

**Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavrepalanchowk**



**A project report
On
“Corp manager”**

[Code no=COMP232]

**Submitted by:
Basanta Khanal
Roll no:26
Level: UNG CE(II/II)**

**Submitted to:
Mr.Sanjog Sigdel
Department of Computer Science and Engineering**

**Submission Date:
9/02/2026**

Introduction

MySQL is an open-source relational database management system (RDBMS) widely used for storing and managing data. It is known for its reliability, performance, scalability, and ease of use, making it a popular choice for developers and organizations of all sizes.

In my mini project, I have created a database named Corp Manager, which refers to a company management system. This project is designed to manage important company information such as departments, employees, company projects, and employee project assignments. Each entity in the database serves a specific purpose. The Departments table stores the different departments available in the company, the Employees table keeps records of all employees, and the Projects and Employee Projects tables manage the company's projects and the employees working on them.

This system is beneficial because it helps organize and maintain company data in an effective and structured way, making it easier to access and manage information efficiently.

Table of content

Contents

Introduction	2
Table of content.....	3
Lab session works	4
• General sql table creation and insertion value	4
• Relational operators	8
• Normalization.....	11
• Transactions	16
Conclusion	18

Lab session works

- General SQL table creation and insertion value

I have created a database called basantadb. In this database I created four tables departments, employees, employees projects and projects having the different attributes.

SQL Query:

```
create database basantadb;
use basantadb;

CREATE TABLE departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100),
    location VARCHAR(100)
);

CREATE TABLE employees (
    emp_id INT,
    emp_name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    contact VARCHAR(15),
    position VARCHAR(50),
    salary DECIMAL(10,2),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);

CREATE TABLE projects (
    project_id INT PRIMARY KEY,
    project_name VARCHAR(100),
    start_date DATE,
    end_date DATE,
    budget DECIMAL(10,2)
);

CREATE TABLE employee_projects (
    ep_id INT ,
    emp_id INT,
    project_id INT,
    role VARCHAR(50),
    assigned_date DATE,
    FOREIGN KEY (emp_id) REFERENCES employees(emp_id),
    FOREIGN KEY (project_id) REFERENCES projects(project_id)
);
```

Outputs:

This shows the tables I have created in MY SQL workbench.

Tables_in_basantadb	
▶	departments
	employee_projects
	employees
	projects

I have inserted some values in the table having the four tables according to their attributes described like table department uses the attributes department id ,department name, location and table employees uses employees name, id, salary, email, contact number and projects has attributes like project id, project name, start date end date, budget and employee project has ep id, emp id, projec id, role, assigned date.

SQL Query:

```
INSERT INTO departments (dept_id,dept_name,location)
VALUES
(1, 'Human Resources Department', 'Building A'),
(2, 'IT Department', 'Building B'),
(3, 'Finance Department', 'Building C'),
(4, 'Web Department', 'Building d');

INSERT INTO employees (emp_id,emp_name,email,contact,position,salary,dept_id)
VALUES
(101, 'Ramesh Thapa', 'ramesh@company.com', '9841000001', 'Manager', 60000.00, 1),
(102, 'Sita Sharma', 'sita@company.com', '9841000002', 'Developer', 45000.00, 2),
(103, 'Binod Singh', 'binod@company.com', '9841000003', 'Accountant', 40000.00, 3),
(104, 'Priya Lama', 'priya@company.com', '9841000004', 'HR Assistant', 35000.00, 1),
(105, 'Saugat Thakuri', 'saugat@company.com', '9841202365', 'Developer', 45000, 4);

INSERT INTO projects (project_id,project_name,start_date,end_date,budget)
VALUES
(201, 'Web Portal System', '2023-01-01', '2023-06-01', 200000.00),
(202, 'Mobile App Development', '2023-03-05', NULL, 300000.00),
(203, 'Payroll Automation', '2022-10-10', '2023-02-15', 150000.00),
(204, 'Animation', '2020-9-11', '2023-06-19', 250000.00),
(205, 'Route Hive', '2024-10-28', NULL, 150000.00);

INSERT INTO employee_projects (ep_id,emp_id,project_id,role,assigned_date)
VALUES
(1, 102, 201, 'Backend Developer', '2023-01-02'),
(2, 102, 202, 'Mobile Developer', '2023-03-06'),
(3, 101, 203, 'Project Manager', '2022-10-11'),
(4, 103, 203, 'Finance Analyst', '2022-10-12'),
(5, 104, 203, 'Frontend Developer', '2024-5-4');
```

Output:

	dept_id	dept_name		location	
▶	1	Human Resources Department		Building A	
	2	IT Department		Building B	
	3	Finance Department		Building C	
	4	Web Department		Building d	
*	NULL	NULL		NULL	

	emp_id	emp_name	email	contact	position	salary	dept_id
▶	101	Ramesh Thapa	ramesh@company.com	9841000001	Manager	60000.00	1
	102	Sita Sharma	sita@company.com	9841000002	Developer	45000.00	2
	103	Binod Singh	binod@company.com	9841000003	9841000002	40000.00	3
	104	Priya Lama	priya@company.com	9841000004	HR Assistant	35000.00	1
	105	Saugat Thakuri	saugat@company.com	9841202365	Developer	45000.00	4

	project_id	project_name		start_date	end_date	budget
▶	201	Web Portal System		2023-01-01	2023-06-01	200000.00
	202	Mobile App Development		2023-03-05	NULL	300000.00
	203	Payroll Automation		2022-10-10	2023-02-15	150000.00
	204	Animation		2020-09-11	2023-06-19	250000.00
	205	Route Hive		2024-10-28	NULL	150000.00
*	NULL	NULL		NULL	NULL	NULL

	ep_id	emp_id	project_id	role	assigned_date
▶	1	102	201	Backend Developer	2023-01-02
	2	102	202	Mobile Developer	2023-03-06
	3	101	203	Project Manager	2022-10-11
	4	103	203	Finance Analyst	2022-10-12
	5	104	203	Frontend Developer	2024-05-04

This shows the output generated by the sql query of different insertion values from above.

- Relational operators

In this line of code I have written to select from department where department id is 1 also I have used union to combine name of employes and projects name. I have use intersect to show the values that are in table projects and employee projects.

```
select * from departments where dept_id=1;  
select emp_name as name from employees  
union  
select project_name as name from projects;  
select project_id as id from projects  
intersect  
select emp_id as id from employees;
```

Output

The screenshot shows a database interface with two tables displayed side-by-side. The first table, 'departments', has columns 'dept_id', 'dept_name', and 'location'. It contains one row with dept_id 1, dept_name 'Human Resources Department', and location 'Building A'. The second table, 'employees', has a single column 'name'. It lists eight names: Ramesh Thapa, Sita Sharma, Binod Singh, Priya Lama, Saugat Thakuri, Web Portal System, Mobile App Development, Payroll Automation, Animation, and Route Hive. The 'employees' table has a header row and several data rows, while the 'departments' table has a header row and a single data row.

	dept_id	dept_name	location
▶	1	Human Resources Department	Building A
*	NULL	NULL	NULL

	name
▶	Ramesh Thapa
	Sita Sharma
	Binod Singh
	Priya Lama
	Saugat Thakuri
	Web Portal System
	Mobile App Development
	Payroll Automation
	Animation
	Route Hive

I used join operation to join the two table employees and projects.it shows every employees assigned to a project and I have used except to show the employees that have no projects assigned to them.

```

select emp_id as id from employees;
select employees.emp_name, employee_projects.assigned_date
from employees
join employee_projects on employees.emp_id=employee_projects.emp_id;
SELECT emp_name FROM employees
EXCEPT
SELECT employees.emp_name
FROM employees
JOIN employee_projects
ON employees.emp_id = employee_projects.emp_id;

```

Output

	emp_name	assigned_date
▶	Sita Sharma	2023-01-02
	Sita Sharma	2023-03-06
	Ramesh Thapa	2022-10-11
	Binod Singh	2022-10-12
	Priya Lama	2024-05-04

	Result Grid	grid icon	refresh icon
	emp_name		
▶	Saugat Thakuri sudeep karn		

The inner join between employees and departments joins only the employees who have a department, showing matched employee and department information. The left join joins all employees with their departments, including employees who are not assigned to any department. The right join joins all departments with their employees, including departments that have no employees, with null for employee fields when there is no match. The full outer join joins all employees and all departments, showing all matches where they exist and filling null for unmatched employees or departments.

```

SELECT departments.dept_id,departments.dept_name,employees.emp_id,employees.emp_name
FROM employees
RIGHT JOIN departments
ON employees.dept_id = departments.dept_id;
SELECT departments.dept_id,departments.dept_name,employees.emp_id,employees.emp_name
FROM departments
LEFT JOIN employees
ON employees.dept_id = departments.dept_id;
SELECT employees.emp_id,employees.emp_name,employees.position,departments.dept_name,departments.location
FROM employees
INNER JOIN departments
ON employees.dept_id = departments.dept_id;
SELECT employees.emp_name,projects.project_name,employee_projects.role,employee_projects.assigned_date
FROM employee_projects
INNER JOIN employees
ON employee_projects.emp_id = employees.emp_id
INNER JOIN projects
ON employee_projects.project_id = projects.project_id;

```

Output

	dept_id	dept_name	emp_id	emp_name
▶	1	Human Resources Department	101	Ramesh Thapa
	1	Human Resources Department	104	Priya Lama
	2	IT Department	102	Sita Sharma
	3	Finance Department	103	Binod Singh
	3	Finance Department	201	sudeep karn
	4	Web Department	105	Saugat Thakuri

	emp_id	emp_name	position	dept_name	location
▶	101	Ramesh Thapa	Manager	Human Resources Department	Building A
	104	Priya Lama	HR Assistant	Human Resources Department	Building A
	102	Sita Sharma	Developer	IT Department	Building B
	103	Binod Singh	Accountant	Finance Department	Building C
	201	sudeep karn	HR	Finance Department	Building C
	105	Saugat Thakuri	Developer	Web Department	Building d

	emp_name	project_name	role	assigned_date
▶	Sita Sharma	Web Portal System	Backend Developer	2023-01-02
	Sita Sharma	Mobile App Development	Mobile Developer	2023-03-06
	Ramesh Thapa	Payroll Automation	Project Manager	2022-10-11
	Binod Singh	Payroll Automation	Finance Analyst	2022-10-12
	Priya Lama	Payroll Automation	Frontend Developer	2024-05-04

- Normalization

We have studied the concept of normalization in databases. To demonstrate this topic, I created a table that contains unnormalized data. Normalization is a technique used to organize data properly and reduce redundancy.

There are three main stages of normalization that we learned:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

Using the table below, I have explained how data is gradually converted from an unstructured form into a well-organized database structure through these normal forms.

```
CREATE TABLE employees_unf (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    email VARCHAR(100),
    contact VARCHAR(100),
    position VARCHAR(50),
    salary DECIMAL(10,2),
    dept_info VARCHAR(100)
);

INSERT INTO employees_unf
(emp_id, emp_name, email, contact, position, salary, dept_info)
VALUES
(989, "Suman Giri", "suman12@gmail.com", "9841202156,985212345", "Developer", 55000, "1-IT"),
(101, "Ramesh Thapa", "ramesh@company.com", "9841000001,9745693120", "Manager", 60000, "1-IT");
select * from employees_unf;
```

	emp_id	emp_name	email	contact	position	salary	dept_info
▶	101	Ramesh Thapa	ramesh@company.com	9841000001,9745693120	Manager	60000.00	1-IT
	989	Suman Giri	suman12@gmail.com	9841202156,985212345	Developer	55000.00	1-IT
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

This is my un normalized table now I have to convert it into 1-nf form.

- 1-nf

A table is in **First Normal Form** when:

- Each column contains only atomic (single) values
- No repeating groups or multi-valued attributes exist

```
CREATE TABLE employees_1nf (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    email VARCHAR(100),
    contact1 VARCHAR(100),
    contact2 varchar(100),
    position VARCHAR(50),
    salary DECIMAL(10,2),
    dept_info VARCHAR(100)
);
```

```
INSERT INTO employees_1nf VALUES
(989, "Suman Giri", "suman12@gmail.com", "9841202156", "985212345", "Developer", 55000, 1),
(101, "Ramesh Thapa", "ramesh@company.com", "9841000001", "9745693120", "Manager", 60000, 1);
```

	emp_id	emp_name	email	contact1	contact2	position	salary	dept_info
▶	101	Ramesh Thapa	ramesh@company.com	9841000001	9745693120	Manager	60000.00	1
	989	Suman Giri	suman12@gmail.com	9841202156	985212345	Developer	55000.00	1
*	HULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

So you can see that data in table are separated in each column have only one data. this is how 1-nf is done. I have separated contacts to contact 1 and contact 2 for converting into 1-nf.

- 2-nf

A table is in **Second Normal Form** when:

- It is already in 1NF
- Every non-key attribute is fully dependent on the entire primary key
- Partial dependency is removed
- 2NF helps separate data into smaller related tables.

```
CREATE TABLE employees_2nf (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100),
    email VARCHAR(100),
    position VARCHAR(50),
    salary DECIMAL(10,2),
    dept_id INT
);

CREATE TABLE employee_contacts_2nf (
    emp_id INT,
    contact1 VARCHAR(15),
    contact2 varchar(15)
);

INSERT INTO employees_2nf VALUES
(989, "Suman Giri", "suman12@gmail.com", "Developer", 55000, 1),
(101, "Ramesh Thapa", "ramesh@company.com", "Manager", 60000, 1);

INSERT INTO employee_contacts_2nf VALUES
(989, "9841202156","985212345"),
(101, "9841000001","9745693120");
```

	emp_id	emp_name	email	position	salary	dept_id
▶	101	Ramesh Thapa	ramesh@company.com	Manager	60000.00	1
▶	989	Suman Giri	suman12@gmail.com	Developer	55000.00	1
*	NULL	NULL	NULL	NULL	NULL	NULL

	emp_id	contact1	contact2
▶	989	9841202156	985212345
▶	101	9841000001	9745693120

So for 2 nf I have created 2 table on for employee and another employee contact. Since employee id is primary key I have used it access the 2 tables data. Employee table to show the info of employee like name, department id , email, salary , position. The employee contact table to show the contact of employees by employee id.

- 3-nf

A table is in Third Normal Form when:

- It is already in 2NF
- There are no transitive dependencies
- 3NF ensures the database is free from unnecessary dependency and duplication.

```

CREATE TABLE departments_3nf (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100)
);

INSERT INTO departments_3nf VALUES
(1, "IT"),
(2, "Finance"),
(3, "HR");

select * from departments_3nf;

```

	dept_id	dept_name
▶	1	IT
	2	Finance
	3	HR
✳	NULL	NULL

So for the 3-nf form I have separated department id to show the department name.

Since my data has no redundancy so I no need to convert it into bcnf.

- Transactions

A transaction in MySQL is a sequence of one or more SQL operations that are executed as a single unit of work. It ensures that database actions are performed safely and correctly. If one query fails, the entire transaction can be rolled back to avoid incorrect data.

```
START TRANSACTION;  
INSERT INTO employees(emp_id, emp_name, email, contact, position, salary, dept_id)  
VALUES (99, 'Amit Shrestha', 'amit@company.com', '9841111111', 'Developer', 50000, 2);  
COMMIT;
```

Result Grid							
	emp_id	emp_name	email	contact	position	salary	dept_id
▶	99	Amit Shrestha	amit@company.com	9841111111	Developer	50000.000000	2

This transaction is performed to insert the data in employees table. Start transaction begins a new transaction tells MySQL that SQL statement should be treated as one unit of work so any changes made after this will not be directly reflected unless commit is used.

```
START TRANSACTION;  
INSERT INTO employees(emp_id, emp_name, email, contact, position, salary, dept_id)  
VALUES (202, 'Wrong Person', 'wrong@company.com', '9800000000', 'Tester', 40000, 3);  
ROLLBACK;
```

Transaction has started and insert statement is executed, at this stage the record is inserted only temporarily because the transaction has not been completed. Rollback command cancels all the changes made during this transaction. So although the data have been inserted it is not reflected in the database.

```
START TRANSACTION;  
UPDATE employees  
SET salary = salary + 500000  
WHERE emp_id = 103;  
COMMIT;
```

emp_id	first_name	email	phone_number	role	salary	version
103	Binod Singh	binod@company.com	9841000003	Accountant	-460000.000000	3

So at beginning emp id negative due to errors. After using transaction it is updated.

emp_id	first_name	email	phone_number	role	salary	version
103	Binod Singh	binod@company.com	9841000003	Accountant	40000.000000	3

Conclusion

This lab report shows the expected skills in constructing and working with a MySQL database that includes techniques like constructing tables, inserting and altering data, the use of relational operators and joins, normalization (normal UNF into 1NF, 2NF, 3NF), and employing transactions like COMMIT and ROLLBACK. These steps are useful tools in making database systems more reliable, maintainable, and valid, which are the lab exercises in the report you are expected to prepare.