AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH-(AIUB)

Department of Computer Science & Engineering

Course: ADVANCE DATABASE MANAGEMENT SYSTEM

SUMMER 2024-2025

Section : A

Mid 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	**100%**

Submission Deadline : 24-08-2025

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.

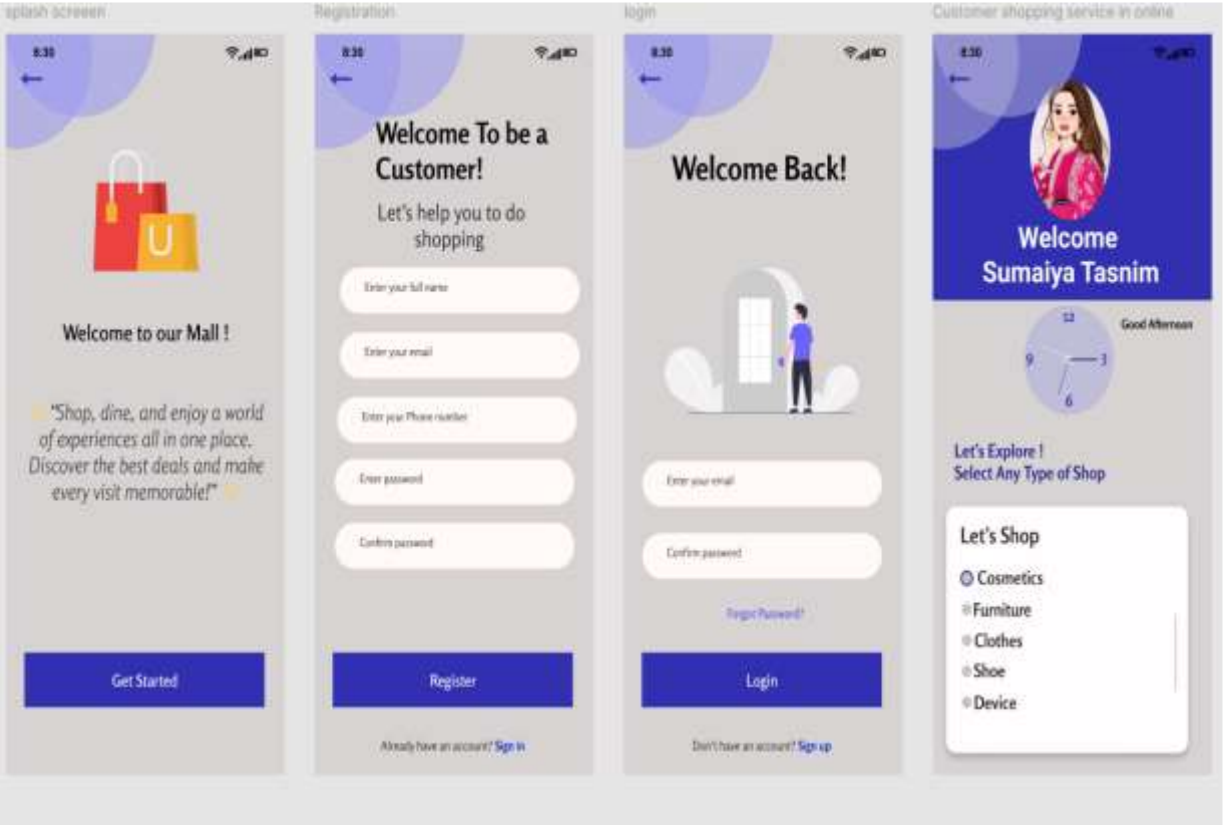
System Features

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.

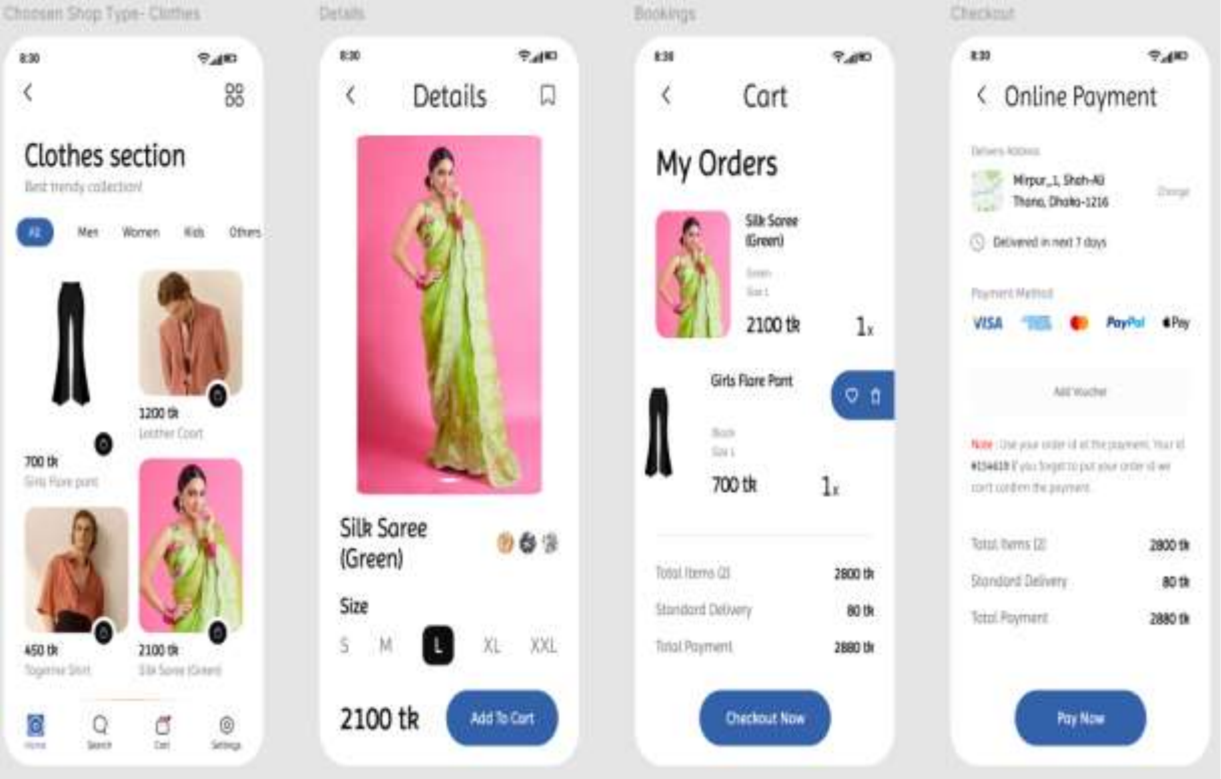
Expected Outcome

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.

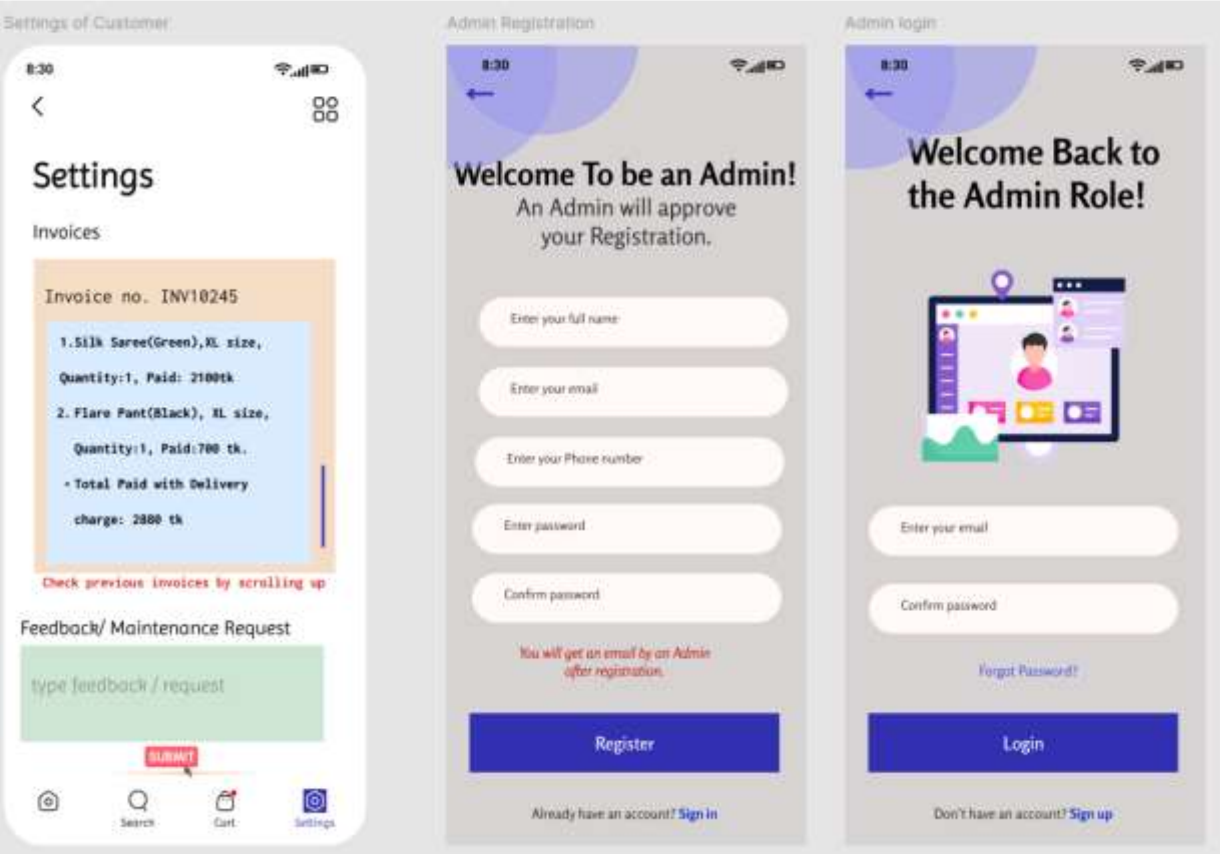
3.User Interface Planning



"Shop, dine, and enjoy a world of experiences all in one place. Discover the best deals and make every visit memorable!"



Note : Use your order id at the payment. Your id #154619 If you forget to put your order id we cant confirm the payment.



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

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!

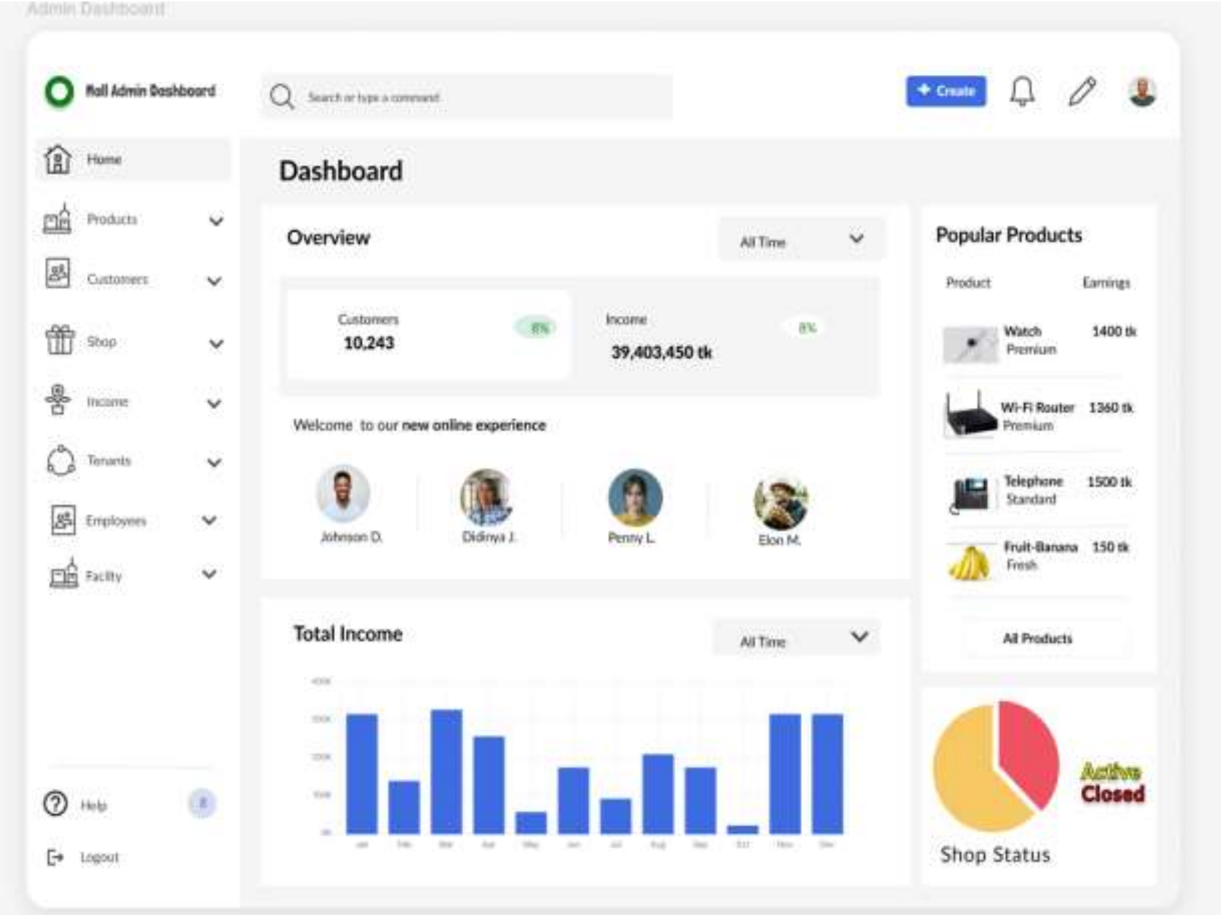
Enter your email

Confirm password

Forgot Password?

Login

Don't have an account? Sign up



Admin Dashboard

Mall Admin Dashboard

Search or type a command

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

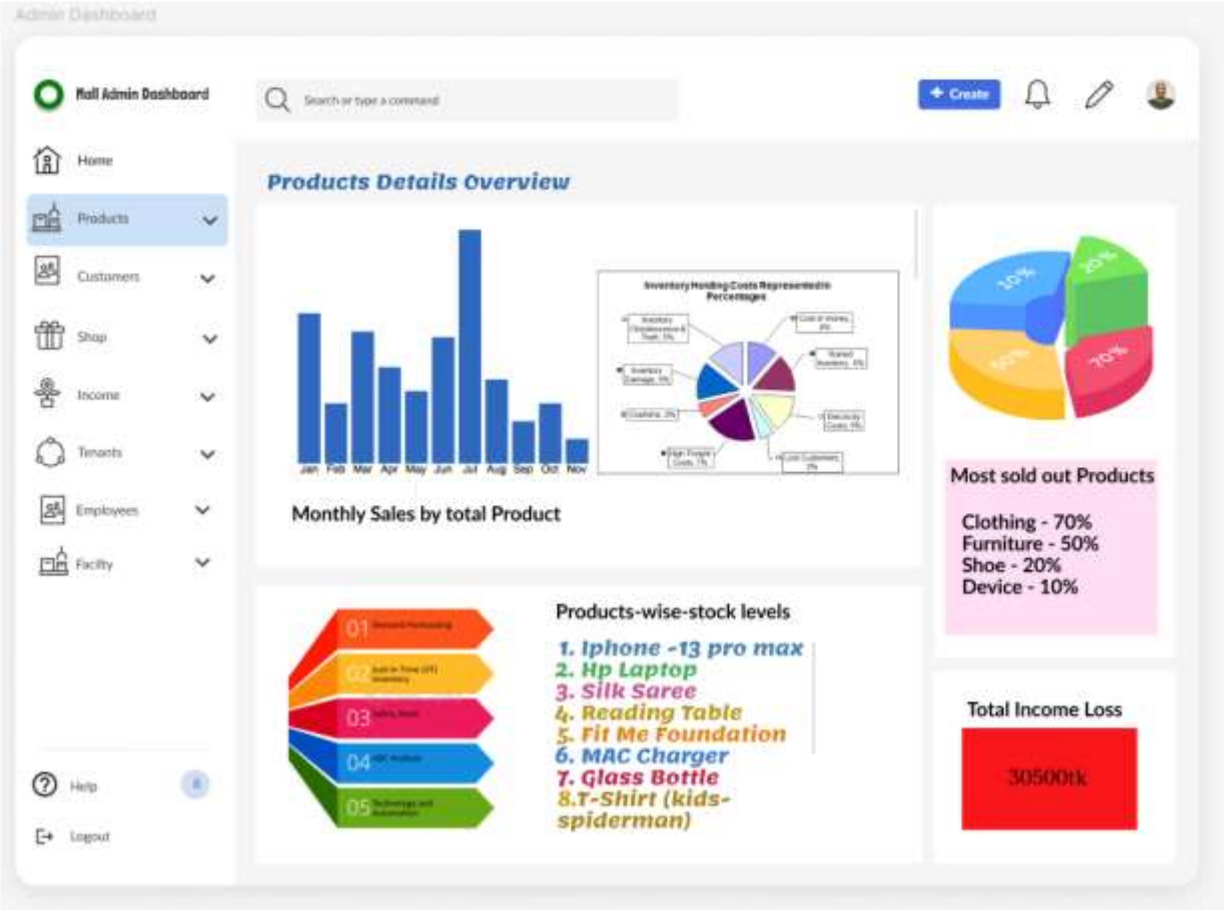
All Time

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

Shop Status



Products Details Overview

Monthly Sales by total Product

Most sold out Products

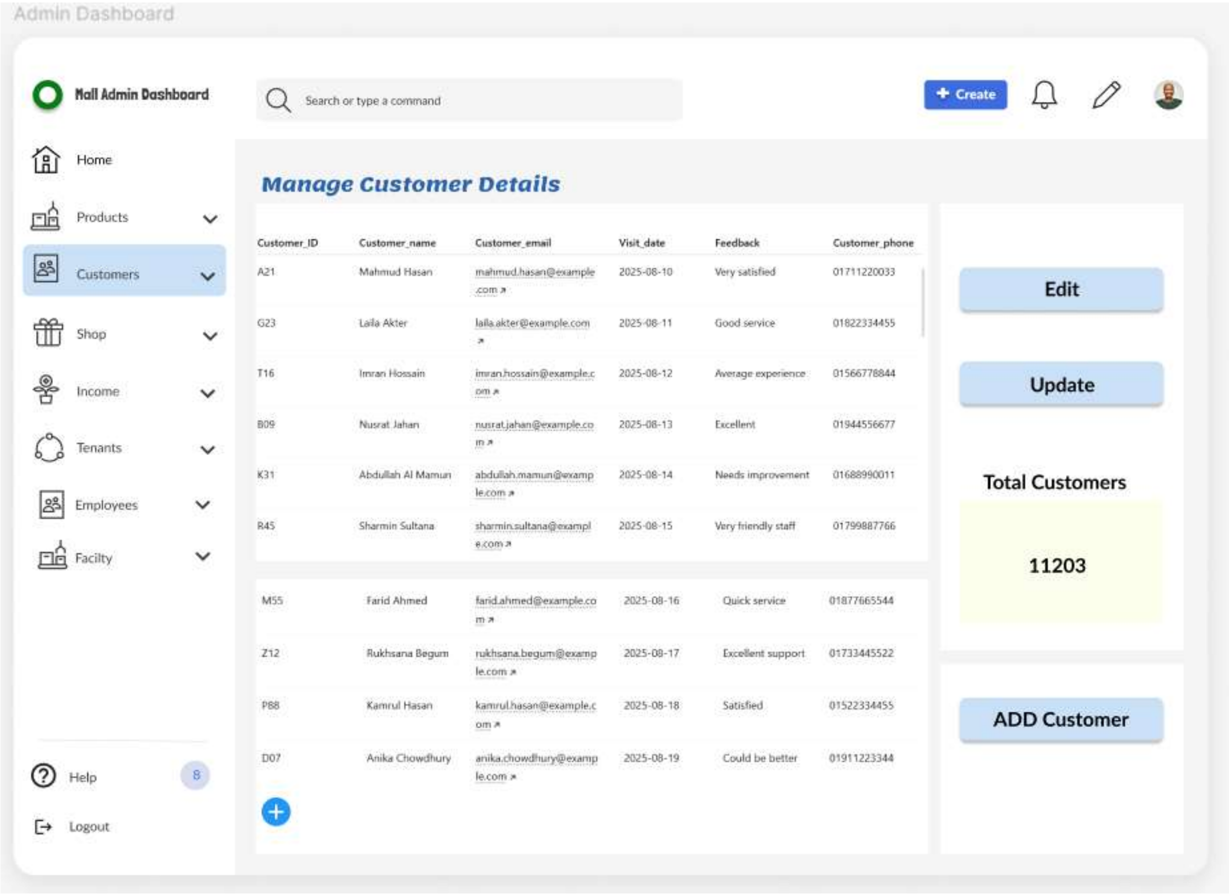
Clothing - 70%
Furniture - 50%
Shoe - 20%
Device - 10%

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

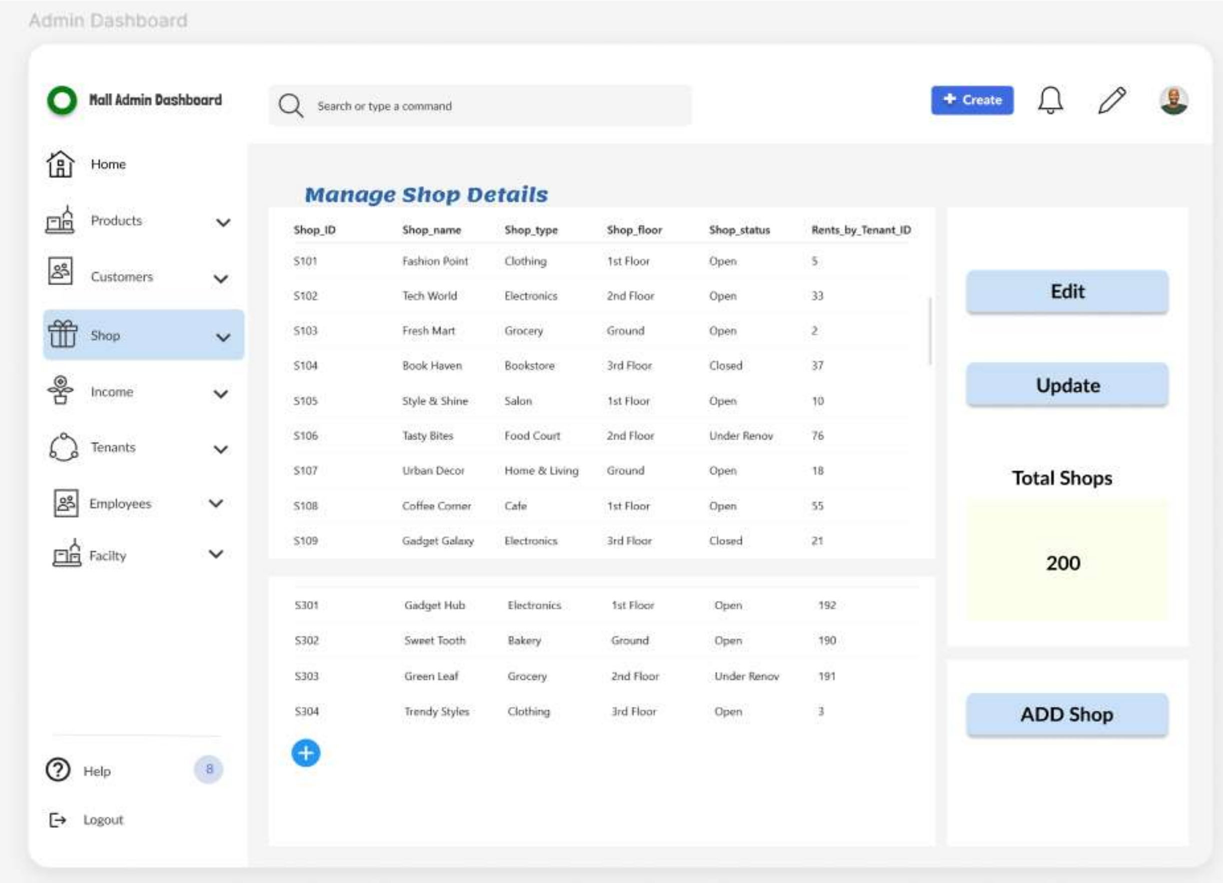


Manage Customer Details

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

Total Customers

11203



Admin Dashboard

Mall Admin Dashboard

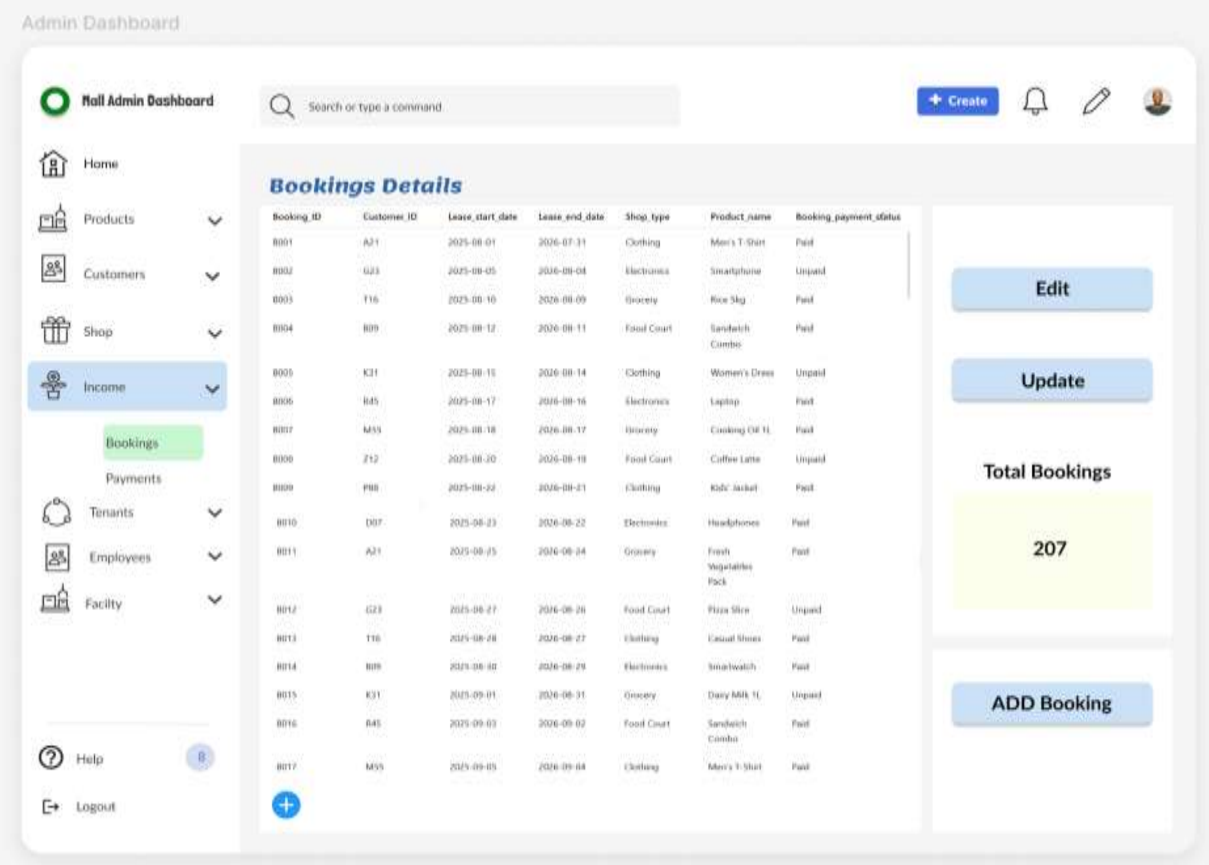
Search or type a command

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

Total Shops

200



Admin Dashboard

Mall Admin Dashboard

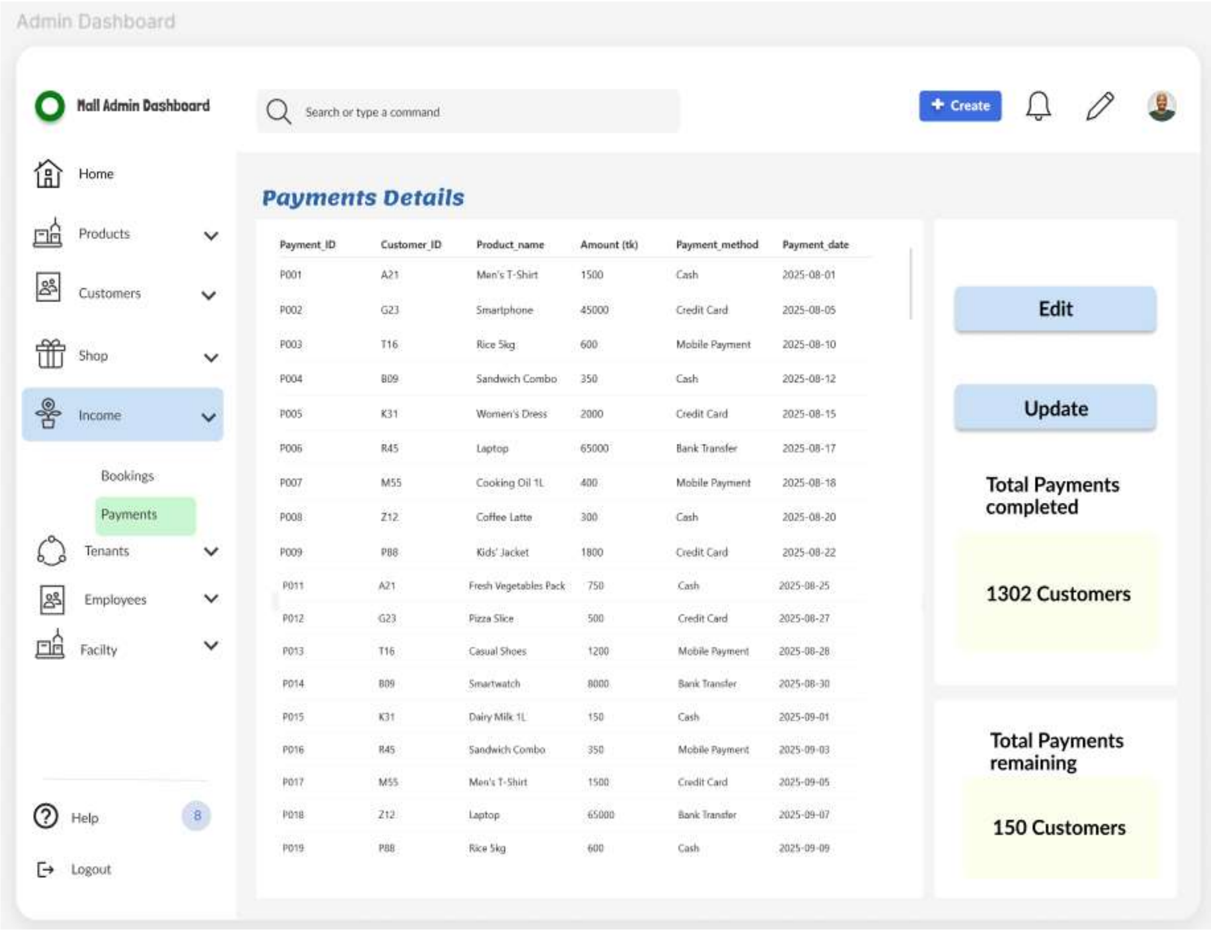
Search or type a command

Bookings Details

Booking_ID	Customer_ID	Lease_start_date	Lease_end_date	Shop_type	Product_name	Booking_payment_status
B001	A21	2025-08-01	2026-07-31	Clothing	Men's T-Shirt	Paid
B002	G23	2025-08-05	2026-08-04	Electronics	Smartphone	Unpaid
B003	T16	2025-08-10	2026-08-09	Grocery	Rice 5kg	Paid
B004	B09	2025-08-12	2026-08-11	Food Court	Sandwich Combo	Paid
B005	K31	2025-08-15	2026-08-14	Clothing	Women's Dress	Unpaid
B006	R45	2025-08-17	2026-08-16	Electronics	Laptop	Paid
B007	M55	2025-08-18	2026-08-17	Grocery	Cooking Oil 1L	Paid
B008	Z12	2025-08-20	2026-08-19	Food Court	Coffee Latte	Unpaid
B009	P88	2025-08-22	2026-08-21	Clothing	Kids' Jacket	Paid
B010	D07	2025-08-23	2026-08-22	Electronics	Headphones	Paid
B011	A21	2025-08-25	2026-08-24	Grocery	Fresh Vegetables Pack	Paid
B012	G23	2025-08-27	2026-08-26	Food Court	Pizza Slice	Unpaid
B013	T16	2025-08-28	2026-08-27	Clothing	Casual Shoes	Paid
B014	B09	2025-08-30	2026-08-29	Electronics	Smartwatch	Paid
B015	K31	2025-09-01	2026-08-31	Grocery	Dairy Milk 1L	Unpaid
B016	R45	2025-09-03	2026-09-02	Food Court	Sandwich Combo	Paid
B017	M55	2025-09-05	2026-09-04	Clothing	Men's T-Shirt	Paid

Total Bookings

207



Payments Details

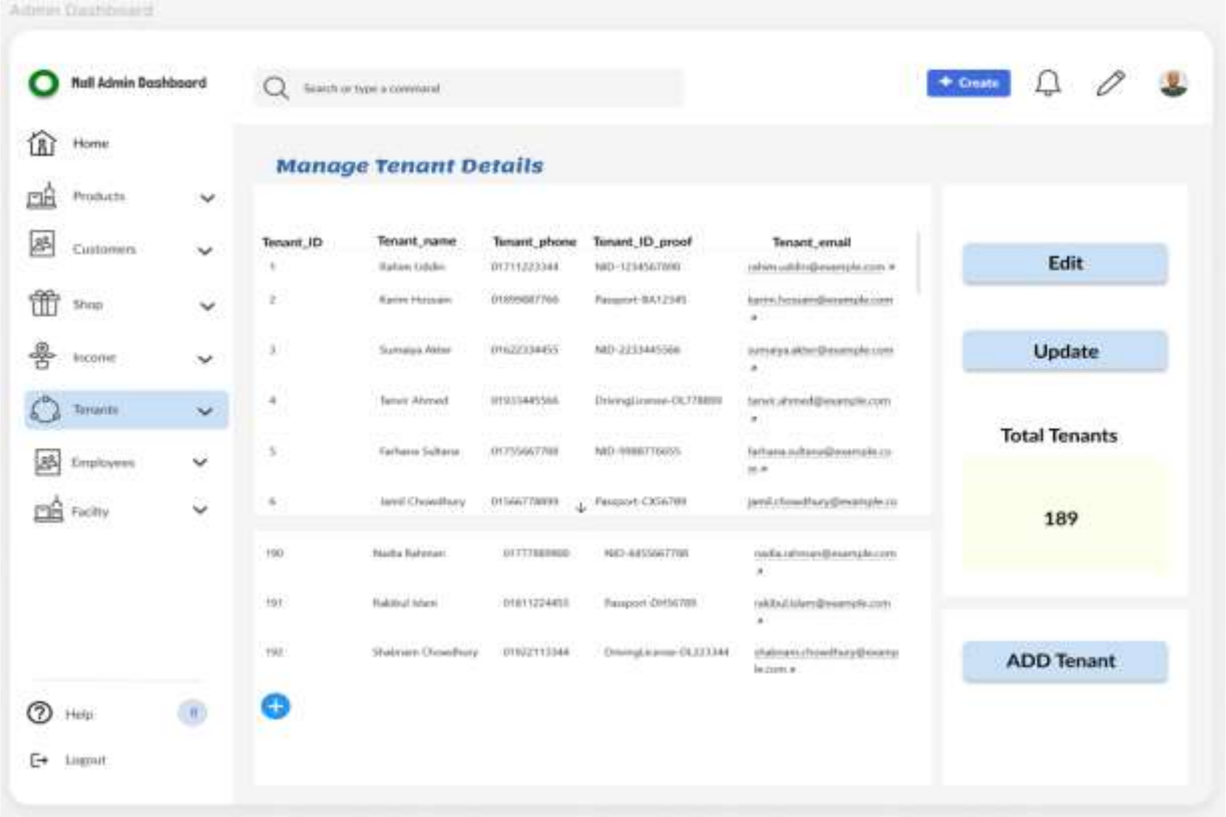
Payment_ID	Customer_ID	Product_name	Amount (tk)	Payment_method	Payment_date
P001	A21	Men's T-Shirt	1500	Cash	2025-08-01
P002	G23	Smartphone	45000	Credit Card	2025-08-05
P003	T16	Rice 5kg	600	Mobile Payment	2025-08-10
P004	B09	Sandwich Combo	350	Cash	2025-08-12
P005	K31	Women's Dress	2000	Credit Card	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

Total Payments completed

1302 Customers

Total Payments remaining

150 Customers

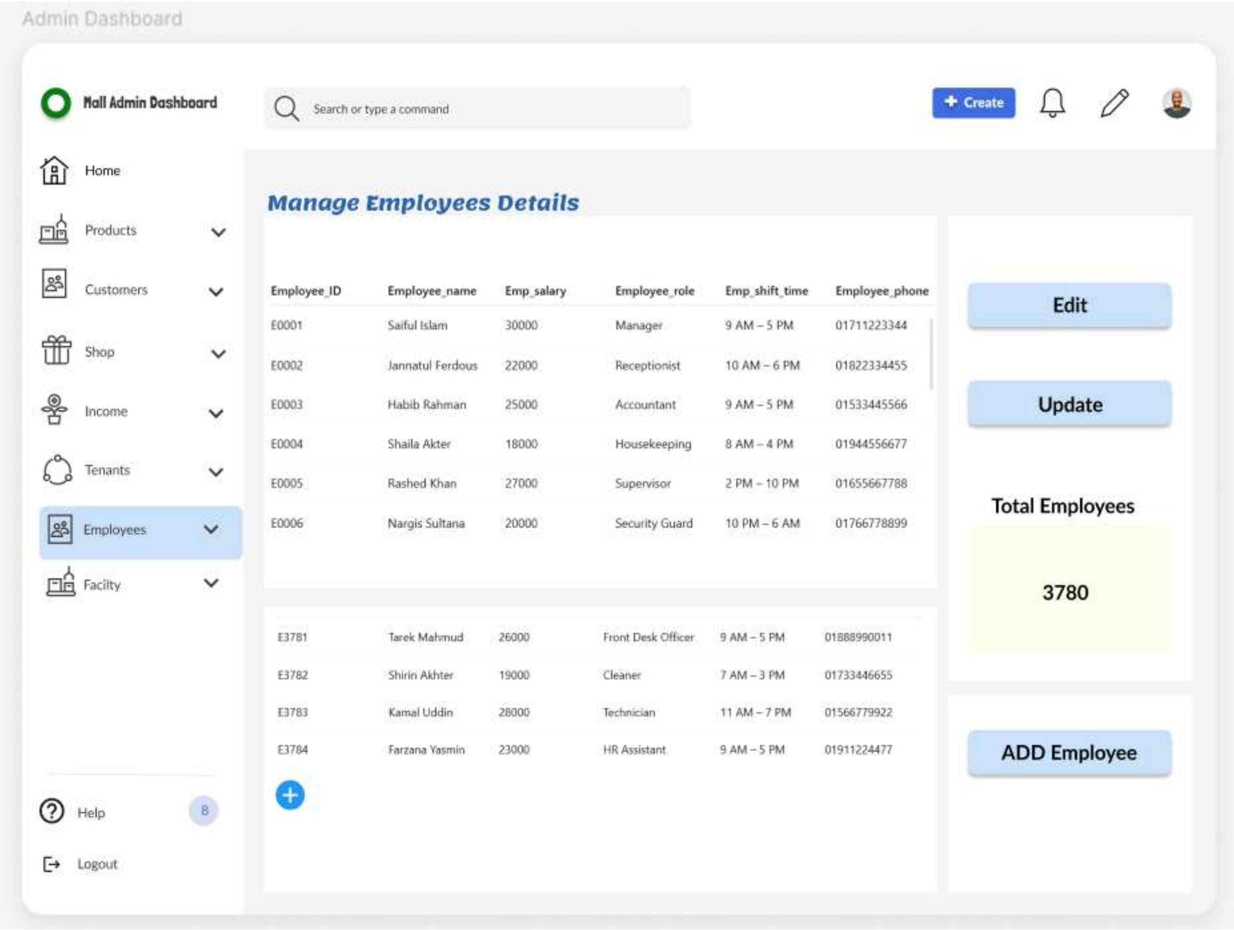


Manage Tenant Details

Tenant_ID	Tenant_name	Tenant_phone	Tenant_ID_proof	Tenant_email
1	Rahim Uddin	01711223344	NID-1234567890	rahim.uddin@example.com
2	Karim Hossain	01899887766	Passport-BA12345	karim.hossain@example.com
3	Sumaiya Akter	01622334455	NID-2233445566	sumaiya.akter@example.com
4	Tanvir Ahmed	01933445566	DrivingLicense-DL778899	tanvir.ahmed@example.com
5	Farhana Sultana	01755667788	NID-9988776655	farhana.sultana@example.com
6	Jamil Chowdhury	01566778899	Passport-CX56789	jamil.chowdhury@example.co
190	Nadia Rahman	01777888900	NID-4455667788	nadia.rahman@example.com
191	Rakibul Islam	01811224455	Passport-DH56789	rakibul.islam@example.com
192	Shabnam Chowdhury	01922113344	DrivingLicense-DL223344	shabnam.chowdhury@example.com

Total Tenants

189

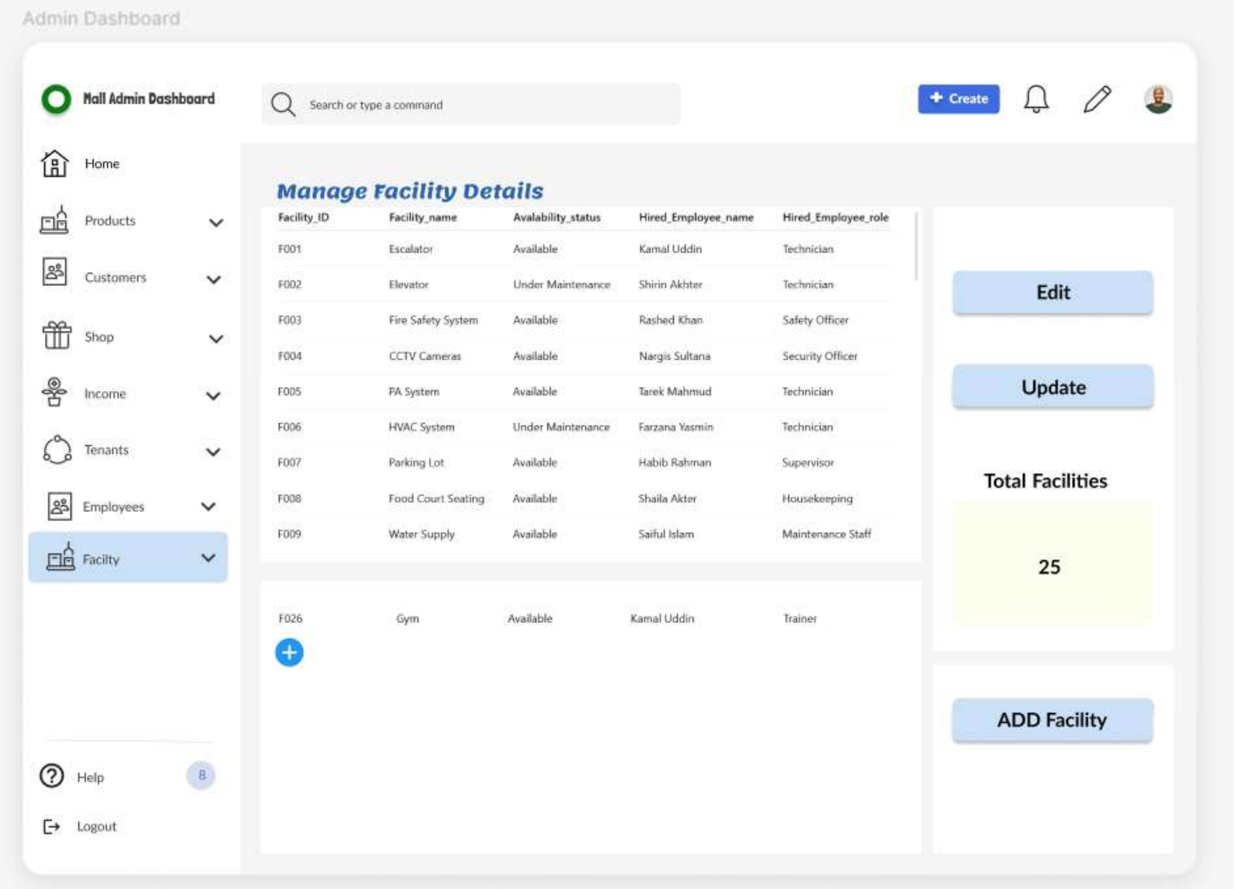


Manage Employees Details

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

Total Employees

3780

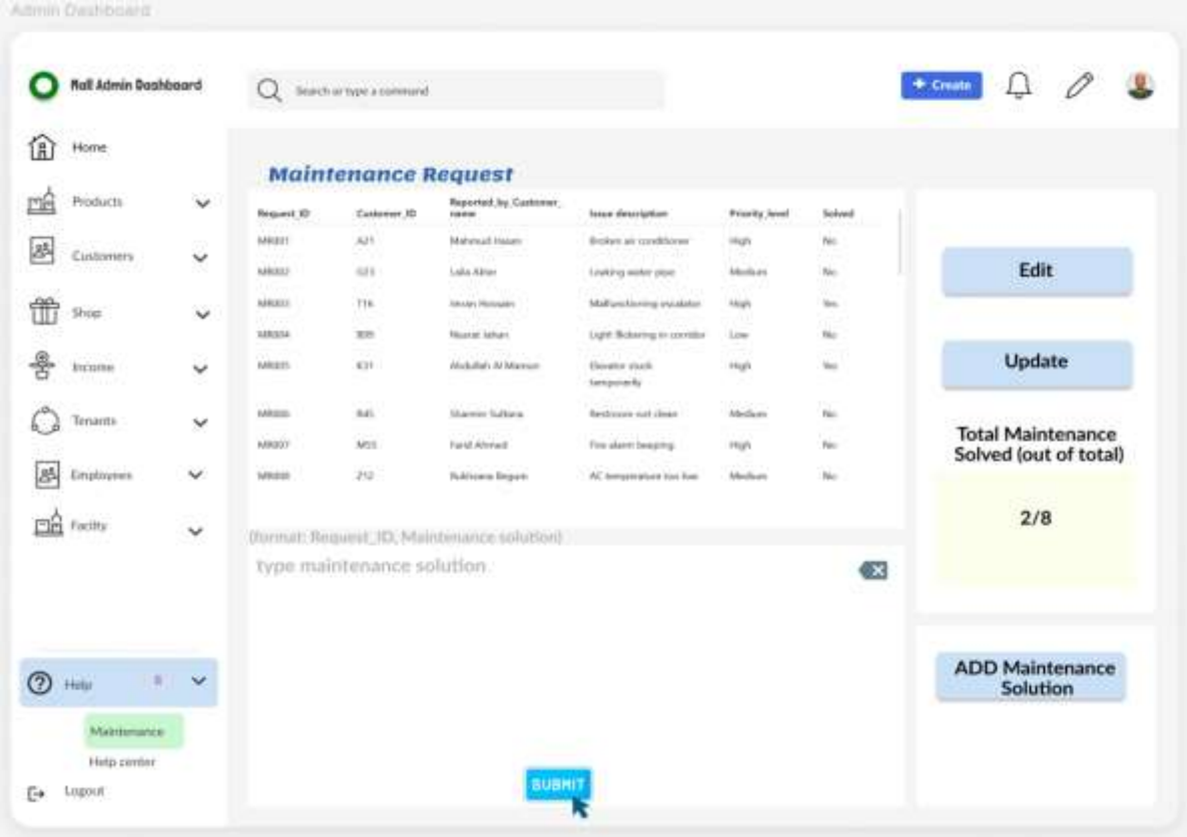


Manage Facility Details

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

Total Facilities

25



Maintenance Request

Request_ID	Customer_ID	Reported_by_Customer_name	Issue description	Priority_level	Solved
MR001	A21	Mahmud Hasan	Broken air conditioner	High	No
MR002	G23	Laila Akter	Leaking water pipe	Medium	No
MR003	T16	Imran Hossain	Malfunctioning escalator	High	Yes
MR004	309	Nusrat Jahan	Light flickering in corridor	Low	No
MR005	K31	Abdullah Al Mamun	Elevator stuck temporarily	High	Yes
MR006	B45	Sharmin Sultana	Restroom not clean	Medium	No
MR007	M55	Farid Ahmed	Fire alarm beeping	High	No
MR008	Z12	Rukhsana Begum	AC temperature too low	Medium	No

(format: Request_ID, Maintenance solution)

type maintenance solution

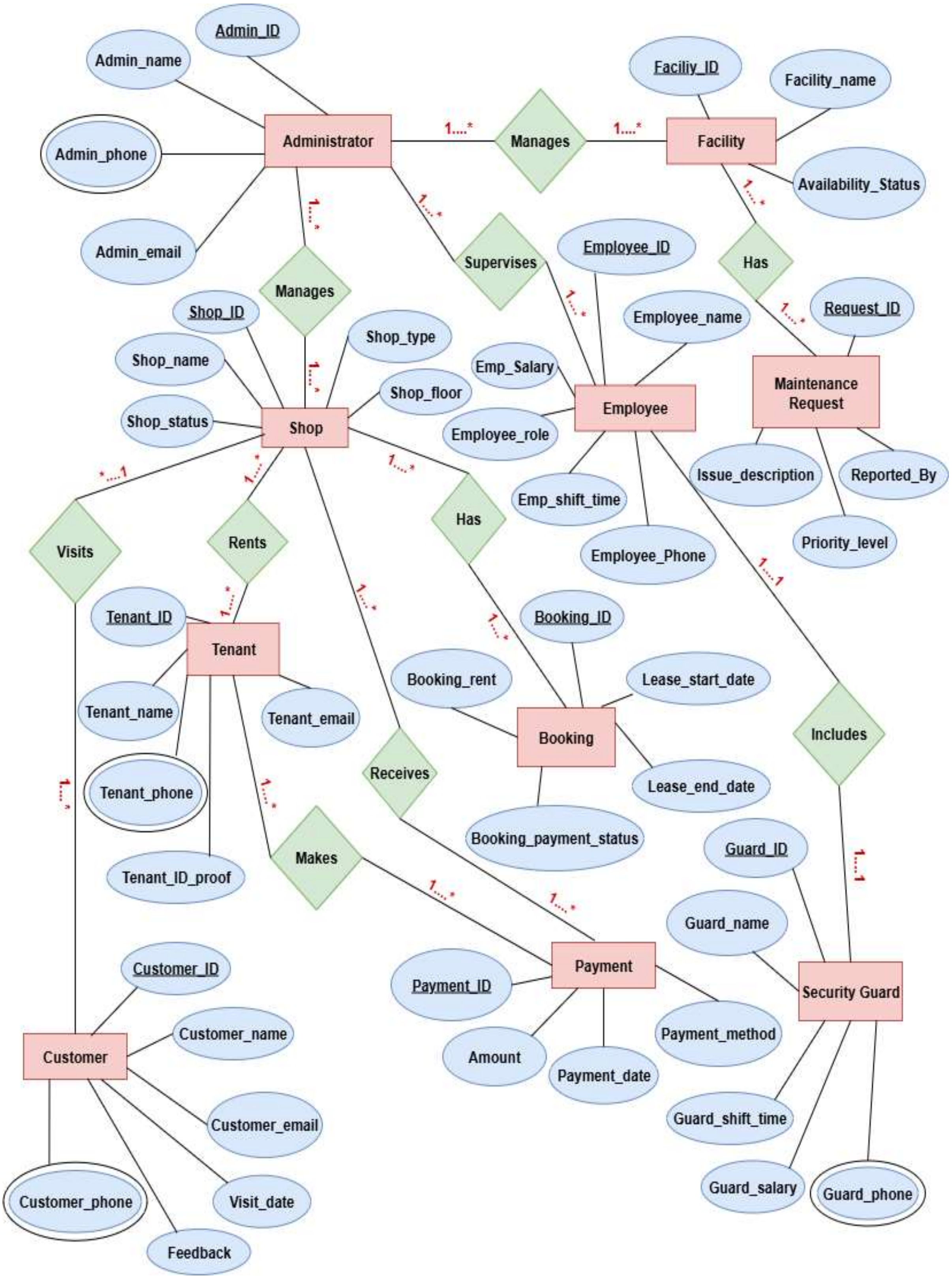
Total Maintenance Solved (out of total)

2/8

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

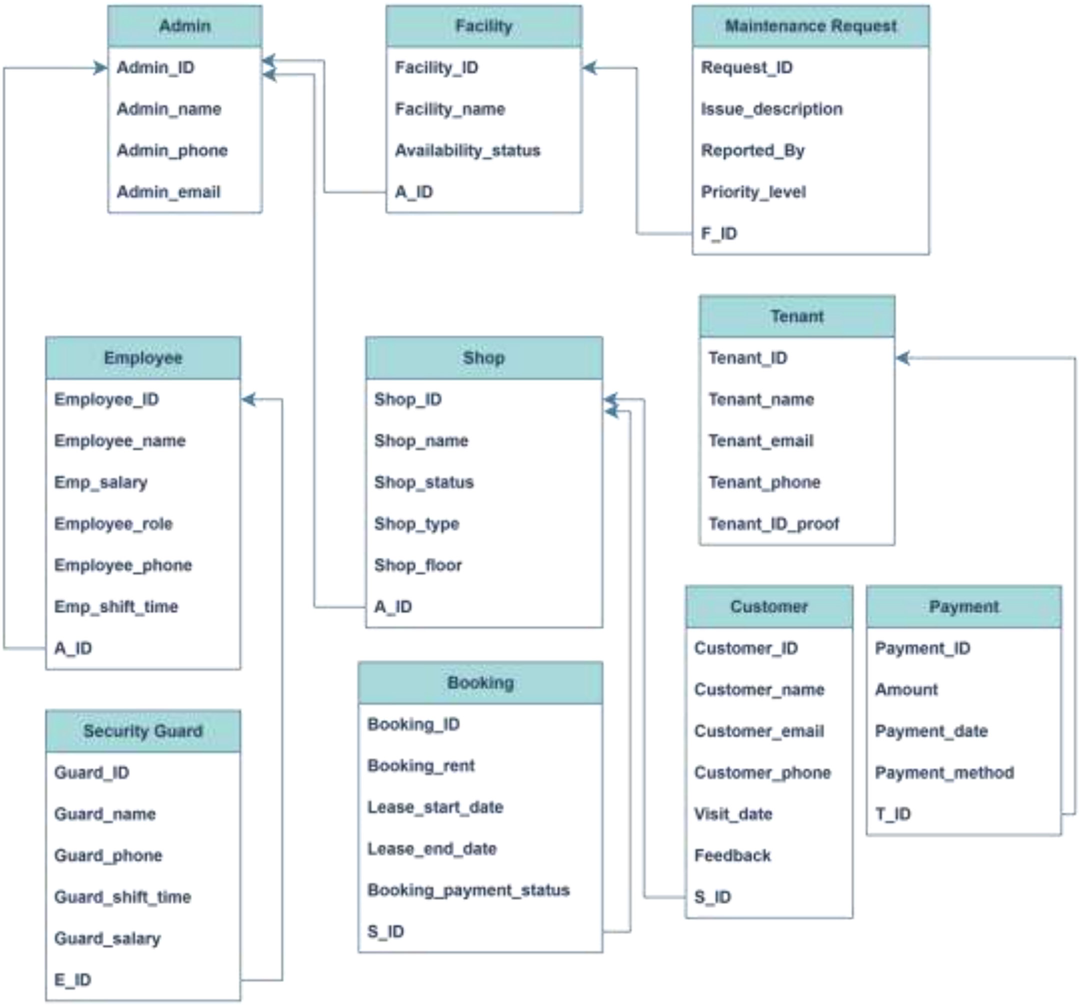
7. Payment_ID, Amount, Payment_date, Payment_method, **T_ID**

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**

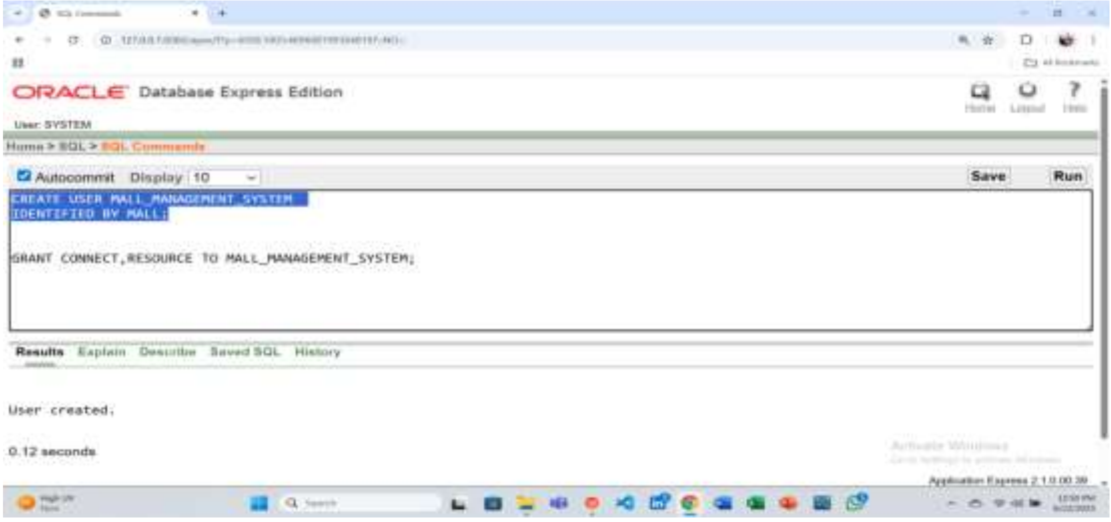
10. Guard_ID, Guard_name, Guard_phone, Guard_shift_time, Guard_salary, **E_ID**

7.Schema Diagram



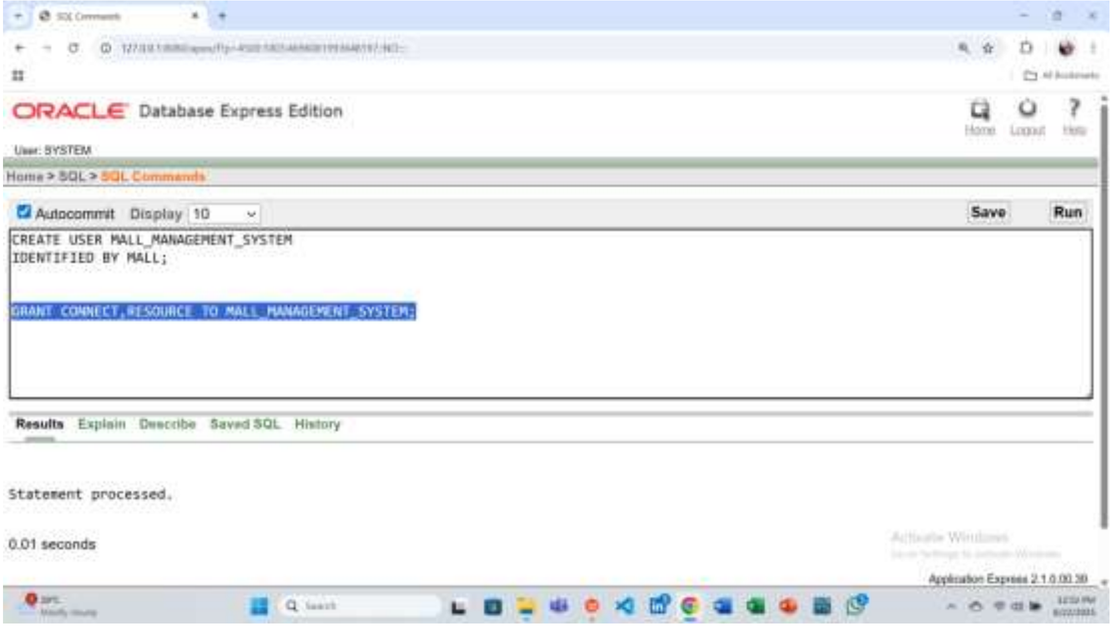
8.Table Creation Using SQL

Created A separate Database/Schema for Mall_management_system:


ORACLE Database Express Edition
User: SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
CREATE USER MALL_MANAGEMENT_SYSTEM
IDENTIFIED BY MALL;


GRANT CONNECT,RESOURCE TO MALL_MANAGEMENT_SYSTEM;
```

```
User created.

0.12 seconds
```


ORACLE Database Express Edition
User: SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History
Application Express 2.1.0.00.39

```
CREATE USER MALL_MANAGEMENT_SYSTEM
IDENTIFIED BY MALL;

GRANT CONNECT,RESOURCE TO MALL_MANAGEMENT_SYSTEM;
```

```
Statement processed.

0.01 seconds
```

Creating all Final Tables:

-- 1. Admin

CREATE TABLE Admin (

Admin_ID INT PRIMARY KEY,

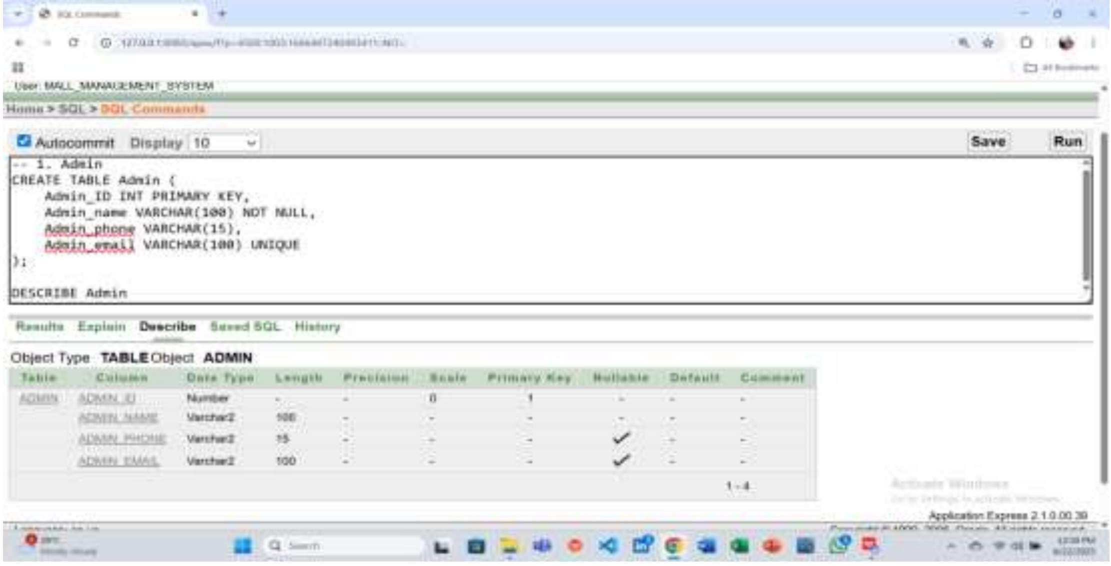
Admin_name VARCHAR(100) NOT NULL,

Admin_phone VARCHAR(15),

Admin_email VARCHAR(100) UNIQUE

);

DESCRIBE Admin


User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History
Application Express 2.1.0.00.39

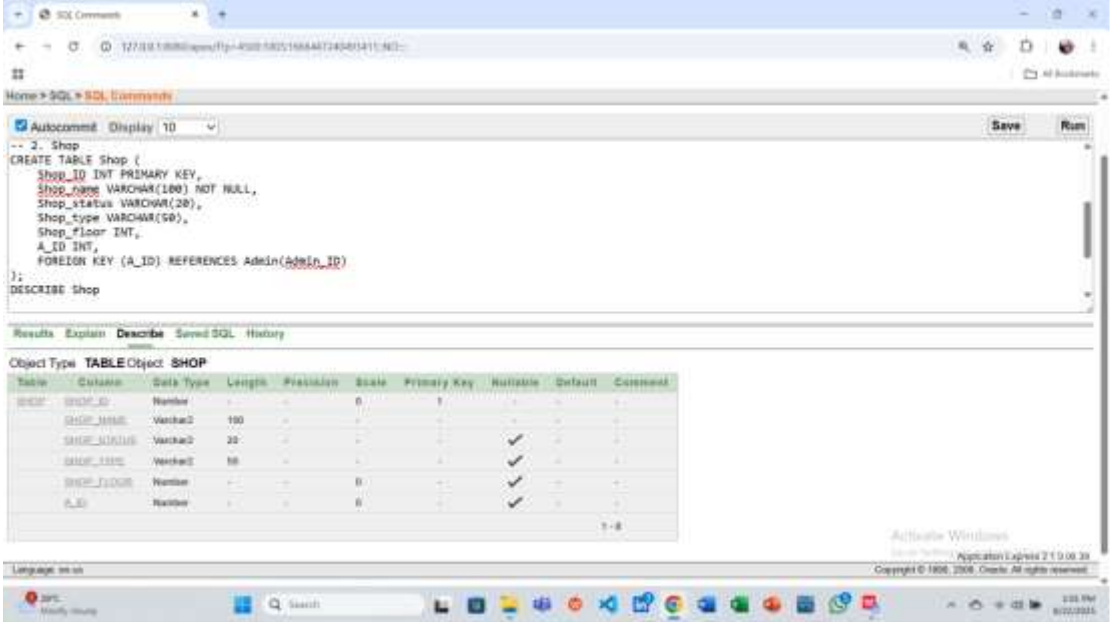
```
-- 1. Admin
CREATE TABLE Admin (
    Admin_ID INT PRIMARY KEY,
    Admin_name VARCHAR(100) NOT NULL,
    Admin_phone VARCHAR(15),
    Admin_email VARCHAR(100) UNIQUE
);

DESCRIBE Admin
```

Object Type **TABLE** Object **ADMIN**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ADMIN	ADMIN_ID	Number	-	-	0	1	-	-	-
	ADMIN_NAME	Varchar2	100	-	-	-	-	-	-
	ADMIN_PHONE	Varchar2	15	-	-	-	✓	-	-
	ADMIN_EMAIL	Varchar2	100	-	-	-	✓	-	-

1 - 4

```
-- 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
```



```
-- 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
```

Object Type **TABLE** Object **SHOP**

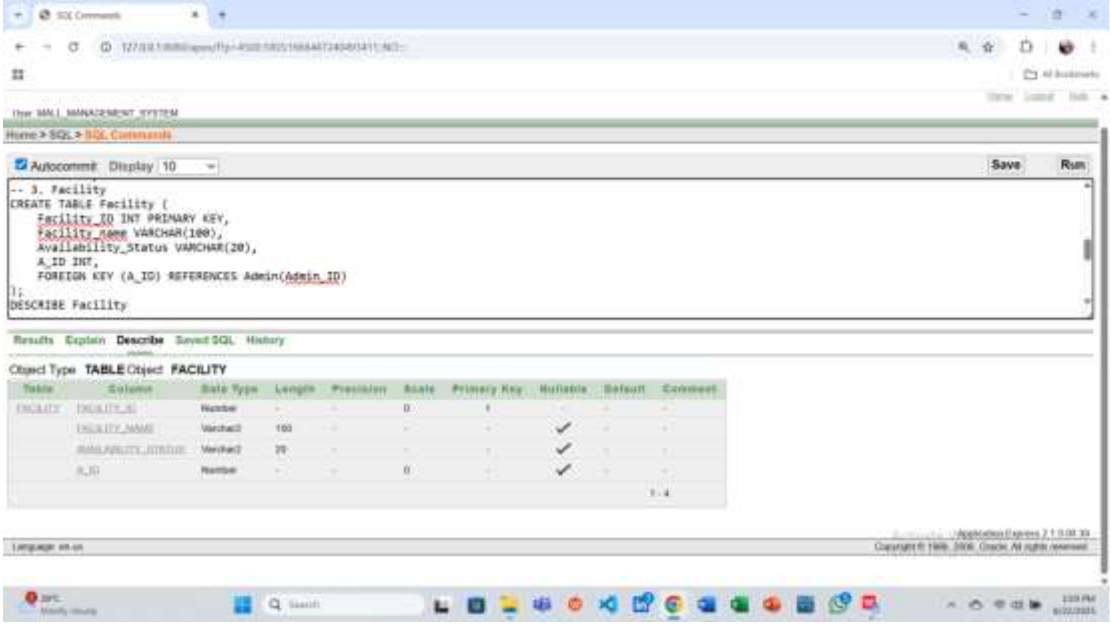
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SHOP	SHOP_ID	Number	-	-	0	1	-	-	-
	SHOP_NAME	Varchar2	100	-	-	-	-	-	-
	SHOP_STATUS	Varchar2	20	-	-	-	✓	-	-
	SHOP_TYPE	Varchar2	50	-	-	-	✓	-	-
	SHOP_FLOOR	Number	-	-	0	-	✓	-	-
	A_ID	Number	-	-	0	-	✓	-	-

1 - 6

```
-- 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



User MALL_MANAGEMENT_SYSTEM

Home > SQL > SQL Commands

☑ Autocommit Display 10

```
-- 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
```

Results Explain Describe Saved SQL History

Object Type TABLE Object FACILITY

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FACILITY	FACILITY_ID	Number	-	-	0	1	-	-	-
	FACILITY_NAME	Varchar2	100	-	-	-	✓	-	-
	AVAILABILITY_STATUS	Varchar2	20	-	-	-	✓	-	-
	A_ID	Number	-	-	0	-	✓	-	-

1 - 4

Language: en-us

-- 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),

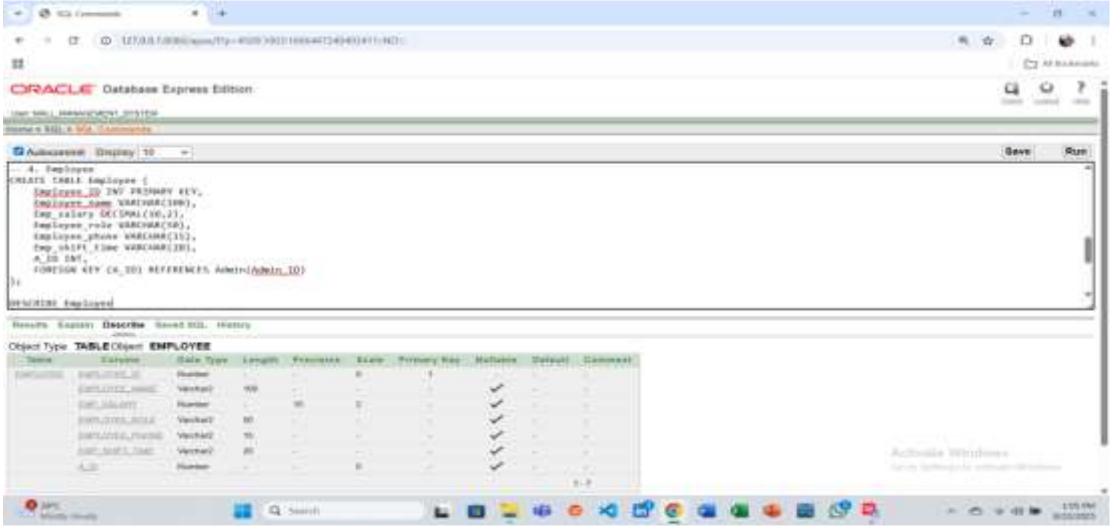
Emp_shift_time VARCHAR(20),

A_ID INT,

FOREIGN KEY (A_ID) REFERENCES Admin(Admin_ID)

);

DESCRIBE Employee



```
-- 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
```

Object Type TABLE Object EMPLOYEE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMPLOYEE_ID	Number	-	-	0	1	-	-	-
	EMPLOYEE_NAME	Varchar2	100	-	-	-	✓	-	-
	EMP_SALARY	Number	-	10	2	-	✓	-	-
	EMPLOYEE_ROLE	Varchar2	50	-	-	-	✓	-	-
	EMPLOYEE_PHONE	Varchar2	15	-	-	-	✓	-	-
	EMP_SHIFT_TIME	Varchar2	20	-	-	-	✓	-	-
	A_ID	Number	-	-	0	-	✓	-	-

1 - 7

-- 5. Request

CREATE TABLE Request (

Request_ID INT PRIMARY KEY,

Issue_description VARCHAR(255),

Reported_by VARCHAR(100),

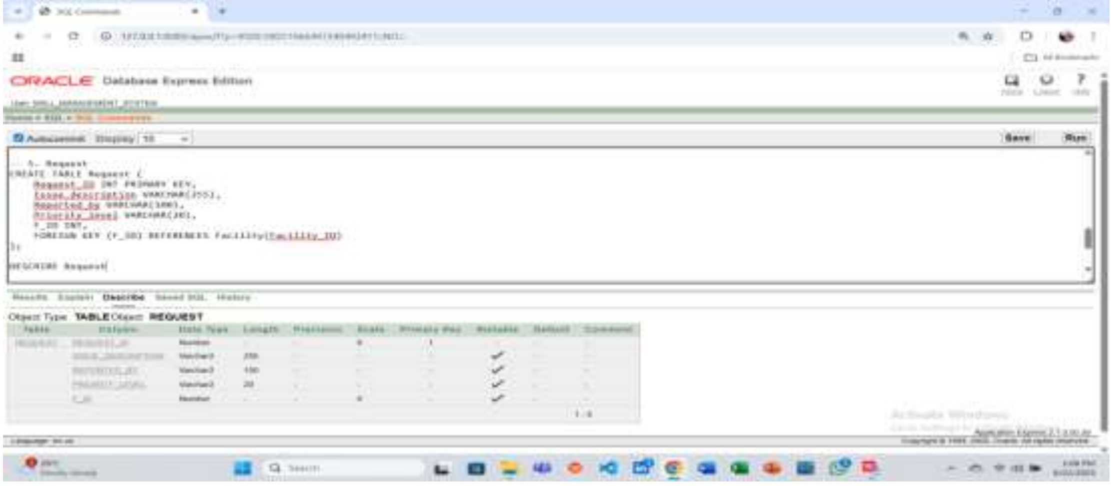
Priority_level VARCHAR(20),

F_ID INT,

FOREIGN KEY (F_ID) REFERENCES Facility(Facility_ID)

);

DESCRIBE Request



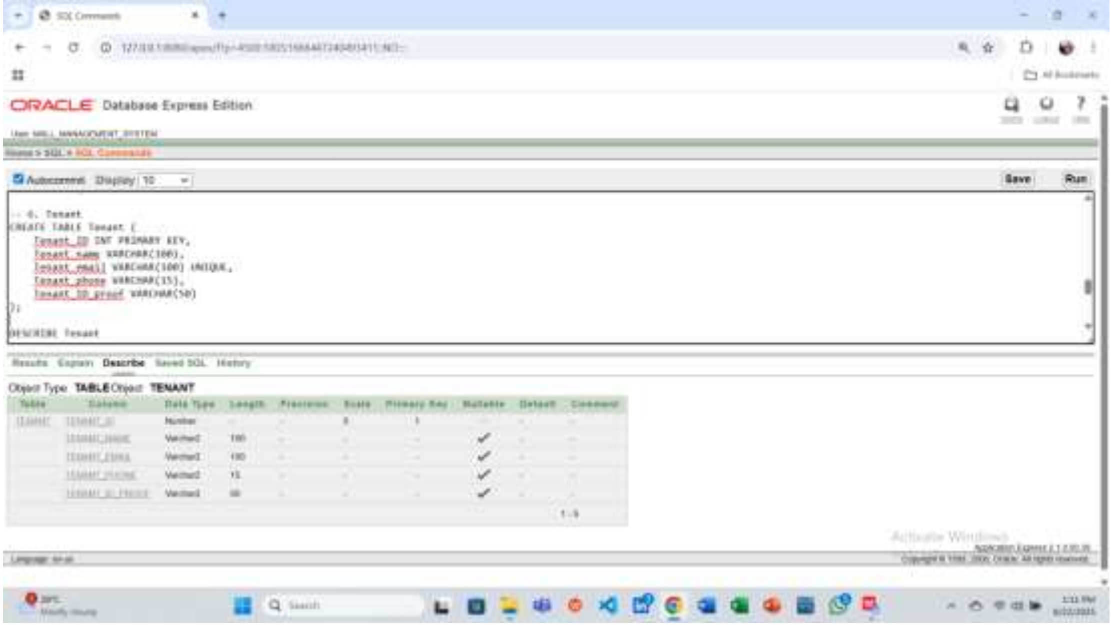
```
-- 5. Request
CREATE TABLE Request (
    Request_ID INT PRIMARY KEY,
    Issue_description VARCHAR(255),
    Reported_by VARCHAR(100),
    Priority_level VARCHAR(20),
    F_ID INT,
    FOREIGN KEY (F_ID) REFERENCES Facility(Facility_ID)
);

DESCRIBE Request
```

Object Type TABLE Object REQUEST

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REQUEST	REQUEST_ID	Number	-	-	0	1	-	-	-
	ISSUE_DESCRIPTION	Varchar2	255	-	-	-	✓	-	-
	REPORTED_BY	Varchar2	100	-	-	-	✓	-	-
	PRIORITY_LEVEL	Varchar2	20	-	-	-	✓	-	-
	F_ID	Number	-	-	0	-	✓	-	-

1 - 5

```
-- 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
```



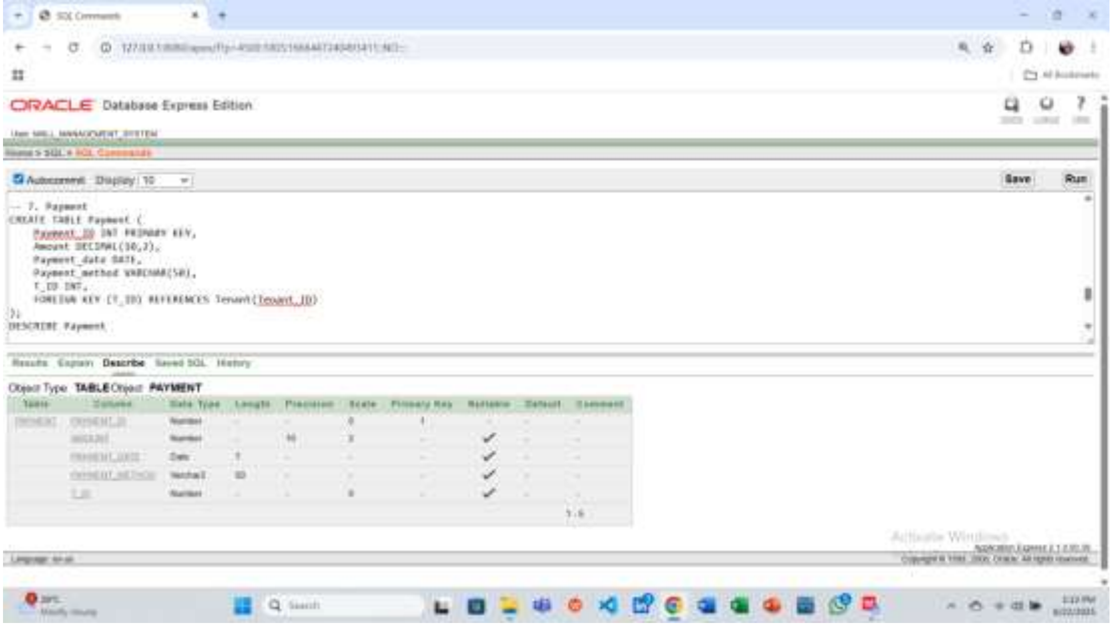
```
-- 6. Tenant
CREATE TABLE Tenant (
    Tenant_ID INT PRIMARY KEY,
    Tenant_name VARCHAR(100),
    Tenant_email VARCHAR(100) UNIQUE,
    Tenant_phone VARCHAR(15),
    Tenant_ID_proof VARCHAR(50)
);

DESCRIBE Tenant
```

Object Type TABLE Object TENANT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TENANT	TENANT_ID	Number	-	-	0	1	-	-	-
	TENANT_NAME	Varchar2	100	-	-	-	✓	-	-
	TENANT_EMAIL	Varchar2	100	-	-	-	✓	-	-
	TENANT_PHONE	Varchar2	15	-	-	-	✓	-	-
	TENANT_ID_PROOF	Varchar2	50	-	-	-	✓	-	-
								1 - 5	

```
-- 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



ORACLE Database Express Edition

User: MALL_MANAGEMENT_SYSTEM

Home > SQL > SQL Commands

```
-- 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
```

Results Explain Describe Saved SQL History

Object Type TABLE Object PAYMENT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT	PAYMENT_ID	Number	-	-	0	1	-	-	-
	AMOUNT	Number	-	10	2	-	✓	-	-
	PAYMENT_DATE	Date	7	-	-	-	✓	-	-
	PAYMENT_METHOD	Varchar2	50	-	-	-	✓	-	-
	T_ID	Number	-	-	0	-	✓	-	-
									1 - 5

-- 8. Booking

CREATE TABLE Booking (

Booking_ID INT PRIMARY KEY,

Booking_rent DECIMAL(10,2),

Lease_start_date DATE,

Lease_end_date DATE,

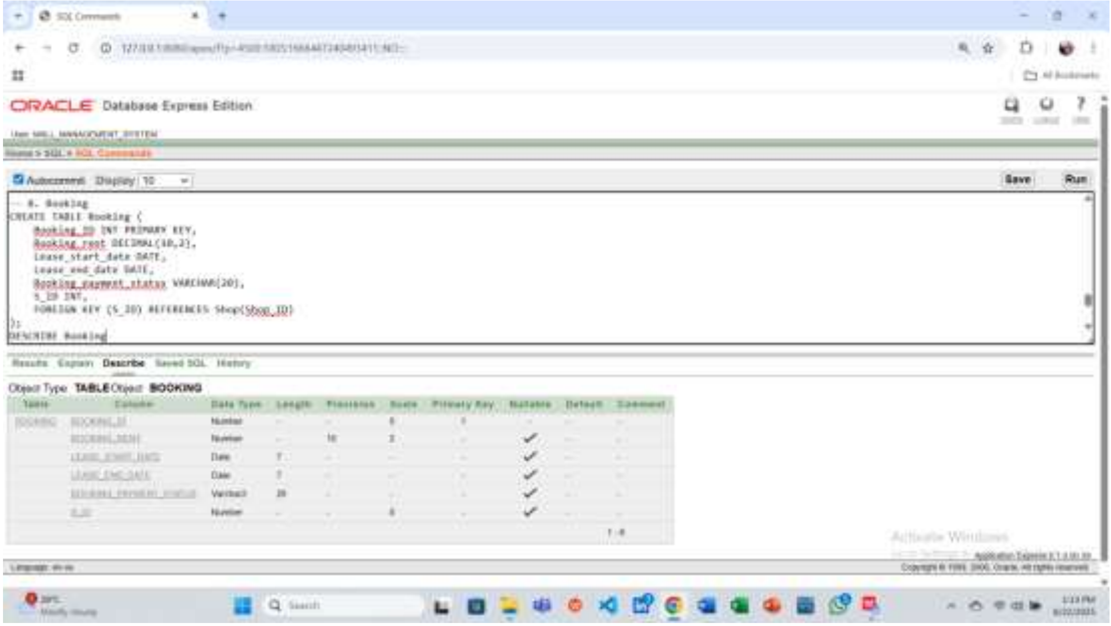
Booking_payment_status VARCHAR(20),

S_ID INT,

FOREIGN KEY (S_ID) REFERENCES Shop(Shop_ID)

);

DESCRIBE Booking



ORACLE Database Express Edition
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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
```

Object Type **TABLE** Object **BOOKING**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKING	BOOKING_ID	Number	-	-	0	1	-	-	-
	BOOKING_RENT	Number	-	10	2	-	✓	-	-
	LEASE_START_DATE	Date	7	-	-	-	✓	-	-
	LEASE_END_DATE	Date	7	-	-	-	✓	-	-
	BOOKING_PAYMENT_STATUS	Varchar2	20	-	-	-	✓	-	-
	S_ID	Number	-	-	0	-	✓	-	-

1 - 6

-- 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,

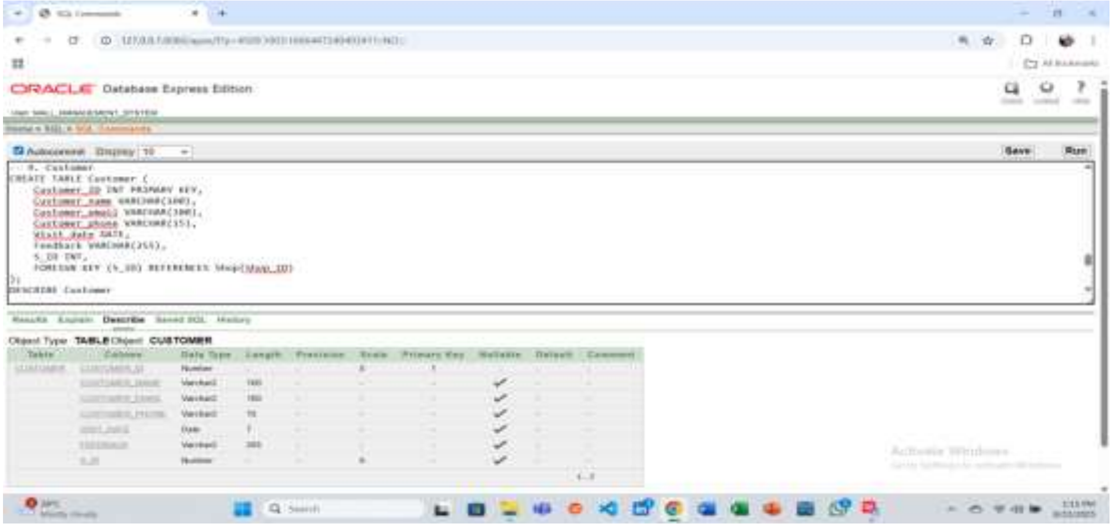
Feedback VARCHAR(255),

S_ID INT,

FOREIGN KEY (S_ID) REFERENCES Shop(Shop_ID)

);

DESCRIBE Customer



ORACLE Database Express Edition
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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
```

Object Type TABLE Object CUSTOMER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CUSTOMER_ID	Number	-	-	0	1	-	-	-
	CUSTOMER_NAME	Varchar2	100	-	-	-	✓	-	-
	CUSTOMER_EMAIL	Varchar2	100	-	-	-	✓	-	-
	CUSTOMER_PHONE	Varchar2	15	-	-	-	✓	-	-
	VISIT_DATE	Date	7	-	-	-	✓	-	-
	FEEDBACK	Varchar2	255	-	-	-	✓	-	-
	S_ID	Number	-	-	0	-	✓	-	-

1 - 7

-- 10. Guard

CREATE TABLE Guard (

Guard_ID INT PRIMARY KEY,

Guard_name VARCHAR(100),

Guard_phone VARCHAR(15),

Guard_shift_time VARCHAR(20),

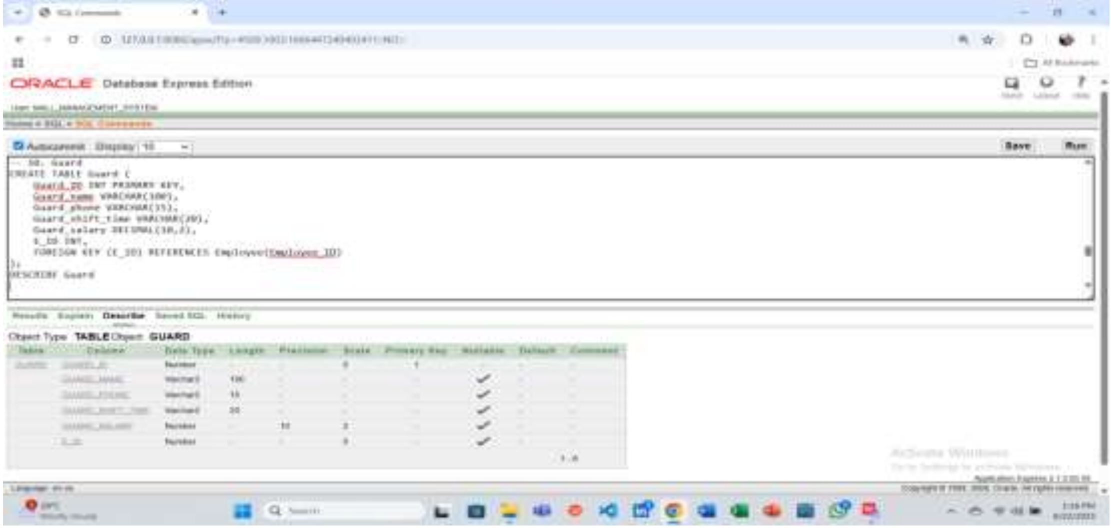
Guard_salary DECIMAL(10,2),

E_ID INT,

FOREIGN KEY (E_ID) REFERENCES Employee(Employee_ID)

);

DESCRIBE Guard



ORACLE Database Express Edition
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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
```

Object Type TABLE Object GUARD

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
GUARD	GUARD_ID	Number	-	-	0	1	-	-	-
	GUARD_NAME	Varchar2	100	-	-	-	✓	-	-
	GUARD_PHONE	Varchar2	15	-	-	-	✓	-	-
	GUARD_SHIFT_TIME	Varchar2	20	-	-	-	✓	-	-
	GUARD_SALARY	Number	-	10	2	-	✓	-	-
	E_ID	Number	-	-	0	-	✓	-	-

1 - 6

9.Data Insertion

-- Admin

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

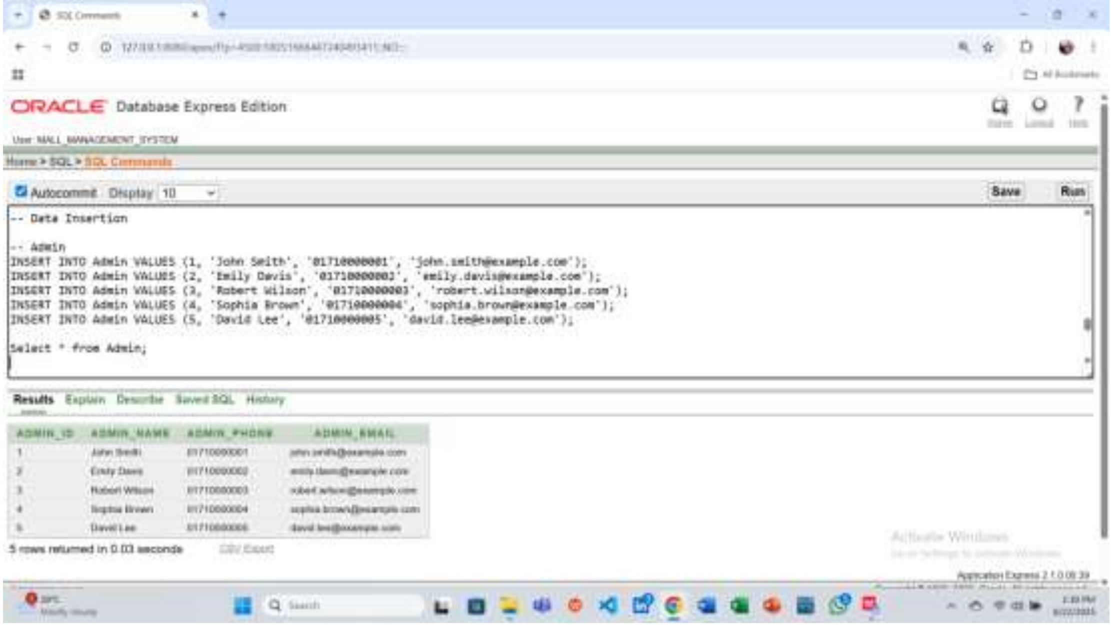
INSERT INTO Admin VALUES (2, 'Emily Davis', '01710000002', 'emily.davis@example.com');

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;


ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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;
```

ADMIN_ID	ADMIN_NAME	ADMIN_PHONE	ADMIN_EMAIL
1	John Smith	01710000001	john.smith@example.com
2	Emily Davis	01710000002	emily.davis@example.com
3	Robert Wilson	01710000003	robert.wilson@example.com
4	Sophia Brown	01710000004	sophia.brown@example.com
5	David Lee	01710000005	david.lee@example.com

5 rows returned in 0.03 seconds

-- Shop

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

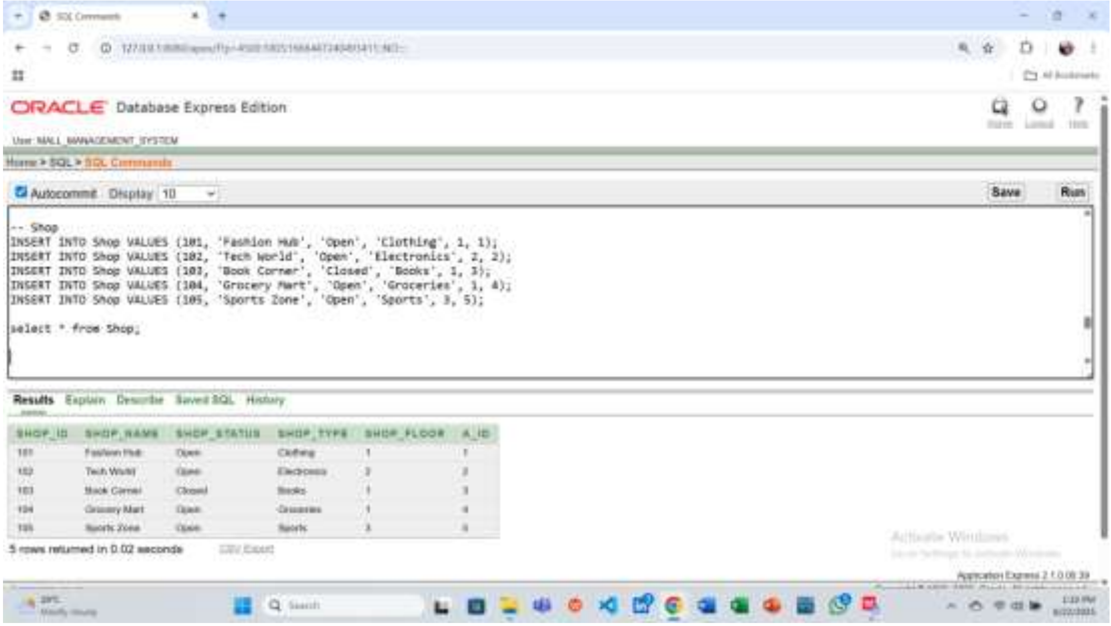
INSERT INTO Shop VALUES (102, 'Tech World', 'Open', 'Electronics', 2, 2);

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;



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit
Display 10
Save
Run
Results
Explain
Describe
Saved SQL
History
CSV Export

```
-- 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;
```

SHOP_ID	SHOP_NAME	SHOP_STATUS	SHOP_TYPE	SHOP_FLOOR	A_ID
101	Fashion Hub	Open	Clothing	1	1
102	Tech World	Open	Electronics	2	2
103	Book Corner	Closed	Books	1	3
104	Grocery Mart	Open	Groceries	1	4
105	Sports Zone	Open	Sports	3	5

5 rows returned in 0.02 seconds

-- Facility

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

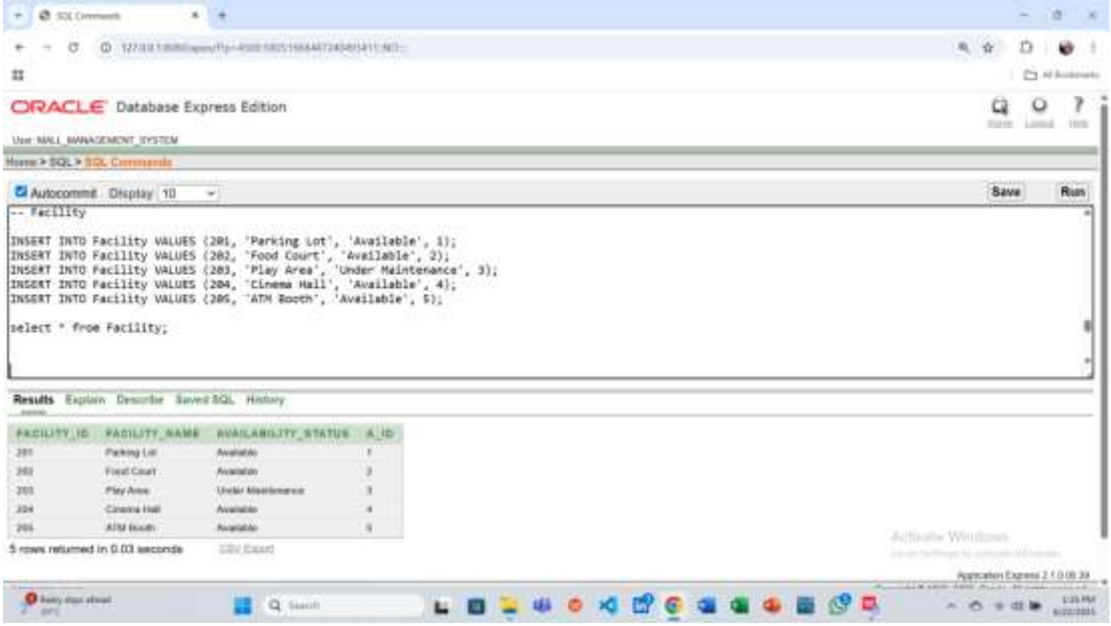
INSERT INTO Facility VALUES (202, 'Food Court', 'Available', 2);

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;



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit
Display 10
Save
Run
Results
Explain
Describe
Saved SQL
History
CSV Export

```
-- 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;
```

FACILITY_ID	FACILITY_NAME	AVAILABILITY_STATUS	A_ID
201	Parking Lot	Available	1
202	Food Court	Available	2
203	Play Area	Under Maintenance	3
204	Cinema Hall	Available	4
205	ATM Booth	Available	5

5 rows returned in 0.03 seconds

-- Employee

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

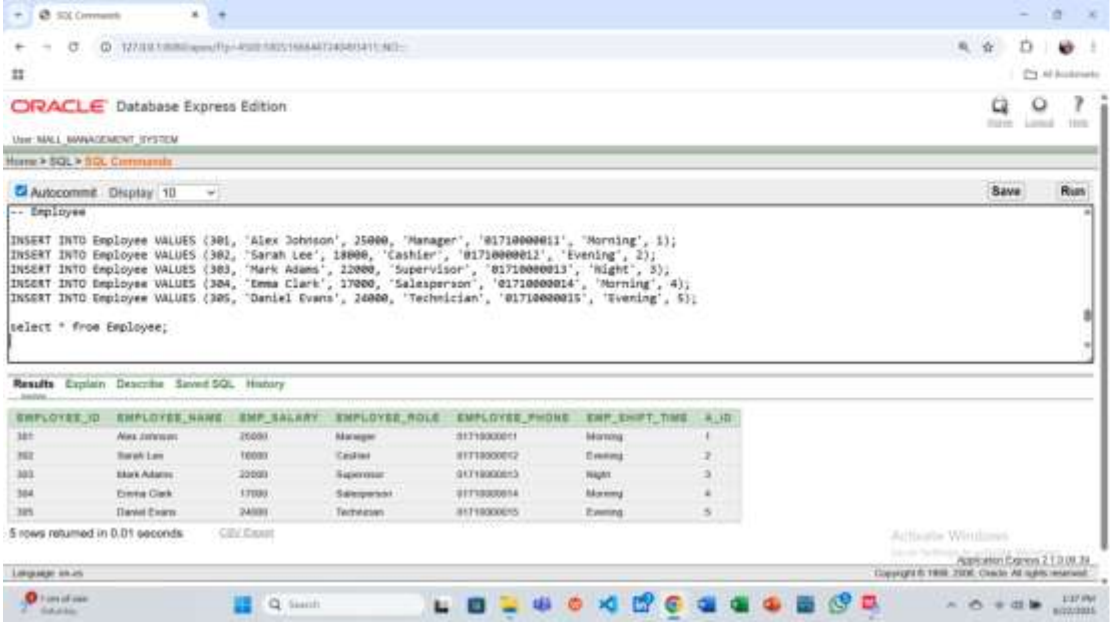
INSERT INTO Employee VALUES (302, 'Sarah Lee', 18000, 'Cashier', '01710000012', 'Evening', 2);

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;


ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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;
```

EMPLOYEE_ID	EMPLOYEE_NAME	EMP_SALARY	EMPLOYEE_ROLE	EMPLOYEE_PHONE	EMP_SHIFT_TIME	A_ID
301	Alex Johnson	25000	Manager	01710000011	Morning	1
302	Sarah Lee	18000	Cashier	01710000012	Evening	2
303	Mark Adams	22000	Supervisor	01710000013	Night	3
304	Emma Clark	17000	Salesperson	01710000014	Morning	4
305	Daniel Evans	24000	Technician	01710000015	Evening	5

5 rows returned in 0.01 seconds

-- Request

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

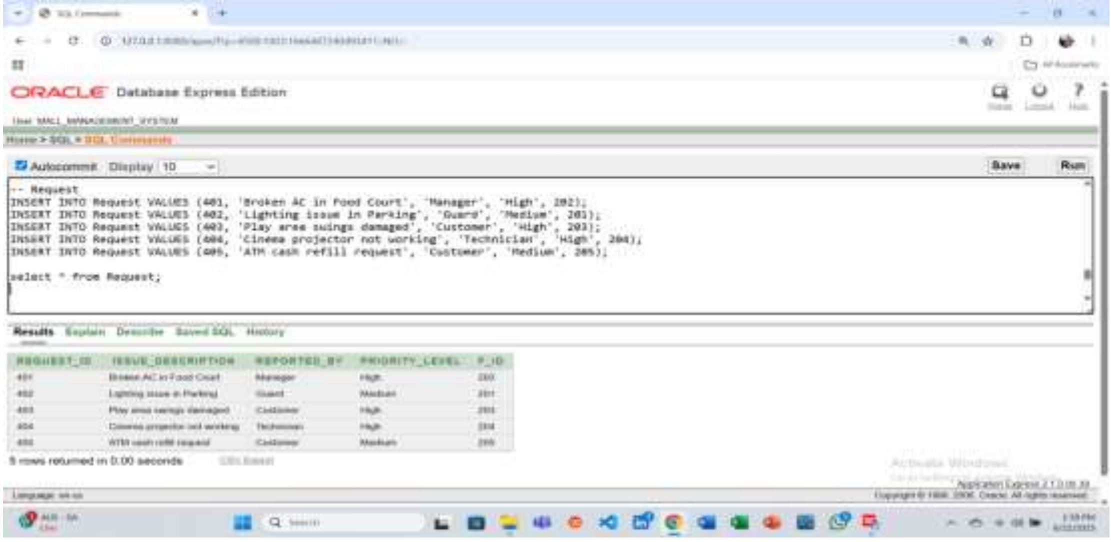
INSERT INTO Request VALUES (402, 'Lighting issue in Parking', 'Guard', 'Medium', 201);

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;



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
Results
Explain
Describe
Saved SQL
History

```
-- 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;
```

REQUEST_ID	ISSUE_DESCRIPTION	REPORTED_BY	PRIORITY_LEVEL	P_ID
401	Broken AC in Food Court	Manager	High	202
402	Lighting issue in Parking	Guard	Medium	201
403	Play area swings damaged	Customer	High	203
404	Cinema projector not working	Technician	High	204
405	ATM cash refill request	Customer	Medium	205

5 rows returned in 0.00 seconds

-- Tenant

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

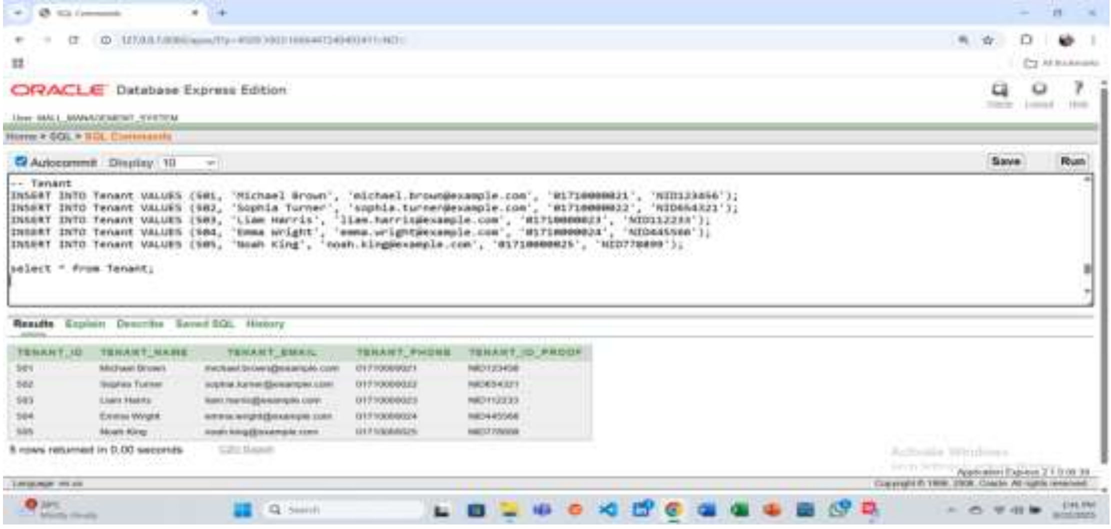
INSERT INTO Tenant VALUES (502, 'Sophia Turner', 'sophia.turner@example.com', '01710000022', 'NID654321');

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;



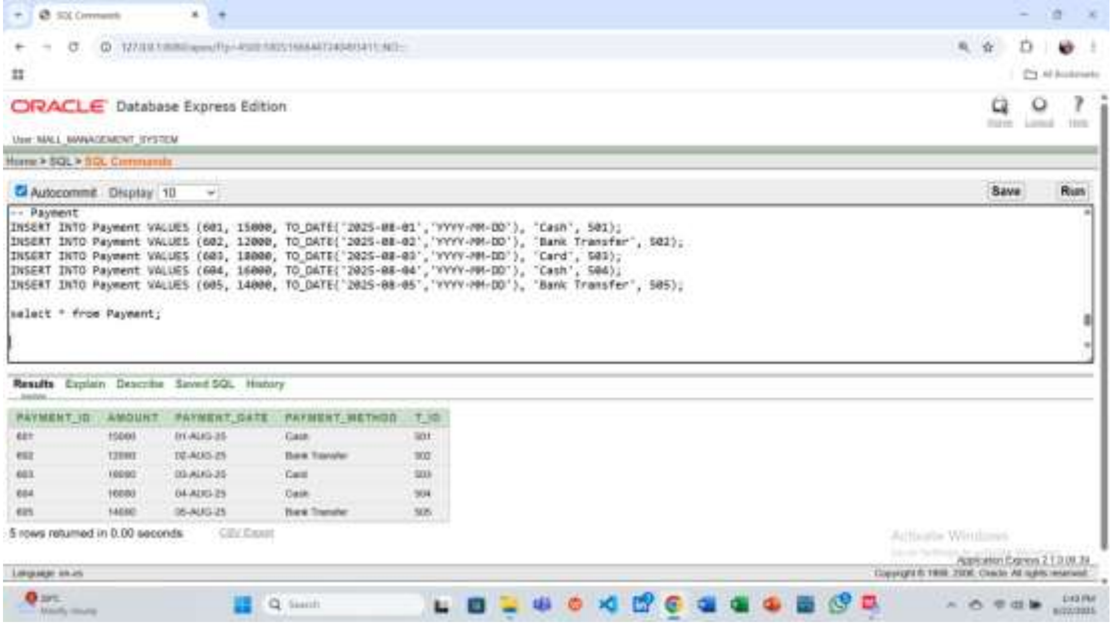
ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
Results
Explain
Describe
Saved SQL
History

```
-- 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;
```

TENANT_ID	TENANT_NAME	TENANT_EMAIL	TENANT_PHONE	TENANT_ID_PROOF
501	Michael Brown	michael.brown@example.com	01710000021	NID123456
502	Sophia Turner	sophia.turner@example.com	01710000022	NID654321
503	Liam Harris	liam.harris@example.com	01710000023	NID112233
504	Emma Wright	emma.wright@example.com	01710000024	NID445566
505	Noah King	noah.king@example.com	01710000025	NID778899

5 rows returned in 0.00 seconds

```
-- 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;
```



```
-- 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;
```

PAYMENT_ID	AMOUNT	PAYMENT_DATE	PAYMENT_METHOD	T_ID
601	15000	01-AUG-25	Cash	501
602	12000	02-AUG-25	Bank Transfer	502
603	18000	03-AUG-25	Card	503
604	16000	04-AUG-25	Cash	504
605	14000	05-AUG-25	Bank Transfer	505

5 rows returned in 0.00 seconds

```
-- 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);

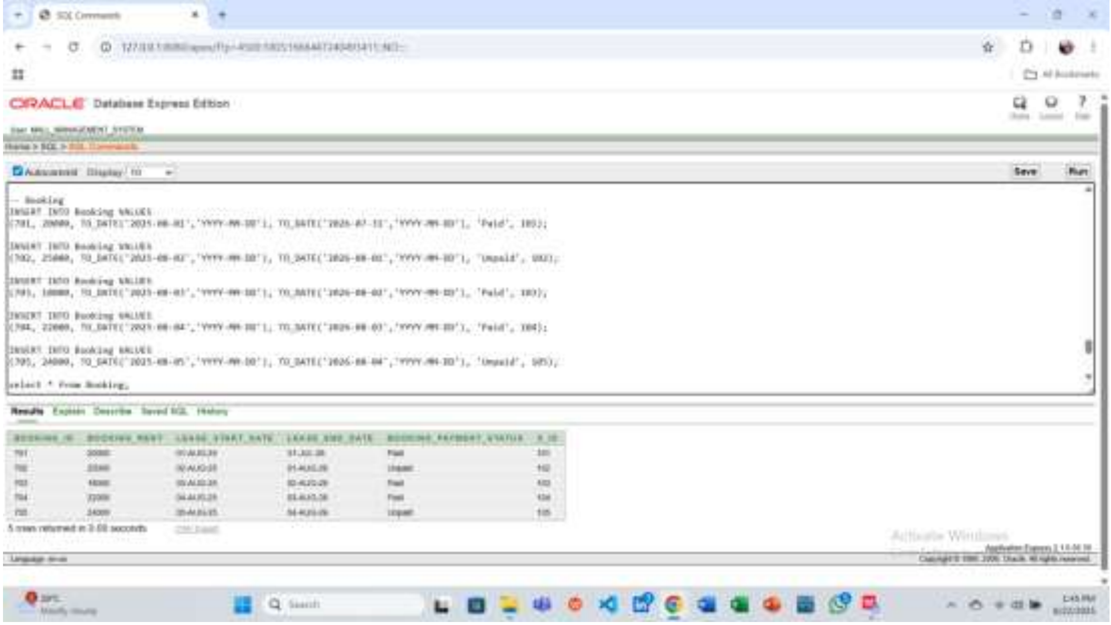
INSERT INTO Booking VALUES

(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;



ORACLE Database Express Edition
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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;
```

BOOKING_ID	BOOKING_RENT	LEASE_START_DATE	LEASE_END_DATE	BOOKING_PAYMENT_STATUS	S_ID
701	20000	01-AUG-25	31-JUL-26	Paid	101
702	25000	02-AUG-25	01-AUG-26	Unpaid	102
703	18000	03-AUG-25	02-AUG-26	Paid	103
704	22000	04-AUG-25	03-AUG-26	Paid	104
705	24000	05-AUG-25	04-AUG-26	Unpaid	105

5 rows returned in 0.00 seconds CSV Export

-- Customer

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

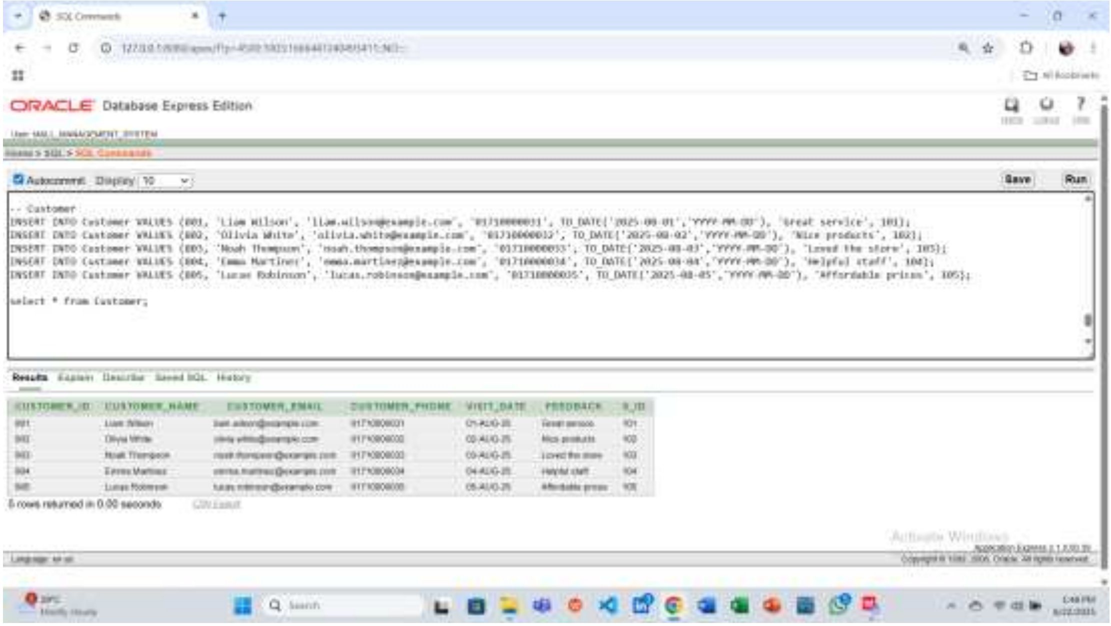
INSERT INTO Customer VALUES (802, 'Olivia White', 'olivia.white@example.com', '01710000032', TO_DATE('2025-08-02','YYYY-MM-DD'), 'Nice products', 102);

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;


ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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;
```

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_EMAIL	CUSTOMER_PHONE	VISIT_DATE	FEEDBACK	S_ID
801	Liam Wilson	liam.wilson@example.com	01710000031	01-AUG-25	Great service	101
802	Olivia White	olivia.white@example.com	01710000032	02-AUG-25	Nice products	102
803	Noah Thompson	noah.thompson@example.com	01710000033	03-AUG-25	Loved the store	103
804	Emma Martinez	emma.martinez@example.com	01710000034	04-AUG-25	Helpful staff	104
805	Lucas Robinson	lucas.robinson@example.com	01710000035	05-AUG-25	Affordable prices	105

5 rows returned in 0.00 seconds CSV Export

-- Guard

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

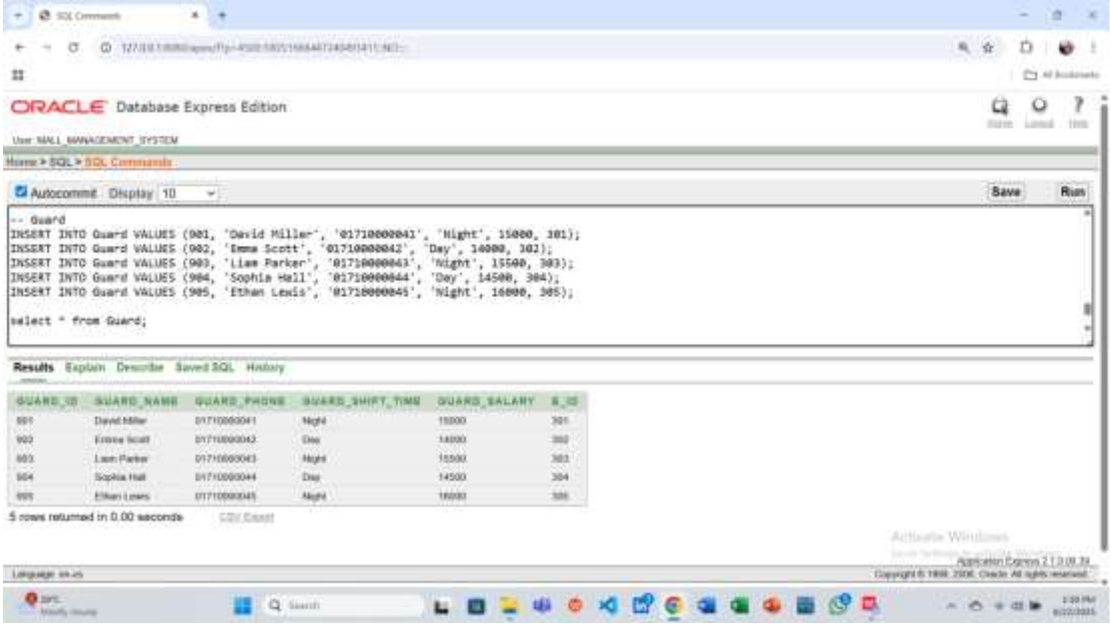
INSERT INTO Guard VALUES (902, 'Emma Scott', '01710000042', 'Day', 14000, 302);

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;


ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- 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;
```

GUARD_ID	GUARD_NAME	GUARD_PHONE	GUARD_SHIFT_TIME	GUARD_SALARY	S_ID
901	David Miller	01710000041	Night	15000	301
902	Emma Scott	01710000042	Day	14000	302
903	Liam Parker	01710000043	Night	15500	303
904	Sophia Hall	01710000044	Day	14500	304
905	Ethan Lewis	01710000045	Night	16000	305

5 rows returned in 0.00 seconds CSV Export

10.Query Writing Using PL/SQL

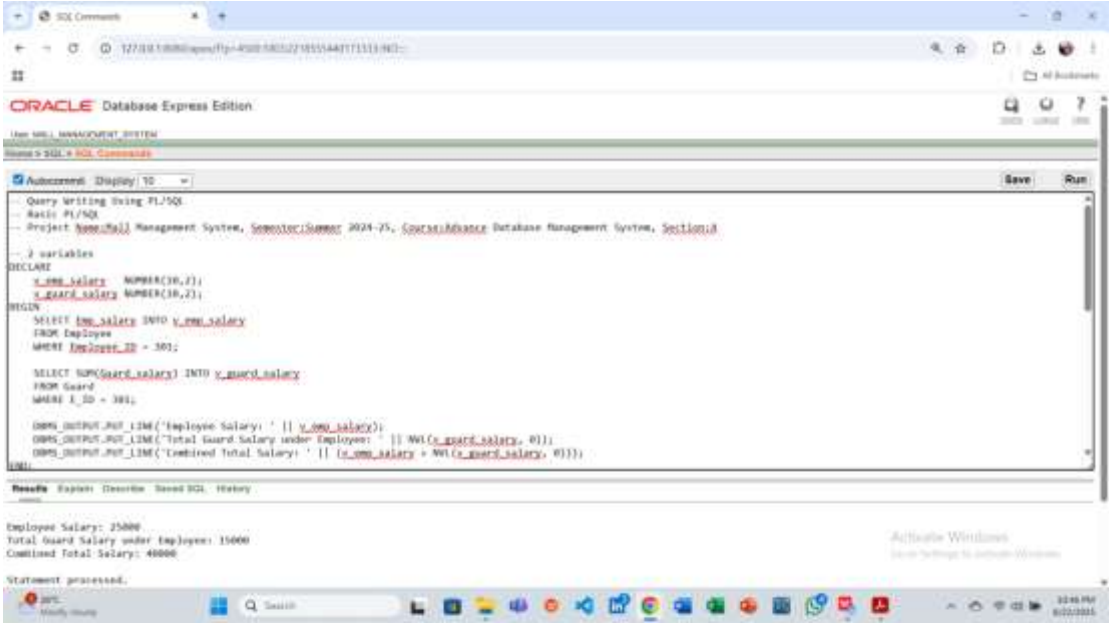
11. Basic PL/SQL

- Using 2 variables

Question: Calculate total salary of an employee and guards reporting to that employee (Employee_ID = 301).

Answer:

```
DECLARE

    v_emp_salary   NUMBER(10,2);

    v_guard_salary NUMBER(10,2);

BEGIN

    SELECT Emp_salary INTO v_emp_salary

    FROM Employee

    WHERE Employee_ID = 301;



    SELECT SUM(Guard_salary) INTO v_guard_salary

    FROM Guard

    WHERE E_ID = 301;



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

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

    DBMS_OUTPUT.PUT_LINE('Combined Total Salary: ' || (v_emp_salary + NVL(v_guard_salary, 0)));

END;

/
```



```
-- Query Writing Using PL/SQL
-- Basic PL/SQL
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A

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

    SELECT SUM(Guard_salary) INTO v_guard_salary
    FROM Guard
    WHERE E_ID = 301;

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

```
Employee Salary: 25000
Total Guard Salary under Employee: 15000
Combined Total Salary: 40000

Statement processed.
```

-Using 2 Operators

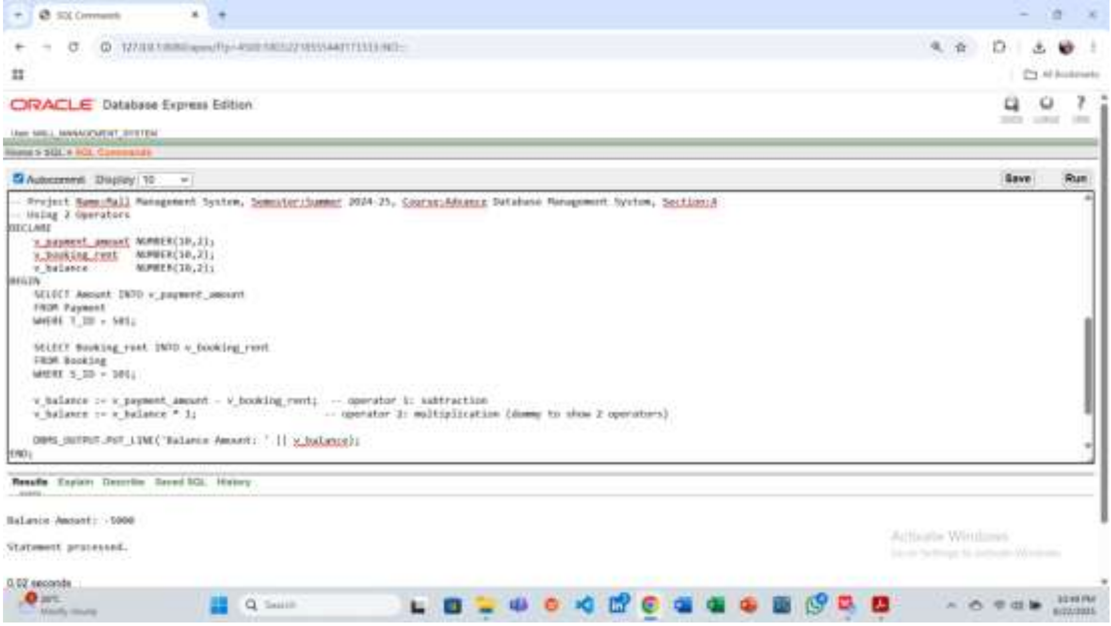
Question: Calculate if total payments made by Tenant_ID = 501 minus Booking rent for Shop_ID = 101 is positive or negative.

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 1: subtraction

    v_balance := v_balance * 1;

    DBMS_OUTPUT.PUT_LINE('Balance Amount: ' || v_balance);
```

END;

/



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Using 2 Operators
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 = 301;

    v_balance := v_payment_amount - v_booking_rent;  -- operator 1: subtraction
    v_balance := v_balance * 1;                      -- operator 2: multiplication (dummy to show 2 operators)

    DBMS_OUTPUT.PUT_LINE('Balance Amount: ' || v_balance);
END;
```

```
Balance Amount: -5000

Statement processed.
```

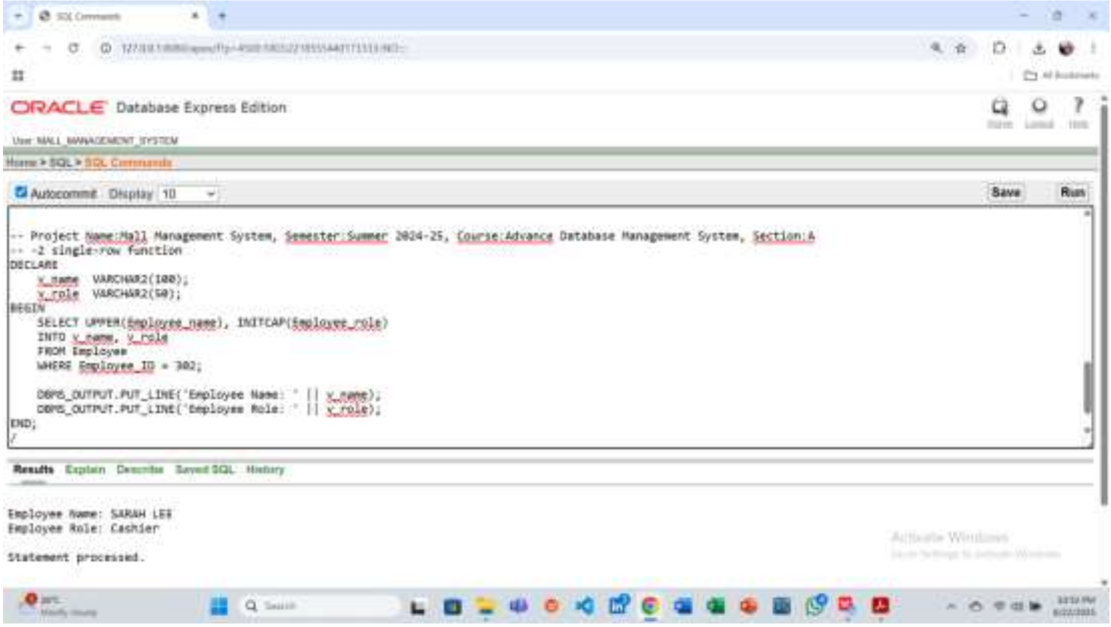
0.02 seconds

-Using 2 single-row function

Question: Display employee name in uppercase and employee role with first letter capital using Employee_ID = 302

Answer:

```
DECLARE

   v_name  VARCHAR2(100);

   v_role  VARCHAR2(50);

BEGIN

   SELECT UPPER(Employee_name), INITCAP(Employee_role)

   INTO v_name, v_role

   FROM Employee

   WHERE Employee_ID = 302;

   DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_name);

   DBMS_OUTPUT.PUT_LINE('Employee Role: ' || v_role);

END;

/
```



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- -2 single-row function
DECLARE
    v_name  VARCHAR2(100);
    v_role  VARCHAR2(50);
BEGIN
    SELECT UPPER(Employee_name), INITCAP(Employee_role)
    INTO v_name, v_role
    FROM Employee
    WHERE Employee_ID = 302;

    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_name);
    DBMS_OUTPUT.PUT_LINE('Employee Role: ' || v_role);
END;
/
```

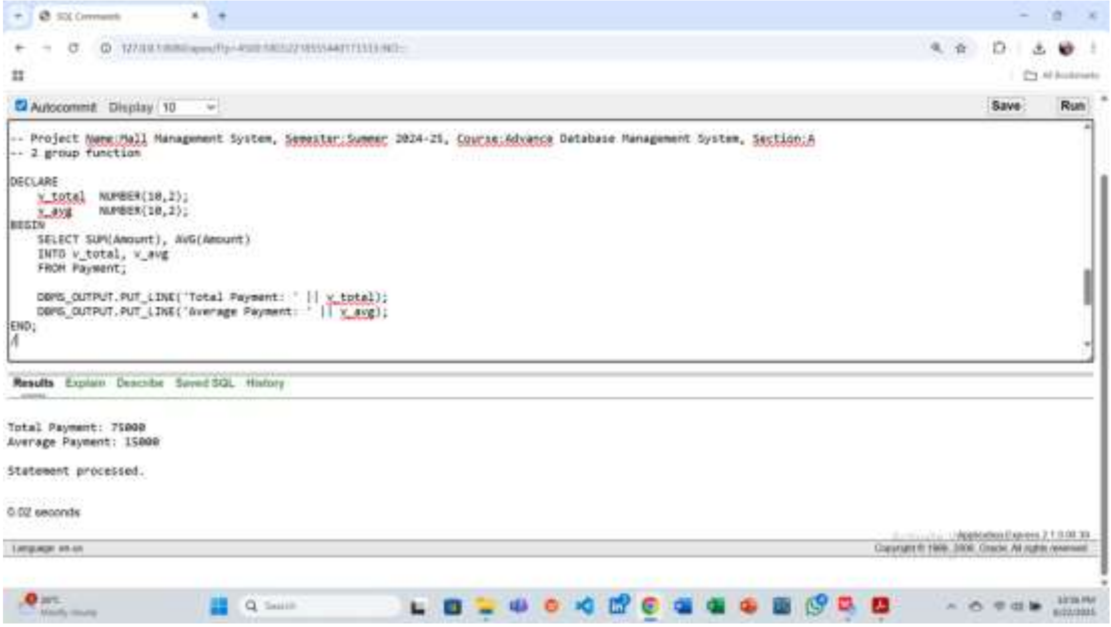
```
Employee Name: SARAH LEE
Employee Role: Cashier

Statement processed.
```

- Using 2 group function

Question: Find total and average Payment amounts.

Answer:

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


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



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 group function

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

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

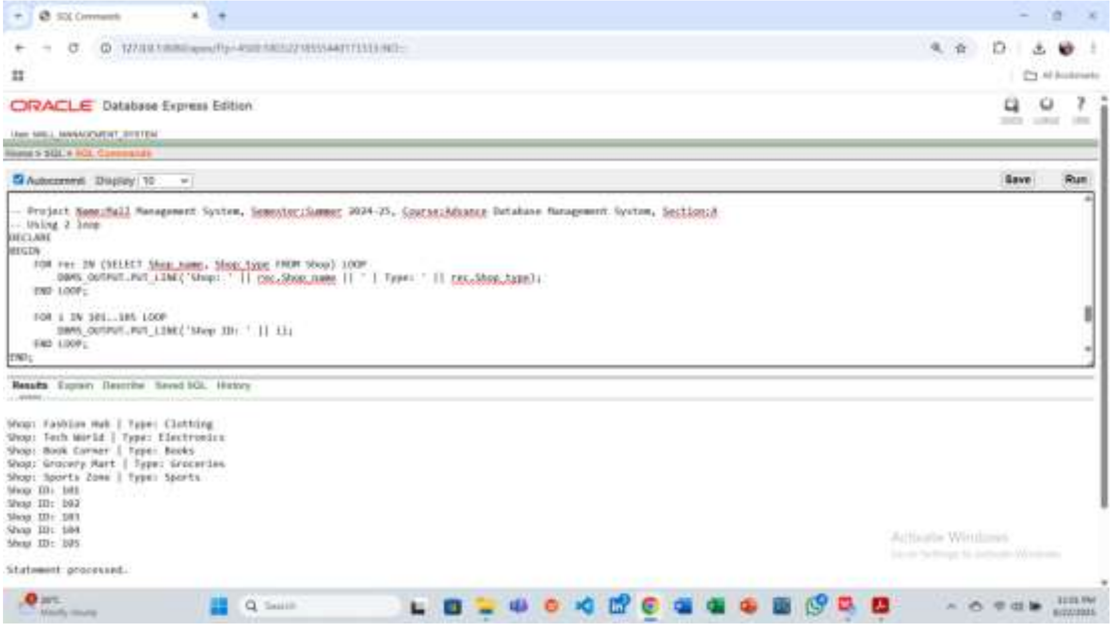
```
Total Payment: 75000
Average Payment: 15000

Statement processed.

0.02 seconds
```

-Using 2 loop

Question: Display all Shop names and their types

Answer:

```
DECLARE

BEGIN

      FOR rec IN (SELECT Shop_name, Shop_type FROM Shop) LOOP

       DBMS_OUTPUT.PUT_LINE('Shop: ' || rec.Shop_name || ' | Type: ' || rec.Shop_type);

   END LOOP;

   FOR i IN 101..105 LOOP

       DBMS_OUTPUT.PUT_LINE('Shop ID: ' || i);

   END LOOP;

END;

/
```



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Using 2 loop
DECLARE
BEGIN
   FOR rec IN (SELECT Shop_name, Shop_type FROM Shop) LOOP
      DBMS_OUTPUT.PUT_LINE('Shop: ' || rec.Shop_name || ' | Type: ' || rec.Shop_type);
   END LOOP;

   FOR i IN 101..105 LOOP
      DBMS_OUTPUT.PUT_LINE('Shop ID: ' || i);
   END LOOP;
END;
```

```
Shop: Fashion Hub | Type: Clothing
Shop: Tech World | Type: Electronics
Shop: Book Corner | Type: Books
Shop: Grocery Mart | Type: Groceries
Shop: Sports Zone | Type: Sports
Shop ID: 101
Shop ID: 102
Shop ID: 103
Shop ID: 104
Shop ID: 105

Statement processed.
```

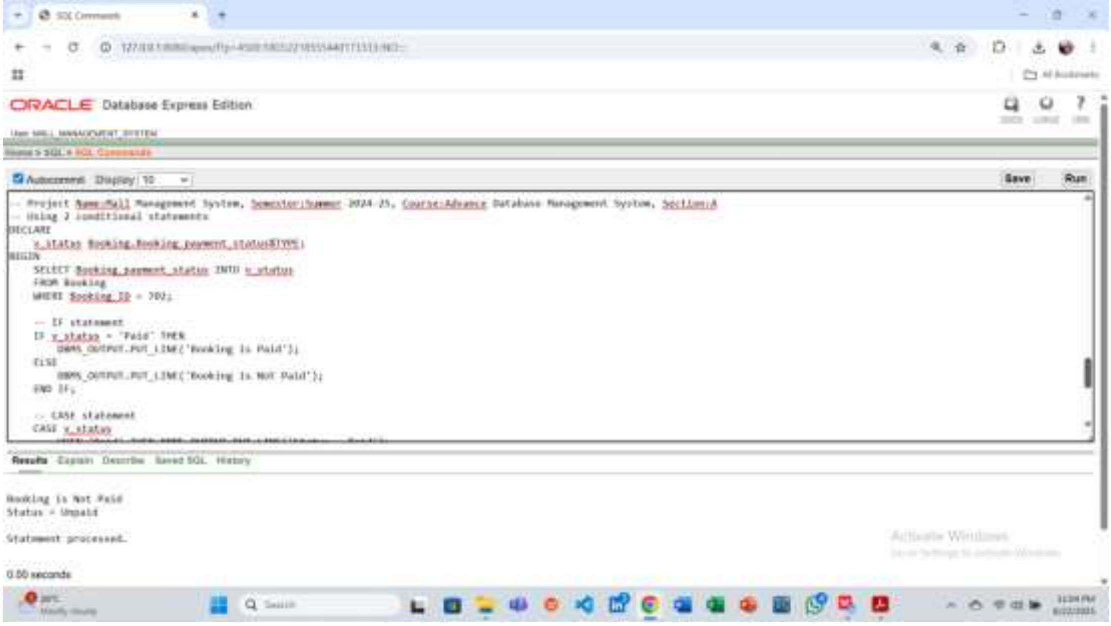
-2 conditional statements

Question: Check Booking payment status and display message

Answer:

```
DECLARE

   v_status Booking.Booking_payment_status%TYPE;

BEGIN

   SELECT Booking_payment_status INTO v_status

   FROM Booking

   WHERE Booking_ID = 702;

   IF v_status = 'Paid' THEN

      DBMS_OUTPUT.PUT_LINE('Booking is Paid');

   ELSE

      DBMS_OUTPUT.PUT_LINE('Booking is Not Paid');

   END IF;

   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;
/
```



ORACLE Database Express Edition

User: MALL_MANAGEMENT_SYSTEM

Home > SQL > SQL Commands

☑ Autocommit Display 10

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Using 2 conditional statements
DECLARE
    v_status Booking.Booking_payment_status%TYPE;
BEGIN
    SELECT Booking_payment_status INTO v_status
    FROM Booking
    WHERE Booking_ID = 702;

    -- IF statement
    IF v_status = 'Paid' THEN
        DBMS_OUTPUT.PUT_LINE('Booking Is Paid');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Booking Is Not Paid');
    END IF;

    -- CASE statement
    CASE v_status
```

```
Booking Is Not Paid
Status = Unpaid

Statement processed.
```

0.00 seconds

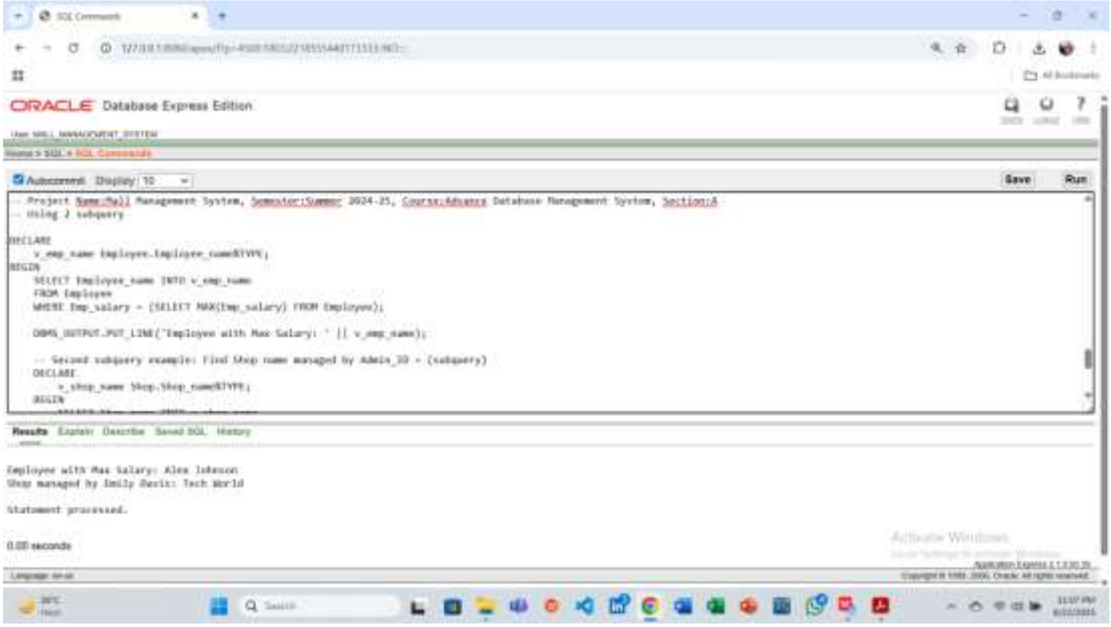
-2 subquery

Question: Find Employee name whose salary is equal to the maximum salary in Employee table.

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);
    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;

END;

/
```



ORACLE Database Express Edition

User: MALL_MANAGEMENT_SYSTEM

Home > SQL > SQL Commands

☑ Autocommit Display 10

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Using 2 subquery

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);

    -- Second subquery example: Find Shop name managed by Admin_ID = (subquery)
    DECLARE
        v_shop_name Shop.Shop_name%TYPE;
    BEGIN
```

Results Explain Describe Saved SQL History

```
Employee with Max Salary: Alex Johnson
Shop managed by Emily Davis: Tech World

Statement processed.
```

0.00 seconds

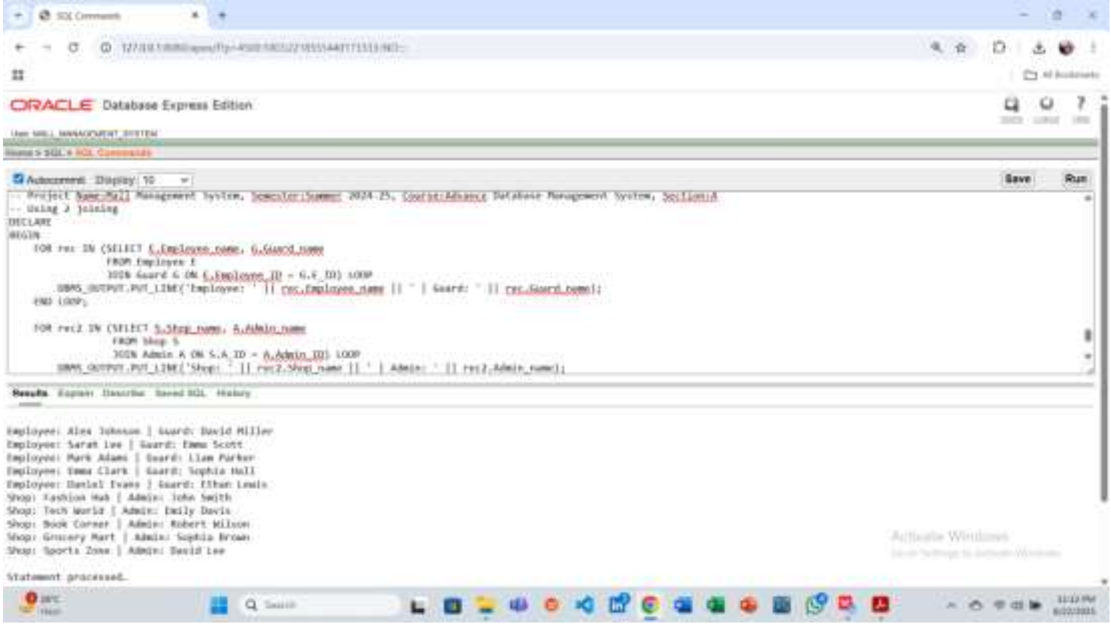
-2 joining

Question: Display Employee name and Guard name under them. Also Display Shop name and Admin name under them.

Answer:

```
DECLARE

BEGIN

      FOR rec IN (SELECT E.Employee_name, G.Guard_name

              FROM Employee E

              JOIN Guard G ON E.Employee_ID = G.E_ID) LOOP

        DBMS_OUTPUT.PUT_LINE('Employee: ' || rec.Employee_name || ' | Guard: ' ||
rec.Guard_name);
```

```
    END LOOP;

    FOR rec2 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: ' || rec2.Shop_name || ' | Admin: ' || rec2.Admin_name);

    END LOOP;

END;

/
```



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Using 2 joining
DECLARE
BEGIN
    FOR rec IN (SELECT E.Employee_name, G.Guard_name
                FROM Employee E
                JOIN Guard G ON E.Employee_ID = G.E_ID) LOOP
        DBMS_OUTPUT.PUT_LINE('Employee: ' || rec.Employee_name || ' | Guard: ' || rec.Guard_name);
    END LOOP;

    FOR rec2 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: ' || rec2.Shop_name || ' | Admin: ' || rec2.Admin_name);
```

```
Employee: Alex Johnson | Guard: David Miller
Employee: Sarah Lee | Guard: Emma Scott
Employee: Mark Adams | Guard: Liam Parker
Employee: Emma Clark | Guard: Sophia Hall
Employee: Daniel Evans | Guard: Ethan Lewis
Shop: Fashion Hub | Admin: John Smith
Shop: Tech World | Admin: Emily Davis
Shop: Book Corner | Admin: Robert Wilson
Shop: Grocery Mart | Admin: Sophia Brown
Shop: Sports Zone | Admin: David Lee

Statement processed.
```

ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

12. Advance PL/SQL

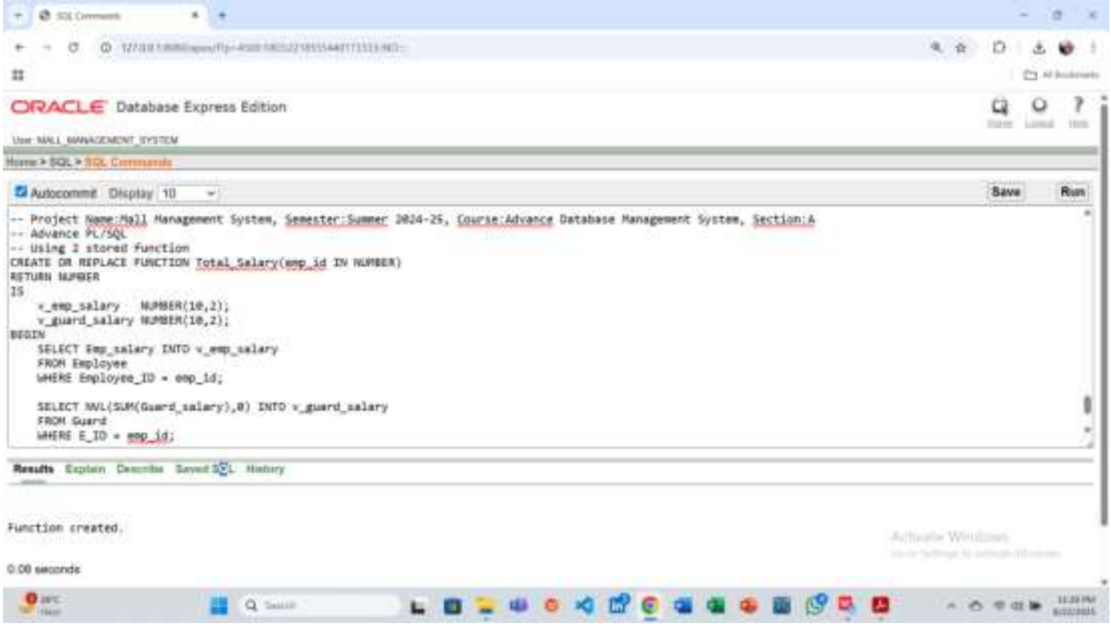
-2 stored function

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);
BEGIN
   SELECT Emp_salary INTO v_emp_salary
   FROM Employee
   WHERE Employee_ID = emp_id;
   SELECT NVL(SUM(Guard_salary),0) INTO v_guard_salary
   FROM Guard
   WHERE E_ID = emp_id;
   RETURN v_emp_salary + v_guard_salary;
END;
/
```

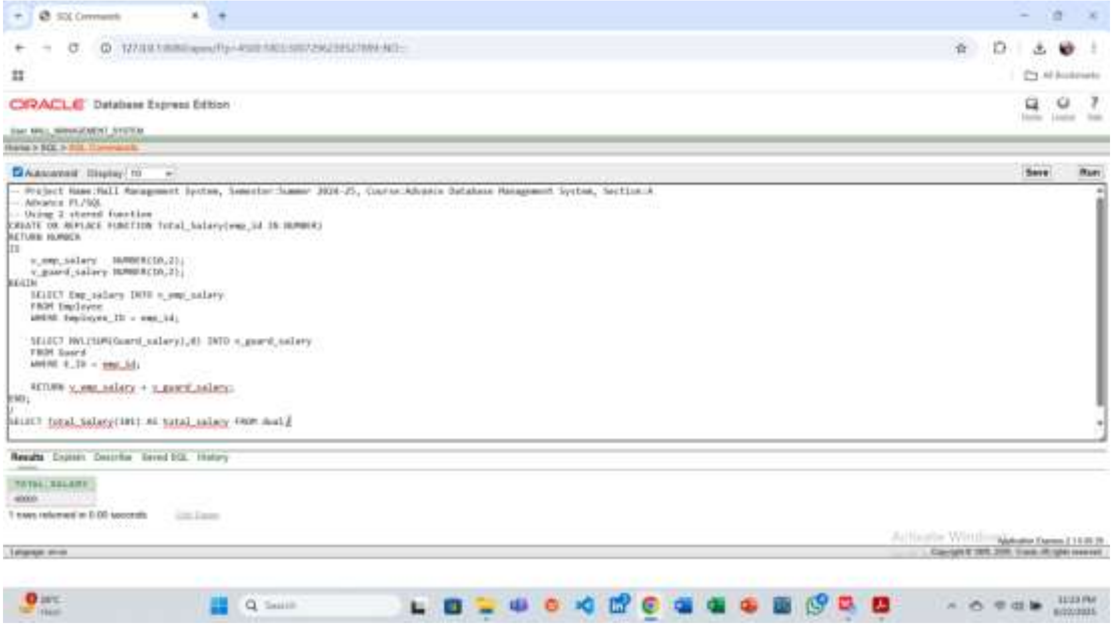


SQL Commands
ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Advance PL/SQL
-- Using 2 stored function
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);
BEGIN
    SELECT Emp_salary INTO v_emp_salary
    FROM Employee
    WHERE Employee_ID = emp_id;

    SELECT NVL(SUM(Guard_salary),0) INTO v_guard_salary
    FROM Guard
    WHERE E_ID = emp_id;
```

```
Function created.

0.08 seconds
```

SELECT Total_Salary(301) AS total_salary FROM dual;



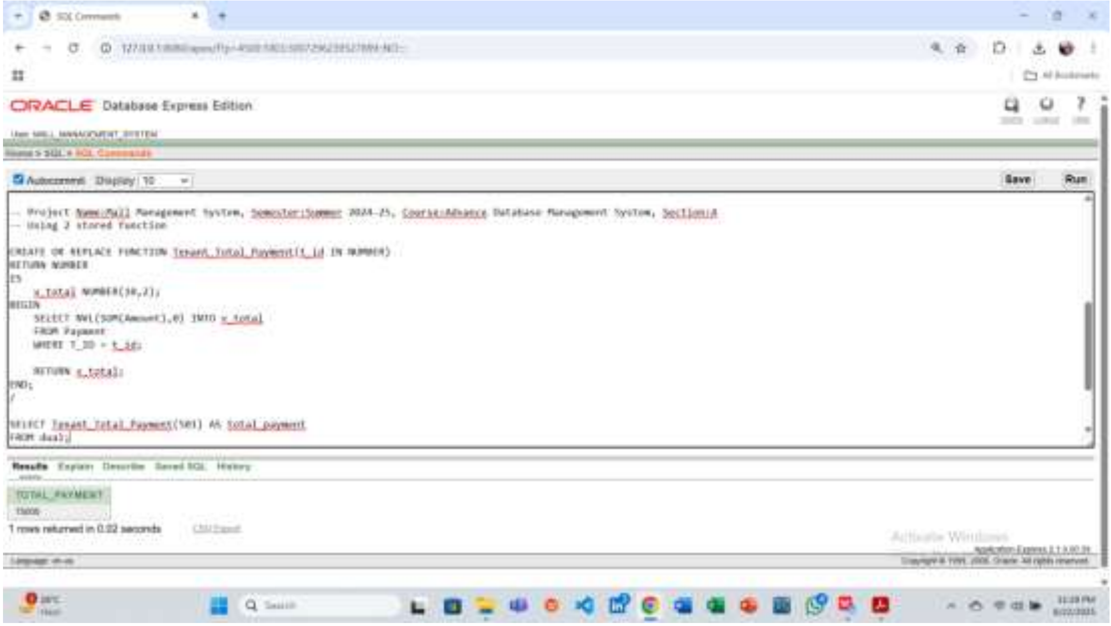
```
-- Project Name:Hall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Advance PL/SQL
-- Using 2 stored function
CREATE OR REPLACE FUNCTION total_salary(emp_id IN NUMBER)
RETURN NUMBER
IS
    v_emp_salary   NUMBER(10,2);
    v_guard_salary NUMBER(10,2);
BEGIN
    SELECT Emp_salary INTO v_emp_salary
    FROM Employee
    WHERE Employee_ID = emp_id;

    SELECT NVL(SUM(Guard_salary),0) INTO v_guard_salary
    FROM Guard
    WHERE E_ID = emp_id;

    RETURN v_emp_salary + v_guard_salary;
END;
/
SELECT total_salary(101) AS total_salary FROM dual;
```

```
TOTAL_SALARY
40000
1 rows returned in 0.00 seconds
```

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

   SELECT NVL(SUM(Amount),0) INTO v_total

   FROM Payment

   WHERE T_ID = t_id;



   RETURN v_total;

END;

/



SELECT Tenant_Total_Payment(501) AS total_payment

FROM dual;
```



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- Using 2 stored function

CREATE OR REPLACE FUNCTION Tenant_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;
END;
/

SELECT Tenant_Total_Payment(501) AS total_payment
FROM dual;
```

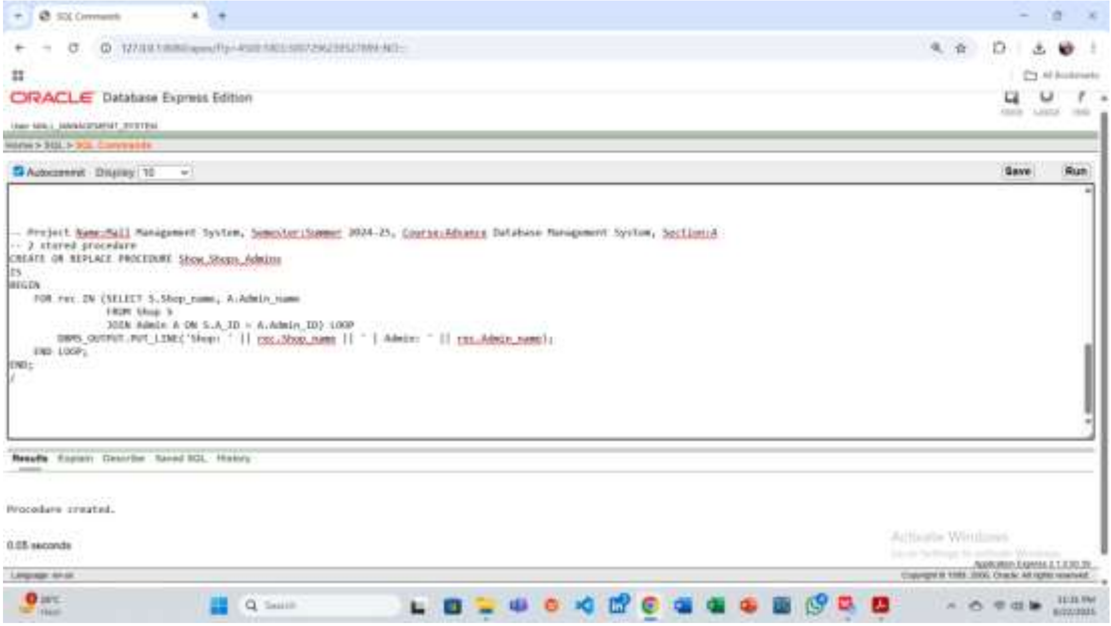
```
TOTAL_PAYMENT
15000
1 rows returned in 0.02 seconds
```

-2 stored procedure

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

Answer:

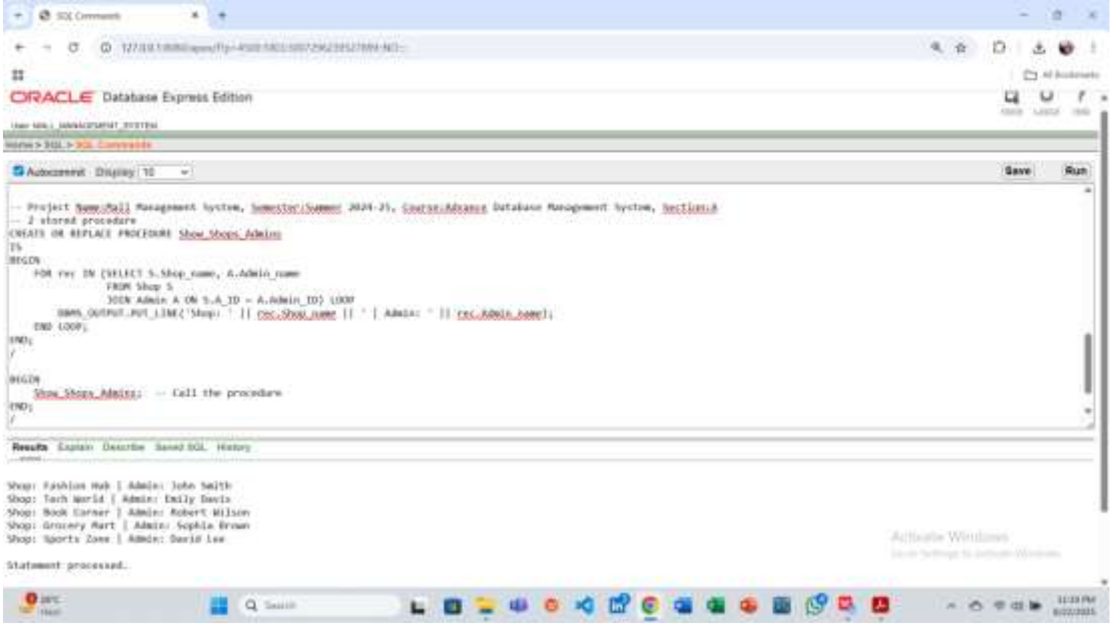
```
CREATE OR REPLACE PROCEDURE Show_Shops_Admins

IS

BEGIN

   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;

END;

/

BEGIN

   Show_Shops_Admins;  -- Call the procedure

END;

/
```



ORACLE Database Express Edition
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 stored procedure
CREATE OR REPLACE PROCEDURE Show_Shops_Admins
IS
BEGIN
    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;
END;
/
```

```
Procedure created.

0.05 seconds
```



ORACLE Database Express Edition
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 stored procedure
CREATE OR REPLACE PROCEDURE Show_Shops_Admins
IS
BEGIN
    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;
END;
/

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

```
Shop: Fashion Hub | Admin: John Smith
Shop: Tech World | Admin: Emily Davis
Shop: Book Corner | Admin: Robert Wilson
Shop: Grocery Mart | Admin: Sophia Brown
Shop: Sports Zone | Admin: David Lee

Statement processed.
```

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

v_total_salary := Total_Salary(emp_id);

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

END;

/

BEGIN

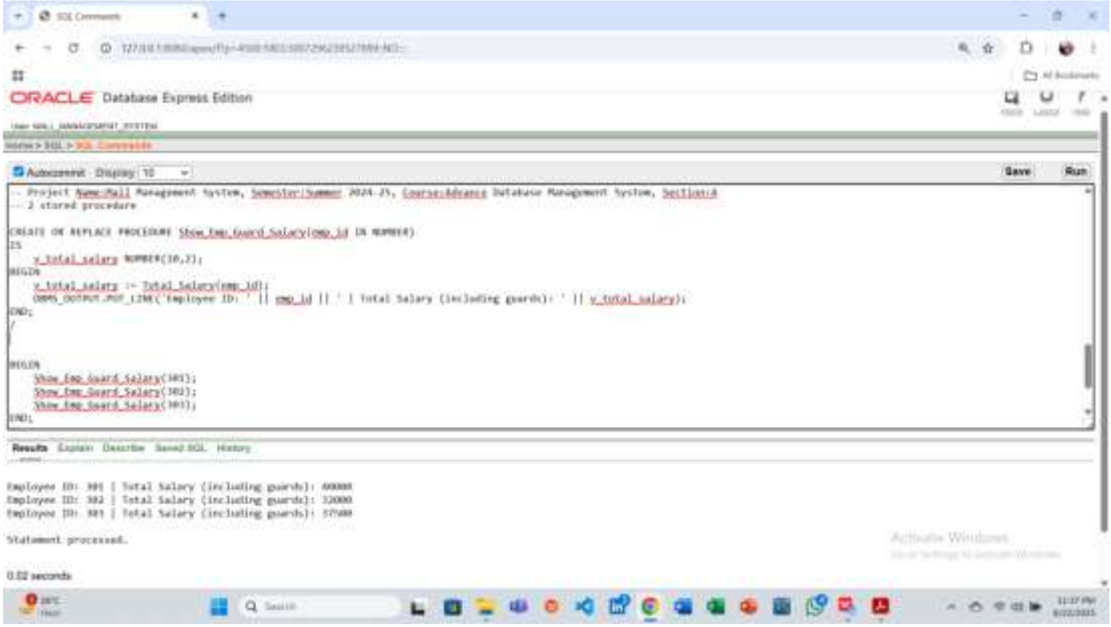
    Show_Emp_Guard_Salary(301);

    Show_Emp_Guard_Salary(302);

    Show_Emp_Guard_Salary(303);

END;

/



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:4
-- 2 stored procedure

CREATE OR REPLACE PROCEDURE Show_Emp_Guard_Salary(emp_id IN NUMBER)
IS
    v_total_salary NUMBER(10,2);
BEGIN
    v_total_salary := Total_Salary(emp_id);
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_id || ' | Total Salary (including guards): ' || v_total_salary);
END;
/


BEGIN
    Show_Emp_Guard_Salary(301);
    Show_Emp_Guard_Salary(302);
    Show_Emp_Guard_Salary(303);
END;
```

```
Employee ID: 301 | Total Salary (including guards): [illegible]
Employee ID: 302 | Total Salary (including guards): 320000
Employee ID: 303 | Total Salary (including guards): [illegible]

Statement processed.

0.02 seconds
```

-2 table-based record

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

Answer:

DECLARE

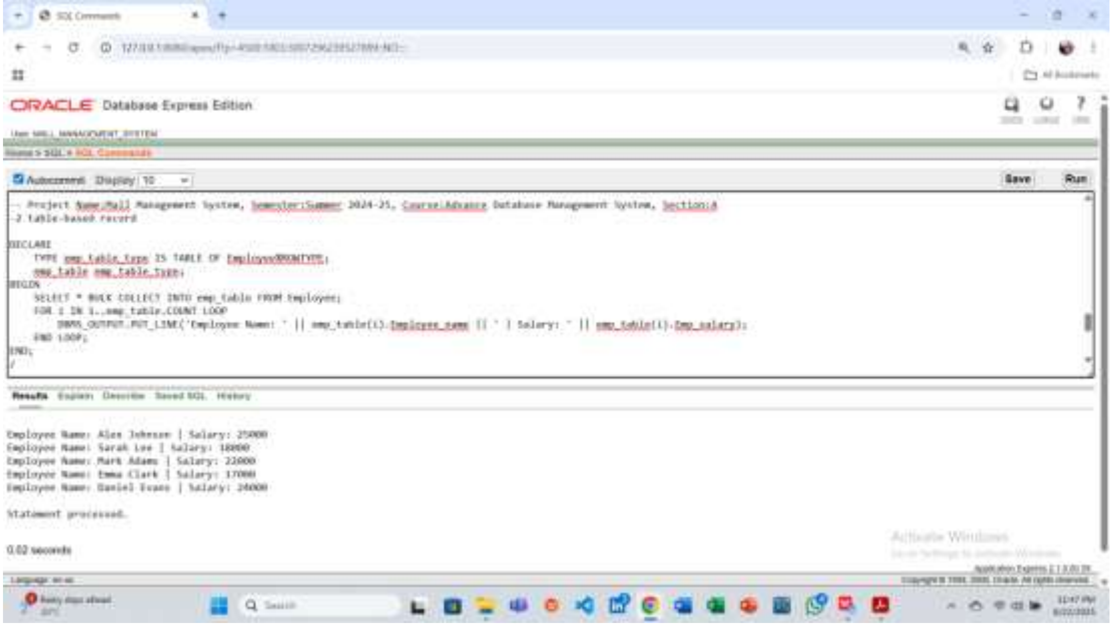
    TYPE emp_table_type IS TABLE OF Employee%ROWTYPE;

    emp_table emp_table_type;

BEGIN

    SELECT * BULK COLLECT INTO emp_table FROM Employee;

```
    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;

END;

/
```



ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

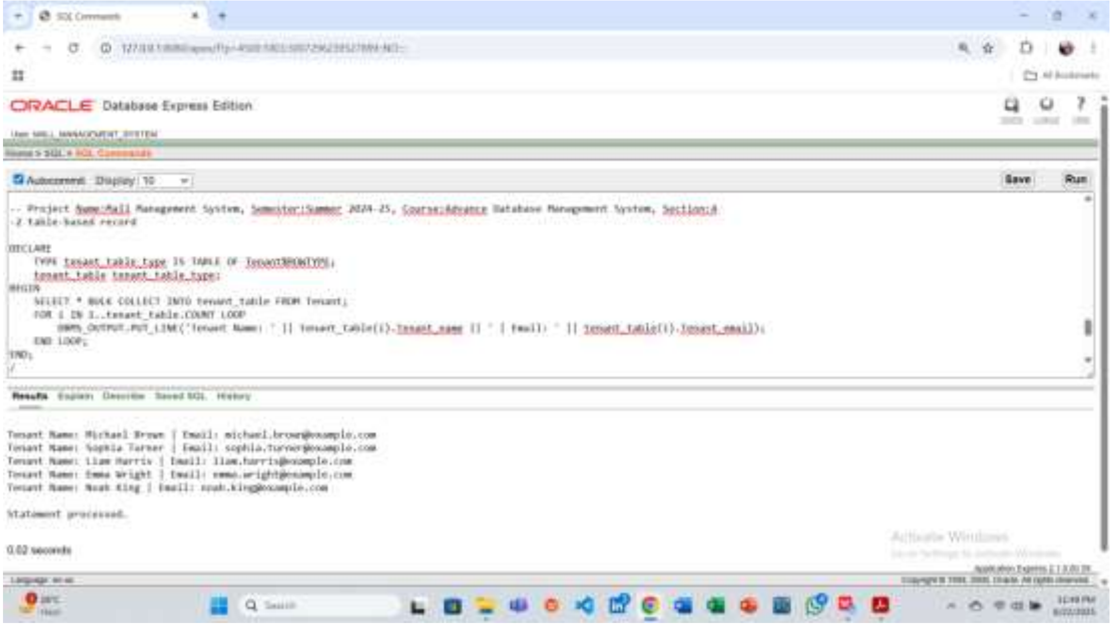
```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-2 table-based record

DECLARE
    TYPE emp_table_type IS TABLE OF Employee%ROWTYPE;
    emp_table emp_table_type;
BEGIN
    SELECT * BULK COLLECT INTO emp_table FROM Employee;
    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;
END;
/
```

```
Employee Name: Alex Johnson | Salary: 25000
Employee Name: Sarah Lee | Salary: 18000
Employee Name: Mark Adams | Salary: 22000
Employee Name: Emma Clark | Salary: 17000
Employee Name: Daniel Evans | Salary: 24000

Statement processed.

0.02 seconds
```

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

    SELECT * BULK COLLECT INTO tenant_table FROM Tenant;

    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;

END;

/
```



ORACLE Database Express Edition
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-2 table-based record

DECLARE
    TYPE tenant_table_type IS TABLE OF Tenant%ROWTYPE;
    tenant_table tenant_table_type;
BEGIN
    SELECT * BULK COLLECT INTO tenant_table FROM Tenant;
    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;
END;
/
```

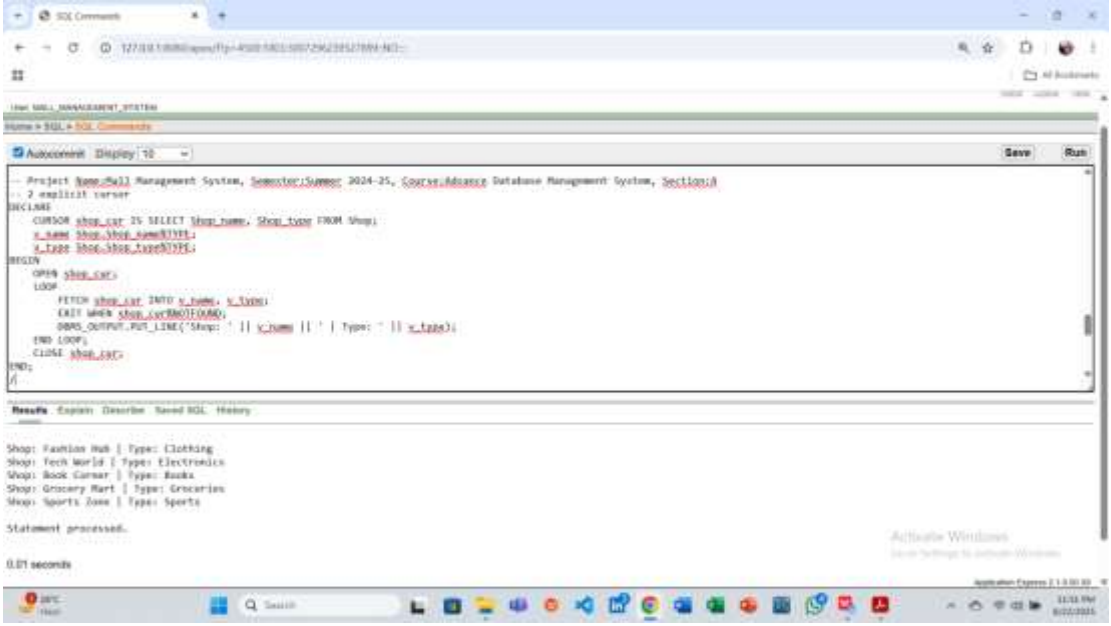
```
Tenant Name: Michael Brown | Email: michael.brown@example.com
Tenant Name: Sophia Turner | Email: sophia.turner@example.com
Tenant Name: Liam Harris | Email: liam.harris@example.com
Tenant Name: Emma Wright | Email: emma.wright@example.com
Tenant Name: Noah King | Email: noah.king@example.com

Statement processed.

0.02 seconds
```

-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;

END;

/
```



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 explicit cursor
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;
END;
/
```

```
Shop: Fashion Hub | Type: Clothing
Shop: Tech World | Type: Electronics
Shop: Book Corner | Type: Books
Shop: Grocery Mart | Type: Groceries
Shop: Sports Zone | Type: Sports

Statement processed.

0.01 seconds
```

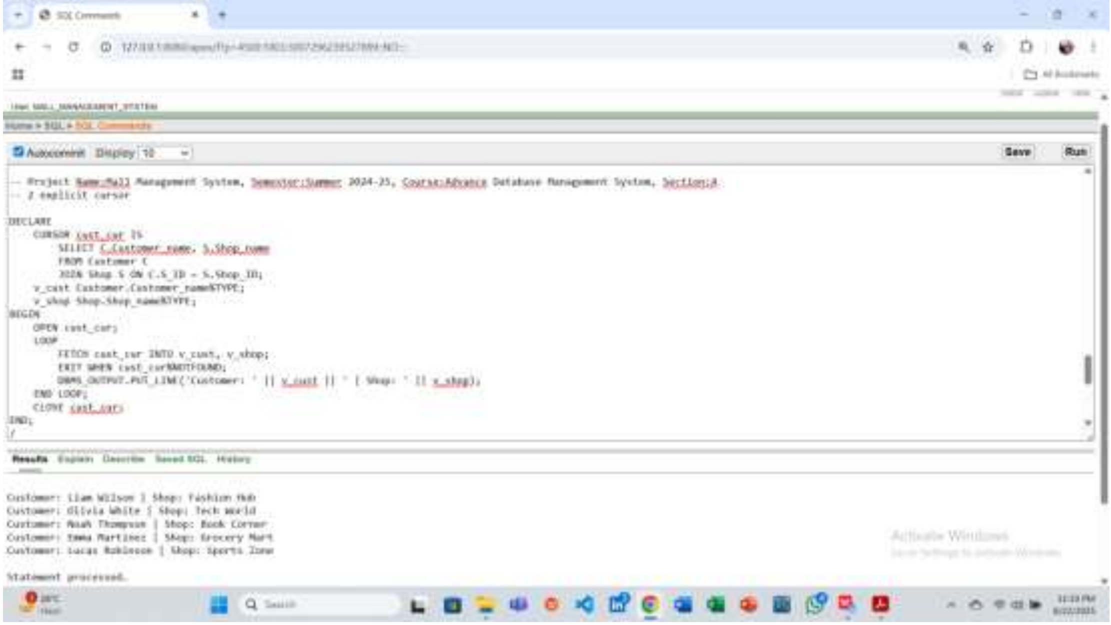
Question -2: Display Customer names and their Shop names using an explicit cursor.

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;
```

END;

/



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 explicit cursor

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;
END;
/
```

```
Customer: Liam Wilson | Shop: Fashion Hub
Customer: Olivia White | Shop: Tech World
Customer: Noah Thompson | Shop: Book Corner
Customer: Emma Martinez | Shop: Grocery Mart
Customer: Lucas Robinson | Shop: Sports Zone

Statement processed.
```

SQL Commands
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History
Activate Windows
Search

-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;

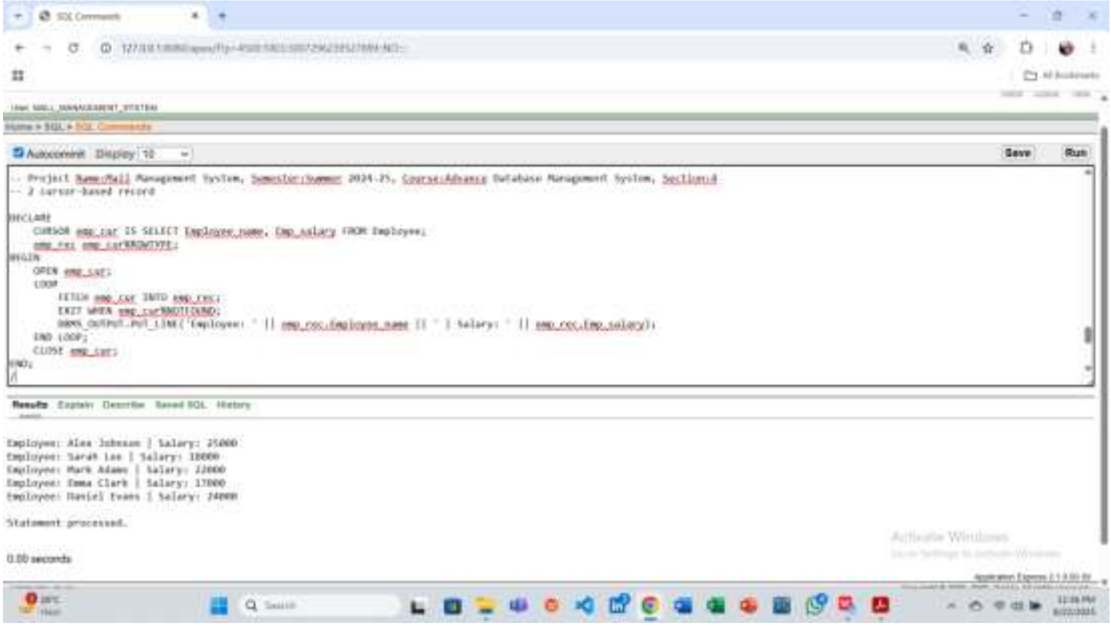
EXIT WHEN emp_cur%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.Employee_name || ' | Salary: ' || emp_rec.Emp_salary);

END LOOP;

CLOSE emp_cur;

END; /



SQL Commands
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

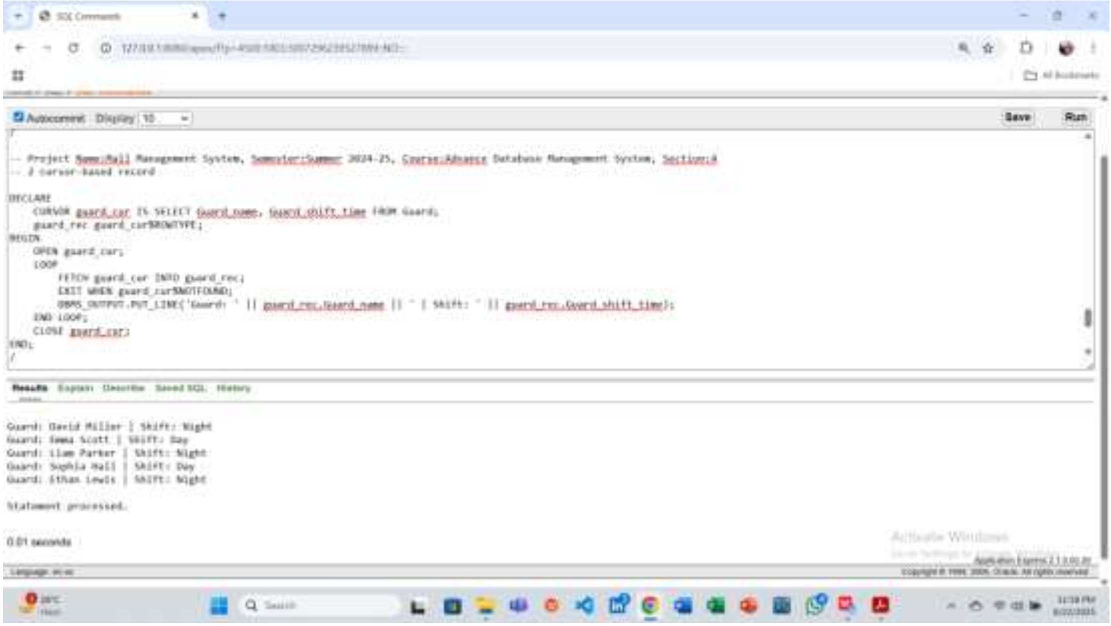
```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:4
-- 2 cursor-based record

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;
END;
/
```

```
Employee: Alex Johnson | Salary: 25000
Employee: Sarah Lee | Salary: 18000
Employee: Mark Adams | Salary: 22000
Employee: Emma Clark | Salary: 17000
Employee: Daniel Evans | Salary: 24000

Statement processed.

0.00 seconds
```

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;

END;

/
```



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 cursor-based record

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;
END;
/
```

```
Guard: David Miller | Shift: Night
Guard: Emma Scott | Shift: Day
Guard: Liam Parker | Shift: Night
Guard: Sophia Hall | Shift: Day
Guard: Ethan Lewis | Shift: Night

Statement processed.

0.01 seconds
```

-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

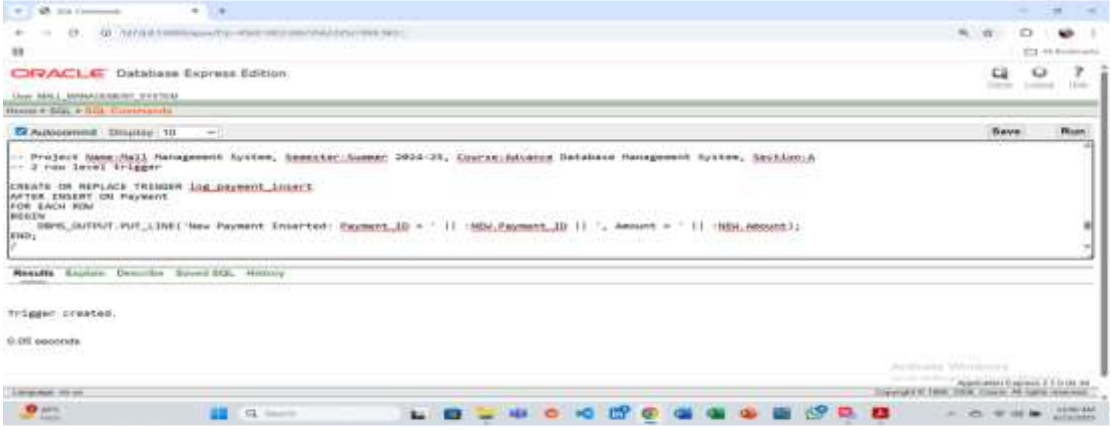
FOR EACH ROW

BEGIN

   DBMS_OUTPUT.PUT_LINE('New Payment Inserted: Payment_ID = ' || :NEW.Payment_ID || ', Amount = ' || :NEW.Amount);

END;

/



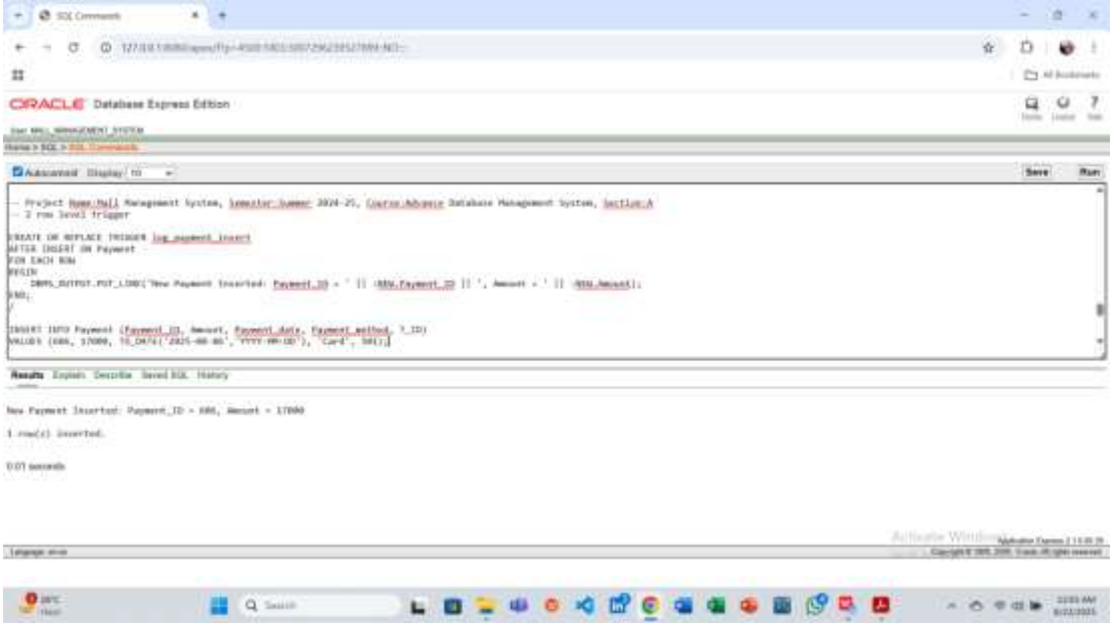
```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 row level trigger

CREATE OR REPLACE TRIGGER log_payment_insert
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('New Payment Inserted: Payment_ID = ' || :NEW.Payment_ID || ', Amount = ' || :NEW.Amount);
END;
/
```

```
Trigger created.

0.05 seconds
```

-- 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);

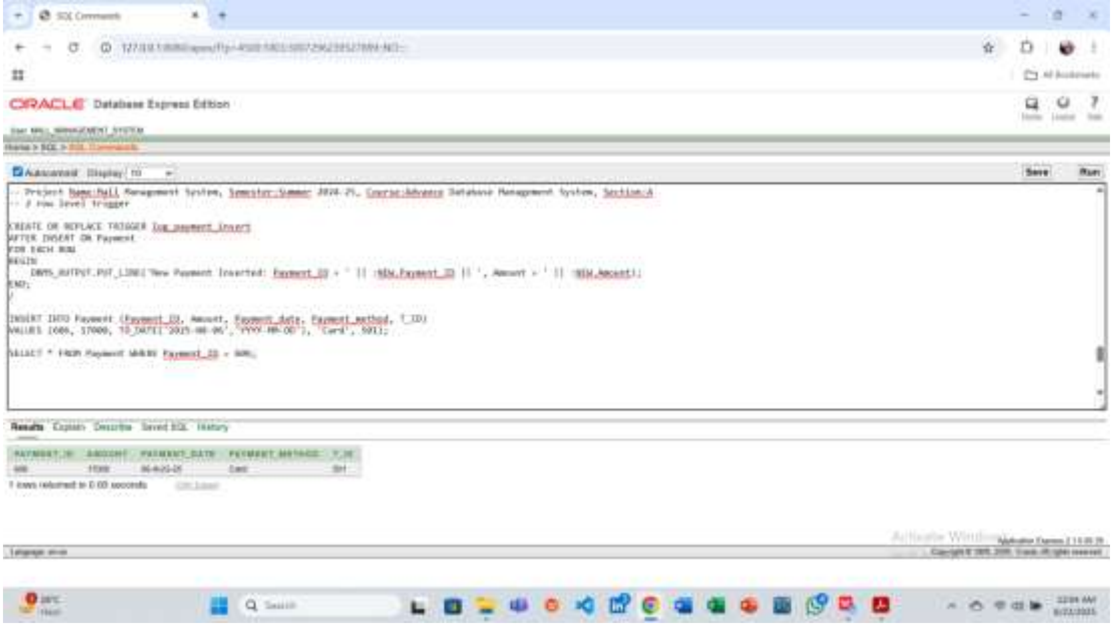


```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 row level trigger

CREATE OR REPLACE TRIGGER log_payment_insert
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
   DBMS_OUTPUT.PUT_LINE('New Payment Inserted: Payment_ID = ' || :NEW.Payment_ID || ', Amount = ' || :NEW.Amount);
END;
/

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);
```

```
New Payment Inserted: Payment_ID = 606, Amount = 17000

1 row(s) inserted.

0.01 seconds
```

-- Verifying if the Row is in Table

SELECT * FROM Payment WHERE Payment_ID = 606;



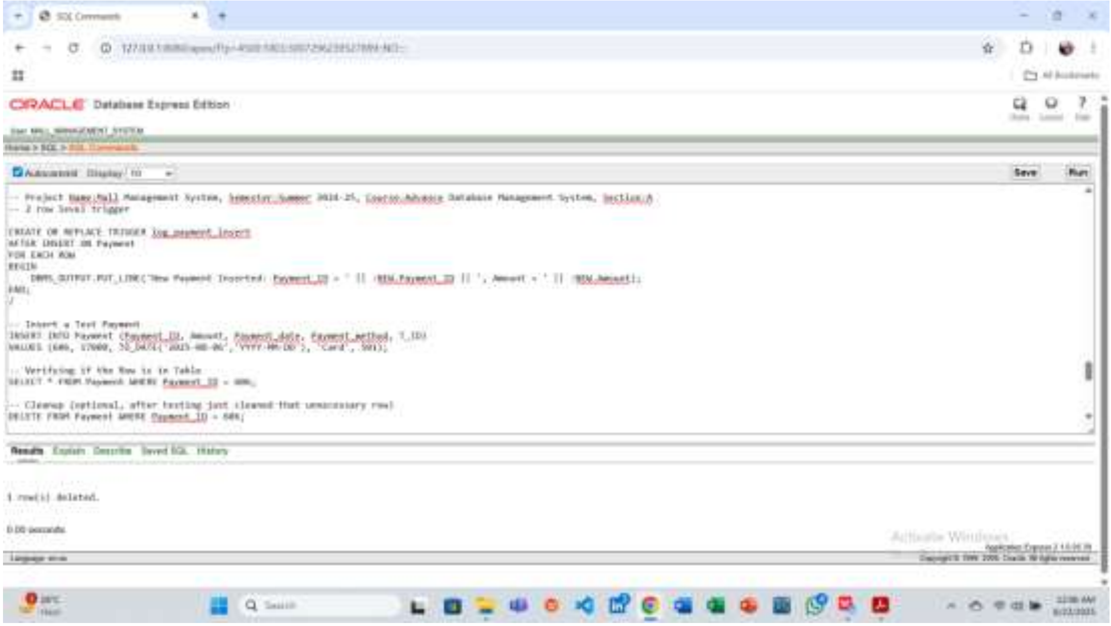
```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 row level trigger

CREATE OR REPLACE TRIGGER log_payment_insert
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
   DBMS_OUTPUT.PUT_LINE('New Payment Inserted: Payment_ID = ' || :NEW.Payment_ID || ', Amount = ' || :NEW.Amount);
END;
/

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);

SELECT * FROM Payment WHERE Payment_ID = 606;
```

PAYMENT_ID	AMOUNT	PAYMENT_DATE	PAYMENT_METHOD	T_ID
606	17000	06-AUG-25	Card	501

1 rows returned in 0.00 seconds

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

DELETE FROM Payment WHERE Payment_ID = 606;



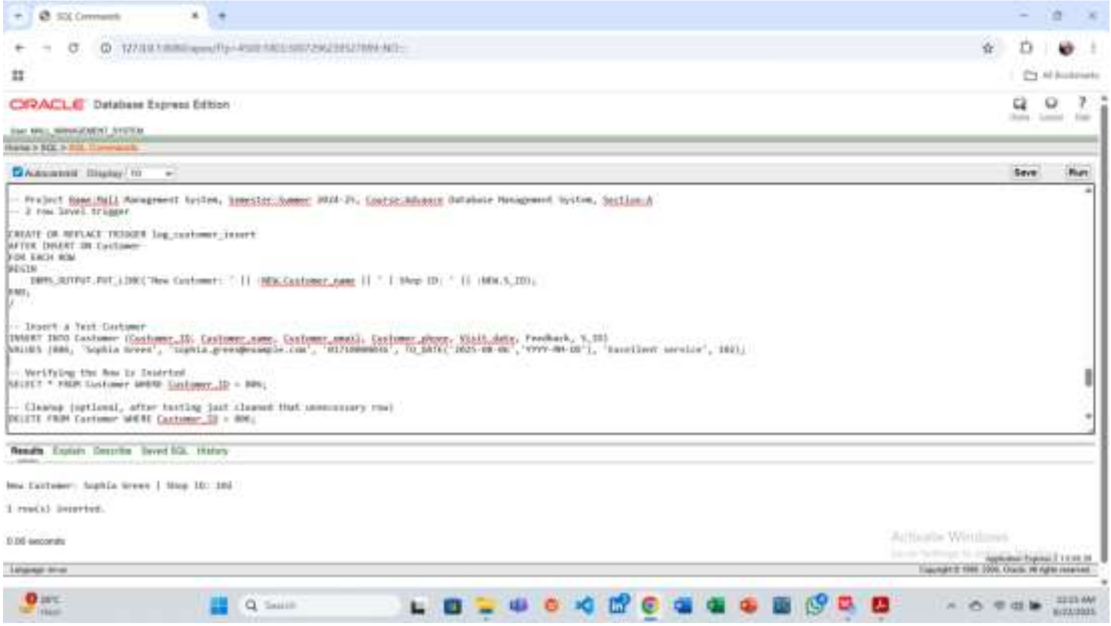
```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 row level trigger

CREATE OR REPLACE TRIGGER log_payment_insert
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('New Payment Inserted: Payment_ID = ' || :NEW.Payment_ID || ', Amount = ' || :NEW.Amount);
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;
```

```
1 row(s) deleted.

0.00 seconds
```

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

    DBMS_OUTPUT.PUT_LINE('New Customer: ' || :NEW.Customer_name || ' | Shop ID: ' || :NEW.S_ID);

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;
```



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 row level trigger

CREATE OR REPLACE TRIGGER log_customer_insert
AFTER INSERT ON Customer
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('New Customer: ' || :NEW.Customer_name || ' | Shop ID: ' || :NEW.S_ID);
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', '01710000046', 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;
```

```
New Customer: Sophia Green | Shop ID: 102

1 row(s) inserted.

0.06 seconds
```

-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

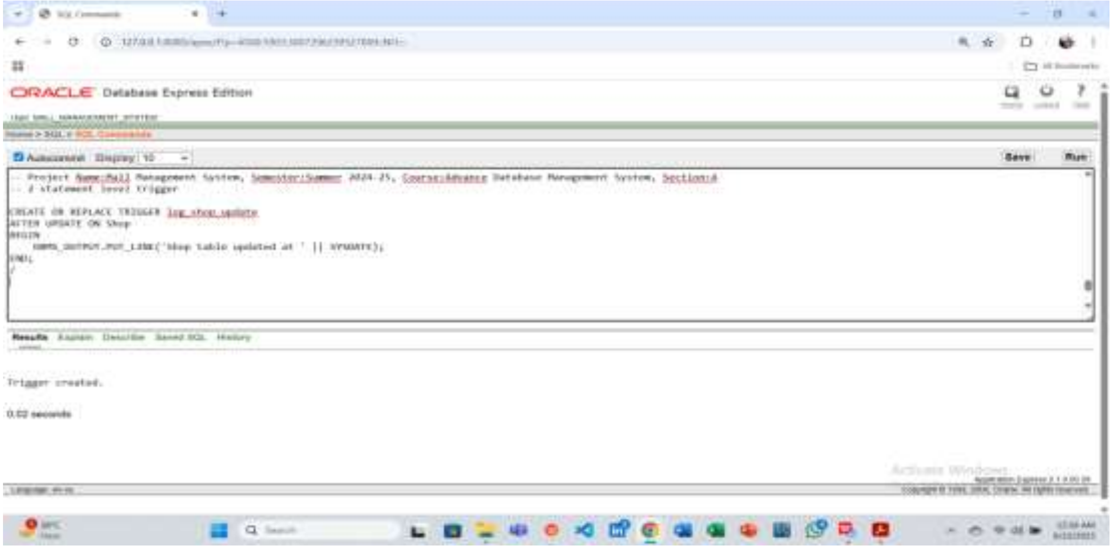
AFTER UPDATE ON Shop

BEGIN

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

END;

/



```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 statement level trigger

CREATE OR REPLACE TRIGGER log_shop_update
AFTER UPDATE ON Shop
BEGIN
    DBMS_OUTPUT.PUT_LINE('Shop table updated at ' || SYSDATE);
END;
/
```

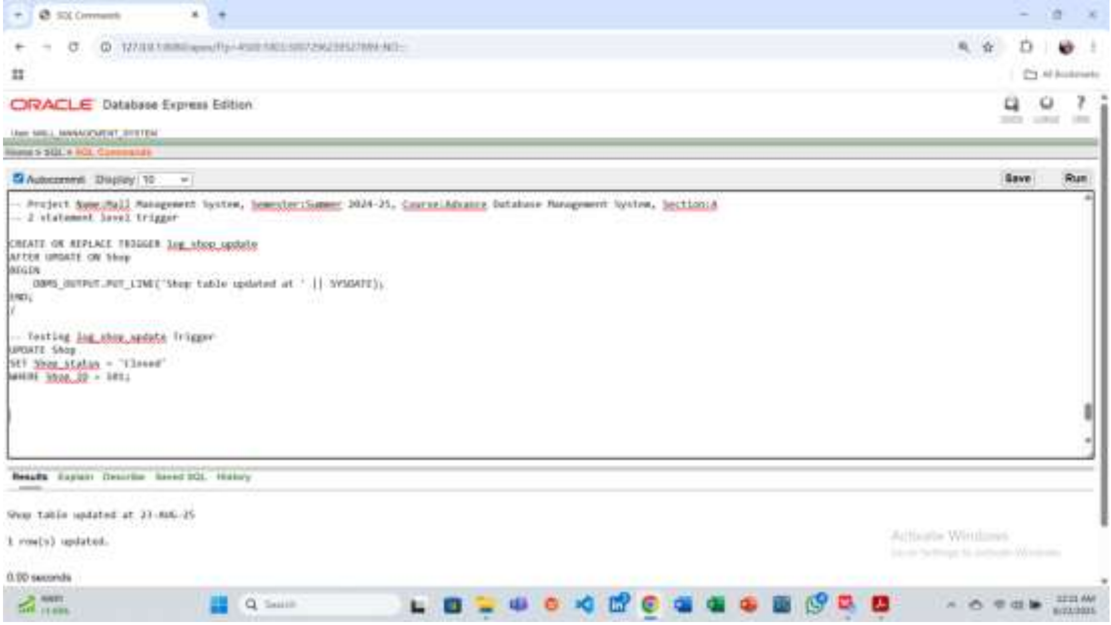
```
Trigger created.

0.02 seconds
```

-- Testing log_shop_update Trigger

UPDATE Shop

SET Shop_status = 'Closed'

WHERE Shop_ID = 101;



SQL Commands
ORACLE Database Express Edition
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 statement level trigger

CREATE OR REPLACE TRIGGER log_shop_update
AFTER UPDATE ON Shop
BEGIN
    DBMS_OUTPUT.PUT_LINE('Shop table updated at ' || SYSDATE);
END;
/

-- Testing log_shop_update Trigger
UPDATE Shop
SET Shop_status = 'Closed'
WHERE Shop_ID = 101;
```

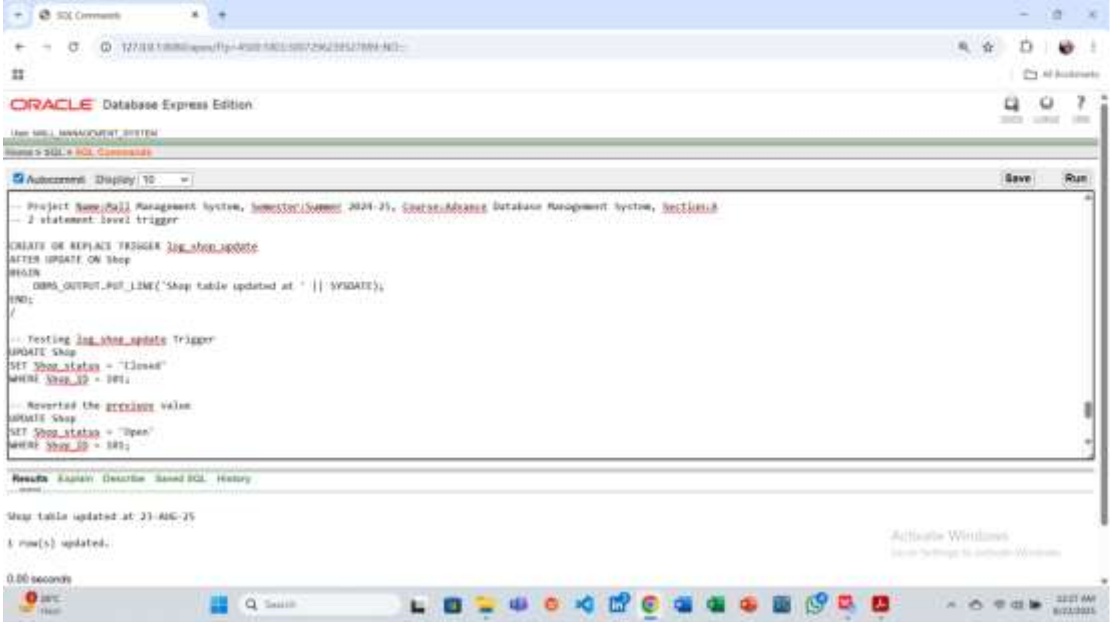
```
Shop table updated at 23-AUG-25

1 row(s) updated.

0.00 seconds
```

-- Reverted the previous value

UPDATE Shop

SET Shop_status = 'Open'

WHERE Shop_ID = 101;



SQL Commands
ORACLE Database Express Edition
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

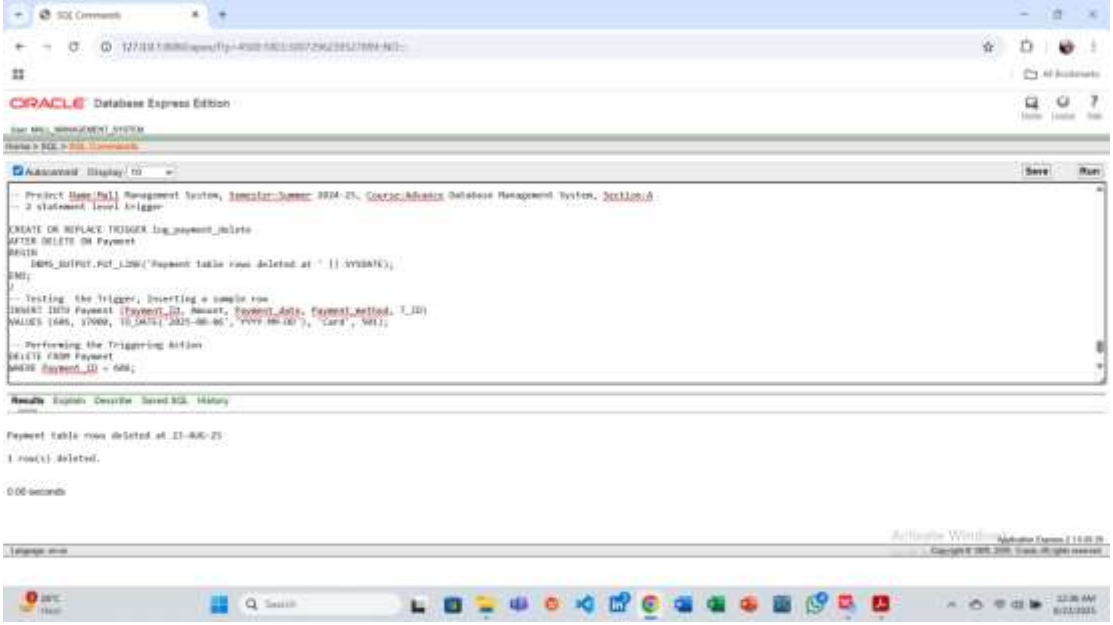
```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 statement level trigger

CREATE OR REPLACE TRIGGER log_shop_update
AFTER UPDATE ON Shop
BEGIN
    DBMS_OUTPUT.PUT_LINE('Shop table updated at ' || SYSDATE);
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;
```

```
Shop table updated at 23-AUG-25

1 row(s) updated.

0.00 seconds
```

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

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

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;
```



SQL Commands
ORACLE Database Express Edition
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 statement level trigger

CREATE OR REPLACE TRIGGER log_payment_delete
AFTER DELETE ON Payment
BEGIN
    DBMS_OUTPUT.PUT_LINE('Payment table rows deleted at ' || SYSDATE);
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;
```

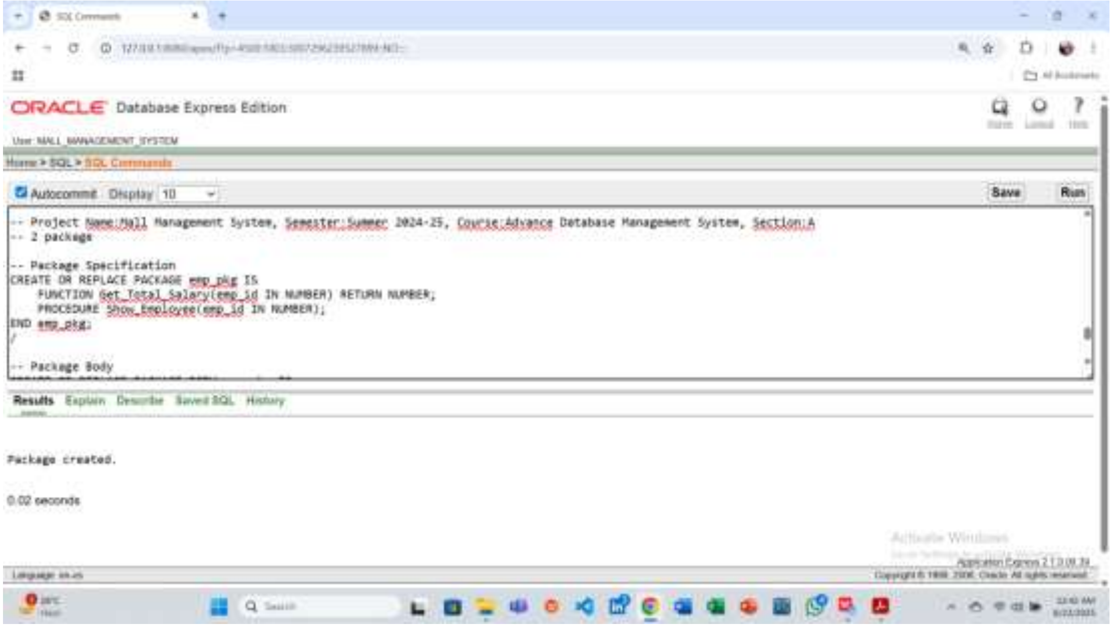
```
Payment table rows deleted at 23-AUG-25

1 row(s) deleted.

0.00 seconds
```

-2 package

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

Answer:

```
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;
/
```



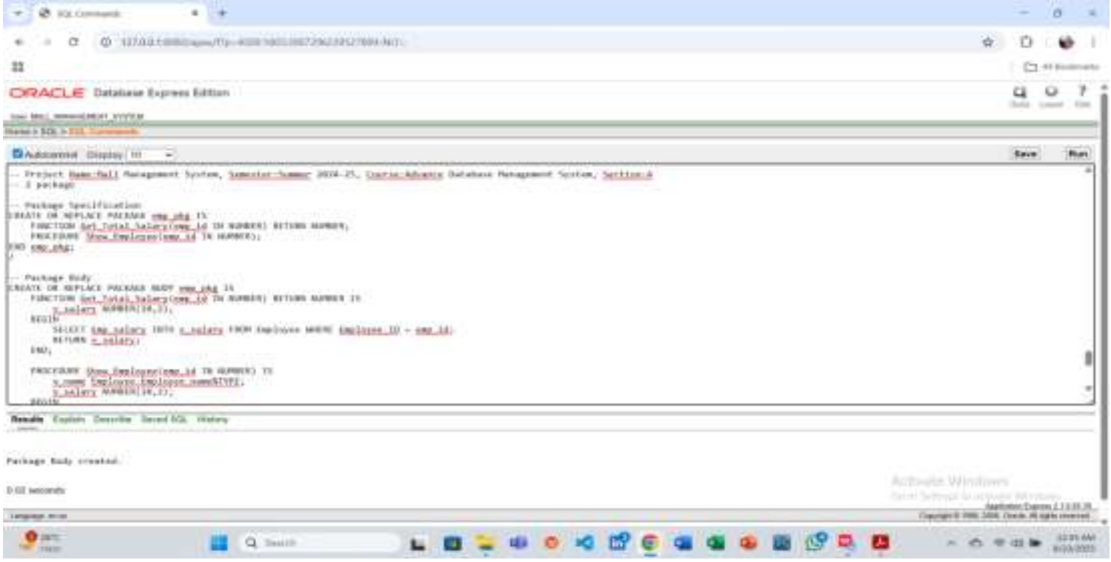
ORACLE Database Express Edition
User: MALL_MANAGEMENT_SYSTEM
Home > SQL > SQL Commands
☑ Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 2 package

-- 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
```

```
Package created.

0.02 seconds
```

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


   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);
    END;
END emp_pkg;
/
```



ORACLE Database Express Edition
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 3 package

-- 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 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;
    END;

    PROCEDURE Show_Employee(emp_id IN NUMBER) IS
        v_name Employee.Employee_name%TYPE;
        v_salary NUMBER(10,2);
    BEGIN
```

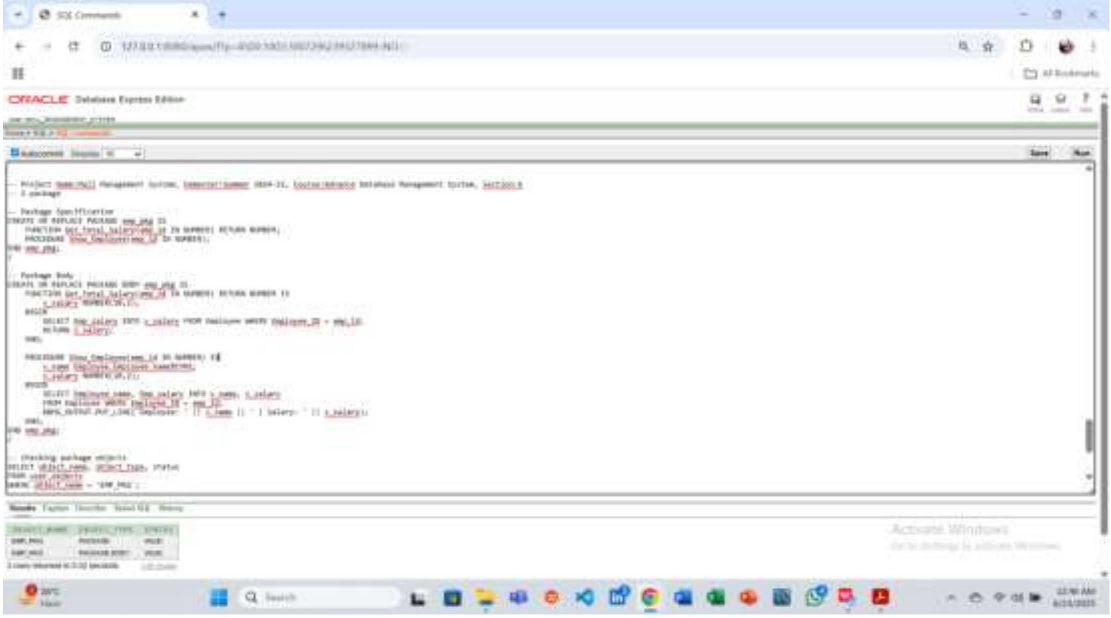
```
Package Body created.

0.02 seconds
```

-- Checking package objects

SELECT object_name, object_type, status

FROM user_objects

WHERE object_name = 'EMP_PKG';



ORACLE Database Express Edition
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 3 package

-- 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 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;
    END;

    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);
    END;
END emp_pkg;
/

-- Checking package objects
SELECT object_name, object_type, status
FROM user_objects
WHERE object_name = 'EMP_PKG';
```

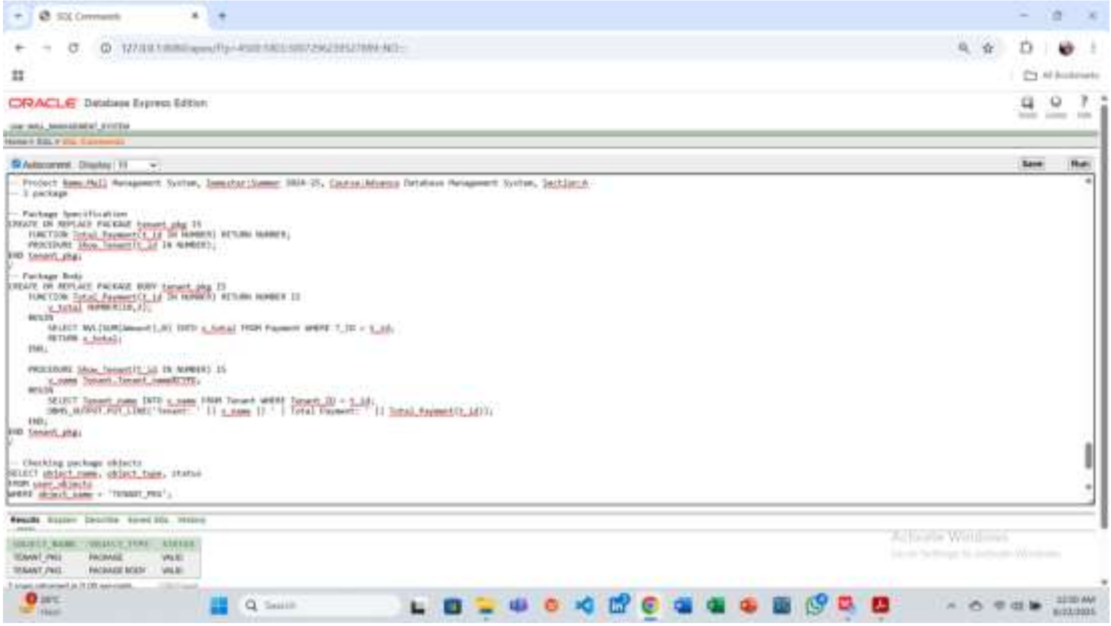
OBJECT_NAME	OBJECT_TYPE	STATUS
EMP_PKG	PACKAGE	VALID
EMP_PKG	PACKAGE BODY	VALID

2 rows returned in 0.02 seconds CSV Export

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 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;
   END;

   PROCEDURE Show_Tenant(t_id IN NUMBER) IS
      v_name Tenant.Tenant_name%TYPE;
   BEGIN
      SELECT Tenant_name INTO v_name FROM Tenant WHERE Tenant_ID = t_id;
      DBMS_OUTPUT.PUT_LINE('Tenant: ' || v_name || ' | Total Payment: ' || Total_Payment(t_id));
   END;
END tenant_pkg;
/


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


ORACLE Database Express Edition
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
-- Project Name:Mall Management System, Semester:Summer 2024-25, Course:Advance Database Management System, Section:A
-- 3 package

-- 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 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;
    END;

    PROCEDURE Show_Tenant(t_id IN NUMBER) IS
        v_name Tenant.Tenant_name%TYPE;
    BEGIN
        SELECT Tenant_name INTO v_name FROM Tenant WHERE Tenant_ID = t_id;
        DBMS_OUTPUT.PUT_LINE('Tenant: ' || v_name || ' | Total Payment: ' || Total_Payment(t_id));
    END;
END tenant_pkg;
/

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

OBJECT_NAME	OBJECT_TYPE	STATUS
TENANT_PKG	PACKAGE	VALID
TENANT_PKG	PACKAGE BODY	VALID

2 rows returned in 0.00 seconds

13. 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.