Project Title

Mini Project Report -Database Lab (DSE 2241)
Department of Data Science & Computer Applications

B. Tech Data Science

4th Semester

Submitted By

Aditya Dwarkani	220968178

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

ABSTRACT

The Garage Database Management System (GDMS) addresses the critical need for efficient management in today's rapidly evolving automotive sector. By centralizing vehicle information, customer data, repair records, costs, garage employees and supplier details, GDMS streamlines workflows and enhances operational efficiency.

This project begins with a systematic examination of the operational needs and objectives, followed by the utilization of SQL Plus for database implementation. The efficient database design facilitates seamless communication, improving process efficiency and service delivery. Implementing a comprehensive database offers several advantages, including optimized inventory tracking, streamlined appointment scheduling, and strategic decision-making through analytics.

In conclusion, the Garage Management System project promises significant outcomes for garages and stakeholders. The centralized database enhances operations, while optimized inventory tracking reduces costs and improves resource allocation. By integrating various facets of garage operations, the project provides a comprehensive solution benefiting both the business and its clientele.

Contents

Chapter 1

Introduction

The domain of garage car management is a critical facet of the automotive industry that plays a pivotal role in maintaining and enhancing the functionality of vehicles. As technology continues to advance, modern vehicles are equipped with complex systems and intricate components, demanding a sophisticated approach to repair, maintenance, and overall management. The domain encapsulates a spectrum of activities within garages, ranging from routine inspections and repairs to meticulous record-keeping, inventory management, and customer service.

In the contemporary automotive landscape, garages are not just places for mechanical repairs; they have evolved into multifaceted service centres dealing with electronic diagnostics, software updates, and intricate system integrations. This shift places a substantial burden on garage owners and mechanics to not only keep pace with technological advancements but also to efficiently manage the vast array of data associated with each vehicle passing through their facilities. The management of vehicle information, service histories, inventory, and employee schedules are becoming increasingly complex, necessitating the implementation of robust database management systems tailored to the unique needs of the garage car management domain.

Existing System Challenges and the Need for Change:

The conventional methods employed in garage car management often rely on manual record-keeping systems, paper-based schedules, and fragmented databases. This approach, while once effective, is proving inadequate in the face of the industry's evolving demands. The limitations of the existing system become apparent in the form of inefficiencies, errors, and a lack of scalability.

One of the primary challenges of the current systems is the difficulty in maintaining accurate and up-to-date records of vehicles, their service histories, and associated customer details. Manual data entry is not only timeconsuming but also prone to human errors, potentially leading to mismanagement of critical information. Additionally, the reliance on paper-based scheduling systems can result in scheduling conflicts, missed appointments, and suboptimal resource utilization.

Inventory management is another critical aspect where traditional methods fall short. Tracking spare parts, managing stock levels, and reordering processes often rely on manual assessments, leading to instances of stockouts or overstocking. This inefficiency can significantly impact the garage's ability to provide timely and efficient services, affecting both customer satisfaction and the bottom line.

In the face of these challenges, there is a compelling need for a paradigm shift towards modern, technologydriven solutions that can streamline operations, enhance data accuracy, and provide actionable insights. The Garage Car Management System project endeavours to address these pain points by introducing a comprehensive database management system specifically tailored to meet the unique requirements of the garage car management domain. This shift towards automation and digitization aims to not only improve the efficiency of garage operations but also to future-proof these establishments in a rapidly evolving automotive landscape.

Chapter 2

Synopsis

2.1 Proposed System

Currently, most of the garages lack proper management systems. The owners have to wait for long durations, and the vehicle delivery date keeps extending, majorly to due to unavailability of parts or workforce at the station. To solve this problem and structure the garage ecosystem we have created a Garage Database Management System.

This system effortlessly handles all aspects of vehicle maintenance, from owner details to supplier information, ensuring smooth operations and happy customers. It stores appointment dates for every visit for every customer so that they can be attended by an employee for effective task assignment and avoid crowd at the garage. It allows easy access to vehicle details like make, model and insurance. It will also track parts availability for timely repairs and hence decreasing waiting period for customers in case of unavailability of parts at the station. By integrating these features, the system automates garage operations, boosting time efficiency, customer satisfaction and profits.

2.2 Objectives

The Main Objectives of the work are:

- To create a **centralised database** system to store information on cars, owners, employees, customer ratings, repair costs, damaged parts, etc.
- Implement systematic methodologies to optimise garage shop operations like **appointment scheduling**.
- Enable tracking of all garage activities like employee assignments to customers. This ensures transparent operations and efficient **monitoring of workflow**.
- With detailed records, garages can ensure compliance with regulations, track **employee performance**, and maintain accountability across all aspects of their operations.
- Incorporate features for managing supplier relationships, like tracking orders and managing vendor contracts.
- Formulation of total bills has been made systematic and easier by mapping the cost of parts ordered and checking for both paid and unpaid bills.

Chapter 3

Functional Requirements

What is a Functional Requirement?
A Functional Requirement (FR) is a description of the service that the software must offer. It is a specification that describes the system's behaviour or function. It outlines what the system should do and how it should perform under certain conditions, defining the system's capabilities and the interaction between its components. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

3.1 Owner and Vehicle Registration Module

Customers when providing their personal as well as vehicle's data, get added to the database. This helps in keeping the data systematically and check for repairs to be done on the vehicle and their status.

3.1.1 Registration of a vehicle into the garage

The data must be entered into the garage database.

INPUT	Owner details, Vehicle details
Processing	The values are entered into the database
OUTPUT	N/A

3.1.2 Previous Repairs Finder

When provided with the registration number of a car, we can check for past records of the car.

Input	Reg. No. of car
Processing	Check if the car has previously visited the garage and if yes, then what repairs have been done.
Output	The previous repairs done on the car are displayed.

3.2 Garage Employees Module

There are several employees working for the garage. Each employee is given a rating after every service. The salary of an employee is incremented according to their ratings periodically.

Input	Employee id
Processing	Check for the number of repairs the employee has made. Update the salary of an employee after every 10^{th} repair if his/her rating is >=8.
Output	Updated salary is displayed.

3.3 Payment of dues

The user is given the total bill according to the repairs and materials ordered for the repairs.

Input	Registration number of car, Owner email id
Processing	The order details are checked and all the amounts are added to give the total amount payable.
Output	The total bill is displayed showing the amount to be paid by the customer.

3.4 Supplier Details Module

All available suppliers are stored in the suppliers table. This helps in ordering adequate materials from the specific suppliers as and when required.

Input	Supplier information
Processing	The information is stored in a table and when orders have been made, the suppliers are informed.
Output	N/A

Chapter 4

Detailed Design

4.1 ER Diagram

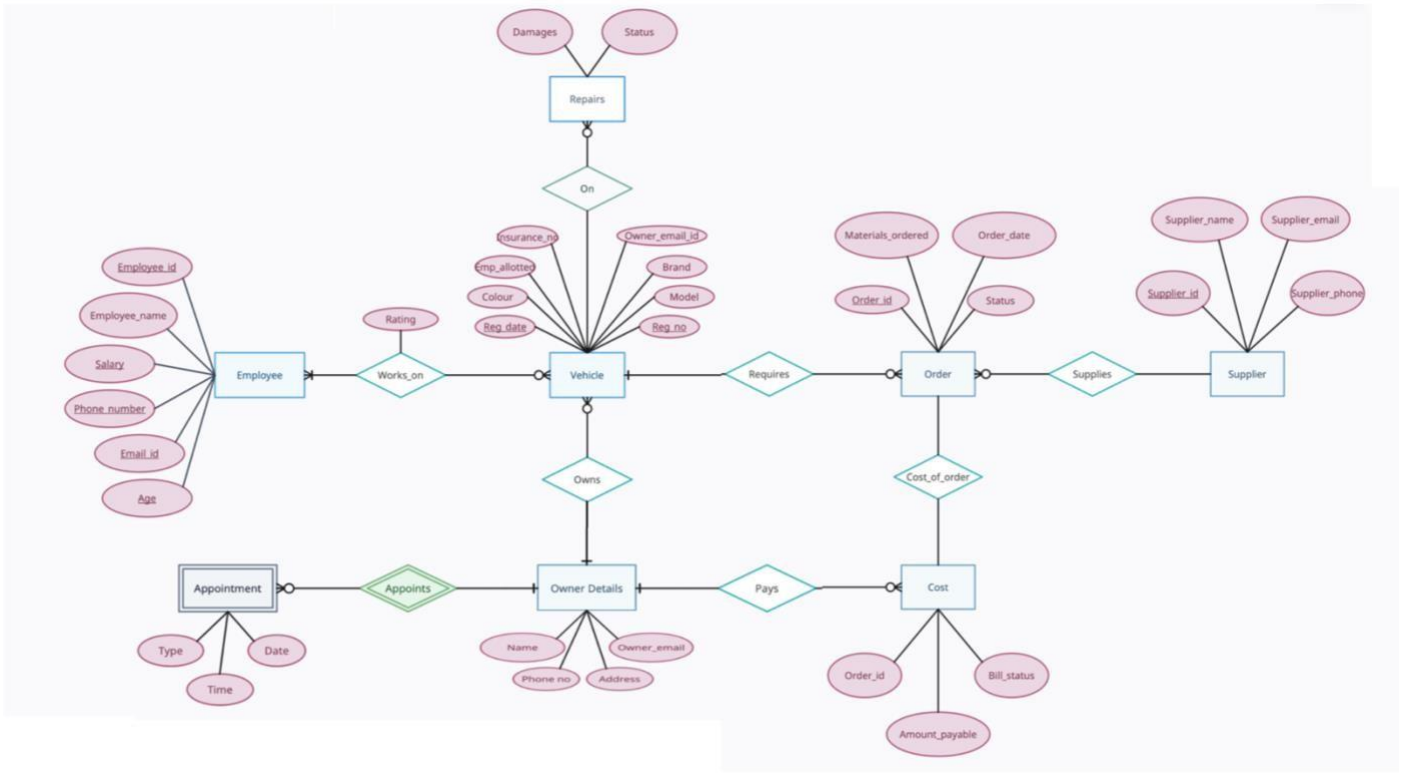

Figure 4.1 ER Diagram

4.2 Schema Diagram

1. Employee (Employee_id , Employee_name, Salary, Phone_number, Email_id, Age)
2. Works_on (Employee_id , Reg_no, Rating)
3. Vehicle (Reg_no, Reg_date, Model, Brand, Insurance_no, Colour, Employee_id, Owner_email_id)
4. Repairs (Reg_no, Reg_date, Damages, Status)
5. Order (Order_id, Material_ordered, Order_date, Owner_status, Reg_no, Supplier_id)
6. Supplier (Supplier_id, Supplier_name, Supplier_email, Supplier_phone)
7. Owner_details (Owner_email_id, Owner_name, Owner_phone_no, Address)
8. Cost (Order_id, Amount_payable, Bill_status, Owner_email_id)
9. Appointment (Owner_email_id, Date, Time, Type)

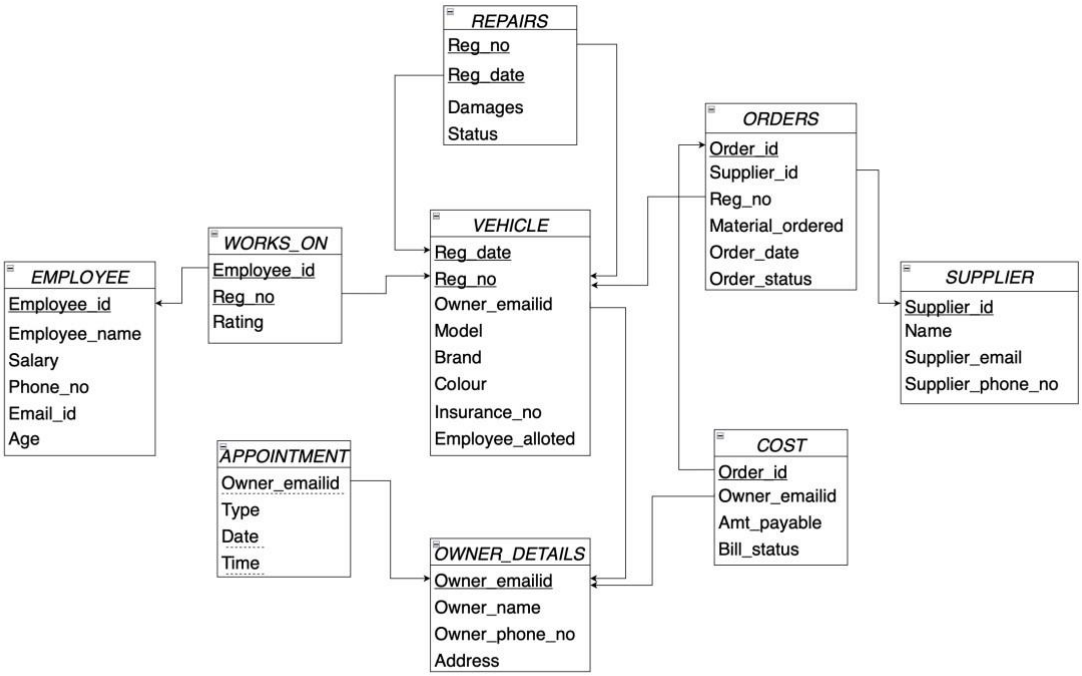

Figure 4.2 Schema Diagram

4.3 Data Dictionary

EMPLOYEE

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Employee_id	Char(4)	Primary key 'E___'	pk_empid, empid_startswith_e
Emp_name	Varchar2(15)	Not null	
Salary	Number(6)	>0	check_valid_salary
Phone_no	Number(10)	Unique	unq_emp_phone
Email_id	Varchar2(15)	Contains '@' Ends with '.com'	valid_email_id
Age	Number(2)	Between 18 and 65	emp_age

WORKS_ON

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Employee_id	Varchar2(4)	References EMPLOYEE	fk_empid
Reg_no	Varchar2(10)	References VEHICLE	fk_vehicle
Rating	Number(2)	Between 0 and 10	check_valid_rating
Reg_date	Date	References Vehicle	

REPAIRS

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Reg_no	Varchar2(10)	Primary key References VEHICLE	fk_vehicle_repair_regno
Damages	Varchar2(20)		
Status	Char(1)	'0' or '1'	
Reg_date	Date	Primary key References VEHICLE	

VEHICLE

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Reg_no	char(10)	Primary key	vehicle_pks
Reg_date	Date	Primary key	
Model	Varchar2(15)		
Brand	Varchar2(15)		
Insurance_no	Varchar2(15)	Unique	unq_insurance
Colour	Varchar2(15)		
Owner_emailid	Varchar2(15)	References OWNER_DETAILS	fk_email_id
Employee_alloted	Varchar2(15)		

SUPPLIER

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Supplier_id	Char(4)	Primary key 'S___'	pk_supp_id starts_with_s
Name	Varchar2(15)		
Supplier_email	Varchar2(15)		
Supplier_phone_no	Number(10)	Unique	phone_unq

COST

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Order_id	Varchar2(5)	Primary key Foreign key referencing ORDERS	cost_pk_owner_id fk_orders_order_id
Amt_payable	Number(5)	>0	
Bill_status	Varchar2(6)	'paid' or 'unpaid'	check_bill_status
Owner_emailid	Varchar2(15)	Foreign key referencing OWNER_DETAILS	fk_cost_owner_email

ORDERS

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Order_id	Varchar2(5)	Primary key	pk_order_id
Reg_no	Varchar2(10)	References VEHICLE	fk_orders_vehicle
Supplier_id	Varchar2(3)	References SUPPLIER	fk_orders_supp_id
Material ordered	Varchar2(20)		
Order_date	Date		
Order_status	char(1)	'0' or '1'	check_order_status

OWNER_DETAILS

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Owner_emailid	Varchar2(15)	Primary key	pk_owner_email
Owner_name	Varchar2(15)		
Owner_phone_no	Number(10)	Unique	unq_owner_phone
Address	Varchar2(50)		

APPOINTMENT

COLUMN NAME	DATA TYPE(SIZE)	CONSTRAINT	CONSTRAINT NAME
Owner_emailid	Varchar2(15)	Primary key	

Type	Varchar2(1)	‘Home’ or ‘Garage’	check_appointment_type
Date	Date	Primary key	
Time	Number(2)	Primary key	

4.4 Relational Model Implementation

4.3.1 Create Commands

```
CREATE TABLE EMPLOYEE(
EMPLOYEE_ID CHAR(4) CONSTRAINT PK_EMPID PRIMARY KEY CONSTRAINT
EMPID_STARTSWITH_E CHECK(EMPLOYEE_ID LIKE 'E___'),
EMP_NAME VARCHAR2(15) NOT NULL,
SALARY NUMBER(7) CONSTRAINT CHECK_VALID_SALARY CHECK(SALARY > 0),
PHONE_NO NUMBER(10) CONSTRAINT UNQ_EMP_PHONE UNIQUE,
EMAIL_ID VARCHAR2(15) CONSTRAINT VALID_EMAIL_ID CHECK(EMAIL_ID LIKE '%@%' AND
EMAIL_ID LIKE '%.com),
AGE NUMBER(2) CONSTRAINT EMP_AGE CHECK(AGE BETWEEN 18 AND 65)
);

CREATE TABLE OWNER_DETAILS(
OWNER_EMAILID VARCHAR2(15) CONSTRAINT PK_OWNER_EMAIL PRIMARY KEY,
OWNER_NAME VARCHAR2(15),
OWNER_PHONE_NO NUMBER(10) CONSTRAINT UNQ_OWNER_PHONE UNIQUE, ADDRESS
VARCHAR2(50)
);

CREATE TABLE VEHICLE(
REG_NO VARCHAR2(10),
REG_DATE DATE,
MODEL VARCHAR2(15),
BRAND VARCHAR2(15),
NSURANCE_NO VARCHAR2(15) CONSTRAINT UNQ_INSURANCE UNIQUE,
COLOUR VARCHAR2(15),
EMPLOYEE_ALLOCATED VARCHAR2(15),
OWNER_EMAIL_ID VARCHAR2(15) CONSTRAINT FK_EMAIL_ID REFERENCES
OWNER_DETAILS,
CONSTRAINT PK_REGNO_REGDATE PRIMARY KEY(REG_NO,REG_DATE)
);


CREATE TABLE WORKS_ON(
EMPLOYEE_ID CHAR(4) CONSTRAINT FK_EMPID REFERENCES EMPLOYEE(EMPLOYEE_ID),
REG_NO VARCHAR2(10),
REG_DATE DATE,
RATING NUMBER(2) CONSTRAINT CHECK_VALID_RATING CHECK(RATING BETWEEN 0
AND 10),
CONSTRAINT FK_VEHICLE FOREIGN KEY (REG_NO, REG_DATE) REFERENCES
VEHICLE(REG_NO, REG_DATE),
PRIMARY KEY(EMPLOYEE_ID, REG_NO, REG_DATE)
);
```

```
CREATE TABLE REPAIRS(
REG_NO VARCHAR2(10),
DAMAGES VARCHAR2(20),
STATUS CHAR(1) CHECK(STATUS IN ('1','0')),
REG_DATE DATE,
PRIMARY KEY(REG_NO,REG_DATE),
CONSTRAINT FK_VEHICLE_REPAIR FOREIGN KEY (REG_NO, REG_DATE) REFERENCES
VEHICLE(REG_NO, REG_DATE)
);

CREATE TABLE SUPPLIER(
SUPPLIER_ID CHAR(4) CONSTRAINT PK_SUPP_ID PRIMARY KEY CONSTRAINT
START_WITH_S CHECK(SUPPLIER_ID LIKE 'S___'),
SUPPLIER_NAME VARCHAR2(15),
SUPPLIER_EMAIL VARCHAR2(15),
SUPPLIER_PHONE_NO NUMBER(10) CONSTRAINT PHONE_UNQ UNIQUE
);

CREATE TABLE ORDERS(
ORDER_ID VARCHAR2(5) CONSTRAINT PK_ORDER_ID PRIMARY KEY,
MATERIAL_ORDERED VARCHAR2(20),
ORDER_DATE DATE,
ORDER_STATUS CHAR(1) CONSTRAINT CHECK_ORDER_STATUS CHECK(ORDER_STATUS IN
('0','1')),
REG_NO VARCHAR2(10),
SUPPLIER_ID CHAR(4) CONSTRAINT FK_ORDER_SUPP REFERENCES SUPPLIER
);

CREATE TABLE COST(
ORDER_ID VARCHAR2(5) CONSTRAINT FK_ORDER REFERENCES ORDERS(ORDER_ID),
AMT_PAYABLE NUMBER(5) CHECK(AMT_PAYABLE>0),
BILL_STATUS VARCHAR2(6) CONSTRAINT CHECK_BILL_STATUS CHECK(BILL_STATUS IN
('PAID','UNPAID')),
OWNER_EMAIL_ID VARCHAR(15) CONSTRAINT FK_COST_OWNER_EMAIL REFERENCES
OWNER_DETAILS
);

CREATE TABLE APPOINTMENT(
OWNER_EMAIL_ID VARCHAR(15),
TYPE VARCHAR2(10) CONSTRAINT CHECK_APPOINMENTS_TYPE CHECK(TYPE IN
('HOME','GARAGE')),
APPOINTMENT_DATE DATE,
TIME NUMBER(2),
PRIMARY KEY(OWNER_EMAIL_ID,APPOINTMENT_DATE,TIME)
);
```

4.3.2 Insert Statements

```
INSERT ALL
INTO EMPLOYEE  VALUES ('E100', 'Amit Kumar', 50000, 9876543210, 'amit@abc.com', 35)
INTO EMPLOYEE  VALUES ('E101', 'Rahul Sharma', 60000, 8765432109, 'rahul@abc.com', 28)
INTO EMPLOYEE  VALUES ('E102', 'Priya Singh', 55000, 7654321098, 'priya@abc.com', 30)
```

```
INTO EMPLOYEE  VALUES ('E103', 'Sneha Patel', 52000, 6543210987, 'sneha@abc.com', 32)
INTO EMPLOYEE  VALUES ('E104', 'Ankit Gupta', 48000, 5432109876, 'ankit@abc.com', 25)
INTO EMPLOYEE  VALUES ('E105', 'Divya Verma', 53000, 4321098765, 'divya@abc.com', 29)
INTO EMPLOYEE  VALUES ('E106', 'Ravi Reddy', 51000, 3210987654, 'ravi@abc.com', 33)
INTO EMPLOYEE  VALUES ('E107', 'Neha Jain', 59000, 2109876543, 'neha@abc.com', 31)
INTO EMPLOYEE  VALUES ('E108', 'Rajesh Kumar', 54000, 1098765432, 'rajesh@abc.com', 27)
INTO EMPLOYEE  VALUES ('E109', 'Pooja Rani', 40000, 9170543211, 'rani@abc.com', 32)
INTO EMPLOYEE  VALUES ('E110', 'Pooja Dwarkani', 59000, 9176543211, 'dwarka@abc.com', 30)
SELECT * FROM DUAL;


INSERT ALL
INTO OWNER_DETAILS VALUES ('owner1@abc.com', 'Vikram Singh', 1111111111, '123, ABC Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner2@abc.com', 'Anjali Patel', 2222222222, '456, DEF Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner3@abc.com', 'Rahul Verma', 3333333333, '789, GHI Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner4@abc.com', 'Sunita Gupta', 4444444444, '101, JKL Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner5@abc.com', 'Ajay Kumar', 5555555555, '234, MNO Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner6@abc.com', 'Sneha Reddy', 6666666666, '567, PQR Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner7@abc.com', 'Rajesh Jain', 7777777777, '890, STU Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner8@abc.com', 'Pooja Sharma', 8888888888, '123, VWX Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner9@abc.com', 'Amit Dubey', 9999999999, '456, YZA Street, XYZ City')
INTO OWNER_DETAILS VALUES ('owner10@abc.com', 'Deepak Verma', 0000000000, '789, BCD Street, XYZ City')
SELECT * FROM DUAL;


INSERT ALL
INTO VEHICLE VALUES ('TN01AB1234', TO_DATE('2024-04-01', 'YYYY-MM-DD'), 'SUV', 'Toyota', 'INS123456', 'Red', 'Amit Kumar', 'owner1@abc.com')
INTO VEHICLE VALUES ('KA02CD5678', TO_DATE('2024-04-02', 'YYYY-MM-DD'), 'Hatchback', 'Maruti', 'INS234567', 'Blue', 'Rahul Sharma', 'owner2@abc.com')
INTO VEHICLE VALUES ('MH03EF9012', TO_DATE('2024-04-03', 'YYYY-MM-DD'), 'Sedan', 'Honda', 'INS345678', 'White', 'Priya Singh', 'owner3@abc.com')
INTO VEHICLE VALUES ('DL04GH3456', TO_DATE('2024-04-04', 'YYYY-MM-DD'), 'SUV', 'Ford', 'INS456789', 'Black', 'Sneha Patel', 'owner4@abc.com')
INTO VEHICLE VALUES ('UP05IJ7890', TO_DATE('2024-04-05', 'YYYY-MM-DD'), 'Hatchback', 'Hyundai', 'INS567890', 'Silver', 'Ankit Gupta', 'owner5@abc.com')
INTO VEHICLE VALUES ('RJ06KL1234', TO_DATE('2024-04-06', 'YYYY-MM-DD'), 'Sedan', 'Tata', 'INS678901', 'Gray', 'Divya Verma', 'owner6@abc.com')
INTO VEHICLE VALUES ('MP07MN5678', TO_DATE('2024-04-07', 'YYYY-MM-DD'), 'SUV', 'Chevrolet', 'INS789012', 'Brown', 'Ravi Reddy', 'owner7@abc.com')
INTO VEHICLE VALUES ('GJ08OP9012', TO_DATE('2024-04-08', 'YYYY-MM-DD'), 'Hatchback', 'Volkswagen', 'INS890123', 'Green', 'Neha Jain', 'owner8@abc.com')
INTO VEHICLE VALUES ('BR09QR3456', TO_DATE('2024-04-09', 'YYYY-MM-DD'), 'Sedan', 'Renault', 'INS901234', 'Yellow', 'Rajesh Kumar', 'owner9@abc.com')
```

```
INTO VEHICLE VALUES ('WB10ST7890', TO_DATE('2024-04-10', 'YYYY-MM-DD'), 'Hatchback', 'Mercedes', 'INS012345', 'Orange', 'Pooja Sharma', 'owner10@abc.com')
INTO VEHICLE VALUES ('TN01AB1234', TO_DATE('2024-05-01', 'YYYY-MM-DD'), 'SUV', 'Toyota', 'INS123457', 'Red', 'Rahul Sharma', 'owner1@abc.com')
INTO VEHICLE VALUES ('KA02CD5678', TO_DATE('2024-06-02', 'YYYY-MM-DD'), 'Hatchback', 'Maruti', 'INS234566', 'Blue', 'Amit Kumar', 'owner2@abc.com')
INTO VEHICLE VALUES ('MH03EF9012', TO_DATE('2024-07-03', 'YYYY-MM-DD'), 'Sedan', 'Honda', 'INS345679', 'White', 'Ravi Reddy', 'owner3@abc.com')
INTO VEHICLE VALUES ('DL04GH3456', TO_DATE('2024-05-04', 'YYYY-MM-DD'), 'SUV', 'Ford', 'INS456790', 'Black', 'Sneha Patel', 'owner4@abc.com')
INTO VEHICLE VALUES ('UP05IJ7890', TO_DATE('2024-06-05', 'YYYY-MM-DD'), 'Hatchback', 'Hyundai', 'INS567880', 'Silver', 'Ankit Gupta', 'owner5@abc.com')
INTO VEHICLE VALUES ('RJ06KL1234', TO_DATE('2024-07-06', 'YYYY-MM-DD'), 'Sedan', 'Tata', 'INS678911', 'Gray', 'Divya Verma', 'owner6@abc.com')
INTO VEHICLE VALUES ('MP07MN5678', TO_DATE('2024-08-07', 'YYYY-MM-DD'), 'SUV', 'Chevrolet', 'INS789011', 'Brown', 'Priya Singh', 'owner7@abc.com')
INTO VEHICLE VALUES ('GJ08OP9012', TO_DATE('2024-09-08', 'YYYY-MM-DD'), 'Hatchback', 'Volkswagen', 'INS890128', 'Green', 'Rajesh Kumar', 'owner8@abc.com')
INTO VEHICLE VALUES ('BR09QR3456', TO_DATE('2024-10-09', 'YYYY-MM-DD'), 'Sedan', 'Renault', 'INS901233', 'Yellow', 'Neha Jain', 'owner9@abc.com')
INTO VEHICLE VALUES ('WB10ST7890', TO_DATE('2024-12-10', 'YYYY-MM-DD'), 'Hatchback', 'Mercedes', 'INS012349', 'Orange', 'Pooja Sharma', 'owner10@abc.com')
SELECT * FROM DUAL;


INSERT ALL
INTO WORKS_ON VALUES ('E100', 'TN01AB1234', TO_DATE('2024-04-01', 'YYYY-MM-DD'), 9)
INTO WORKS_ON VALUES ('E101', 'KA02CD5678', TO_DATE('2024-04-02', 'YYYY-MM-DD'), 8)
INTO WORKS_ON VALUES ('E102', 'MH03EF9012', TO_DATE('2024-04-03', 'YYYY-MM-DD'), 7)
INTO WORKS_ON VALUES ('E103', 'DL04GH3456', TO_DATE('2024-04-04', 'YYYY-MM-DD'), 8)
INTO WORKS_ON VALUES ('E104', 'UP05IJ7890', TO_DATE('2024-04-05', 'YYYY-MM-DD'), 9)
INTO WORKS_ON VALUES ('E105', 'RJ06KL1234', TO_DATE('2024-04-06', 'YYYY-MM-DD'), 7)
INTO WORKS_ON VALUES ('E106', 'MP07MN5678', TO_DATE('2024-04-07', 'YYYY-MM-DD'), 8)
INTO WORKS_ON VALUES ('E107', 'GJ08OP9012', TO_DATE('2024-04-08', 'YYYY-MM-DD'), 9)
INTO WORKS_ON VALUES ('E108', 'BR09QR3456', TO_DATE('2024-04-09', 'YYYY-MM-DD'), 8)
INTO WORKS_ON VALUES ('E109', 'WB10ST7890', TO_DATE('2024-04-10', 'YYYY-MM-DD'), 9)
INTO WORKS_ON VALUES ('E100', 'TN01AB1234', TO_DATE('2024-05-01', 'YYYY-MM-DD'), 6)
INTO WORKS_ON VALUES ('E101', 'KA02CD5678', TO_DATE('2024-06-02', 'YYYY-MM-DD'), 7)
INTO WORKS_ON VALUES ('E102', 'MH03EF9012', TO_DATE('2024-07-03', 'YYYY-MM-DD'), 5)
INTO WORKS_ON VALUES ('E103', 'DL04GH3456', TO_DATE('2024-05-04', 'YYYY-MM-DD'), 9)
INTO WORKS_ON VALUES ('E104', 'UP05IJ7890', TO_DATE('2024-06-05', 'YYYY-MM-DD'), 7)
INTO WORKS_ON VALUES ('E105', 'RJ06KL1234', TO_DATE('2024-07-06', 'YYYY-MM-DD'), 5)
INTO WORKS_ON VALUES ('E106', 'MP07MN5678', TO_DATE('2024-08-07', 'YYYY-MM-DD'), 7)
INTO WORKS_ON VALUES ('E107', 'GJ08OP9012', TO_DATE('2024-09-08', 'YYYY-MM-DD'), 8)
INTO WORKS_ON VALUES ('E108', 'BR09QR3456', TO_DATE('2024-10-09', 'YYYY-MM-DD'), 2)
INTO WORKS_ON VALUES ('E109', 'WB10ST7890', TO_DATE('2024-12-10', 'YYYY-MM-DD'), 7)
SELECT * FROM DUAL;

INSERT ALL
INTO REPAIRS VALUES ('TN01AB1234', 'Minor Scratches', '1', TO_DATE('2024-04-01', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('KA02CD5678', 'Engine Issue', '0', TO_DATE('2024-04-02', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('MH03EF9012', 'Flat Tire', '1', TO_DATE('2024-04-03', 'YYYY-MM-DD'))
INTO REPAIRS VALUES ('DL04GH3456', 'Brake Problem', '0', TO_DATE('2024-04-04', 'YYYY-MMDD'))
```

```
INTO REPAIRS VALUES ('UP05IJ7890', 'Battery Dead', '1', TO_DATE('2024-04-05', 'YYYY-MM-DD'))
INTO REPAIRS VALUES ('RJ06KL1234', 'Headlight Broken', '0', TO_DATE('2024-04-06', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('MP07MN5678', 'AC Not Working', '1', TO_DATE('2024-04-07', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('GJ08OP9012', 'Scratches on Door', '0', TO_DATE('2024-04-08', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('BR09QR3456', 'Dent on Bonnet', '1', TO_DATE('2024-04-09', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('WB10ST7890', 'Broken Mirror', '0', TO_DATE('2024-04-10', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('TN01AB1234', 'Minor Scratches', '0', TO_DATE('2024-05-01', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('KA02CD5678', 'Flat Tyre', '1', TO_DATE('2024-06-02', 'YYYY-MM-DD'))
INTO REPAIRS VALUES ('MH03EF9012', 'Engine Issue', '0', TO_DATE('2024-07-03', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('DL04GH3456', 'Brake Problem', '0', TO_DATE('2024-05-04', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('UP05IJ7890', 'Battery Dead', '1', TO_DATE('2024-06-05', 'YYYY-MM-DD'))
INTO REPAIRS VALUES ('RJ06KL1234', 'Headlight Broken', '1', TO_DATE('2024-07-06', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('MP07MN5678', 'AC Not Working', '0', TO_DATE('2024-08-07', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('GJ08OP9012', 'Scratches on Door', '1', TO_DATE('2024-09-08', 'YYYY-MMDD'))
INTO REPAIRS VALUES ('BR09QR3456', 'Flat tyre', '0', TO_DATE('2024-10-09', 'YYYY-MM-DD'))
INTO REPAIRS VALUES ('WB10ST7890', 'Broken Mirror', '1', TO_DATE('2024-12-10', 'YYYY-MMDD'))
SELECT * FROM DUAL;


INSERT ALL
INTO SUPPLIER VALUES ('S100', 'ABC Parts', 'abc@abc.com', 1212121212)
INTO SUPPLIER VALUES ('S101', 'XYZ Motors', 'xyz@abc.com', 2323232323)
INTO SUPPLIER VALUES ('S102', 'PQR Acc', 'pqr@abc.com', 4545454545)
INTO SUPPLIER VALUES ('S103', 'LMN Services', 'lmn@abc.com', 6767676767)
INTO SUPPLIER VALUES ('S104', 'DEF Supplies', 'def@abc.com', 8989898989)
INTO SUPPLIER VALUES ('S105', 'GHI Solutions', 'ghi@abc.com', 0909090909)
INTO SUPPLIER VALUES ('S106', 'MNO Tools', 'mno@abc.com', 1313131313)
INTO SUPPLIER VALUES ('S107', 'UVW Tech', 'uvw@abc.com', 4343434343)
INTO SUPPLIER VALUES ('S108', 'QRS Parts', 'qrs@abc.com', 5454545454)
INTO SUPPLIER VALUES ('S109', 'JKL Motors', 'jkl@abc.com', 6565656565)
SELECT * FROM DUAL;

INSERT ALL
INTO ORDERS VALUES ('ORD1', 'Brake Pads', TO_DATE('2024-04-01', 'YYYY-MM-DD'), '1', 'TN01AB1234', 'S100')
INTO ORDERS VALUES ('ORD2', 'Engine Oil', TO_DATE('2024-04-02', 'YYYY-MM-DD'), '0', 'KA02CD5678', 'S101')
INTO ORDERS VALUES ('ORD3', 'Tire Tubes', TO_DATE('2024-04-03', 'YYYY-MM-DD'), '1', 'MH03EF9012', 'S102')
INTO ORDERS VALUES ('ORD4', 'Battery', TO_DATE('2024-04-04', 'YYYY-MM-DD'), '0', 'DL04GH3456', 'S103')
INTO ORDERS VALUES ('ORD5', 'Headlights', TO_DATE('2024-04-05', 'YYYY-MM-DD'), '1', 'UP05IJ7890', 'S104')
INTO ORDERS VALUES ('ORD6', 'AC Compressor', TO_DATE('2024-04-06', 'YYYY-MM-DD'), '0', 'RJ06KL1234', 'S105')
INTO ORDERS VALUES ('ORD7', 'Brake Discs', TO_DATE('2024-04-07', 'YYYY-MM-DD'), '1', 'MP07MN5678', 'S106')
```

```
INTO ORDERS VALUES ('ORD8', 'Mirrors', TO_DATE('2024-04-08', 'YYYY-MM-DD'), '0', 'GJ08OP9012', 'S107')
INTO ORDERS VALUES ('ORD9', 'Filters', TO_DATE('2024-04-09', 'YYYY-MM-DD'), '1', 'BR09QR3456', 'S108')
INTO ORDERS VALUES ('ORD10', 'Wipers', TO_DATE('2024-04-10', 'YYYY-MM-DD'), '0', 'WB10ST7890', 'S109')
INTO ORDERS VALUES ('ORD11', 'Brake Pads', TO_DATE('2024-05-01', 'YYYY-MM-DD'), '0', 'TN01AB1234', 'S100')
INTO ORDERS VALUES ('ORD12', 'Engine Oil', TO_DATE('2024-06-02', 'YYYY-MM-DD'), '1', 'KA02CD5678', 'S101')
INTO ORDERS VALUES ('ORD13', 'Tire Tubes', TO_DATE('2024-07-03', 'YYYY-MM-DD'), '0', 'MH03EF9012', 'S102')
INTO ORDERS VALUES ('ORD14', 'Battery', TO_DATE('2024-05-14', 'YYYY-MM-DD'), '1', 'DL04GH3456', 'S103')
INTO ORDERS VALUES ('ORD15', 'Headlights', TO_DATE('2024-06-05', 'YYYY-MM-DD'), '0', 'UP05IJ7890', 'S104')
INTO ORDERS VALUES ('ORD16', 'AC Compressor', TO_DATE('2024-08-06', 'YYYY-MM-DD'), '1', 'RJ06KL1234', 'S105')
INTO ORDERS VALUES ('ORD17', 'Brake Discs', TO_DATE('2024-09-07', 'YYYY-MM-DD'), '0', 'MP07MN5678', 'S106')
INTO ORDERS VALUES ('ORD18', 'Mirrors', TO_DATE('2024-10-10', 'YYYY-MM-DD'), '1', 'GJ08OP9012', 'S107')
INTO ORDERS VALUES ('ORD19', 'Filters', TO_DATE('2024-10-09', 'YYYY-MM-DD'), '0', 'BR09QR3456', 'S108')
INTO ORDERS VALUES ('ORD20', 'Wipers', TO_DATE('2025-01-01', 'YYYY-MM-DD'), '1', 'WB10ST7890', 'S109')
SELECT * FROM DUAL;

INSERT ALL
INTO COST VALUES ('ORD1', 2000, 'PAID', 'owner1@abc.com')
INTO COST VALUES ('ORD2', 1500, 'UNPAID', 'owner2@abc.com')
INTO COST VALUES ('ORD3', 1800, 'PAID', 'owner3@abc.com')
INTO COST VALUES ('ORD4', 1200, 'UNPAID', 'owner4@abc.com')
INTO COST VALUES ('ORD5', 2500, 'PAID', 'owner5@abc.com')
INTO COST VALUES ('ORD6', 3000, 'UNPAID', 'owner6@abc.com')
INTO COST VALUES ('ORD7', 2200, 'PAID', 'owner7@abc.com')
INTO COST VALUES ('ORD8', 1700, 'UNPAID', 'owner8@abc.com')
INTO COST VALUES ('ORD9', 1900, 'PAID', 'owner9@abc.com')
INTO COST VALUES ('ORD10', 2800, 'UNPAID', 'owner10@abc.com')
INTO COST VALUES ('ORD11', 2000, 'UNPAID', 'owner1@abc.com')
INTO COST VALUES ('ORD12', 1500, 'PAID', 'owner2@abc.com')
INTO COST VALUES ('ORD13', 1800, 'UNPAID', 'owner3@abc.com')
INTO COST VALUES ('ORD14', 1200, 'PAID', 'owner4@abc.com')
INTO COST VALUES ('ORD15', 2500, 'UNPAID', 'owner5@abc.com')
INTO COST VALUES ('ORD16', 3000, 'PAID', 'owner6@abc.com')
INTO COST VALUES ('ORD17', 2200, 'PAID', 'owner7@abc.com')
INTO COST VALUES ('ORD18', 1700, 'PAID', 'owner8@abc.com')
INTO COST VALUES ('ORD19', 1900, 'UNPAID', 'owner9@abc.com')
INTO COST VALUES ('ORD20', 2800, 'UNPAID', 'owner10@abc.com')
SELECT * FROM DUAL;

INSERT ALL
INTO APPOINTMENT VALUES ('owner1@abc.com', 'HOME', TO_DATE('2024-04-01', 'YYYY-MM-DD'), 10)
```

```
INTO APPOINTMENT VALUES ('owner2@abc.com', 'GARAGE', TO_DATE('2024-04-02', 'YYYY-MM-DD'), 11)
INTO APPOINTMENT VALUES ('owner3@abc.com', 'HOME', TO_DATE('2024-04-03', 'YYYY-MM-DD'), 12)
INTO APPOINTMENT VALUES ('owner4@abc.com', 'GARAGE', TO_DATE('2024-04-04', 'YYYY-MM-DD'), 13)
INTO APPOINTMENT VALUES ('owner5@abc.com', 'HOME', TO_DATE('2024-04-05', 'YYYY-MM-DD'), 14)
INTO APPOINTMENT VALUES ('owner6@abc.com', 'GARAGE', TO_DATE('2024-04-06', 'YYYY-MM-DD'), 15)
INTO APPOINTMENT VALUES ('owner7@abc.com', 'HOME', TO_DATE('2024-04-07', 'YYYY-MM-DD'), 16)
INTO APPOINTMENT VALUES ('owner8@abc.com', 'GARAGE', TO_DATE('2024-04-08', 'YYYY-MM-DD'), 17)
INTO APPOINTMENT VALUES ('owner9@abc.com', 'HOME', TO_DATE('2024-04-09', 'YYYY-MM-DD'), 18)
INTO APPOINTMENT VALUES ('owner10@abc.com', 'GARAGE', TO_DATE('2024-04-10', 'YYYYMM-DD'), 19)
INTO APPOINTMENT VALUES ('owner1@abc.com', 'GARAGE', TO_DATE('2024-05-01', 'YYYY-MM-DD'), 10)
INTO APPOINTMENT VALUES ('owner2@abc.com', 'HOME', TO_DATE('2024-06-02', 'YYYY-MM-DD'), 11)
INTO APPOINTMENT VALUES ('owner3@abc.com', 'GARAGE', TO_DATE('2024-07-03', 'YYYY-MM-DD'), 12)
INTO APPOINTMENT VALUES ('owner4@abc.com', 'HOME', TO_DATE('2024-05-04', 'YYYY-MM-DD'), 13)
INTO APPOINTMENT VALUES ('owner5@abc.com', 'GARAGE', TO_DATE('2024-06-05', 'YYYY-MM-DD'), 14)
INTO APPOINTMENT VALUES ('owner6@abc.com', 'HOME', TO_DATE('2024-07-06', 'YYYY-MM-DD'), 15)
INTO APPOINTMENT VALUES ('owner7@abc.com', 'GARAGE', TO_DATE('2024-08-07', 'YYYY-MM-DD'), 16)
INTO APPOINTMENT VALUES ('owner8@abc.com', 'HOME', TO_DATE('2024-09-08', 'YYYY-MM-DD'), 17)
INTO APPOINTMENT VALUES ('owner9@abc.com', 'HOME', TO_DATE('2024-10-09', 'YYYY-MM-DD'), 18)
INTO APPOINTMENT VALUES ('owner10@abc.com', 'GARAGE', TO_DATE('2024-12-10', 'YYYYMM-DD'), 19)
SELECT * FROM DUAL;
```

Chapter 5- **Implementation**

5.1 Queries

5.1.1. Find all employees who have worked on vehicles that came to the garage between 03-april-2024 and 06-april-2024.

```
SELECT EMP_NAME FROM EMPLOYEE WHERE EMPLOYEE_ID IN (SELECT EMPLOYEE_ID FROM WORKS_ON WHERE REG_NO IN (SELECT REG_NO FROM VEHICLE WHERE REG_DATE BETWEEN TO_DATE('03-APR-2024','DD-MON-YYYY') AND TO_DATE('06-APRIL-2024','DD-MONYYYY')));
```

5.1.2. Find the total amount payable of every customer and the amount already paid.

```
WITH TABLE1 AS (
   SELECT OWNER_EMAIL_ID, SUM(AMT_PAYABLE) AS PAIDAMT
   FROM COST
   WHERE BILL_STATUS = 'PAID'
   GROUP BY OWNER_EMAIL_ID),
TABLE2 AS (
   SELECT OWNER_EMAIL_ID, SUM(AMT_PAYABLE) AS TOTALAMT
   FROM COST
   GROUP BY OWNER_EMAIL_ID)
SELECT T1.OWNER_EMAIL_ID, T1.PAIDAMT, T2.TOTALAMT, T2.TOTALAMT-T1.PAIDAMT AS PENDING_AMOUNT
FROM TABLE1 T1 JOIN TABLE2 T2 ON T1.OWNER_EMAIL_ID = T2.OWNER_EMAIL_ID;
```

5.1.3. Find the rating of each employee.

```
SELECT EMPLOYEE_ID, AVG(RATING) FROM WORKS_ON GROUP BY EMPLOYEE_ID;
```

5.1.4. Find the supplier who provided the parts for vehicle on which employee with employee id 100 has worked.

```
SELECT DISTINCT(SUPPLIER_NAME) FROM SUPPLIER NATURAL JOIN ORDERS O JOIN WORKS_ON W ON O.REG_NO=W.REG_NO WHERE EMPLOYEE_ID = 'E101';
```

5.1.5. Find the description of all pending repairs.

```
SELECT REG_NO, DAMAGES, EMPLOYEE_ALLOCATED FROM VEHICLE NATURAL JOIN REPAIRS WHERE STATUS=0;
```

5.1.6. Find the employees with rating>8

```
SELECT EMP_NAME, EMPLOYEE_ID, AVG(RATING) FROM WORKS_ON NATURAL JOIN EMPLOYEE GROUP BY EMP_NAME,EMPLOYEE_ID HAVING AVG(WORKS_ON.RATING)>8;
```

5.1.7. Display the detailed bill of the owner with owner_email_id=’owner1@abc.com’

```
SELECT ORDER_ID, MATERIAL_ORDERED, AMT_PAYABLE FROM ORDERS NATURAL JOIN
COST WHERE BILL_STATUS='UNPAID' AND OWNER_EMAIL_ID='OWNER1@ABC.COM';
```

5.2 Triggers

5.2.1. A customer gets 10% discount on every 5th visit to the garage. Write a trigger to automatically reduce cost of repairs by 10% on every 5th visit to the garage.

```
CREATE OR REPLACE TRIGGER TRG_DISCOUNT_GARAGE_VISIT
AFTER INSERT ON APPOINTMENT
FOR EACH ROW
DECLARE
V_VISIT_COUNT NUMBER;
BEGIN
IF :NEW.TYPE = 'GARAGE' THEN
SELECT COUNT(*)
INTO V_VISIT_COUNT
FROM APPOINTMENT
WHERE OWNER_EMAIL_ID = :NEW.OWNER_EMAIL_ID
AND TYPE = 'GARAGE';
IF MOD(V_VISIT_COUNT, 2) = 0 THEN
UPDATE COST
SET AMT_PAYABLE = AMT_PAYABLE * 0.9
WHERE OWNER_EMAIL_ID = :NEW.OWNER_EMAIL_ID AND BILL_STATUS = 'UNPAID'; END
IF;
END IF;
END;
/
```

5.2.2. Whenever an order has been made, insert the cost values into the cost table automatically by asking the amount as input itself.

```
CREATE OR REPLACE TRIGGER trg_insert_cost
AFTER INSERT ON ORDERS
FOR EACH ROW DECLARE
v_amt_payable NUMBER;
v_email VARCHAR2(50); BEGIN
v_amt_payable := costing_for_order();
SELECT OWNER_EMAIL_ID INTO v_email FROM VEHICLE WHERE REG_NO= :NEW.REG_NO;
INSERT INTO COST (ORDER_ID, AMT_PAYABLE, BILL_STATUS, OWNER_EMAIL_ID) VALUES
(:NEW.ORDER_ID, v_amt_payable, 'UNPAID', v_email);
END;
/

CREATE OR REPLACE FUNCTION COSTING_FOR_ORDER
RETURN NUMBER AS
V_INPUT NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER THE AMOUNT OF ORDERS');
V_INPUT:=&V_INPUT;
```

```
RETURN V_INPUT;
END;
/
```

5.3 Stored Procedure

5.3.1. Write a procedure to take email id as input and display the previous damages on his/her vehicle.

```
CREATE OR REPLACE PROCEDURE DISPLAY_PREVIOUS_DAMAGES( P_OWNER_EMAIL IN
VARCHAR2) IS
CURSOR C_REG IS SELECT DISTINCT(REG_NO) FROM VEHICLE WHERE OWNER_EMAIL_ID =
P_OWNER_EMAIL;
BEGIN
FOR I IN C_REG LOOP
DBMS_OUTPUT.PUT_LINE('THE OWNER HAS CAR WITH REGISTRATION NUMBER ' || I.REG_NO
||':');
FOR J IN ( SELECT DAMAGES, STATUS FROM REPAIRS WHERE REG_NO = I.REG_NO) LOOP
DBMS_OUTPUT.PUT_LINE('THE DAMAGES ARE: ' || J.DAMAGES);
DBMS_OUTPUT.PUT_LINE('REPAIR STATUS: ' || CASE J.STATUS WHEN '1' THEN 'COMPLETED'
ELSE 'PENDING' END);
END LOOP;
END LOOP;
END;
/
```

5.4 Functions

5.4.1. Write a function to return the rating of an employee. Increment the salary of an employee by 15% if the employee has repaired 10 vehicles and has a rating>=8.

```
CREATE OR REPLACE FUNCTION GET_OVERALL_RATING(p_emp_id IN CHAR)
RETURN NUMBER
IS
V_TOTAL_RATING NUMBER := 0;
V_TOTAL_COUNT NUMBER := 0;
V_AVG_RATING NUMBER := 0;
BEGIN
SELECT SUM(RATING), COUNT(*)
INTO V_TOTAL_RATING, V_TOTAL_COUNT
FROM WORKS_ON
WHERE EMPLOYEE_ID = P_EMP_ID;
IF V_TOTAL_COUNT > 0 THEN
V_AVG_RATING := V_TOTAL_RATING / V_TOTAL_COUNT;
END IF;
RETURN V_AVG_RATING;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN NULL;
END;
/
CREATE OR REPLACE TRIGGER SALARY_INCREMENT
AFTER INSERT ON WORKS_ON
FOR EACH ROW
```

```
DECLARE
V_EMP_ID EMPLOYEE.EMPLOYEE_ID%TYPE;
V_OVERALL_RATING NUMBER;
V_SERVICE_COUNT NUMBER;
BEGIN
V_EMP_ID := :NEW.EMPLOYEE_ID;
V_OVERALL_RATING := GET_OVERALL_RATING(V_EMP_ID);
SELECT COUNT(*)
INTO V_SERVICE_COUNT
FROM WORKS_ON
WHERE EMPLOYEE_ID = V_EMP_ID;
IF V_OVERALL_RATING >= 8 AND MOD(V_SERVICE_COUNT, 10) = 0 THEN
UPDATE EMPLOYEE
SET SALARY = SALARY * 1.15
WHERE EMPLOYEE_ID = V_EMP_ID;
COMMIT;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
NULL;
END;
/
```

5.5 PL/SQL Block

5.5.1. Write a PL/SQL block to take owner details and vehicle details as input. If owner already exists, do not insert the values again into the owner_details table.

```
SET SERVEROUTPUT ON;
DECLARE
OEI VARCHAR2(15);
OWNERNAME VARCHAR2(15);
OWNERPHNO NUMBER(10);
ADDRESS VARCHAR2(50);
CNT NUMBER:=0;
REGNO VARCHAR2(10);
MODEL VARCHAR2(15);
BRAND VARCHAR2(15);
NSUR VARCHAR2(15);
COLOUR VARCHAR2(15);
BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER THE OWNER EMAIL ID'); OEI:=&OEI;
SELECT NVL(COUNT(*), 0) INTO CNT FROM OWNER_DETAILS WHERE OWNER_EMAILID =
OEI;IF CNT=1 THEN
NULL;
ELSE
OWNERNAME:=&OWNERNAME;
OWNERPHNO:=&OWNERPHNO;
ADDRESS:=&ADDRESS;
INSERT INTO OWNER_DETAILS VALUES (OEI, OWNERNAME, OWNERPHNO, ADDRESS); END
IF;
REGNO:=&ENTERREGNO;
MODEL:=& MODEL;
```

```
NSUR:=& NSUR;
COLOUR:=& COLOUR;
INSERT INTO VEHICLE VALUES(REGNO, TO_DATE(SYSDATE,'DD-MM-YYYY'), MODEL, BRAND,
NSUR, COLOUR, NULL, OEI);
END;
```

Chapter 6- **Results**

6.1.1.

```
SQL> SELECT EMP_NAME FROM EMPLOYEE WHERE EMPLOYEE_ID IN (SELECT EMPLOYEE_ID FROM WORKS_ON WHERE REG_NO IN (SELECT REG_NO FROM VEHICLE WHERE REG_DATE BETWEEN
 TO_DATE('03-APR-2024','DD-MON-YYYY') AND TO_DATE('06-APRIL-2024','DD-MON-YYYY')));

EMP_NAME
---------------
Priya Singh
Sneha Patel
Ankit Gupta
Divya Verma

SQL>
```

6.1.2.

```
SQL> WITH TABLE1 AS (
  2      SELECT OWNER_EMAIL_ID, SUM(AMT_PAYABLE) AS PAIDAMT
  3      FROM COST
  4      WHERE BILL_STATUS = 'PAID'
  5      GROUP BY OWNER_EMAIL_ID),
  6  TABLE2 AS (
  7      SELECT OWNER_EMAIL_ID, SUM(AMT_PAYABLE) AS TOTALAMT
  8      FROM COST
  9      GROUP BY OWNER_EMAIL_ID)
 10  SELECT T1.OWNER_EMAIL_ID, T1.PAIDAMT, T2.TOTALAMT, T2.TOTALAMT-T1.PAIDAMT AS PENDING_AMOUNT
 11  FROM TABLE1 T1 JOIN TABLE2 T2 ON T1.OWNER_EMAIL_ID = T2.OWNER_EMAIL_ID;

OWNER_EMAIL_ID     PAIDAMT   TOTALAMT PENDING_AMOUNT
--------------- ---------- ---------- --------------
owner1@abc.com        2000       4000           2000
owner2@abc.com        1500       3000           1500
owner3@abc.com        1800       3600           1800
owner4@abc.com        1200       2400           1200
owner5@abc.com        2500       5000           2500
owner6@abc.com        3000       6000           3000
owner7@abc.com        4400       4400              0
owner8@abc.com        1700       3400           1700
owner9@abc.com        1900       3800           1900

9 rows selected.
```

6.1.3.

```
SQL> SELECT EMPLOYEE_ID, AVG(RATING) FROM WORKS_ON GROUP BY EMPLOYEE_ID;

EMPL AVG(RATING)
---- -----------
E100         7.5
E101         7.5
E102           6
E103         8.5
E104           8
E105           6
E106         7.5
E107         8.5
E108           5
E109           8

10 rows selected.
```

6.1.4.

```
SQL> SELECT DISTINCT(SUPPLIER_NAME) FROM SUPPLIER NATURAL JOIN ORDERS O JOIN WORKS_ON W ON O.REG_NO=W.REG_NO WHERE EMPLO
YEE_ID = 'E101';

SUPPLIER_NAME
---------------
XYZ Motors
```

6.1.5.

```
SQL> SELECT REG_NO, DAMAGES, EMPLOYEE_ALLOCATED FROM VEHICLE NATURAL JOIN REPAIRS WHERE STATUS=0;

REG_NO     DAMAGES              EMPLOYEE_ALLOCA
---------- -------------------- ---------------
KA02CD5678 Engine Issue         Rahul Sharma
DL04GH3456 Brake Problem        Sneha Patel
RJ06KL1234 Headlight Broken     Divya Verma
GJ08OP9012 Scratches on Door    Neha Jain
WB10ST7890 Broken Mirror        Pooja Sharma
TN01AB1234 Minor Scratches      Rahul Sharma
MH03EF9012 Engine Issue         Ravi Reddy
DL04GH3456 Brake Problem        Sneha Patel
MP07MN5678 AC Not Working       Priya Singh
BR09QR3456 Flat tyre            Neha Jain

10 rows selected.
```

6.1.6.

```
SQL> SELECT EMP_NAME, EMPLOYEE_ID, AVG(RATING) FROM WORKS_ON NATURAL JOIN EMPLOYEE GROUP BY EMP_NAME,EMPLOYEE_ID HAVING
AVG(WORKS_ON.RATING)>8;

EMP_NAME        EMPL AVG(RATING)
--------------- ---- -----------
Sneha Patel     E103         8.5
Neha Jain       E107         8.5
```

6.1.7.

```
SQL> select order_id, material_ordered, amt_payable from orders natural join cost where bill_status='UNPAID' and owner_e
mail_id='owner1@abc.com';

ORDER MATERIAL_ORDERED     AMT_PAYABLE
----- -------------------- -----------
ORD11 Brake Pads                  2000
```

6.2.1.

```
SQL> CREATE OR REPLACE TRIGGER TRG_DISCOUNT_GARAGE_VISIT
  2  AFTER INSERT ON APPOINTMENT
  3  FOR EACH ROW
  4  DECLARE
  5      V_VISIT_COUNT NUMBER;
  6  BEGIN
  7      IF :NEW.TYPE = 'GARAGE' THEN
  8          SELECT COUNT(*)
  9          INTO V_VISIT_COUNT
 10          FROM APPOINTMENT
 11          WHERE OWNER_EMAIL_ID = :NEW.OWNER_EMAIL_ID
 12          AND TYPE = 'GARAGE';
 13          IF MOD(V_VISIT_COUNT, 5) = 0 THEN
 14              UPDATE COST
 15              SET AMT_PAYABLE = AMT_PAYABLE * 0.9
 16              WHERE OWNER_EMAIL_ID = :NEW.OWNER_EMAIL_ID
 17              AND BILL_STATUS = 'UNPAID';
 18          END IF;
 19      END IF;
 20  END;
 21  /

Trigger created.
```

6.2.2

```
SQL> Create or replace function costing_for_order
  2  Return number as
  3  v_input number;
  4  Begin
  5  Dbms_output.put_line('Enter the amount of orders');
  6  v_input:=&v_input;
  7  Return v_input;
  8  End;
  9  /
Enter value for v_input: 5
old   6: v_input:=&v_input;
new   6: v_input:=5;

Function created.

SQL> CREATE OR REPLACE TRIGGER trg_insert_cost
  2  AFTER INSERT ON ORDERS
  3  FOR EACH ROW
  4  DECLARE
  5      v_amt_payable NUMBER;
  6      v_email VARCHAR2(50);
  7  BEGIN
  8      v_amt_payable := costing_for_order();
  9      SELECT OWNER_EMAIL_ID INTO v_email FROM VEHICLE WHERE REG_NO= :NEW.REG_NO;
 10      INSERT INTO COST (ORDER_ID, AMT_PAYABLE, BILL_STATUS, OWNER_EMAIL_ID)
 11      VALUES (:NEW.ORDER_ID, v_amt_payable, 'UNPAID', v_email);
 12  END;
 13  /

Trigger created.
```

6.3.1.

```
SQL> set serveroutput on;
SQL> CREATE OR REPLACE PROCEDURE DISPLAY_PREVIOUS_DAMAGES ( P_OWNER_EMAIL IN VARCHAR2 )
  2  IS
  3      CURSOR C_REG IS SELECT DISTINCT(REG_NO) FROM VEHICLE WHERE OWNER_EMAIL_ID = P_OWNER_EMAIL;
  4  BEGIN
  5      FOR I IN C_REG LOOP
  6          DBMS_OUTPUT.PUT_LINE('THE OWNER HAS CAR WITH REGISTRATION NUMBER ' || I.REG_NO ||':');
  7
  8          FOR J IN (
  9              SELECT DAMAGES, STATUS
 10              FROM REPAIRS
 11              WHERE REG_NO = I.REG_NO
 12          )
 13          LOOP
 14              DBMS_OUTPUT.PUT_LINE('THE DAMAGES ARE: ' || J.DAMAGES);
 15              DBMS_OUTPUT.PUT_LINE('REPAIR STATUS: ' || CASE J.STATUS WHEN '1' THEN 'COMPLETED' ELSE 'PENDING' END);
 16          END LOOP;
 17      END LOOP;
 18  END;
 19  /

Procedure created.

SQL> exec display_previous_damages('owner2@abc.com');
THE OWNER HAS CAR WITH REGISTRATION NUMBER KA02CD5678:
THE DAMAGES ARE: Engine Issue
REPAIR STATUS: PENDING
THE DAMAGES ARE: Flat Tyre
REPAIR STATUS: COMPLETED

PL/SQL procedure successfully completed.
```

6.4.1.

```
SQL> CREATE OR REPLACE FUNCTION GET_OVERALL_RATING(p_emp_id IN CHAR)
  2  RETURN NUMBER
  3  IS
  4      V_TOTAL_RATING NUMBER := 0;
  5      V_TOTAL_COUNT NUMBER := 0;
  6      V_AVG_RATING NUMBER := 0;
  7  BEGIN
  8      SELECT SUM(RATING), COUNT(*)
  9      INTO V_TOTAL_RATING, V_TOTAL_COUNT
 10      FROM WORKS_ON
 11      WHERE EMPLOYEE_ID = P_EMP_ID;
 12      IF V_TOTAL_COUNT > 0 THEN
 13          V_AVG_RATING := V_TOTAL_RATING / V_TOTAL_COUNT;
 14      END IF;
 15      RETURN V_AVG_RATING;
 16  EXCEPTION
 17      WHEN NO_DATA_FOUND THEN
 18          RETURN NULL;
 19  END;
 20  /

Function created.

SQL> Begin
  2  dbms_output.put_line(get_overall_rating('E101'));
  3  end;
  4  /
7.5

PL/SQL procedure successfully completed.
```

```
SQL> CREATE OR REPLACE TRIGGER SALARY_INCREMENT
  2  AFTER INSERT ON WORKS_ON
  3  FOR EACH ROW
  4  DECLARE
  5      V_EMP_ID EMPLOYEE.EMPLOYEE_ID%TYPE;
  6      V_OVERALL_RATING NUMBER;
  7      V_SERVICE_COUNT NUMBER;
  8  BEGIN
  9      V_EMP_ID := :NEW.EMPLOYEE_ID;
 10      V_OVERALL_RATING := GET_OVERALL_RATING(V_EMP_ID);
 11      SELECT COUNT(*)
 12      INTO V_SERVICE_COUNT
 13      FROM WORKS_ON
 14      WHERE EMPLOYEE_ID = V_EMP_ID;
 15      IF V_OVERALL_RATING >= 8 AND MOD(V_SERVICE_COUNT, 10) = 0 THEN
 16          UPDATE EMPLOYEE
 17          SET SALARY = SALARY * 1.15
 18          WHERE EMPLOYEE_ID = V_EMP_ID;
 19          COMMIT;
 20      END IF;
 21  EXCEPTION
 22      WHEN NO_DATA_FOUND THEN
 23          NULL;
 24  END;
 25  /

Trigger created.
```

Chapter 7- **Conclusion and Future Work**

7.1 Conclusion

The garage management system represents a robust solution for efficiently handling various aspects of garage operations. It has tables including EMPLOYEE, OWNER_DETAILS, VEHICLE, WORKS_ON, REPAIRS, SUPPLIER, ORDERS, COST, and APPOINTMENT. This system enables coordination and management across all facets of garage management activities.

By leveraging the functionalities embedded within each table, garage owners can streamline employee management, customer relations, vehicle servicing, inventory procurement, and appointment scheduling, thereby optimizing resource utilization. The triggers integrated into the system ensure timely automation of critical processes such as cost calculation for orders and incentivizing customer loyalty through discounts.

7.2 Scope for Future Work

Looking towards the future, the garage management system presents exciting opportunities for expansion and innovation. Some of them are:

- **Advanced Reporting**: Implementing advanced reporting features to provide detailed insights into operational performance, revenue generation and employee productivity.
- **Predictive Maintenance**: Develop predictive maintenance algorithms to anticipate vehicle servicing needs, optimize scheduling, and minimize downtime by addressing issues before they escalate.
- **Mobile Application Development**: Creating a mobile application to facilitate appointment booking, order tracking and access to important information for customers and employees on the go.
- **Feedback Mechanisms**: Implementing feedback mechanisms to capture customer feedback and reviews, facilitating continuous improvement efforts and maintaining high service quality standards.
- **Enhanced Inventory Management**: Implementing advanced inventory management features to optimize stock levels and minimize inventory holding costs.

Overall, the garage management system empowers businesses to deliver exceptional service experiences by harnessing the power of technology.