Payment Amount

Answer:

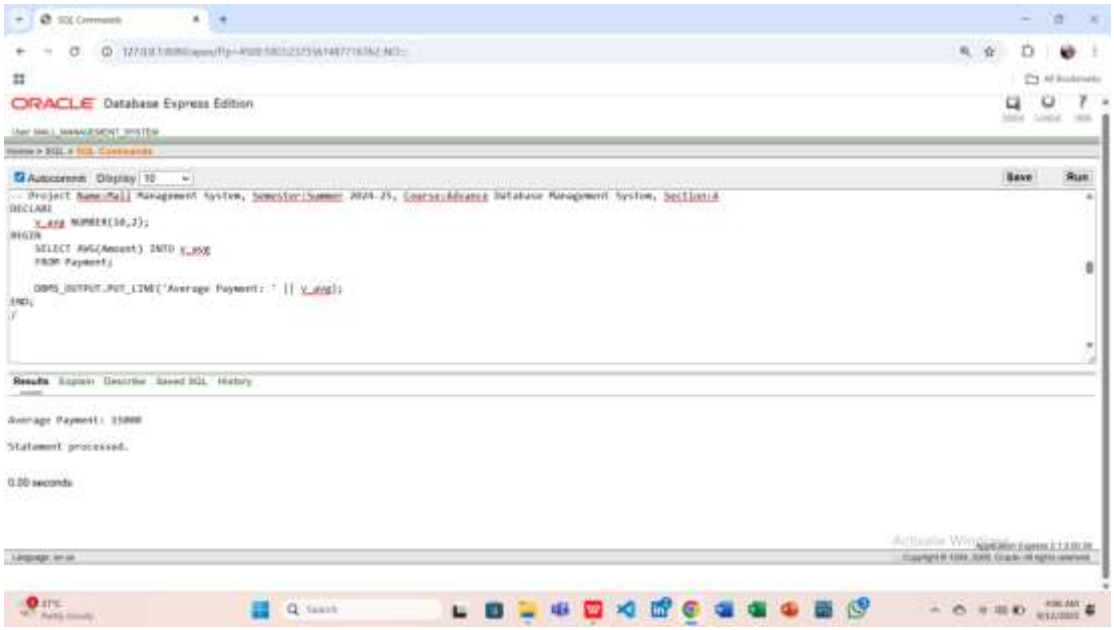
```
DECLARE
   v_total NUMBER(10,2);
BEGIN
   SELECT SUM(Amount) INTO v_total
   FROM Payment;

   DBMS_OUTPUT.PUT_LINE('Total Payment: ' || v_total);
END;
/
```



Question 2: Find the Average Payment Amount

Answer:

```
DECLARE
   v_avg NUMBER(10,2);
BEGIN
   SELECT AVG(Amount) INTO v_avg
   FROM Payment;


   DBMS_OUTPUT.PUT_LINE('Average Payment: ' || v_avg);
END;
/
```

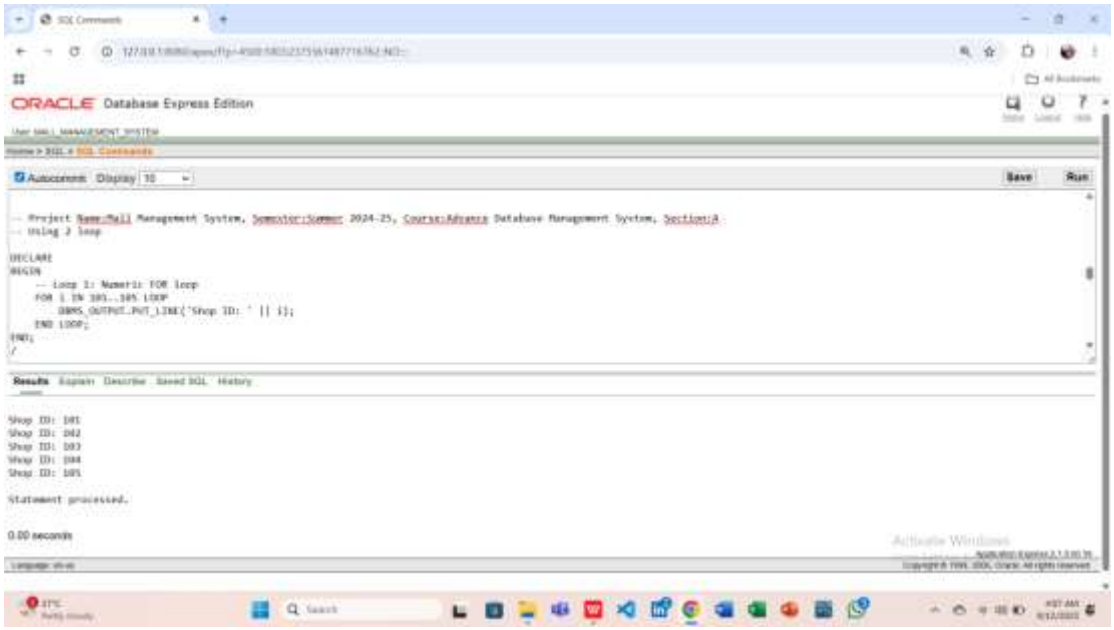


-Using 2 loop

Question 1: Display Shop IDs from 101 to 105 (Numeric FOR Loop)

Answer:

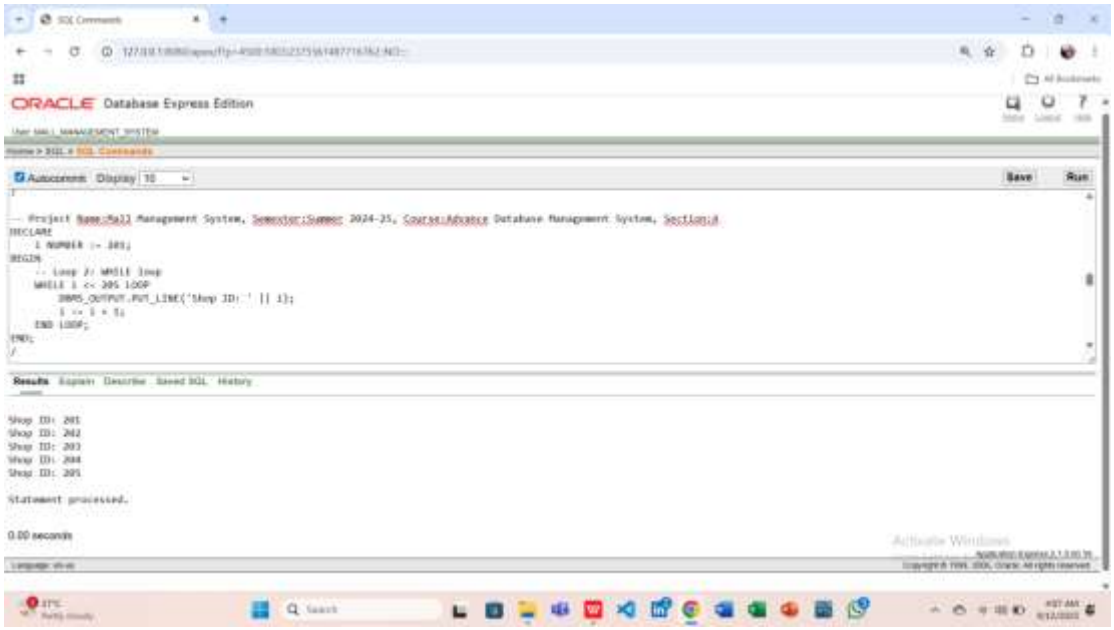
```
DECLARE
BEGIN
   -- Loop 1: Numeric FOR loop
   FOR i IN 101..105 LOOP
      DBMS_OUTPUT.PUT_LINE('Shop ID: ' || i);
   END LOOP;
END;
/
```



Question 2: Display Shop IDs from 201 to 205 (WHILE Loop)

Answer:

```
DECLARE
   i NUMBER := 201;
BEGIN
   -- Loop 2: WHILE loop
   WHILE i <= 205 LOOP
      DBMS_OUTPUT.PUT_LINE('Shop ID: ' || i);
      i := i + 1;
   END LOOP;
END;
/
```



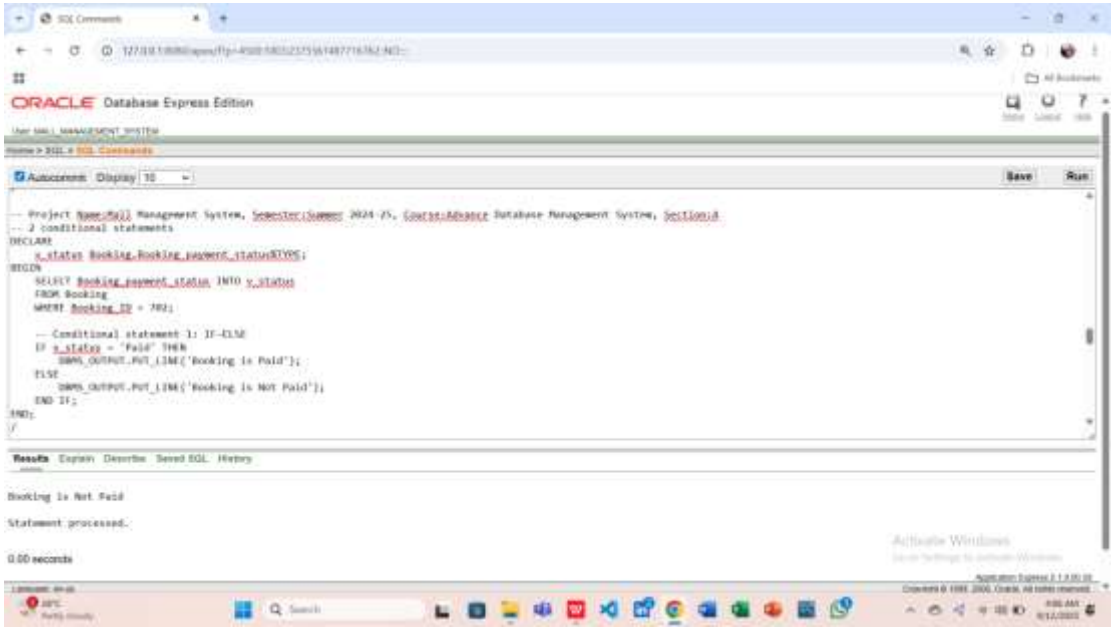
-2 conditional statements

Question 1: Check Booking Payment Status using IF–ELSE

Answer:

```
DECLARE
   v_status Booking.Booking_payment_status%TYPE;
BEGIN
   SELECT Booking_payment_status INTO v_status
   FROM Booking
   WHERE Booking_ID = 702;


   -- Conditional statement 1: IF–ELSE
   IF v_status = 'Paid' THEN
      DBMS_OUTPUT.PUT_LINE('Booking is Paid');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Booking is Not Paid');
   END IF;
END;
```

/

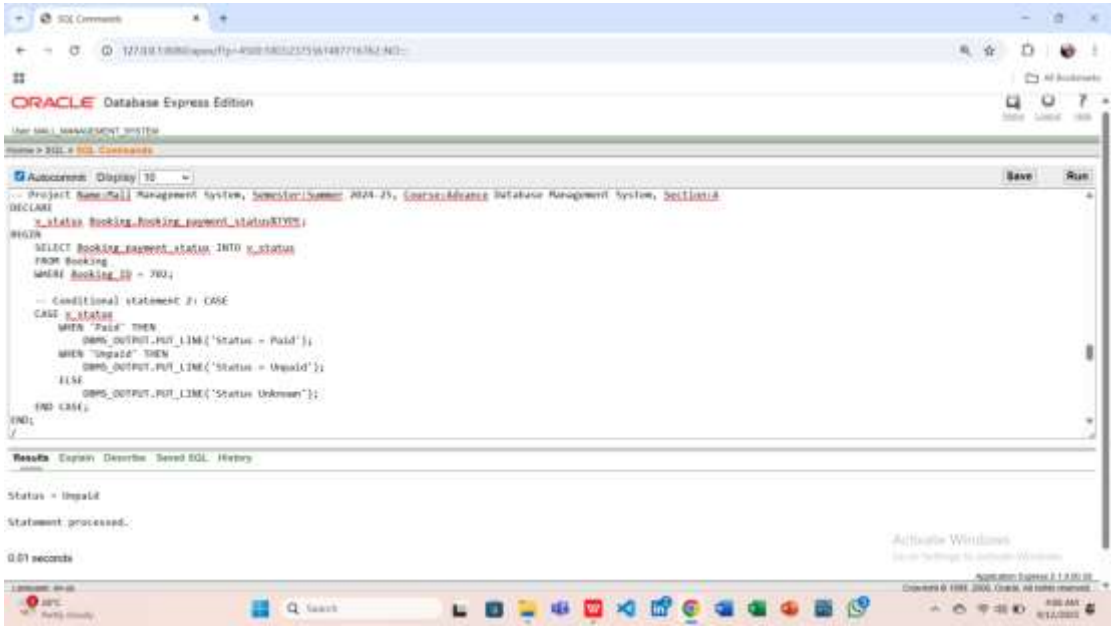


Question 2: Check Booking Payment Status using CASE

Answer:

```
DECLARE
   v_status Booking.Booking_payment_status%TYPE;
BEGIN
   SELECT Booking_payment_status INTO v_status
   FROM Booking
   WHERE Booking_ID = 702;


   -- Conditional statement 2: CASE
   CASE v_status
      WHEN 'Paid' THEN
         DBMS_OUTPUT.PUT_LINE('Status = Paid');
      WHEN 'Unpaid' THEN
         DBMS_OUTPUT.PUT_LINE('Status = Unpaid');
      ELSE
         DBMS_OUTPUT.PUT_LINE('Status Unknown');
```

```
    END CASE;

END;

/
```

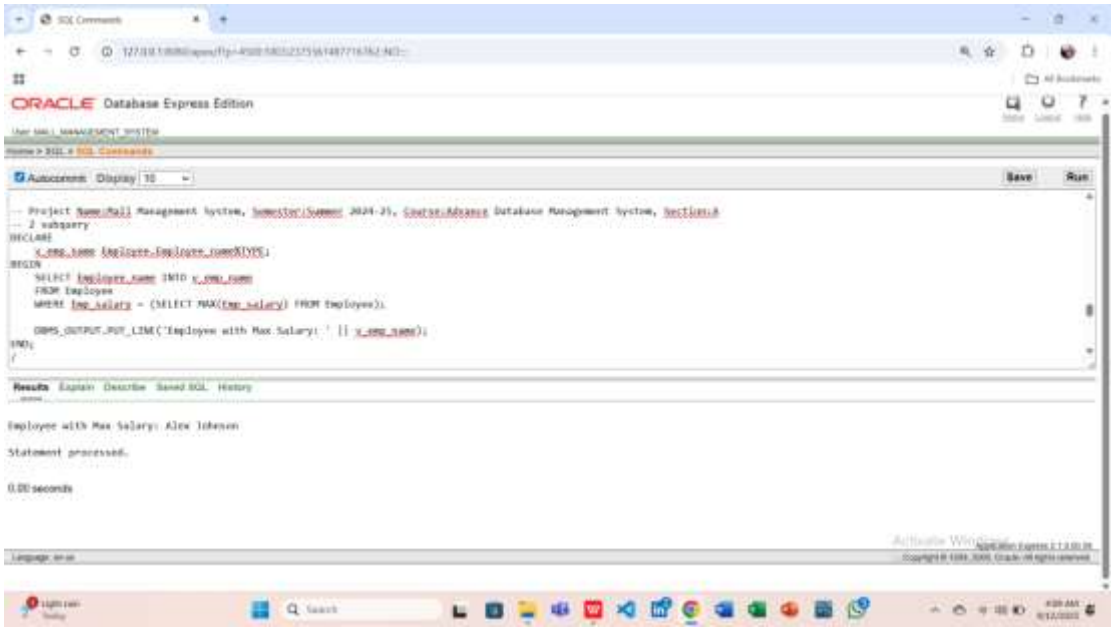


-2 subquery

Question 1: Find Employee Name with Maximum Salary (Subquery in WHERE clause)

Answer:

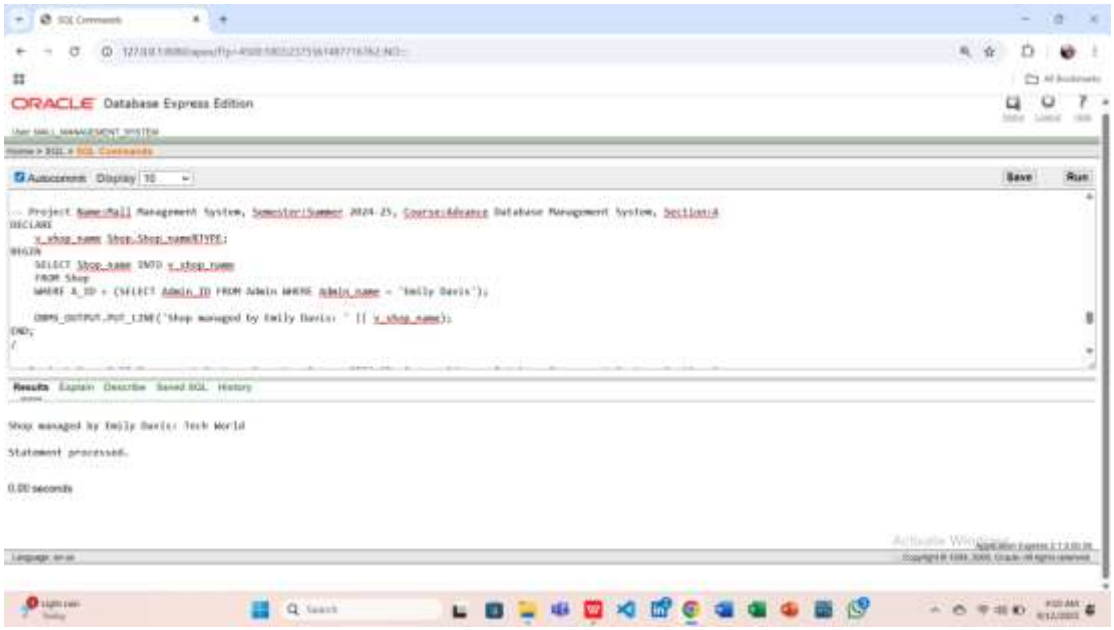
```
DECLARE

   v_emp_name Employee.Employee_name%TYPE;

BEGIN

   SELECT Employee_name INTO v_emp_name

   FROM Employee

   WHERE Emp_salary = (SELECT MAX(Emp_salary) FROM Employee);


   DBMS_OUTPUT.PUT_LINE('Employee with Max Salary: ' || v_emp_name);

END;

/
```



Question 2: Find Shop Managed by Admin 'Emily Davis' (Subquery in WHERE clause)

Answer:

```
DECLARE
   v_shop_name Shop.Shop_name%TYPE;
BEGIN
   SELECT Shop_name INTO v_shop_name
   FROM Shop
   WHERE A_ID = (SELECT Admin_ID FROM Admin WHERE Admin_name = 'Emily Davis');

   DBMS_OUTPUT.PUT_LINE('Shop managed by Emily Davis: ' || v_shop_name);
END;
/
```

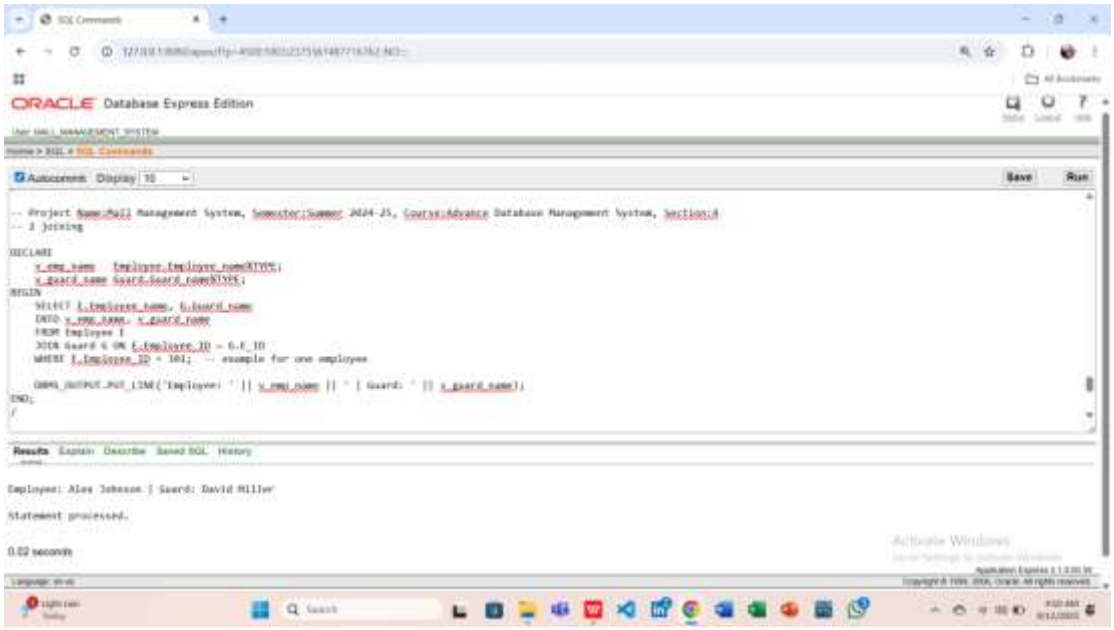


-2 joining

Question 1: Display One Employee and Guard Name (JOIN)

Answer:

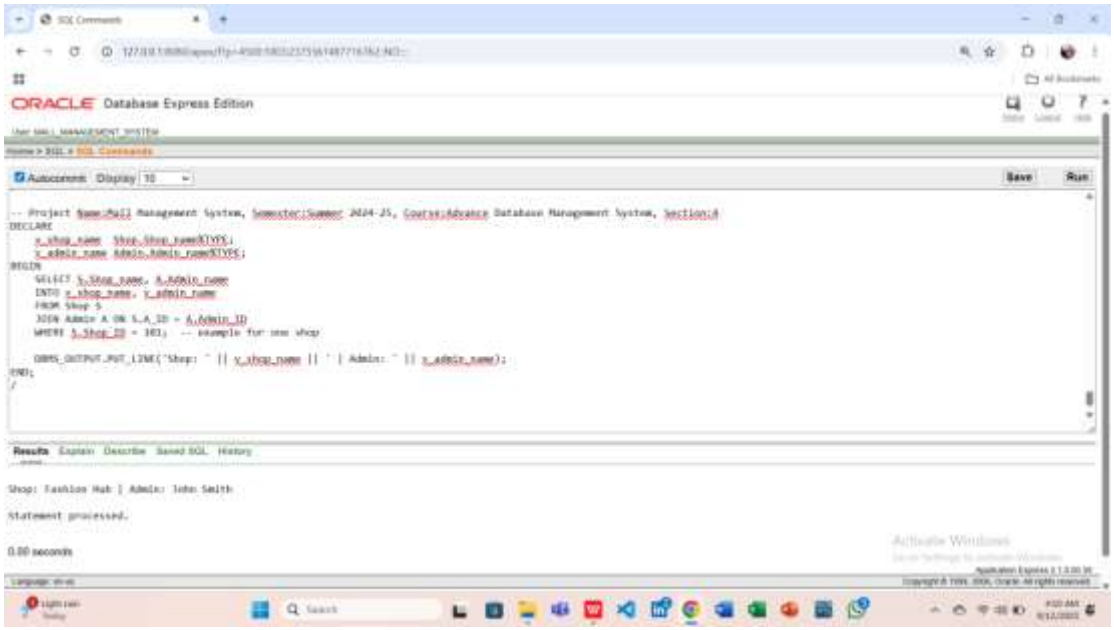
```
DECLARE
   v_emp_name   Employee.Employee_name%TYPE;
   v_guard_name Guard.Guard_name%TYPE;
BEGIN
   SELECT E.Employee_name, G.Guard_name
   INTO v_emp_name, v_guard_name
   FROM Employee E
   JOIN Guard G ON E.Employee_ID = G.E_ID
   WHERE E.Employee_ID = 301;  -- example for one employee

   DBMS_OUTPUT.PUT_LINE('Employee: ' || v_emp_name || ' | Guard: ' || v_guard_name);
END;
/
```



Question 2: Display One Shop and Admin Name (JOIN)

Answer:

```
DECLARE
   v_shop_name  Shop.Shop_name%TYPE;
   v_admin_name Admin.Admin_name%TYPE;
BEGIN
   SELECT S.Shop_name, A.Admin_name
   INTO v_shop_name, v_admin_name
   FROM Shop S
   JOIN Admin A ON S.A_ID = A.Admin_ID
   WHERE S.Shop_ID = 101;  -- example for one shop

   DBMS_OUTPUT.PUT_LINE('Shop: ' || v_shop_name || ' | Admin: ' || v_admin_name);
END;
/
```



12. Advance PL/SQL (with Exception Handling)

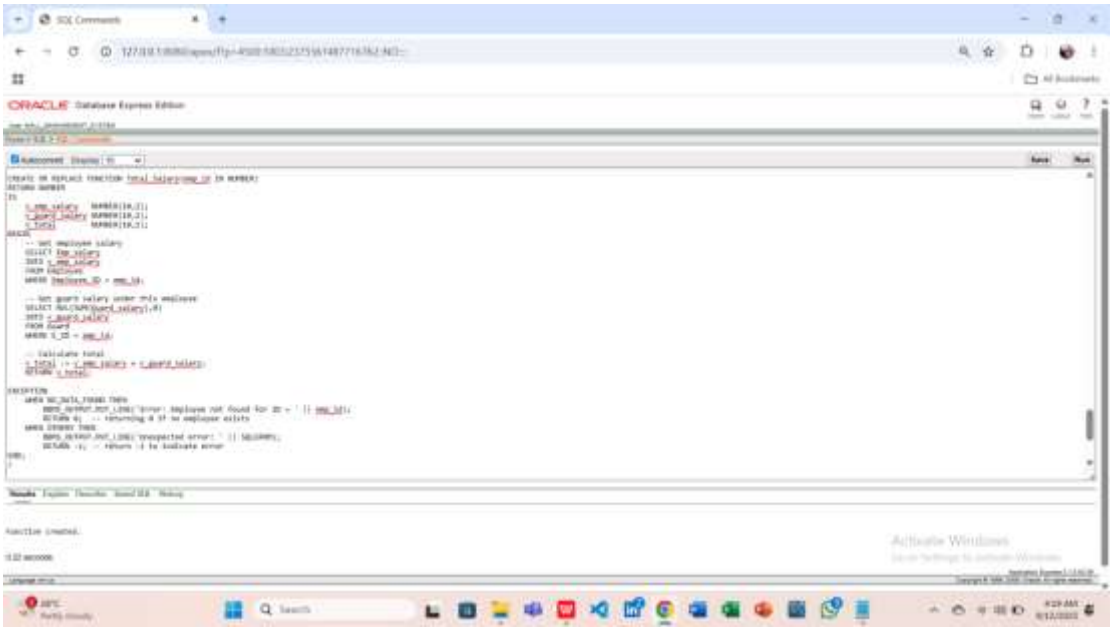
-2 stored function

Question-1: Write a stored function to calculate total salary of an Employee including Guards under them.

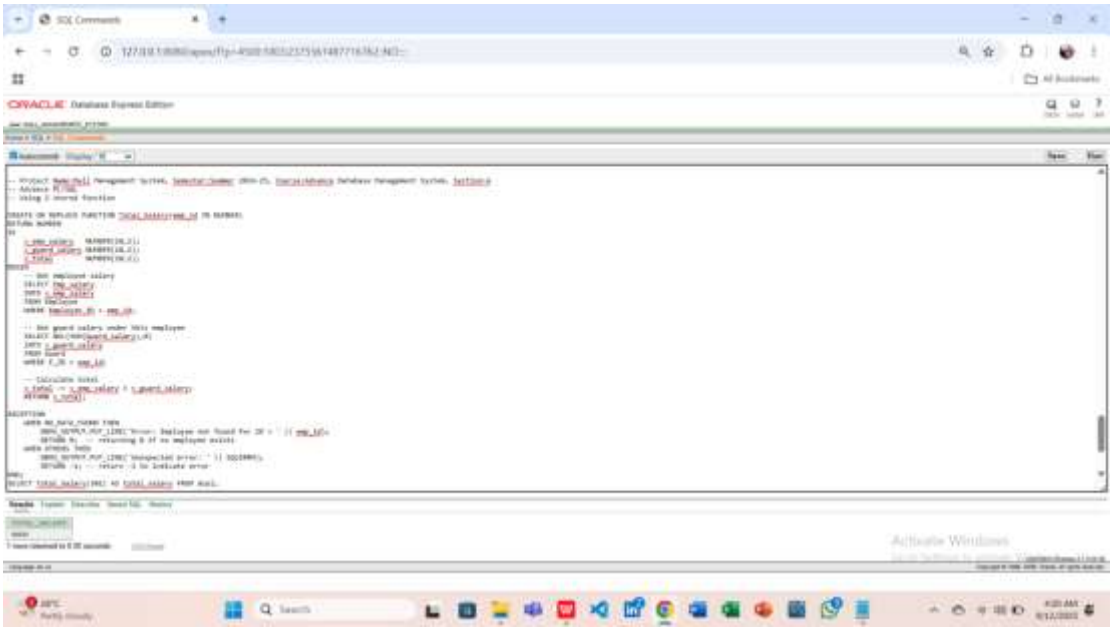
Answer:

```
CREATE OR REPLACE FUNCTION Total_Salary(emp_id IN NUMBER)

RETURN NUMBER

IS

   v_emp_salary   NUMBER(10,2);

   v_guard_salary NUMBER(10,2);

   v_total        NUMBER(10,2);

BEGIN

   -- Get employee salary

   SELECT Emp_salary

   INTO v_emp_salary

   FROM Employee
```

```
    WHERE Employee_ID = emp_id;

    -- Get guard salary under this employee
    SELECT NVL(SUM(Guard_salary),0)
    INTO v_guard_salary
    FROM Guard
    WHERE E_ID = emp_id;

    -- Calculate total
    v_total := v_emp_salary + v_guard_salary;
    RETURN v_total;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee not found for ID = ' || emp_id);
        RETURN 0;  -- returning 0 if no employee exists
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
        RETURN -1; -- return -1 to indicate error
END;
/
```



SELECT Total_Salary(301) AS total_salary FROM dual;

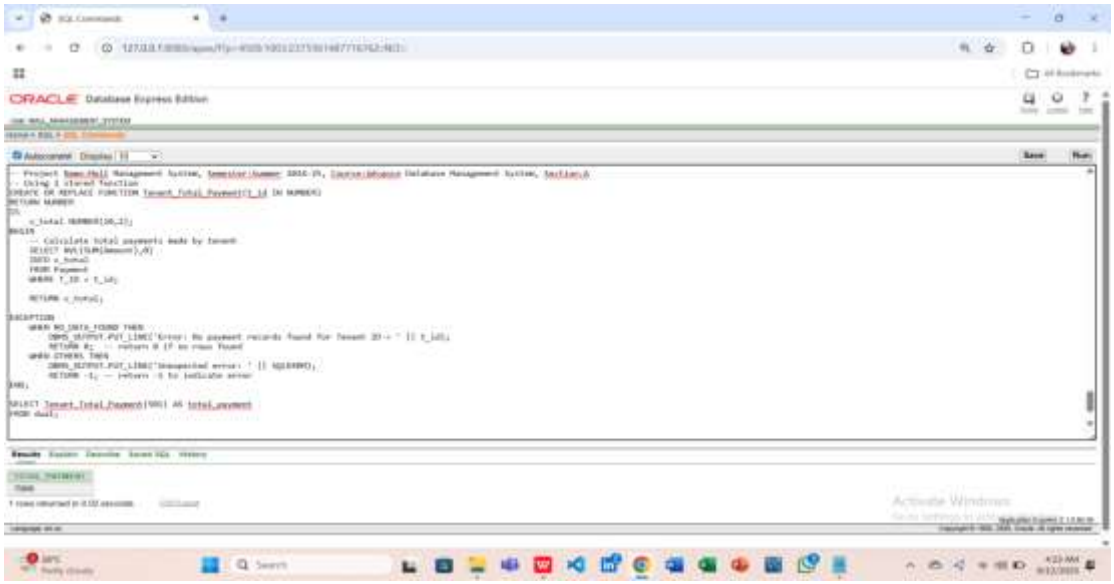


Question-2: Write a stored function to calculate total payment made by a tenant.

Answer:

```
CREATE OR REPLACE FUNCTION Tenant_Total_Payment(t_id IN NUMBER)

RETURN NUMBER

IS

   v_total NUMBER(10,2);

BEGIN
```

```
    -- Calculate total payments made by tenant
    SELECT NVL(SUM(Amount),0)
    INTO v_total
    FROM Payment
    WHERE T_ID = t_id;


    RETURN v_total;


EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: No payment records found for Tenant ID = ' || t_id);
        RETURN 0;  -- return 0 if no rows found
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
        RETURN -1; -- return -1 to indicate error
END;
SELECT Tenant_Total_Payment(501) AS total_payment
FROM dual;
```

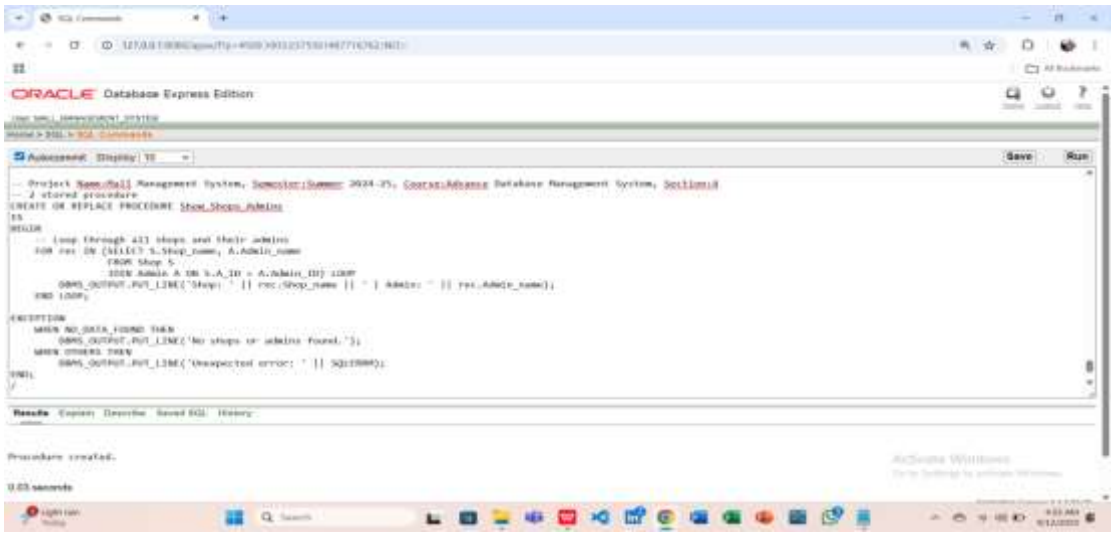


-2 stored procedure

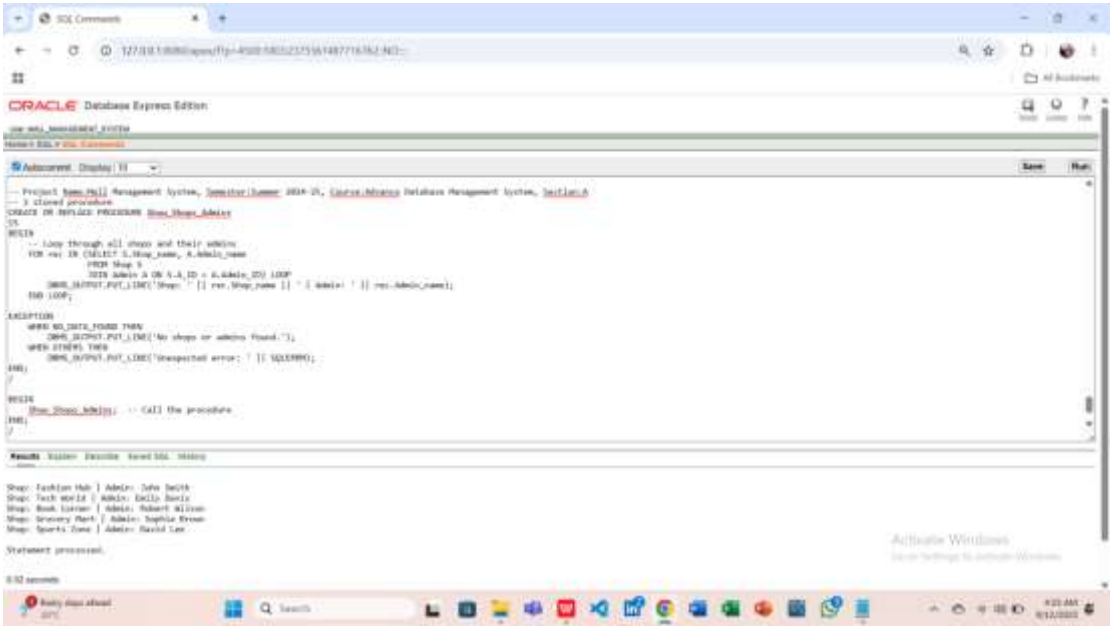
Question-1: Write a stored procedure to display all Shops and their Admin names.

Answer:

```
CREATE OR REPLACE PROCEDURE Show_Shops_Admins
IS
BEGIN
   -- Loop through all shops and their admins
   FOR rec IN (SELECT S.Shop_name, A.Admin_name
               FROM Shop S
               JOIN Admin A ON S.A_ID = A.Admin_ID) LOOP
      DBMS_OUTPUT.PUT_LINE('Shop: ' || rec.Shop_name || ' | Admin: ' || rec.Admin_name);
   END LOOP;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No shops or admins found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
END;
```



```
BEGIN
    Show_Shops_Admins;  -- Call the procedure
END;
```

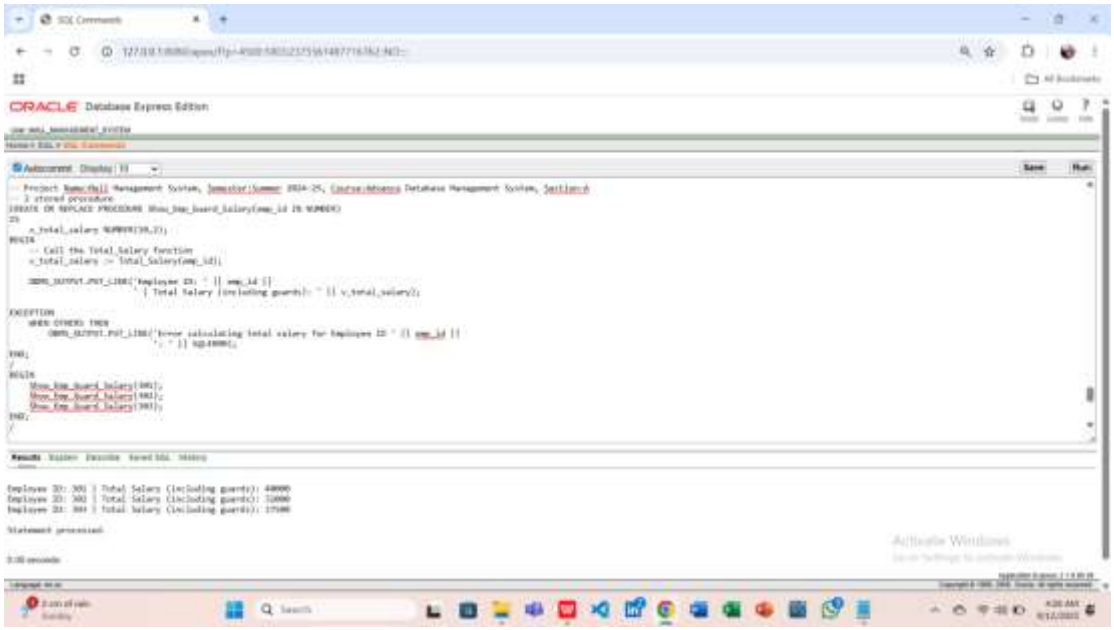


Question-2: Write a stored procedure to display Employee and total Guard salary under them.

Answer:

```
CREATE OR REPLACE PROCEDURE Show_Emp_Guard_Salary(emp_id IN NUMBER)
IS
    v_total_salary NUMBER(10,2);
BEGIN
    -- Call the Total_Salary function
    v_total_salary := Total_Salary(emp_id);

    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_id ||
                         ' | Total Salary (including guards): ' || v_total_salary);

EXCEPTION
    WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Error calculating total salary for Employee ID ' || emp_id ||
                             ': ' || SQLERRM);
END;
BEGIN
    Show_Emp_Guard_Salary(301);
    Show_Emp_Guard_Salary(302);
    Show_Emp_Guard_Salary(303);
END;
```



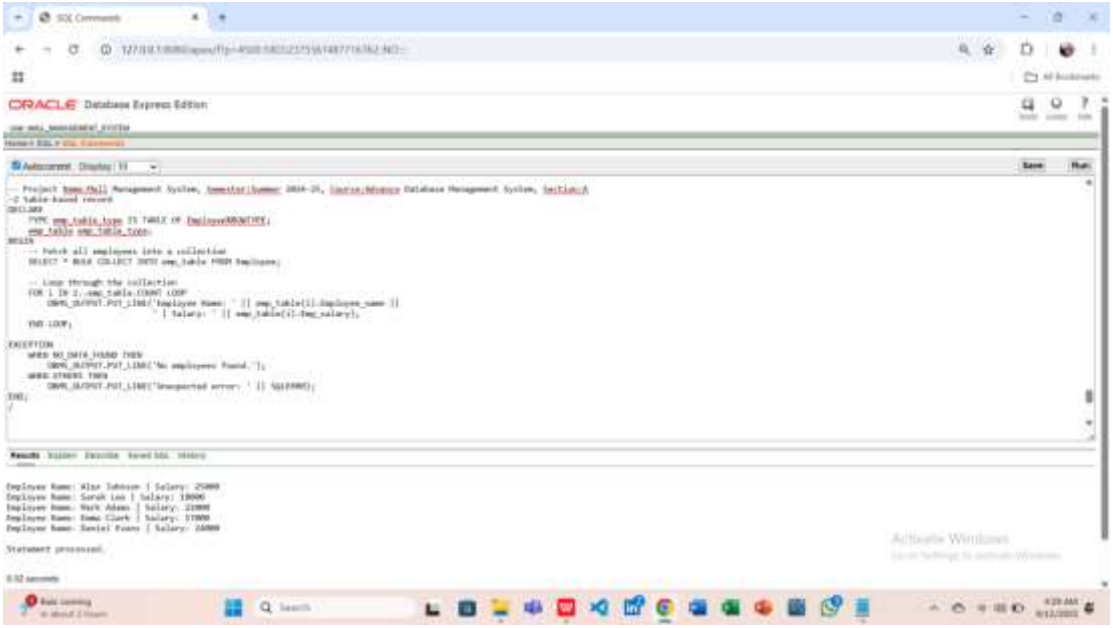
-2 table-based record

Question-1:Display all Employees using a table-based record type.

Answer:

```
DECLARE
    TYPE emp_table_type IS TABLE OF Employee%ROWTYPE;
    emp_table emp_table_type;
BEGIN
    -- Fetch all employees into a collection
    SELECT * BULK COLLECT INTO emp_table FROM Employee;
```

```
  -- Loop through the collection
  FOR i IN 1..emp_table.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_table(i).Employee_name ||
                         ' | Salary: ' || emp_table(i).Emp_salary);
  END LOOP;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employees found.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
END;
/
```

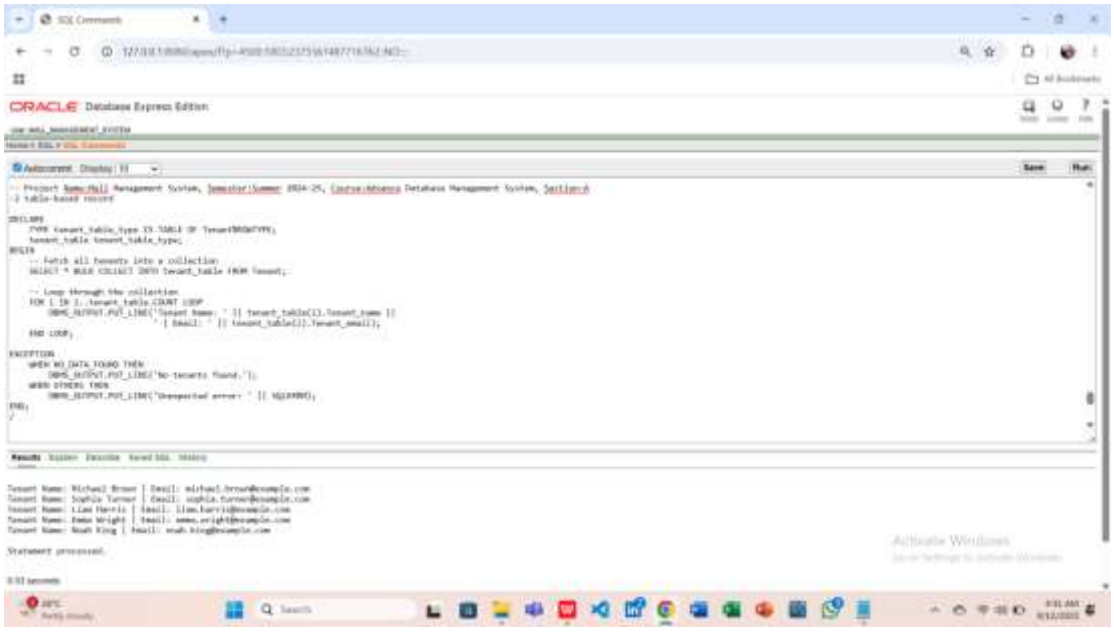


Question 2: Display all Tenants using a table-based record type.

Answer:

```
DECLARE
   TYPE tenant_table_type IS TABLE OF Tenant%ROWTYPE;
```

```
    tenant_table tenant_table_type;
BEGIN
    -- Fetch all tenants into a collection
    SELECT * BULK COLLECT INTO tenant_table FROM Tenant;

    -- Loop through the collection
    FOR i IN 1..tenant_table.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('Tenant Name: ' || tenant_table(i).Tenant_name ||
                             ' | Email: ' || tenant_table(i).Tenant_email);
    END LOOP;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No tenants found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
END;
/
```



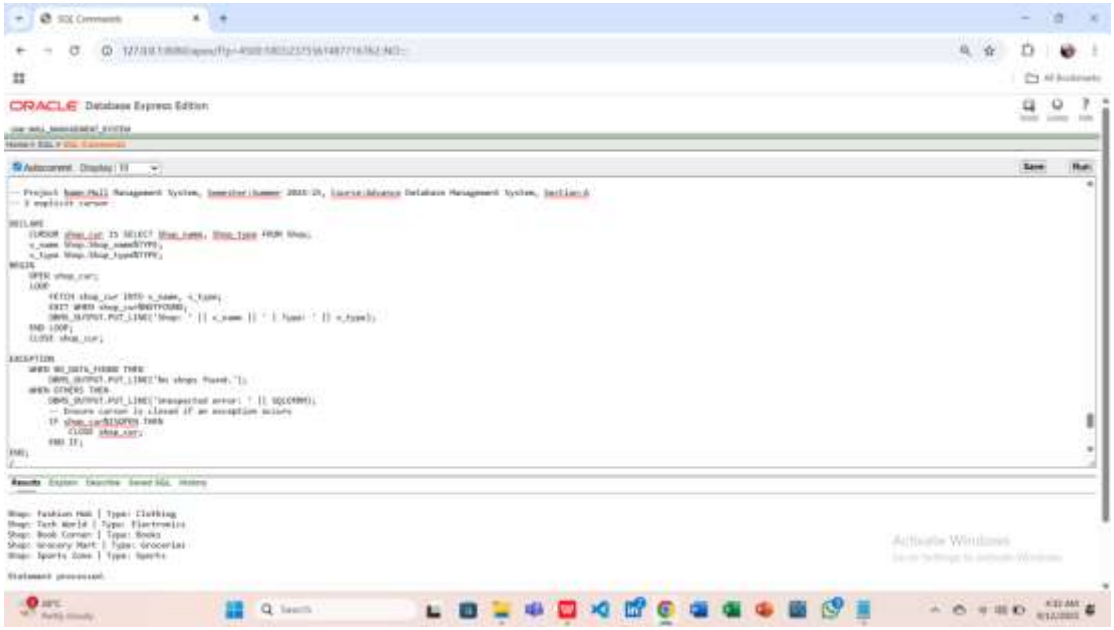
-2 explicit cursor

Question 1: Display Shop names and types using an explicit cursor.

Answer:

```
DECLARE
   CURSOR shop_cur IS SELECT Shop_name, Shop_type FROM Shop;
   v_name Shop.Shop_name%TYPE;
   v_type Shop.Shop_type%TYPE;
BEGIN
   OPEN shop_cur;
   LOOP
      FETCH shop_cur INTO v_name, v_type;
      EXIT WHEN shop_cur%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Shop: ' || v_name || ' | Type: ' || v_type);
   END LOOP;
   CLOSE shop_cur;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No shops found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
      -- Ensure cursor is closed if an exception occurs
      IF shop_cur%ISOPEN THEN
         CLOSE shop_cur;
      END IF;
END;
```

/

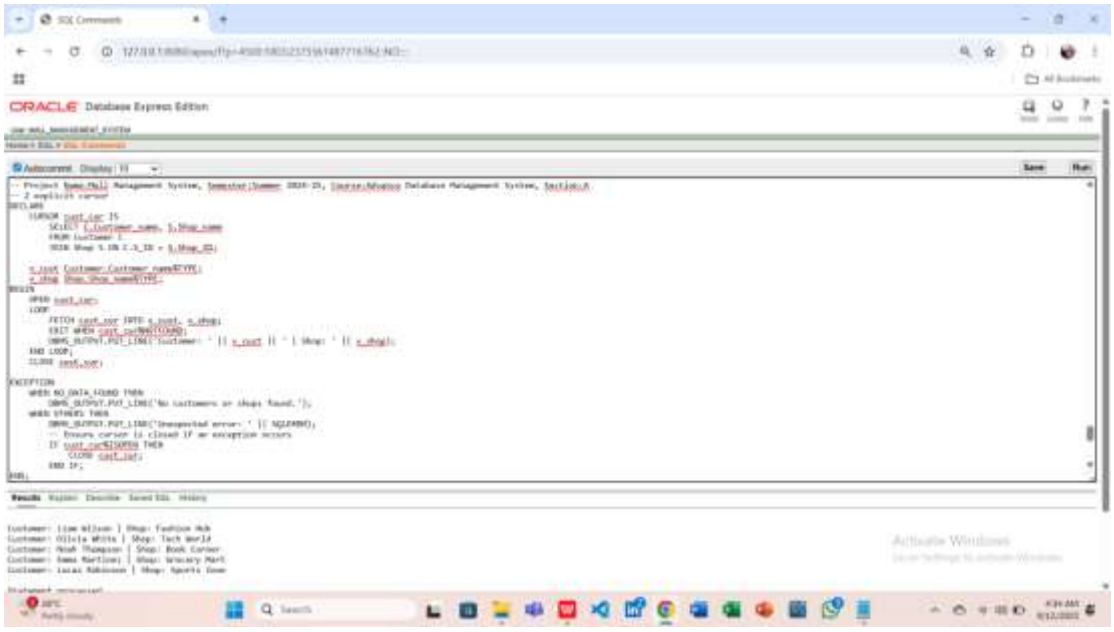


Question -2: Display Customer names and their Shop names using an explicit cursor.

Answer:

```
DECLARE
   CURSOR cust_cur IS
      SELECT C.Customer_name, S.Shop_name
      FROM Customer C
      JOIN Shop S ON C.S_ID = S.Shop_ID;

   v_cust Customer.Customer_name%TYPE;
   v_shop Shop.Shop_name%TYPE;
BEGIN
   OPEN cust_cur;
   LOOP
      FETCH cust_cur INTO v_cust, v_shop;
      EXIT WHEN cust_cur%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Customer: ' || v_cust || ' | Shop: ' || v_shop);
```

```
    END LOOP;

    CLOSE cust_cur;


EXCEPTION

    WHEN NO_DATA_FOUND THEN

        DBMS_OUTPUT.PUT_LINE('No customers or shops found.');

    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);

        -- Ensure cursor is closed if an exception occurs

        IF cust_cur%ISOPEN THEN

            CLOSE cust_cur;

        END IF;

END;

/
```

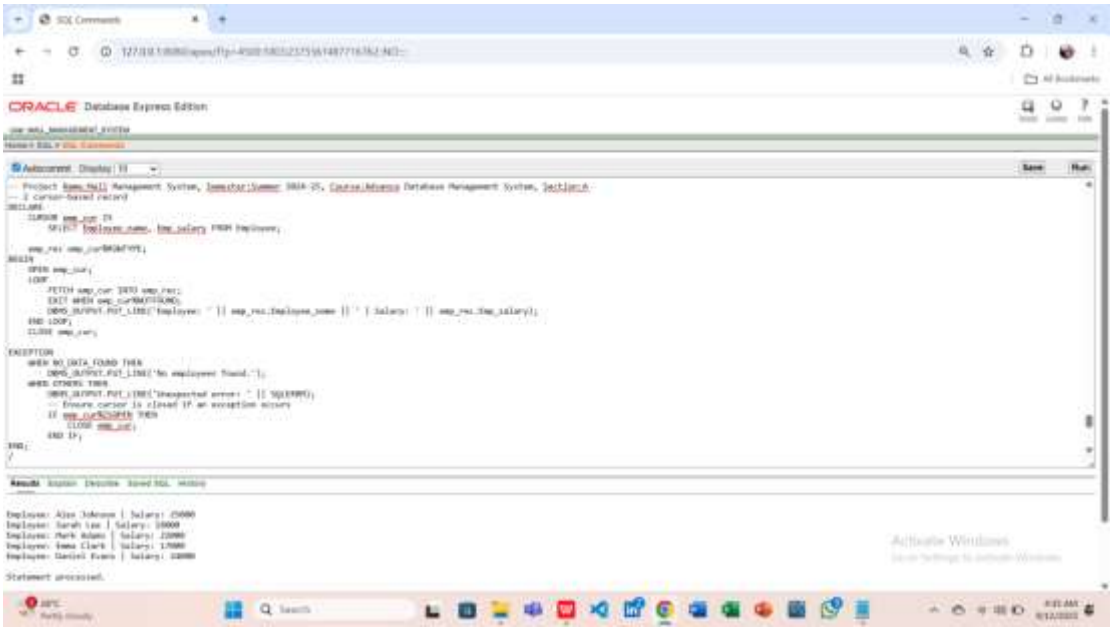


-2 cursor-based record

Question 1:Display Employee and salary using a cursor-based record.

Answer:

```
DECLARE
   CURSOR emp_cur IS
      SELECT Employee_name, Emp_salary FROM Employee;

   emp_rec emp_cur%ROWTYPE;
BEGIN
   OPEN emp_cur;
   LOOP
      FETCH emp_cur INTO emp_rec;
      EXIT WHEN emp_cur%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.Employee_name || ' | Salary: ' ||
emp_rec.Emp_salary);
   END LOOP;
   CLOSE emp_cur;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No employees found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
      -- Ensure cursor is closed if an exception occurs
      IF emp_cur%ISOPEN THEN
         CLOSE emp_cur;
      END IF;
END;
/
```

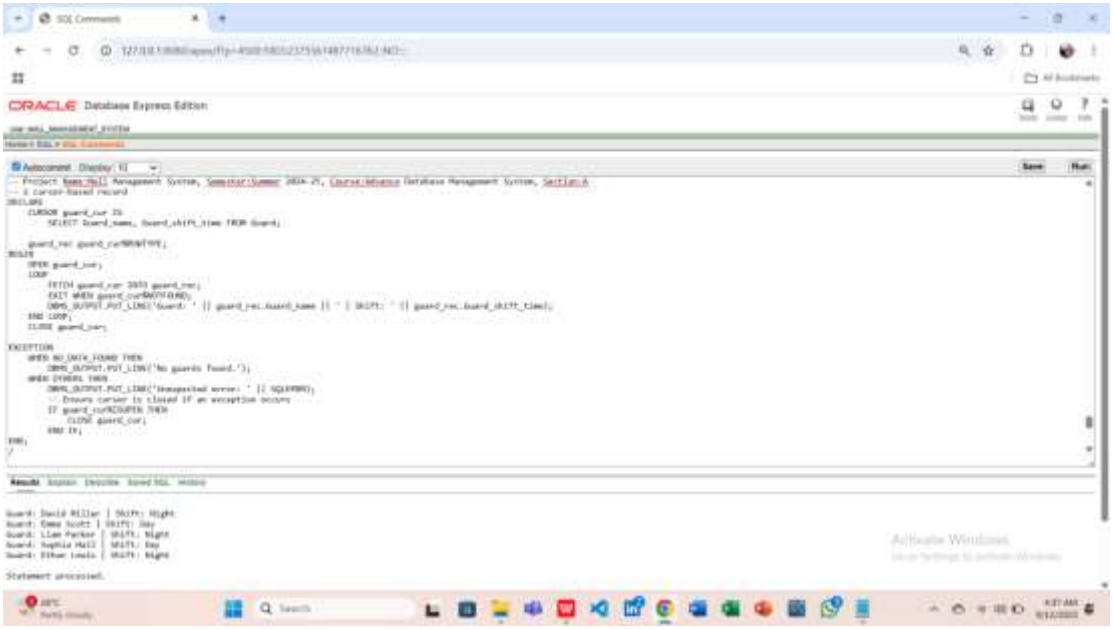


Question-2:Display Guards and their shift times using a cursor-based record.

Answer:

```
DECLARE
   CURSOR guard_cur IS
      SELECT Guard_name, Guard_shift_time FROM Guard;

   guard_rec guard_cur%ROWTYPE;
BEGIN
   OPEN guard_cur;
   LOOP
      FETCH guard_cur INTO guard_rec;
      EXIT WHEN guard_cur%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Guard: ' || guard_rec.Guard_name || ' | Shift: ' ||
guard_rec.Guard_shift_time);
   END LOOP;
   CLOSE guard_cur;

EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No guards found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
        -- Ensure cursor is closed if an exception occurs
        IF guard_cur%ISOPEN THEN
            CLOSE guard_cur;
        END IF;
END;
/
```



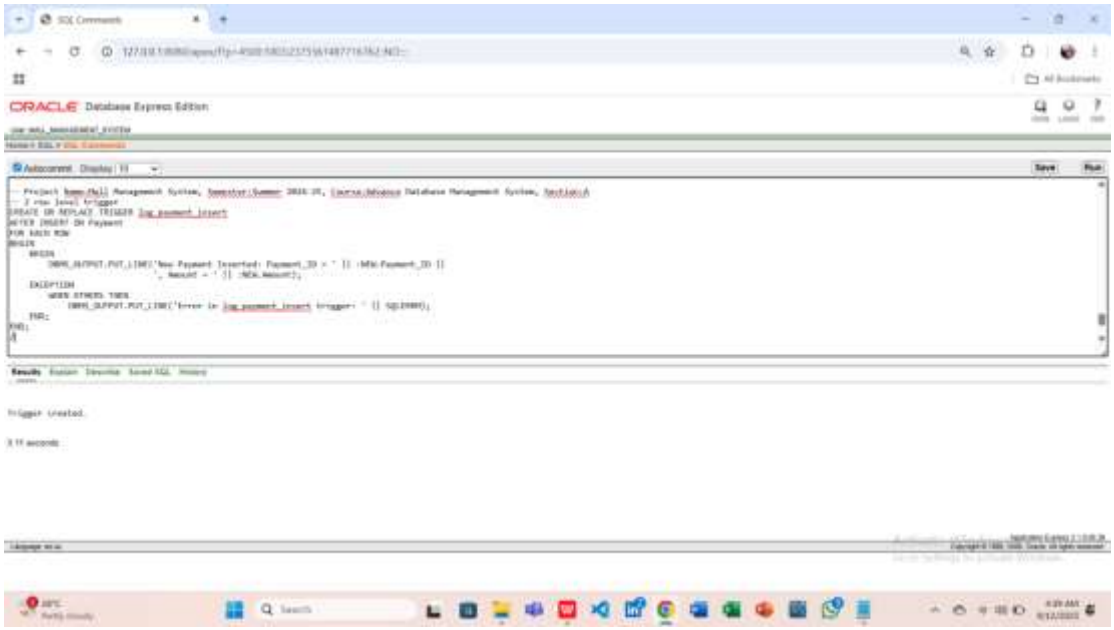
-2 row level trigger

Question-1: Create a row-level trigger to log any insert in Payment table.

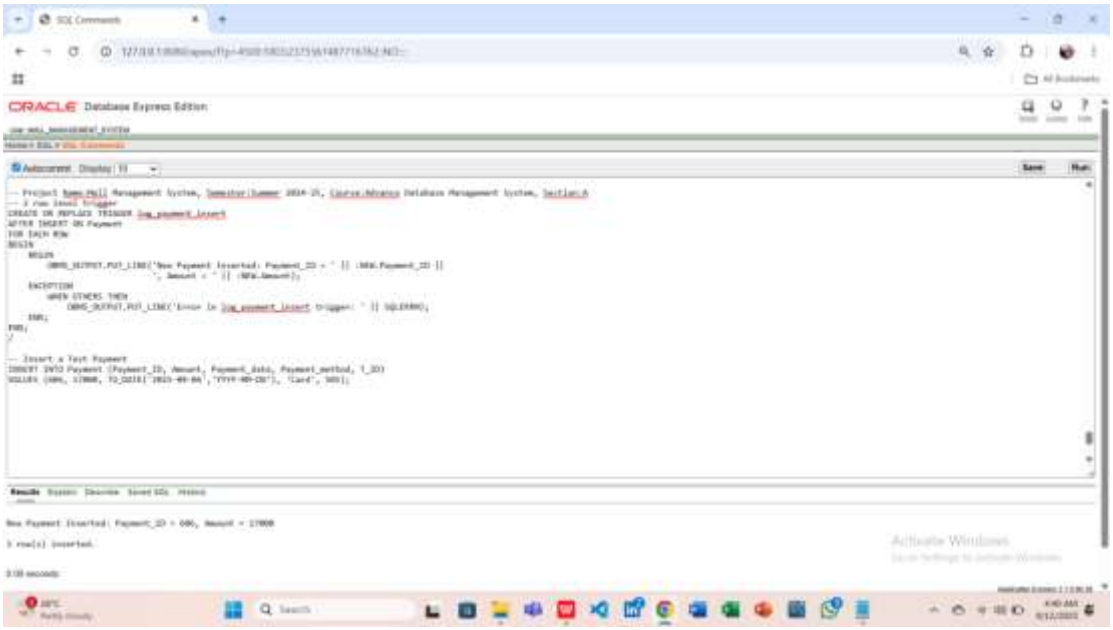
Answer:

```
CREATE OR REPLACE TRIGGER log_payment_insert
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
```

```
  BEGIN
    DBMS_OUTPUT.PUT_LINE('New Payment Inserted: Payment_ID = ' || :NEW.Payment_ID ||
                         ', Amount = ' || :NEW.Amount);
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error in log_payment_insert trigger: ' || SQLERRM);
  END;
END;
/
```

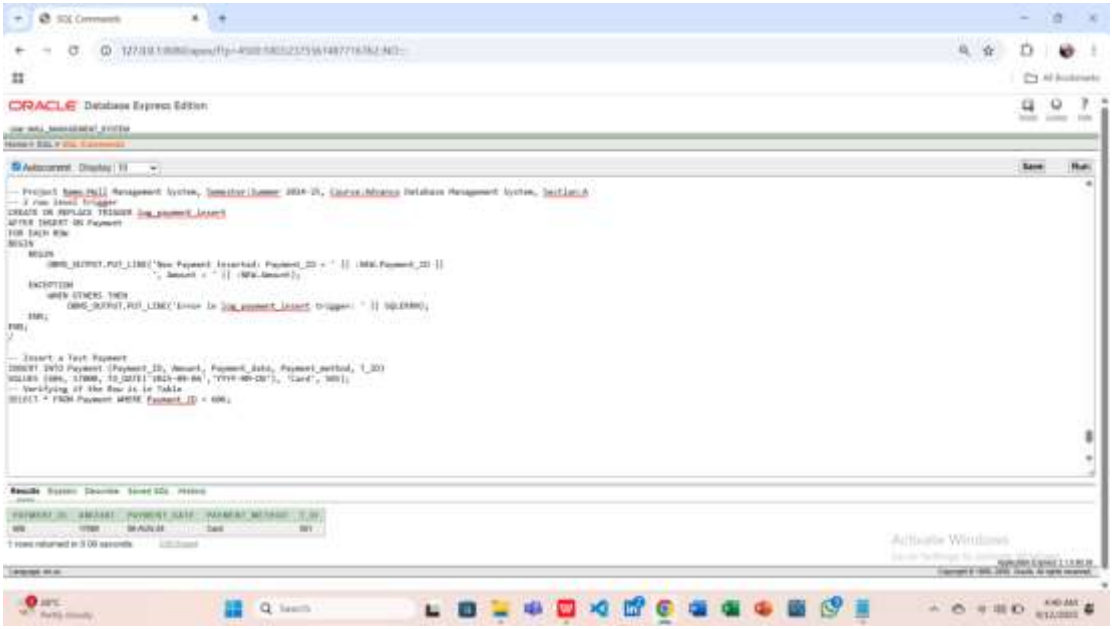



```
-- Insert a Test Payment
INSERT INTO Payment (Payment_ID, Amount, Payment_date, Payment_method, T_ID)
VALUES (606, 17000, TO_DATE('2025-08-06','YYYY-MM-DD'), 'Card', 501);
```



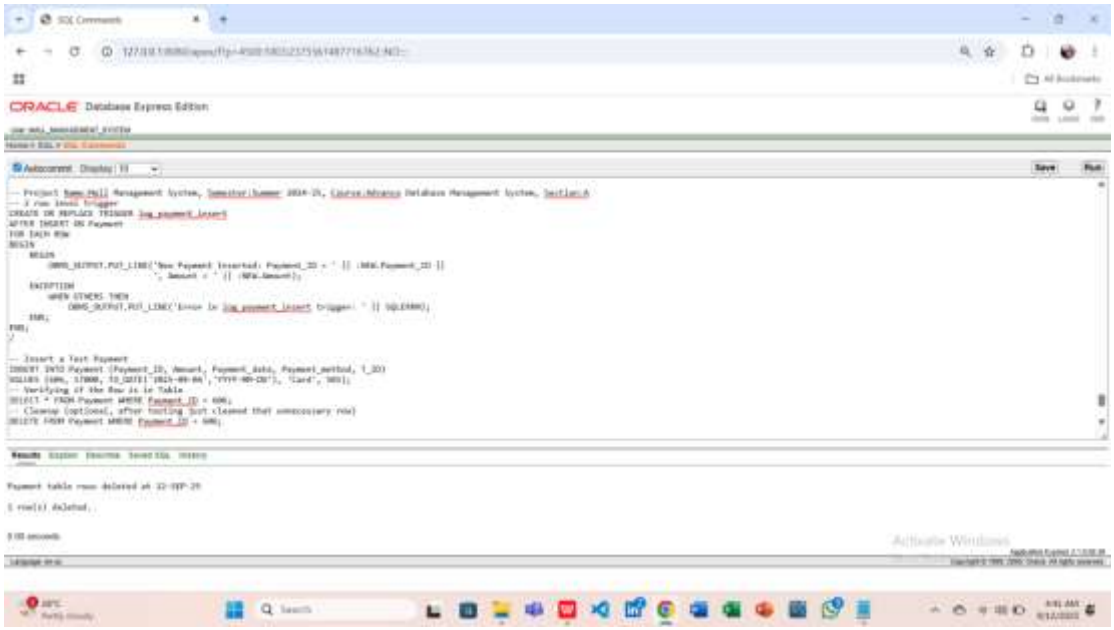
-- Verifying if the Row is in Table

SELECT * FROM Payment WHERE Payment_ID = 606;



-- Cleanup (optional, after testing just cleaned that unnecessary row)

DELETE FROM Payment WHERE Payment_ID = 606;



Question-2: Create a row-level trigger to log new Customer insertion.

Answer:

```
CREATE OR REPLACE TRIGGER log_customer_insert

AFTER INSERT ON Customer

FOR EACH ROW

BEGIN

  BEGIN

    DBMS_OUTPUT.PUT_LINE('New Customer: ' || :NEW.Customer_name ||

                  ' | Shop ID: ' || :NEW.S_ID);

  EXCEPTION

    WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('Error in log_customer_insert trigger: ' || SQLERRM);

  END;

END;

/

-- Insert a Test Customer

INSERT INTO Customer (Customer_ID, Customer_name, Customer_email, Customer_phone,
Visit_date, Feedback, S_ID)
```

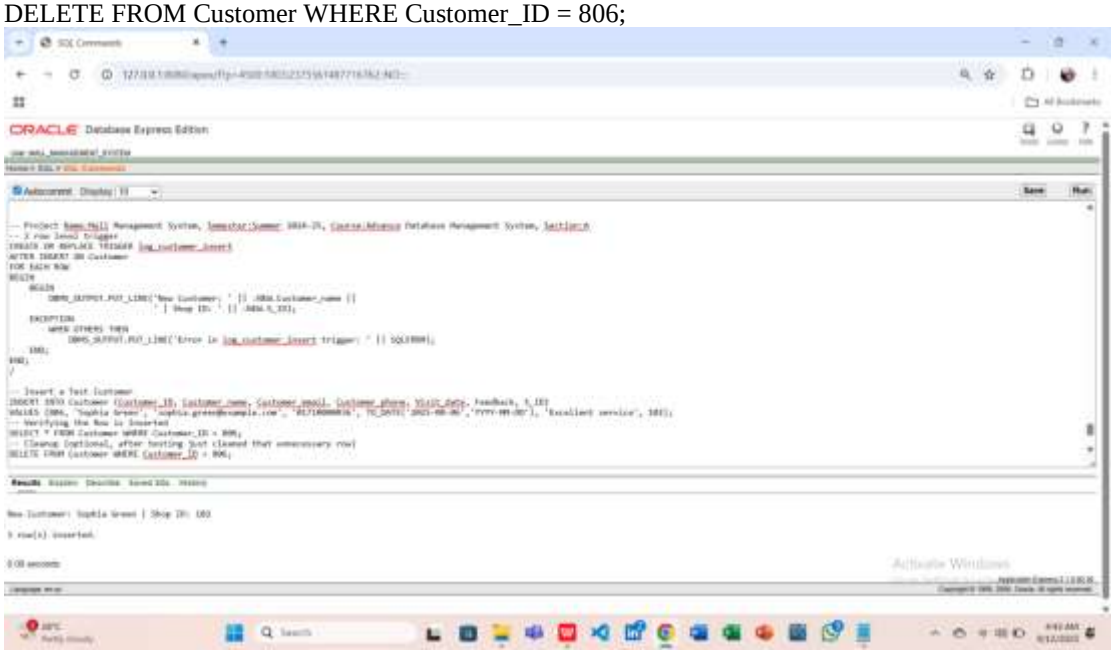
VALUES (806, 'Sophia Green', 'sophia.green@example.com', '01710000036', TO_DATE('2025-08-06','YYYY-MM-DD'), 'Excellent service', 102);

-- Verifying the Row is Inserted

SELECT * FROM Customer WHERE Customer_ID = 806;

-- Cleanup (optional, after testing just cleaned that unnecessary row)

DELETE FROM Customer WHERE Customer_ID = 806;



-2 statement level trigger

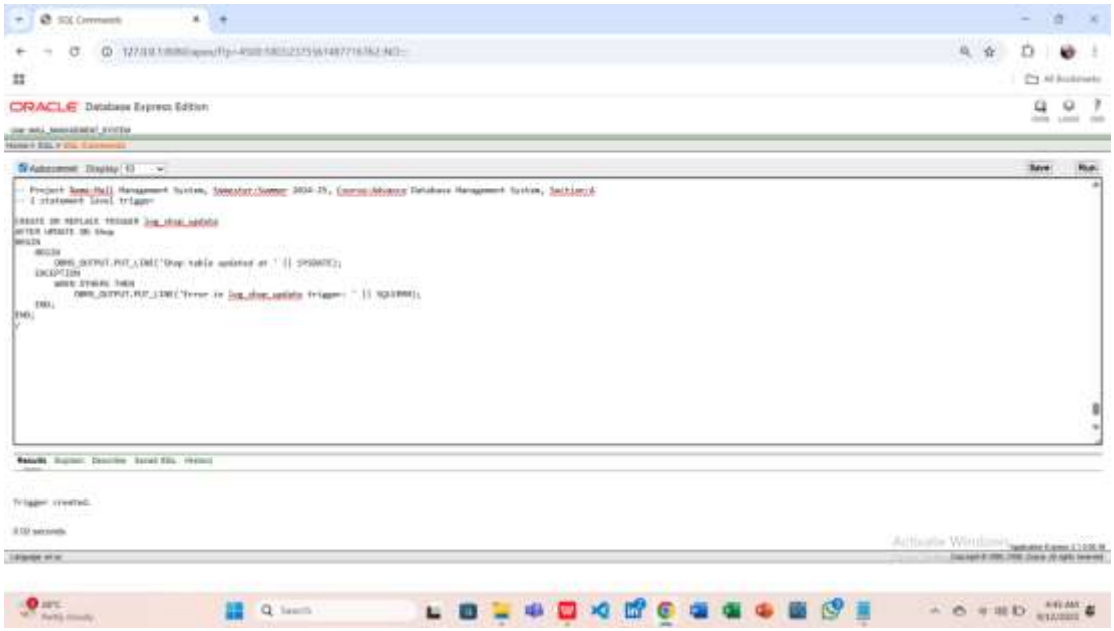
Question-1: Create a statement-level trigger to log any update in Shop table.

Answer:

```
CREATE OR REPLACE TRIGGER log_shop_update

AFTER UPDATE ON Shop

BEGIN

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Shop table updated at ' || SYSDATE);

  EXCEPTION

    WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('Error in log_shop_update trigger: ' || SQLERRM);

  END;
```

END;

/

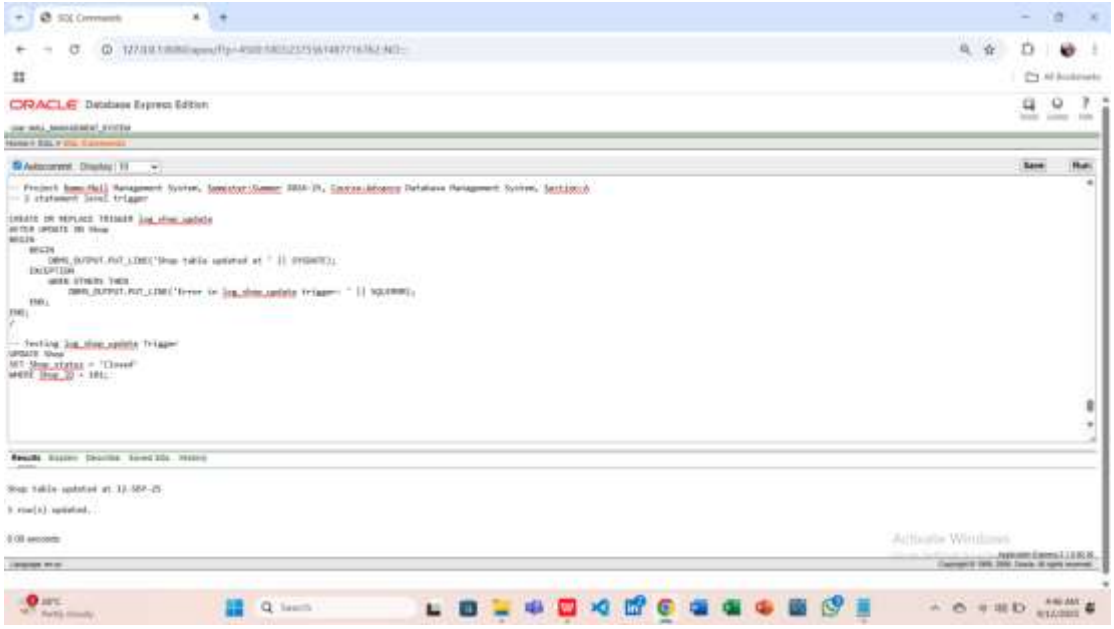


-- Testing log_shop_update Trigger

UPDATE Shop

SET Shop_status = 'Closed'

WHERE Shop_ID = 101;

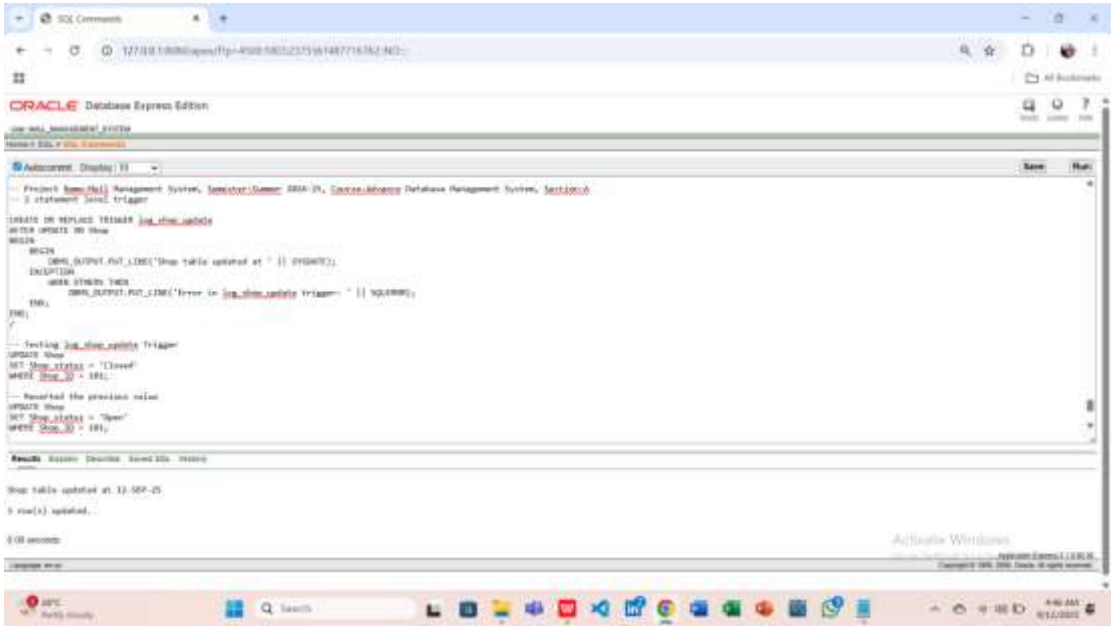


-- Reverted the previous value

UPDATE Shop

SET Shop_status = 'Open'

WHERE Shop_ID = 101;



Question-2: Create a statement-level trigger to log any delete from Payment table.

Answer:

CREATE OR REPLACE TRIGGER log_payment_delete

AFTER DELETE ON Payment

BEGIN

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Payment table rows deleted at ' || SYSDATE);

  EXCEPTION

    WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('Error in log_payment_delete trigger: ' || SQLERRM);

  END;

END;

/-- Testing the Trigger, Inserting a sample row

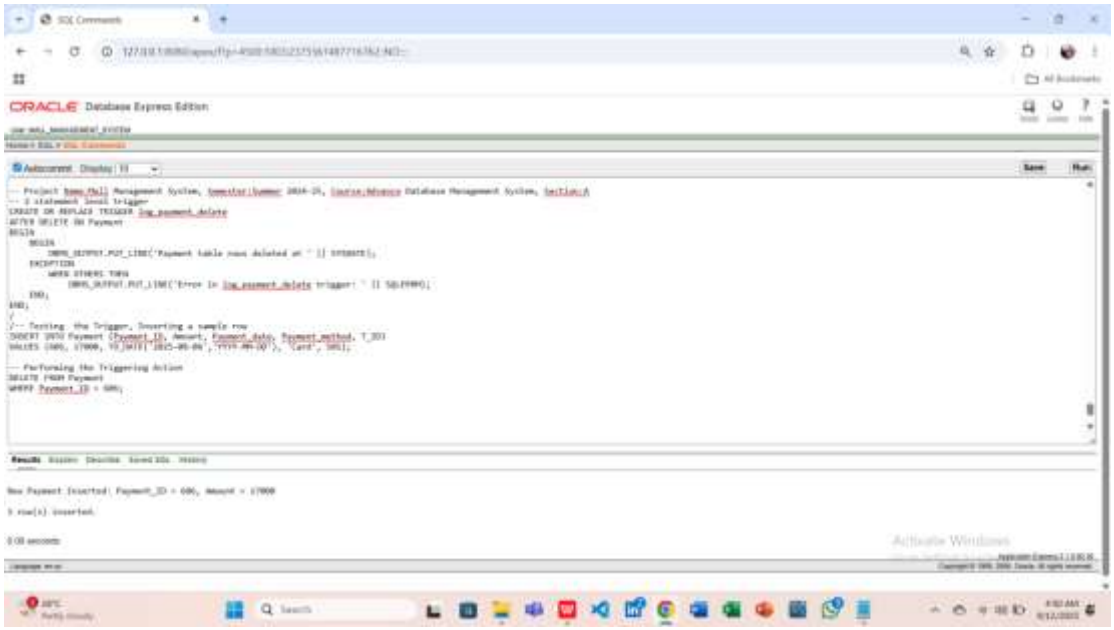
INSERT INTO Payment (Payment_ID, Amount, Payment_date, Payment_method, T_ID)

VALUES (606, 17000, TO_DATE('2025-08-06','YYYY-MM-DD'), 'Card', 501);

-- Performing the Triggering Action

DELETE FROM Payment

WHERE Payment_ID = 606;



-2 package

Question-1:Create a package to handle Employee salary queries.

Answer:

-- Package Specification

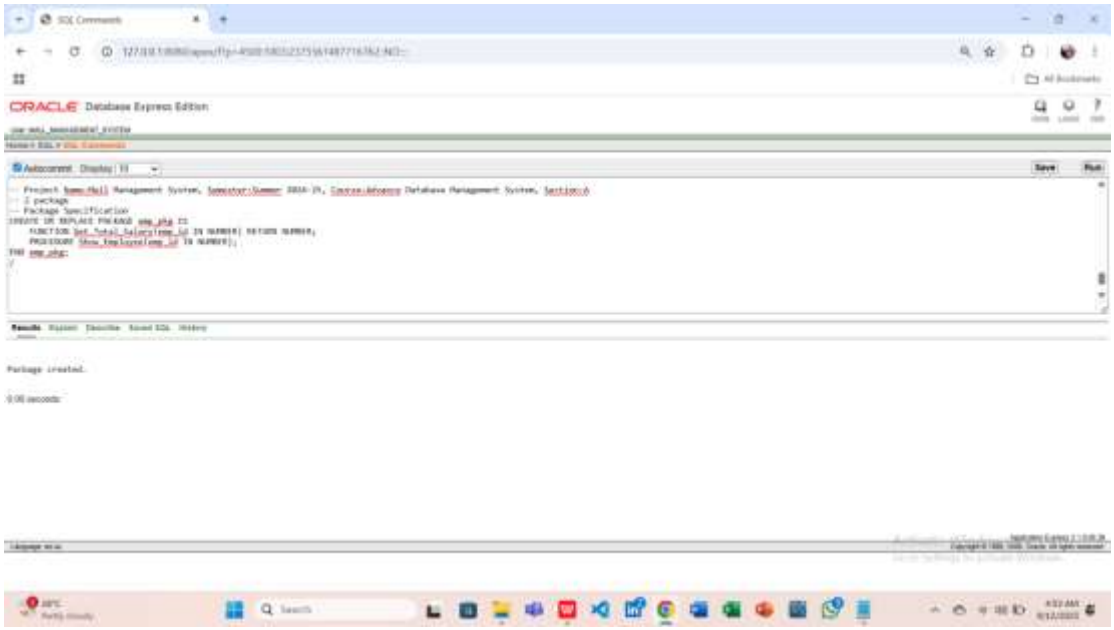
CREATE OR REPLACE PACKAGE emp_pkg IS

    FUNCTION Get_Total_Salary(emp_id IN NUMBER) RETURN NUMBER;

    PROCEDURE Show_Employee(emp_id IN NUMBER);

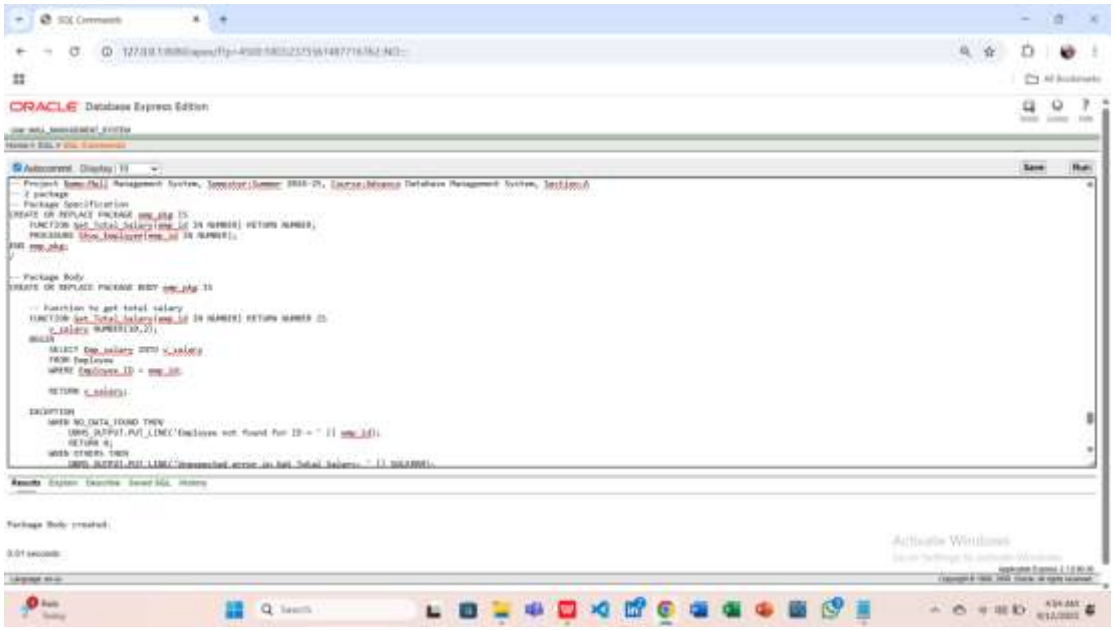
END emp_pkg;

/



```
-- Package Body
CREATE OR REPLACE PACKAGE BODY emp_pkg IS

   -- Function to get total salary
   FUNCTION Get_Total_Salary(emp_id IN NUMBER) RETURN NUMBER IS
      v_salary NUMBER(10,2);
   BEGIN
      SELECT Emp_salary INTO v_salary
      FROM Employee
      WHERE Employee_ID = emp_id;

      RETURN v_salary;

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE('Employee not found for ID = ' || emp_id);
         RETURN 0;
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Unexpected error in Get_Total_Salary: ' || SQLERRM);
         RETURN -1;
```

```
  END Get_Total_Salary;

  -- Procedure to display employee info
  PROCEDURE Show_Employee(emp_id IN NUMBER) IS
    v_name Employee.Employee_name%TYPE;
    v_salary NUMBER(10,2);
  BEGIN
    SELECT Employee_name, Emp_salary INTO v_name, v_salary
    FROM Employee
    WHERE Employee_ID = emp_id;

    DBMS_OUTPUT.PUT_LINE('Employee: ' || v_name || ' | Salary: ' || v_salary);

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Employee not found for ID = ' || emp_id);
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Unexpected error in Show_Employee: ' || SQLERRM);
  END Show_Employee;

END emp_pkg;
/
```

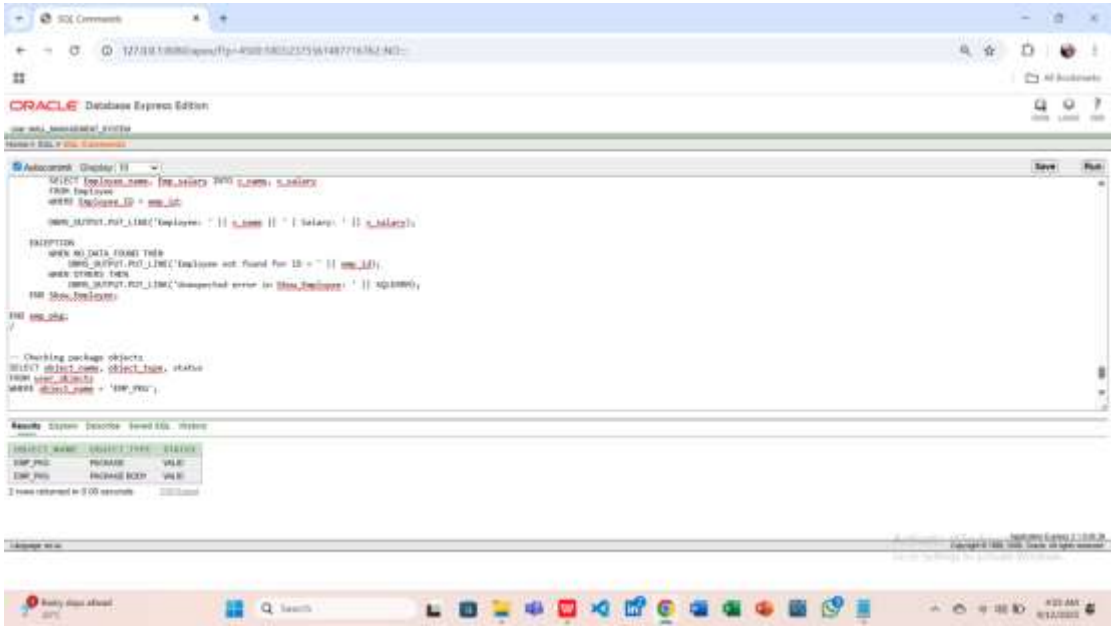


-- Checking package objects

SELECT object_name, object_type, status

FROM user_objects

WHERE object_name = 'EMP_PKG';



Question-2:Create a package to handle Tenant payment queries.

Answer:

-- Package Specification

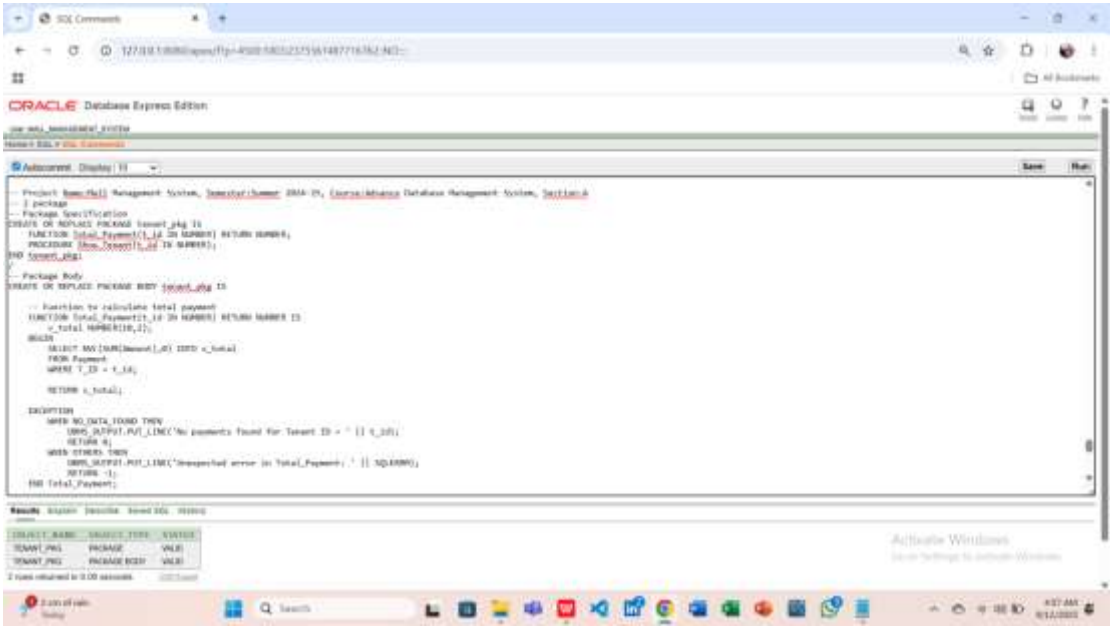
CREATE OR REPLACE PACKAGE tenant_pkg IS

    FUNCTION Total_Payment(t_id IN NUMBER) RETURN NUMBER;

    PROCEDURE Show_Tenant(t_id IN NUMBER);

```
END tenant_pkg;
/
-- Package Body
CREATE OR REPLACE PACKAGE BODY tenant_pkg IS

   -- Function to calculate total payment
   FUNCTION Total_Payment(t_id IN NUMBER) RETURN NUMBER IS
      v_total NUMBER(10,2);
   BEGIN
      SELECT NVL(SUM(Amount),0) INTO v_total
      FROM Payment
      WHERE T_ID = t_id;

      RETURN v_total;

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE('No payments found for Tenant ID = ' || t_id);
         RETURN 0;
      WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Unexpected error in Total_Payment: ' || SQLERRM);
         RETURN -1;
   END Total_Payment;

   -- Procedure to show tenant details
   PROCEDURE Show_Tenant(t_id IN NUMBER) IS
      v_name Tenant.Tenant_name%TYPE;
      v_total NUMBER(10,2);
   BEGIN
      SELECT Tenant_name INTO v_name
      FROM Tenant
      WHERE Tenant_ID = t_id;

      v_total := Total_Payment(t_id);
```

```
        DBMS_OUTPUT.PUT_LINE('Tenant: ' || v_name || ' | Total Payment: ' || v_total);

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Tenant not found for ID = ' || t_id);
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Unexpected error in Show_Tenant: ' || SQLERRM);
    END Show_Tenant;


END tenant_pkg;
/
-- Checking package objects
SELECT object_name, object_type, status
FROM user_objects
WHERE object_name = 'TENANT_PKG';
```



13. Relational Algebra

Question 1: List the names and salaries of all Employees.

Answer : $\pi_{Employee\_name,Emp\_salary}(Employee)$

Question 2: Find the Guard working under Employee_ID = 301.

Answer : $\pi_{Guard\_name}(\sigma_{E\_ID=301}(Guard))$

Question 3: List all Shops and the name of the Admin managing them.

Answer : $\pi_{Shop\_name,Admin\_name}(Shop \bowtie_{Shop.A\_ID=Admin.Admin\_ID} Admin)$

Question 4: Find the total payment made by Tenant_ID = 501.

Answer : $\gamma_{SUM(Amount)\rightarrow Total\_Payment}(\sigma_{T\_ID=501}(Payment))$

Question 5: List all Tenants who have made payments greater than 15000.

Answer :

$\pi_{Tenant\_name}(\sigma_{Amount>15000}(Payment \bowtie_{Payment.T\_ID=Tenant.Tenant\_ID} Tenant))$

14. User Interface Design To Code Implementation (Screenshots)

15. Oracle 10g Database Connection Process

Based on the midterm UI designs, we implemented the corresponding user interfaces in **C# programming language**.

The following procedure was followed to connect Oracle 10g Database using C# programming language:

1. Oracle 10g Installation and Configuration

First, we ensured that Oracle 10g Database was properly installed and running (It was already ensured in mid-term, as we created database & also table creation, data insertion were done). We checked the listener service and noted the host name, port number (default 1521), and database SID which would be required for the connection.

2. Preparing the Development Environment

Next, we prepared the development environment in Visual Studio. A new C# Console Application project was created where we would implement the connection code.

3. Adding Oracle Data Access Library

After creating the project, we added the required Oracle Data Access library. For this, the **Oracle.ManagedDataAccess.dll** file was included in the project references. This library is necessary for establishing communication between the C# application and the Oracle database.

4. Constructing the Connection String

Then we constructed the database connection string. In the connection string, we specified the database username, password, and the data source, which contained the host, port, and Oracle SID.

Example Format:
User Id=**MALL_MANGEMENT_SYSTEM**; Password=**mall**; Data Source=localhost:1521/orcl;

5. Writing the Connection Code

After preparing the connection string, we wrote C# code to connect with the Oracle 10g database. Inside the code, we created an OracleConnection object using the connection string and opened the connection inside a try-catch block to handle any possible errors.

6. Executing a Sample SQL Query

Once the connection was successful, we executed a sample SQL query. For verification, a simple query was executed on the Oracle system table dual to fetch the current system date. The returned result confirmed that the connection was working correctly.

7. Testing the Process

Finally, we tested the entire process. The project was built and executed. On successful execution, a confirmation message displayed that the application was connected to the Oracle 10g database along with the current database date.

8. Troubleshooting and Error Handling

In case of errors, we reviewed common issues. For example, if the Oracle listener was not running, we started it using the Oracle tools. If the credentials were incorrect, we corrected them in the connection string. If the Oracle assembly was not found, we ensured that the Oracle Data Provider for .NET was properly installed.

This way, we successfully connected a C# application with the Oracle 10g Database and verified the connection by executing a query.

Issues Faced During Oracle 10g Database Connection

Listener Not Running : Sometimes the Oracle listener service was not active, which caused connection failures. We had to manually start it using Oracle tools.

Invalid Credentials : Connection attempts failed when incorrect usernames or passwords were entered. Updating the connection string with correct credentials solved this.

Missing Oracle Assembly : The project could not run if the Oracle Data Access library (Oracle.ManagedDataAccess.dll or Oracle.DataAccess.dll) was not properly installed or referenced.

Port or Network Issues: The default Oracle port (1521) was sometimes blocked by firewall or network restrictions, which prevented the C# application from connecting.

Configuration Mismatches : If the SID or service name in the connection string did not match the Oracle database configuration, the connection could not be established.

Outdated Oracle 10g Version : Since Oracle 10g is an older version, compatibility issues were faced with newer development tools and drivers. Some modern libraries do not fully support Oracle 10g, which required extra configuration and adjustments.

16. Conclusion

The Mall Management System successfully demonstrates how database-driven applications can streamline and organize mall operations. By integrating core functions such as shop and tenant management, customer tracking, inventory monitoring, billing, employee supervision, and facility maintenance, the system reduces manual errors, enhances efficiency, and ensures secure and consistent data handling. The use of proper database design principles, normalization, and PL/SQL queries ensures data integrity and reliability, while the centralized system provides quick access to information, supporting effective decision-making for mall administrators and shopkeepers. Overall, the project highlights the potential of database systems to improve real-world operations by saving time, minimizing errors, and providing structured insights.

Future Work

Although the system meets its primary objectives, there are several areas where it can be improved and expanded in the future:

1. **Web-Based User Interface** – Develop a web or mobile application connected to the database to allow mall staff, tenants, and customers to interact with the system in real time.

2. **Automation & Notifications** – Add features such as automated reminders for rent payments, stock replenishment alerts, and maintenance scheduling.

3. **Analytics & Reporting** – Integrate advanced reporting dashboards and data visualization tools for sales forecasting, customer trends, and performance analysis.

4. **Security Enhancements** – Implement stronger authentication, role-based access control, and encryption to protect sensitive tenant and customer data.

5. **Integration with IoT Devices** – Connect the system with sensors (e.g., for energy monitoring, security cameras, or foot traffic tracking) to enable smart mall management.

6. **Scalability Improvements** – Optimize the database for handling larger malls with thousands of shops, tenants, and customers without performance issues.

7. **Cloud Deployment** – Move the system to a cloud-based environment for better accessibility, reliability, and scalability.

By addressing these improvements, the Mall Management System can evolve into a fully functional, intelligent, and scalable solution capable of meeting the growing demands of modern shopping complexes.