INSTITUTE OF ENGINEERING
ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT
KALANKI,KATHMANDU
(AFFILIATED TO TRIBHUVAN UNIVERSITY)

PROJECT REPORT
ON
Equipment Rental Management System

SUBMITTED TO:

Sailesh Singh

Lecturer,Department of Computer and Electronic Engineering

SUBMITTED BY:

Name: Gautam Jayswal

Roll_No: ACE080BCT029

SUBMISSION DATE: Falgun 15, 2082

INTRODUCTION

The Equipment Rental Management System is a database-driven application developed to streamline the operations of a rental business. In modern commerce, tracking high-value assets like cameras, laptops, and audio gear requires a robust system to prevent inventory loss and ensure timely returns. This project implements a relational database model that automates customer registration, tracks equipment stock levels, and logs every financial transaction with precision.

OBJECTIVES

The primary objectives of this database project are:

- → To design a normalized relational schema with a maximum of 5 entities to ensure efficiency.
- → To implement strong data integrity using Primary Keys, Foreign Keys, and Unique constraints.
- → To demonstrate advanced data retrieval techniques using complex joins and aggregations.
- → To provide a real-time monitoring tool via Database Views and ensure atomic operations using Transactions.

SYSTEM FEATURES

This system is packed with features designed to meet business needs:

- → **Automated Inventory Tracking:** The system automatically reduces the amount available_quantity of an item when a rental is confirmed.
- → **Financial Transparency:** Every rental is linked to a unique payment record, allowing for easy revenue auditing and status tracking.
- → **Real-time Monitoring:** The Active_Reverential allows staff to see exactly which items are currently out with customers and when they are due back.
- → **Customer Insights:** Built-in queries identify "Power Users" (customers who rent frequently), enabling targeted marketing strategies.

BUSINESS RULES & CONSTRAINTS

To ensure the database operates realistically, the following logic was applied:

- → **Customer Integrity:** Every customer must provide a unique phone number and email to prevent duplicate profiles.
- → **Inventory Control:** Equipment cannot be rented if the available_quantity is zero.
- → **Rental Tracking:** Each rental must have a defined rental_date and a return_date to calculate the total rental duration.
- → **Payment Uniqueness:** To avoid double-billing, the Payment table uses a Unique constraint on rental_id, ensuring each rental event is paid for only once.
- → **Cascading Logic:** Relationships are established using Foreign Keys to ensure that a rental cannot exist without a valid customer.

DATA DICTIONARY

The following tables define the structure and constraints of the five implemented entities.

Table: Customer	Description: Stores personal details of clients
customer_id	INT (Primary Key): Unique ID for the customer.
name	VARCHAR(100): Full name of the client.
phone	VARCHAR(15): Primary contact number.
email	VARCHAR(100): Contact email for notifications.

Table: Equipment	Description: Manages the inventory of profitable gear
equipment_id	INT (Primary Key): Unique ID for each piece of gear.
name	VARCHAR(100): Description of the item (eg, Canon DSLR).
rental_price_per_day	DECIMAL(10,2): Cost to rent the item for 24 hours.
available_quantity	INT: Current number of units in stock.

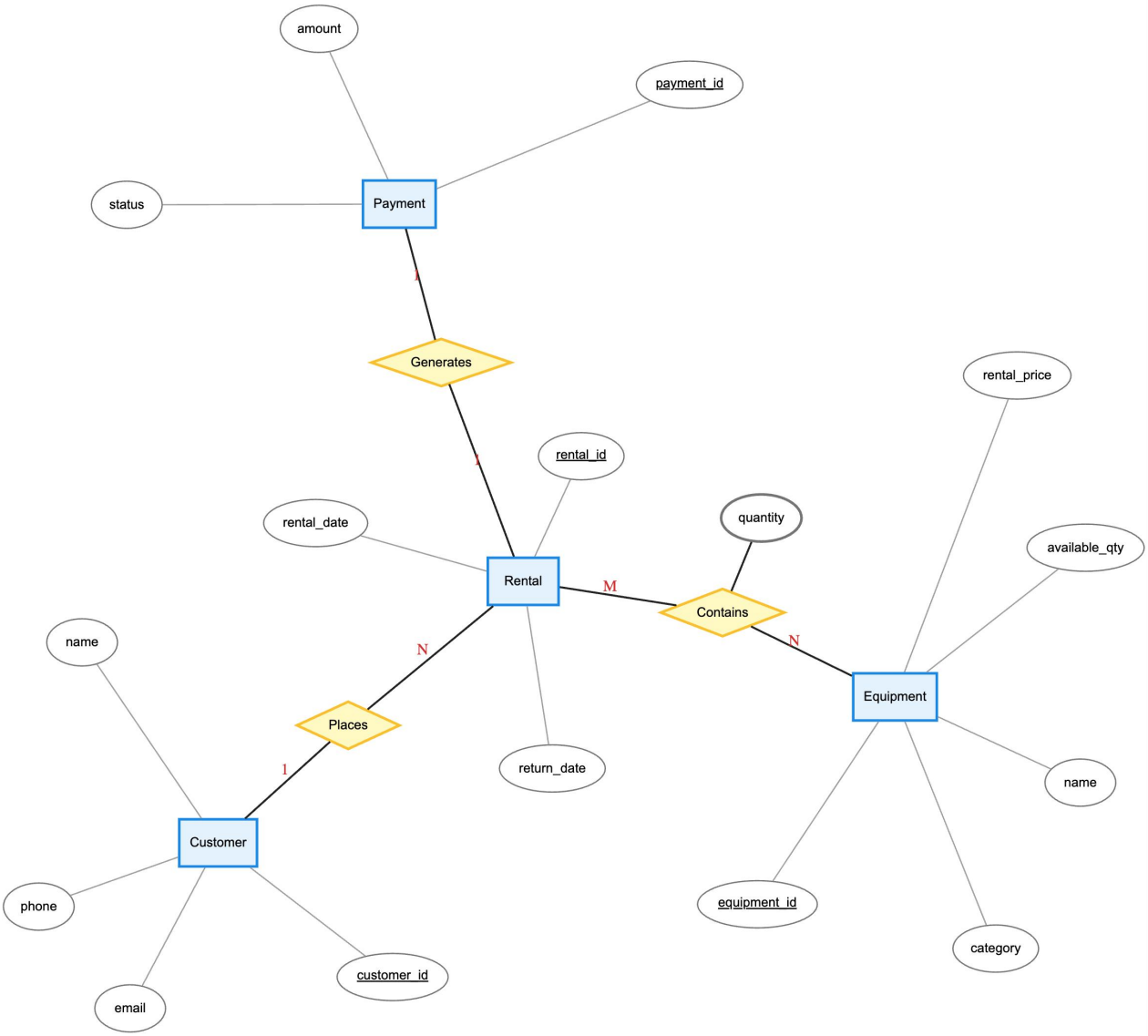
Table: Rental	Description: Master record for a rental transaction
rental_id	INT (Primary Key): Unique ID for the rental event.

Table: Rental	Description: Master record for a rental transaction
customer_id	INT (Foreign Key): References the Customer table.
rental_date	DATE: The date the equipment was picked up.
return_date	DATE: The date the equipment is due back.

Table: Rental_Detail	Description: Junction table for M:N Relationship
rental_id	INT (PK/FK): References the rental record.
equipment_id	INT (PK/FK): References the Equipment record.
quantity	INT: Number of units of that item in the rental.

Table: Payment	Description: Tracks financial status per rental
payment_id	INT (Primary Key): Unique ID for the payment.
rental_id	INT (FK, UNIQUE): Ensures exactly one payment per rental.
amount	DECIMAL(10,2): Total fees paid.
payment_status	VARCHAR(20): Status (eg, 'Paid', 'Pending').

ENTITY-RELATIONSHIP (ER) DIAGRAM



SQL IMPLEMENTATION ANALYSIS

The system was implemented using SQL scripts that demonstrate comprehensive database operations:

- → **Relational Joins:** We used INNER JOIN to generate customer rental histories and LEFT JOIN to audit inventory that has never been rented.
- → **Aggregate Functions:** COUNT and GROUP BY are used for analytic, while SUM calculating the total revenue from the Payment table.
- → **Sub-query:** A nested query was implemented to identify "Loyal Customers" (those with more than one rental).
- → **Database View:** The Active_Rentals view provides a dynamic window into the database, showing only current rentals that are due for return.
- → **Transaction Control:** Atomic operations were ensured using START TRANSACTION, COMMIT, and ROLLBACK logic during inventory updates

APPENDIX: SOURCE CODE

Below is the complete SQL script designed for this project.

```
-- Create Database
Create Database DBMS;
use DBMS;

-- 1. Customer Table
CREATE TABLE Customer (
   customer_id INT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   phone VARCHAR(15),
   email VARCHAR(100)
);

-- 2. Equipment Table
CREATE TABLE Equipment (
   equipment_id INT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   category VARCHAR(50),
   rental_price_per_day DECIMAL(10,2),
   available_quantity INT
```

```
);

-- 3. Rental Table
CREATE TABLE Rental (
    rental_id INT PRIMARY KEY,
    customer_id INT,
    rental_date DATE,
    return_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

-- 4. Rental_Detail Table
CREATE TABLE Rental_Detail (
    rental_id INT,
    equipment_id INT,
    quantity INT,
    PRIMARY KEY (rental_id, equipment_id),
    FOREIGN KEY (rental_id) REFERENCES Rental(rental_id),
    FOREIGN KEY (equipment_id) REFERENCES Equipment(equipment_id)
);

-- 5. Payment Table
CREATE TABLE Payment (
    payment_id INT PRIMARY KEY,
    rental_id INT UNIQUE,
    amount DECIMAL(10,2),
    payment_date DATE,
    payment_status VARCHAR(20),
    FOREIGN KEY (rental_id) REFERENCES Rental(rental_id)
);

-- Insert Sample Data

INSERT INTO Customer VALUES
(1, 'Amit Sharma', '9800000001', 'amit@gmail.com'),
(2, 'Sita Rai', '9800000002', 'sita@gmail.com'),
(3, 'Ramesh Thapa', '9800000003', 'ramesh@gmail.com'),
(4, 'Nisha Karki', '9800000004', 'nisha@gmail.com'),
(5, 'Bikash Gurung', '9800000005', 'bikash@gmail.com');

INSERT INTO Equipment VALUES
(1, 'Canon DSLR Camera', 'Camera', 1500.00, 5),
(2, 'Projector Epson X200', 'Projector', 2000.00, 3),
(3, 'Dell Laptop i7', 'Laptop', 2500.00, 4),
(4, 'Sound System JBL', 'Audio', 1800.00, 2),
(5, 'GoPro Hero 10', 'Camera', 1200.00, 6);

INSERT INTO Rental VALUES
```

```
(101, 1, '2026-02-01', '2026-02-05'),
(102, 2, '2026-02-02', '2026-02-06'),
(103, 1, '2026-02-10', '2026-02-12'),
(104, 3, '2026-02-11', '2026-02-15'),
(105, 4, '2026-02-15', '2026-02-18');

INSERT INTO Rental_Detail VALUES
(101,1,1), (101,2,1), (102,3,1),
 (103,5,2), (104,4,1), (105,1,1);


INSERT INTO Payment VALUES
(1, 101, 7000.00, '2026-02-05', 'Paid'),
(2, 102, 10000.00, '2026-02-06', 'Paid'),
(3, 103, 3600.00, '2026-02-12', 'Pending'),
(4, 104, 7200.00, '2026-02-15', 'Paid'),
(5, 105, 4500.00, '2026-02-18', 'Paid');

-- Required Queries

-- INNER JOIN

SELECT c.name, r.rental_id, r.rental_date FROM Customer c INNER JOIN Rental r ON
c.customer_id = r.customer_id;

-- AGGREGATE with GROUP BY

SELECT customer_id, COUNT(rental_id) AS total_rentals FROM Rental GROUP BY
customer_id;

-- SUB-QUERY

SELECT name FROM Customer WHERE customer_id IN (SELECT customer_id FROM Rental
GROUP BY customer_id HAVING COUNT(rental_id) > 1);

-- VIEW

CREATE VIEW Active_Rentals AS SELECT r.rental_id, c.name, r.return_date FROM Rental r
JOIN Customer c ON r.customer_id = c.customer_id WHERE r.return_date >= CURDATE();

-- TRANSACTION

START TRANSACTION;
INSERT INTO Rental VALUES (106, 5, '2026-02-20', '2026-02-22');
UPDATE Equipment SET available_quantity = available_quantity - 1 WHERE equipment_id
= 3;
COMMIT;
```

CONCLUSION

The Equipment Rental Management System fulfills all technical and documentation requirements for the DBMS Practical Internal Evaluation. The design is normalized, the queries are optimized, and the system provides a reliable platform for managing business data.