A

SYNOPSIS REPORT ON

COMPLETION

OF

CAR DEALERSHIP APPLICATION

IN

anudip

Life. Transformed.

SUBMITTED BY:-

Jayant Agrawal

SUBMITTED TO:-

ANUJ KUMAR

GNIOT
Group of Institutions
Since 2001

GNIOT
GROUP OF INSTITUTIONS
Approved by AICTE & PCI, New Delhi and Affiliated to AKTU-Lucknow, GGSIPU-New Delhi & BTE-Lucknow

ABSTRACT

This project aims to develop a console-based car dealership application using Java and Hibernate, focusing on entities such as Car, Customer, Salesperson, and Payment. The application facilitates efficient management of car inventory, customer interactions, sales transactions, and payment processing within a dealership environment.

Leveraging object-relational mapping with Hibernate, the system ensures seamless data persistence and retrieval, enabling users to perform various operations like adding new cars, registering customers, assigning salespersons, recording sales, and managing payments.

The console interface provides a user-friendly experience, allowing dealership personnel to interact with the system intuitively. Additionally, the application incorporates features for generating comprehensive reports, empowering dealership managers to analyze sales performance, customer trends, and financial insights for informed decision-making. Overall, this project serves as a robust solution for automating and streamlining the operations of a car dealership, enhancing efficiency and productivity while delivering an optimal experience for both staff and customers.

OVERVIEW

The console-based car dealership application developed in Java and Hibernate serves as a comprehensive solution for managing various aspects of dealership operations. The system revolves around key entities: Car, Customer, Salesperson, and Payment, each playing a crucial role in the sales process.

Car Management:

Enables dealership staff to add new cars to the inventory, update car details, and remove sold or obsolete vehicles. Tracks essential information such as make, model, year, price, and availability status for effective inventory management.

Customer Management:

Facilitates the registration of new customers, storing their details including name, contact information, and preferences. Allows retrieval of customer information for personalized service and targeted marketing strategies.

Salesperson Assignment:

Associates salespersons with specific sales transactions, ensuring accountability and efficient sales process management. Enables tracking of salesperson performance and commission calculations based on sales contributions. Sales

Transaction Handling:

Records sales transactions, capturing details such as the car sold, customer involved, salesperson assigned, and payment information. Validates customer eligibility, availability of the selected car, and payment processing for a smooth transaction flow.

Payment Processing:

Manages payment processing for completed sales transactions, supporting various payment methods such as cash, credit card, or financing options. Calculates total transaction amounts, handles partial payments, and generates receipts for customers.

REQUIREMENTS

Language: JAVA

Framework: Hibernate

IDE: Eclipse

SYSTEM REQUIREMENTS: 512MB RAM 2GB at least ROM

ENTITY

The Project includes the following entities:

1. Car
2. Customer
3. Salesperson
4. Payment

DATABASE DESIGN

Step - 1

Create Database

```
create database cardealership;
```

Step - 2

Use Current Database

```
use cardealership;
```

Step - 3

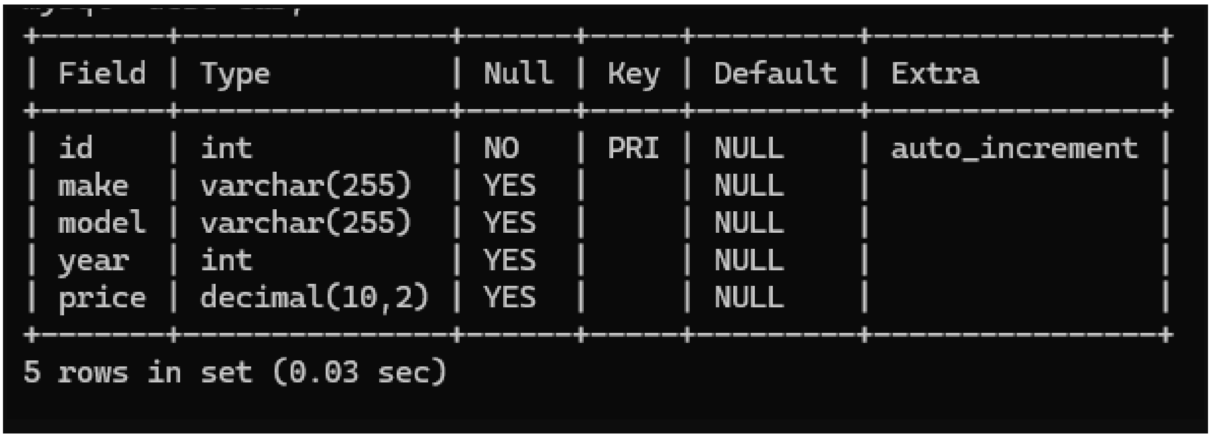
Create Car Table:

```
CREATE TABLE Car ( id INT PRIMARY KEY
AUTO_INCREMENT, make VARCHAR(255), model
VARCHAR(255), year INT, price DECIMAL(10, 2)
);
```

Step - 4

View Structure of Car table

```
desc Car;
```



```
+-------+---------------+------+-----+---------+----------------+
| Field | Type          | Null | Key | Default | Extra          |
+-------+---------------+------+-----+---------+----------------+
| id    | int           | NO   | PRI | NULL    | auto_increment |
| make  | varchar(255)  | YES  |     | NULL    |                |
| model | varchar(255)  | YES  |     | NULL    |                |
| year  | int           | YES  |     | NULL    |                |
| price | decimal(10,2) | YES  |     | NULL    |                |
+-------+---------------+------+-----+---------+----------------+
5 rows in set (0.03 sec)
```

Step - 5

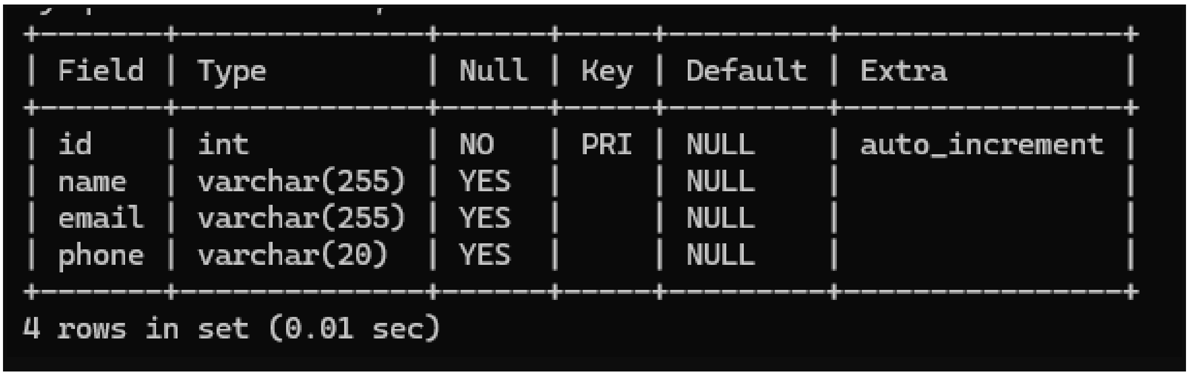
Create Customer Table:

```
CREATE TABLE Customer ( id INT PRIMARY KEY
AUTO_INCREMENT, name VARCHAR(255), email
VARCHAR(255), phone VARCHAR(20) );
```

Step - 6

View Structure of Customer table

```
desc Customer;
```



```
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  |     | NULL    |                |
| email | varchar(255) | YES  |     | NULL    |                |
| phone | varchar(20)  | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Step - 7

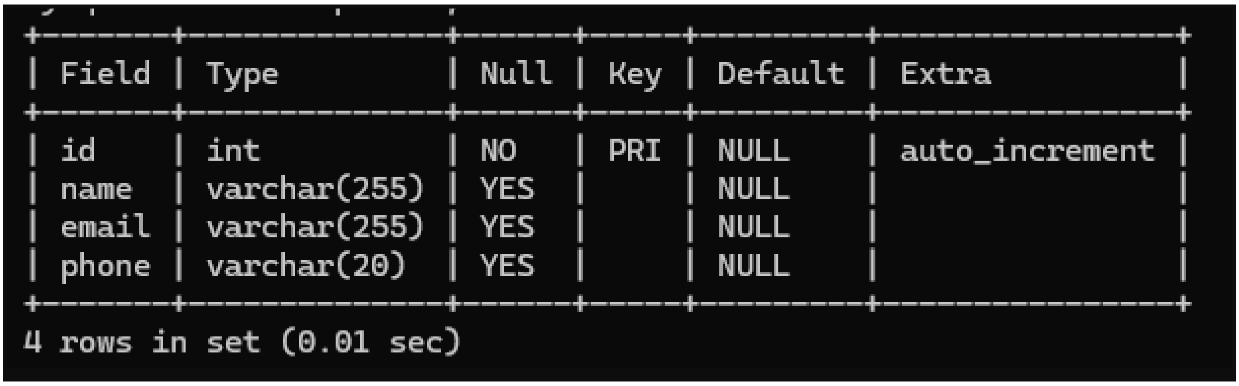
Create Salesperson Table:

```
CREATE TABLE Salesperson ( id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255), email VARCHAR(255), phone VARCHAR(20) );
```

Step - 8

View Structure of Salesperson table

```
desc Salesperson;
```



```
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int          | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  |     | NULL    |                |
| email | varchar(255) | YES  |     | NULL    |                |
| phone | varchar(20)  | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Step - 9

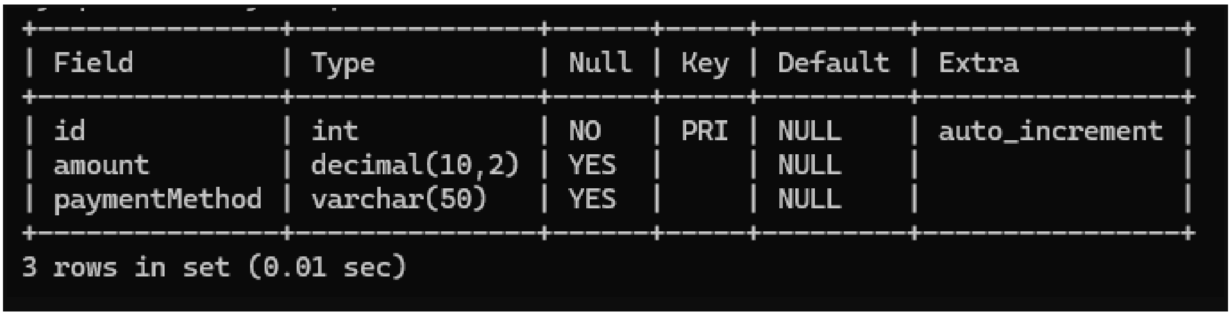
Create Payment Table:

```
CREATE TABLE Payment ( id INT PRIMARY KEY
AUTO_INCREMENT, amount DECIMAL(10, 2),
paymentMethod VARCHAR(50) );
```

Step - 10

View Structure of Payment table

```
desc Payment;
```



```
+---------------+---------------+------+-----+---------+----------------+
| Field         | Type          | Null | Key | Default | Extra          |
+---------------+---------------+------+-----+---------+----------------+
| id            | int           | NO   | PRI | NULL    | auto_increment |
| amount        | decimal(10,2) | YES  |     | NULL    |                |
| paymentMethod | varchar(50)   | YES  |     | NULL    |                |
+---------------+---------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

Step - 11

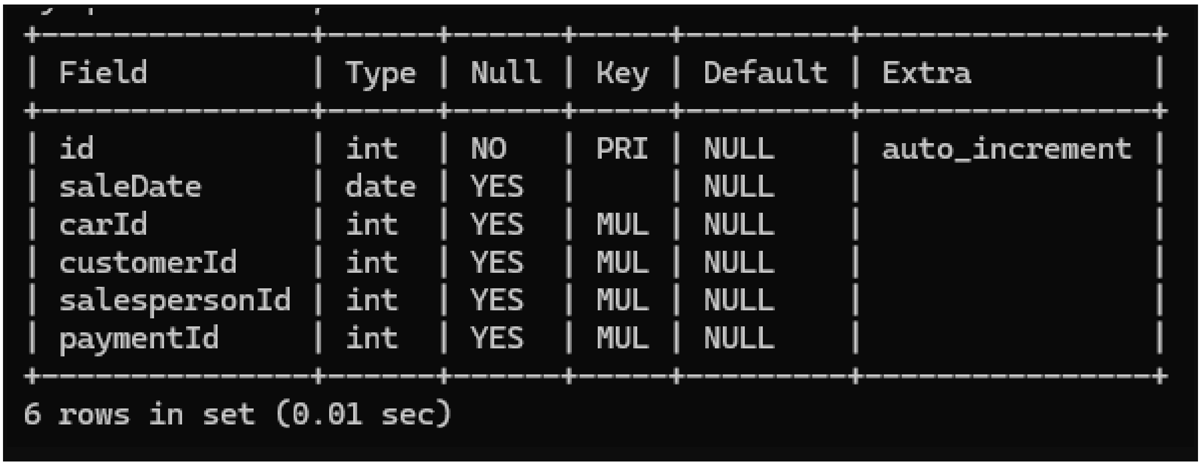
Create Sale Table:

```
CREATE TABLE Sale ( id INT PRIMARY KEY
AUTO_INCREMENT, saleDate DATE, carId INT,
customerId INT, salespersonId INT, paymentId
INT, FOREIGN KEY (carId) REFERENCES Car(id),
FOREIGN KEY (customerId) REFERENCES
Customer(id), FOREIGN KEY (salespersonId)
REFERENCES Salesperson(id), FOREIGN KEY
(paymentId) REFERENCES Payment(id) );
```

Step - 12

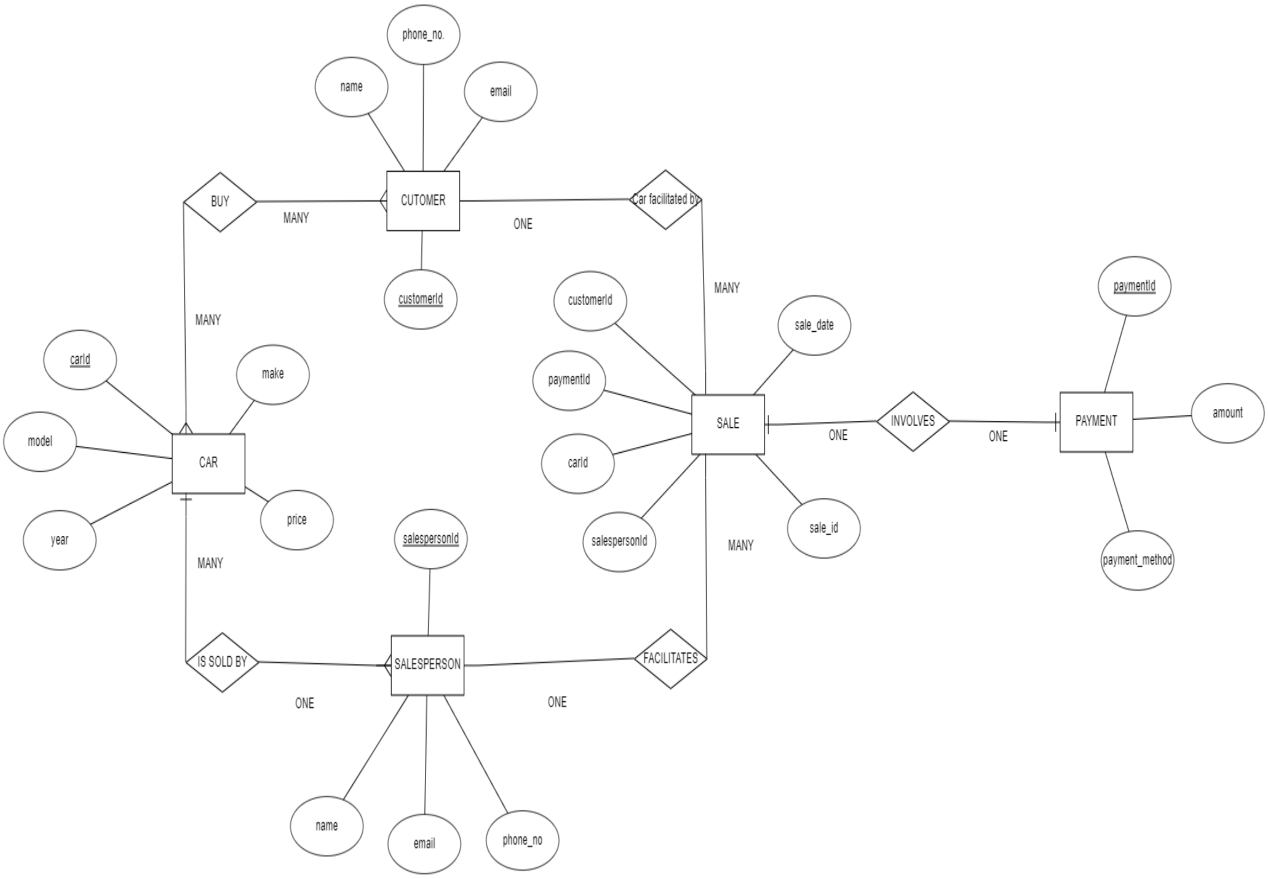
View Structure of Sale table

```
desc Sale;
```



```
+---------------+------+------+-----+---------+----------------+
| Field         | Type | Null | Key | Default | Extra          |
+---------------+------+------+-----+---------+----------------+
| id            | int  | NO   | PRI | NULL    | auto_increment |
| saleDate      | date | YES  |     | NULL    |                |
| carId         | int  | YES  | MUL | NULL    |                |
| customerId    | int  | YES  | MUL | NULL    |                |
| salespersonId | int  | YES  | MUL | NULL    |                |
| paymentId     | int  | YES  | MUL | NULL    |                |
+---------------+------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

ENTITY RELATIONSHIP DIAGRAM


phone_no.
name
email
BUY
MANY
CUTOMER
ONE
Car facilitated by
customerId
MANY
MANY
customerId
sale_date
paymentId
carId
make
model
CAR
price
year
paymentId
SALE
INVOLVES
PAYMENT
amount
ONE
ONE
carId
salespersonId
sale_id
MANY
salespersonId
MANY
payment_method
IS SOLD BY
SALESPERSON
FACILITATES
ONE
ONE
name
email
phone_no

SUMMARY

Hospitals are bustling communities with numerous patients, doctors, and tasks to handle. To streamline operations, Hospital Management Systems (HMS) act as intelligent assistants, aiding in patient management, appointment scheduling, financial transactions, inventory control, and staff administration. This paper explores the functionalities and benefits of an HMS, focusing on its role in enhancing hospital efficiency and patient care.

The requirements for developing an HMS include using Java language, Hibernate framework, and IntelliJ IDEA as the Integrated Development Environment (IDE). The system's minimum system requirements are 512MB RAM and at least 2GB of ROM.

Entities within the HMS encompass Patients, Doctors, Staff, Bills, and Payments. Patients' attributes include personal details, disease status, and admission status. The relationship between patients and billing records is one-to-one, while patients can have multiple payment records. Doctors' attributes consist of name, qualifications, specialization, and availability, with a many-to-many relationship between doctors and patients. Staff attributes include personal details and designation, with staff-patient and staff-doctor relationships established as many- to-many.

The Bill entity records various charges associated with patient care, while Payments track payment details, both tied to patients and bills. The ER Diagram, Class Diagram, and Sequence Diagram visually represent the relationships and interactions between these entities.

The output of the HMS includes menus for Patient, Doctor, Staff, Bill, and Payment management, ensuring smooth navigation and user-friendly interaction with the system. Through these menus, hospital staff can efficiently manage various aspects of hospital operations, ultimately leading to improved patient care and organizational effectiveness.