DBMS MINI PROJECT

PAYROLL MANAGEMENT SYSTEM

NAME: SAMARTH RAJENDRA

SRN: PES1UG20CS370

SEC: G

Description and Scope

The proposed project “Payroll Management System” has been developed to overcome the problems faced in the practicing of manual system. This software is built to eliminate and in some cases reduce the hardships faced by the existing system. Moreover this system is designed for particular need of the company to carry out its operations in a smooth and effective manner. This web application is reduced as much as possible to avoid errors while entering data. It also provides error message while entering invalid data. It is user-friendly as no formal knowledge is required to use the system. Human resource challenges are faced by every organization which has to be overcome by the organization. Every organization has different employee and payroll management needs. Therefore I have design exclusive Employee and payroll Management System that are adapted to the organization’s Managerial Requirements.

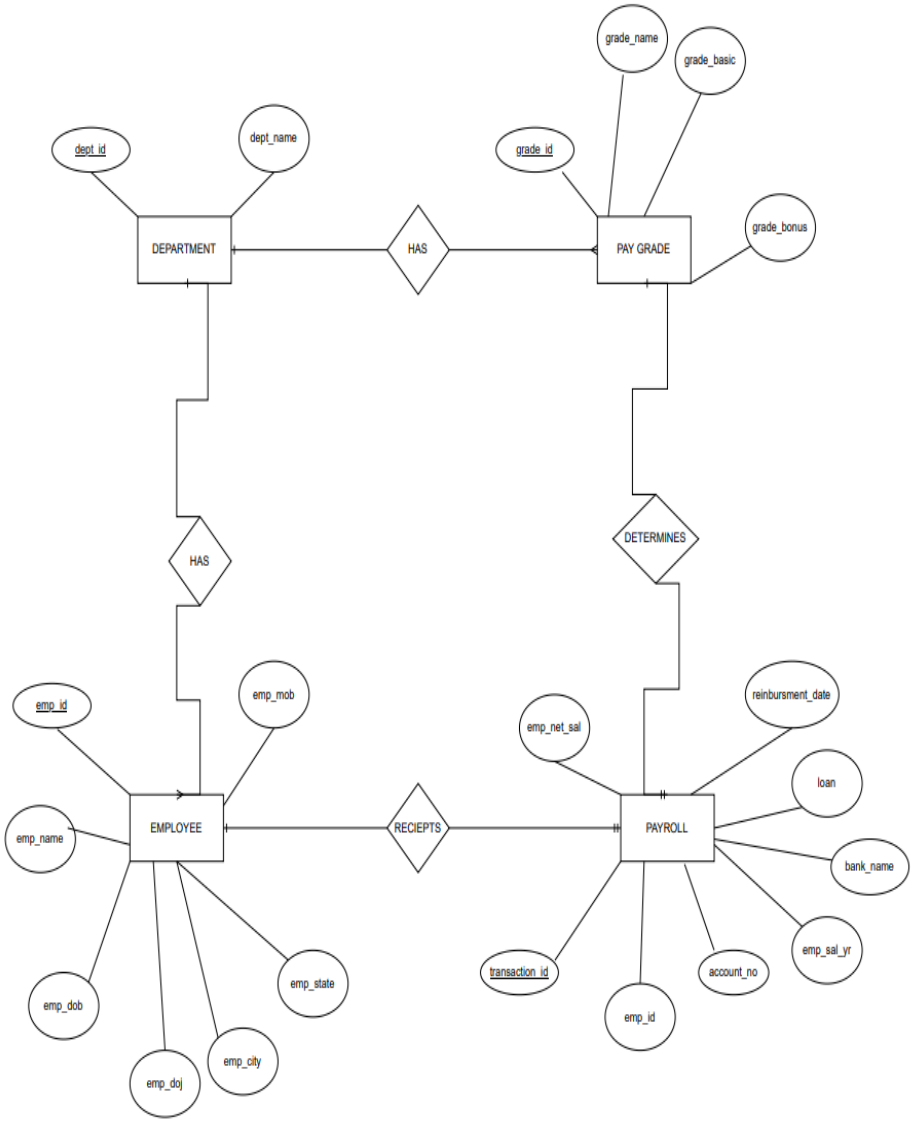
ABSTRACT AND PURPOSE:

The purpose of this document is to describe the functionality and specifications of the design of a web application for Managing Employees and their payroll. The expected audiences of this document are the developers and the admin of the web application. Now with the help of this system the admin has the information on his finger tips and can easily prepare a good record based on their requirements. Finally, we can say that this system will not only automate the process but save the valuable time of the manager or the admin, which can be well utilized buy his institute. This will be an additional advantage and management of power based on their free time from his normal duty.

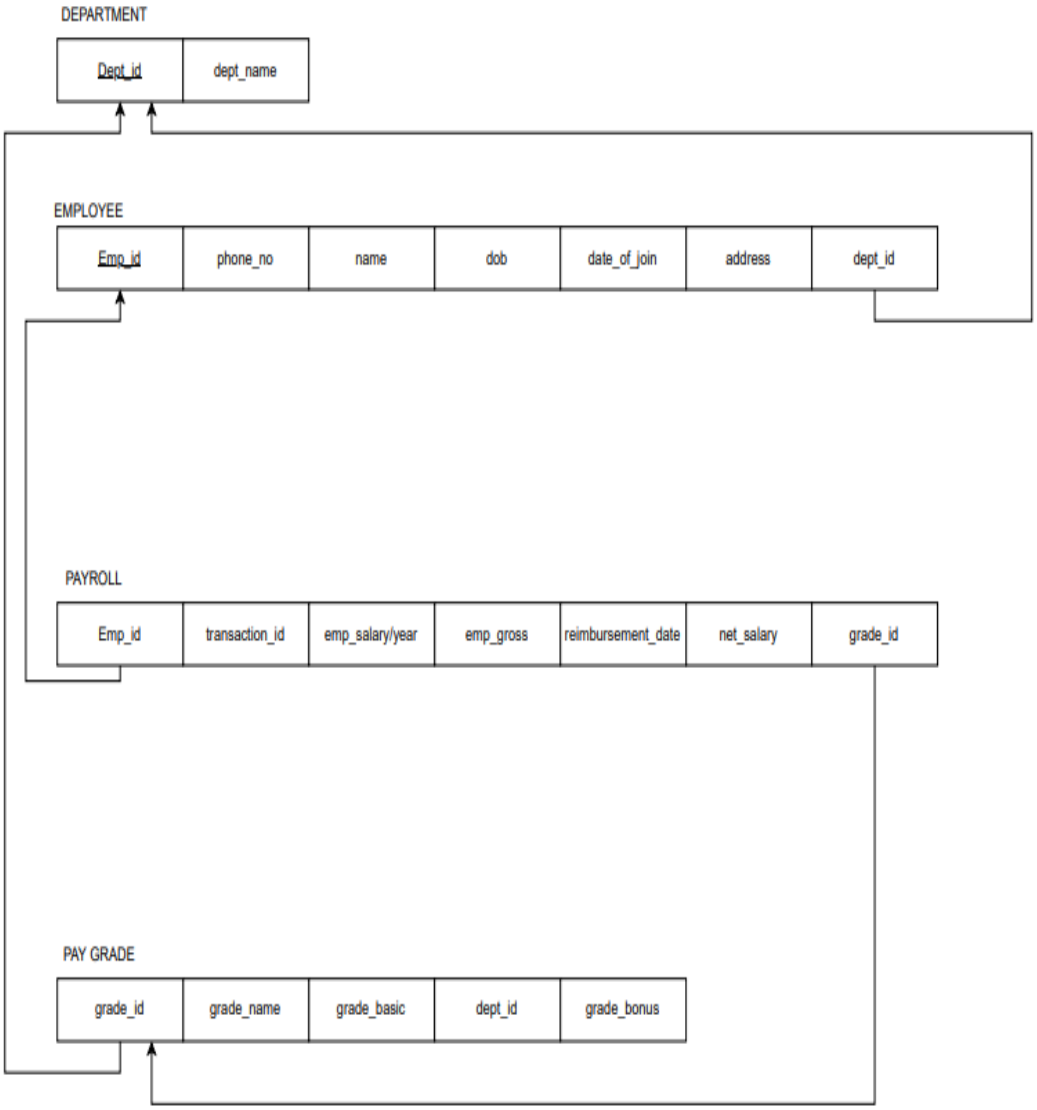
MODULES:

- Streamlit for frontend
- Pickle and streamlit-authenticator for login
- Sql and maria db for backend database

ER – DIAGRAM


grade_name
grade_basic
dept_id
dept_name
grade_id
DEPARTMENT
HAS
PAY GRADE
grade_bonus
HAS
DETERMINES
emp_id
emp_mob
emp_net_sal
reinbursment_date
loan
emp_name
EMPLOYEE
RECIEPTS
PAYROLL
bank_name
emp_sal_yr
emp_dob
emp_state
transaction_id
account_no
emp_id
emp_doj
emp_city

RELATIONAL SCHEMA



DDL Statements – Building the database

```
create table Employee(Emp_id varchar(10),
                    fname varchar(15) NOT NULL,
                    lname varchar(15) NOT NULL,
                    dob varchar(30)  NOT NULL,
                    phone varchar(10) NOT NULL ,
                    email varchar(40) NOT NULL,
                    country varchar(20) NOT NULL,
                    city varchar(30),
                    pincode int(6),
                    bank_name varchar(20) NOT NULL,
                    department_id varchar(10) NOT NULL,
                    primary key(Emp_id),
                    FOREIGN KEY (department_id) REFERENCES
department(department_id) ON DELETE CASCADE);


create table payroll(transaction_id varchar(20)  NOT NULL,
                    Emp_id varchar(20)   NOT NULL,
                    account_no varchar(20)  NOT NULL,
                    bank_name varchar(30)   NOT NULL,
                    Emp_net_sal varchar(20) NOT NULL,
                    Emp_sal_yr varchar(20) NOT NULL,
                    Reinbursment_date varchar(20) NOT NULL,
                    loan varchar(10) default NULL,
                    grade_id varchar(10) NOT NULL,
                    primary key(transaction_id),
                    FOREIGN KEY (grade_id) REFERENCES paygrade(grade_id) ON
DELETE CASCADE,
                    FOREIGN KEY (Emp_id) REFERENCES employee(Emp_id) ON DELETE
CASCADE);



create table paygrade(grade_id varchar(10),
                    grade_name varchar(30) NOT NULL,
                    grade_basic varchar(30) NOT NULL,
                    grade_bonus varchar(30) NOT NULL,
                    department_id varchar(10) NOT NULL,
                    primary key(grade_id),
                    FOREIGN KEY (department_id) REFERENCES
department(department_id) ON DELETE CASCADE);


create table department(department_id varchar(10),
                                 department_name varchar(30),
                                 primary key (department_id,department_name));
```

Populating the Database

NO Sql queries are being inserted in the backend to insert the data, instead using the SQL queries in the frontend , I have given the input to the dataset, so this is the way of populating the dataset

```
def add_data_Employee(Emp_id,fname,lname ,dob ,phone ,email ,country ,city
,pincode,bank_name,department_id):
        c.execute('INSERT INTO Employee(Emp_id,fname,lname ,dob ,phone ,email
,country ,city ,pincode,bank_name,department_id) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)',
                  (Emp_id,fname,lname ,dob ,phone ,email ,country ,city
,pincode,bank_name,department_id))
        mydb.commit()

def add_data_payroll(transaction_id ,Emp_id ,account_no , bank_name
,Emp_net_sal,Emp_sal_yr ,loan,grade_id,Reinbursment_date):
        c.execute('INSERT INTO payroll(transaction_id ,Emp_id ,account_no ,
bank_name ,Emp_net_sal,Emp_sal_yr ,loan,grade_id,Reinbursment_date) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s)',
                  (transaction_id ,Emp_id ,account_no , bank_name
,Emp_net_sal,Emp_sal_yr ,loan,grade_id,Reinbursment_date))
        mydb.commit()

def add_data_paygrade(grade_id , grade_name ,grade_basic , grade_bonus
,department_id):
        c.execute('INSERT INTO paygrade(grade_id , grade_name ,grade_basic ,
grade_bonus , department_id ) VALUES (%s,%s,%s,%s,%s)',
                  (grade_id , grade_name ,grade_basic ,
grade_bonus,department_id))
        mydb.commit()


def add_data_department(department_id, department_name):
        c.execute('INSERT INTO department(department_id, department_name)
VALUES (%s,%s)',
                  (department_id, department_name))
        mydb.commit()
```

JOIN Queries

```
def joining():
    c.execute('SELECT payroll.Emp_id,
Employee.bank_name,payroll.transaction_id,Employee.city,payroll.loan FROM
payroll JOIN Employee ON payroll.Emp_id = Employee.Emp_id;')
    data = c.fetchall()
    return data

def joining_2():
    c.execute('SELECT department.department_id,department.department_name,
paygrade.grade_id, paygrade.grade_name, paygrade.grade_bonus  FROM paygrade
JOIN department ON paygrade.department_id= department.department_id;')
    data = c.fetchall()
    return data

def joining_3():
    c.execute('SELECT department.department_id, Employee.Emp_id,
Employee.fname,Employee.lname FROM Employee JOIN department ON
Employee.department_id=department.department_id')
    data = c.fetchall()
    return data
```

join:

Display Join On payroll and Employee

	Emp_id	bank_name	transaction_id	city	loan
0	1	CANARA BANK	9874563210	Davanagere	9000
1	2	SBI	9874563211	ROBERTO	5000
2	3	CANARA BANK	9874563212	LA	9999

Display Join On Employee and department

	department_id	Emp_id	fname	lname
0	2	1	samarth	rajendra
1	3	2	leo	messi
2	2	3	taylor	swift

Display Join On paygrade and department

	department_id	department_name	grade_id	grade_name	grade_bonus
0	1	COMPUTERS	1	HIGH_SAL	90000
1	3	EEE	2	MID_SAL	70000
2	4	ME	3	LOW_SAL	50000

Aggregate Functions

```
def aggregate():
    c.execute('SELECT bank_name, COUNT(bank_name) FROM Employee GROUP BY
bank_name;')
    data = c.fetchall()
    return data


def aggregate_2():
    c.execute('SELECT MIN(loan) highest_loan FROM Employee JOIN payroll USING
(Emp_id);')
    #c.execute('SELECT Employee.fname, MIN(loan) LEAST_loan FROM Employee JOIN
payroll ON (Employee.Emp_id=payroll.Emp_id) WHERE payroll.loan=MIN(loan);')
    data = c.fetchall()
    return data

def aggregate_3():
    c.execute('SELECT MAX(loan) highest_loan FROM Employee JOIN payroll USING
(Emp_id);')
    data = c.fetchall()
    return data
```

Aggregate:

Display Bank Count

	Bank_Name	Count
0	CANARA BANK	2
1	SBI	1

Display Max loan by employee

	loan
0	9999

Display MIN loan by employee

	loan
0	5000

SET Operations

```
def union_1():
    c.execute('SELECT payroll.Emp_id FROM payroll UNION SELECT Employee.Emp_id
FROM Employee ')
    data = c.fetchall()
    return data

def intersection_1():
    c.execute('SELECT payroll.Emp_id FROM payroll INTERSECT SELECT
Employee.Emp_id FROM Employee ')
    data = c.fetchall()
    return data
```

Intersection:

Display intersection On payroll and Employee

	Emp_id
0	1
1	2
2	3

UNION:

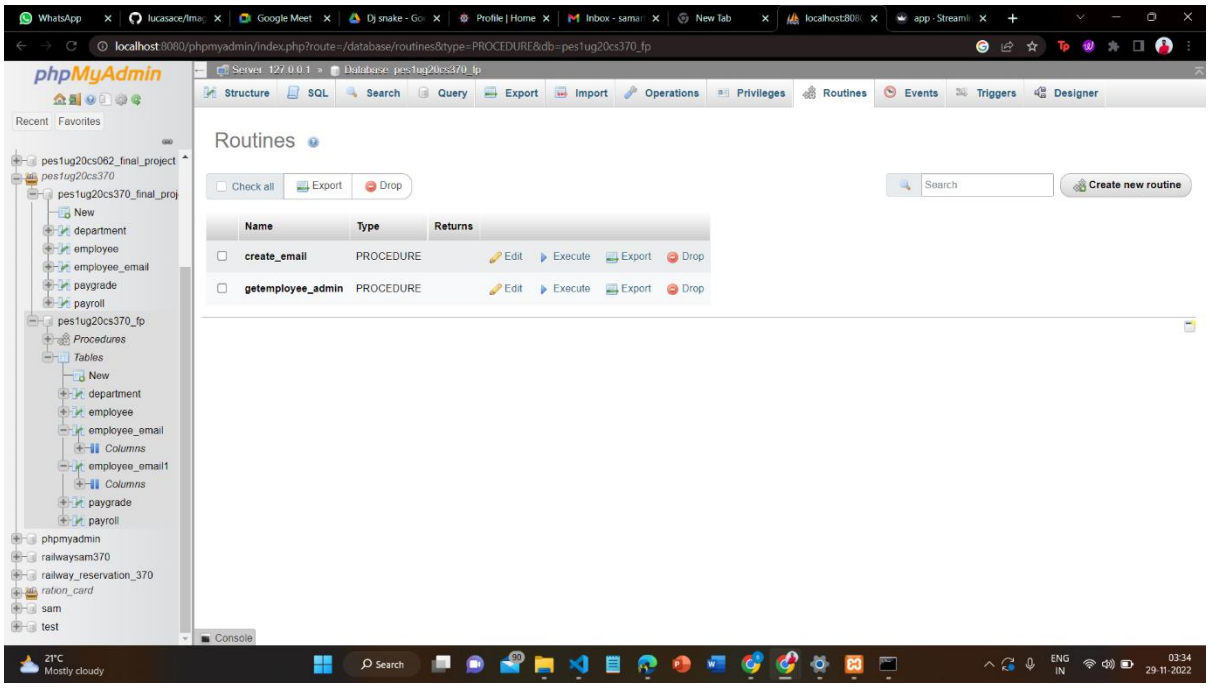
Display union On payroll and Employee

	Emp_id
0	1
1	2
2	3

Functions and Procedures

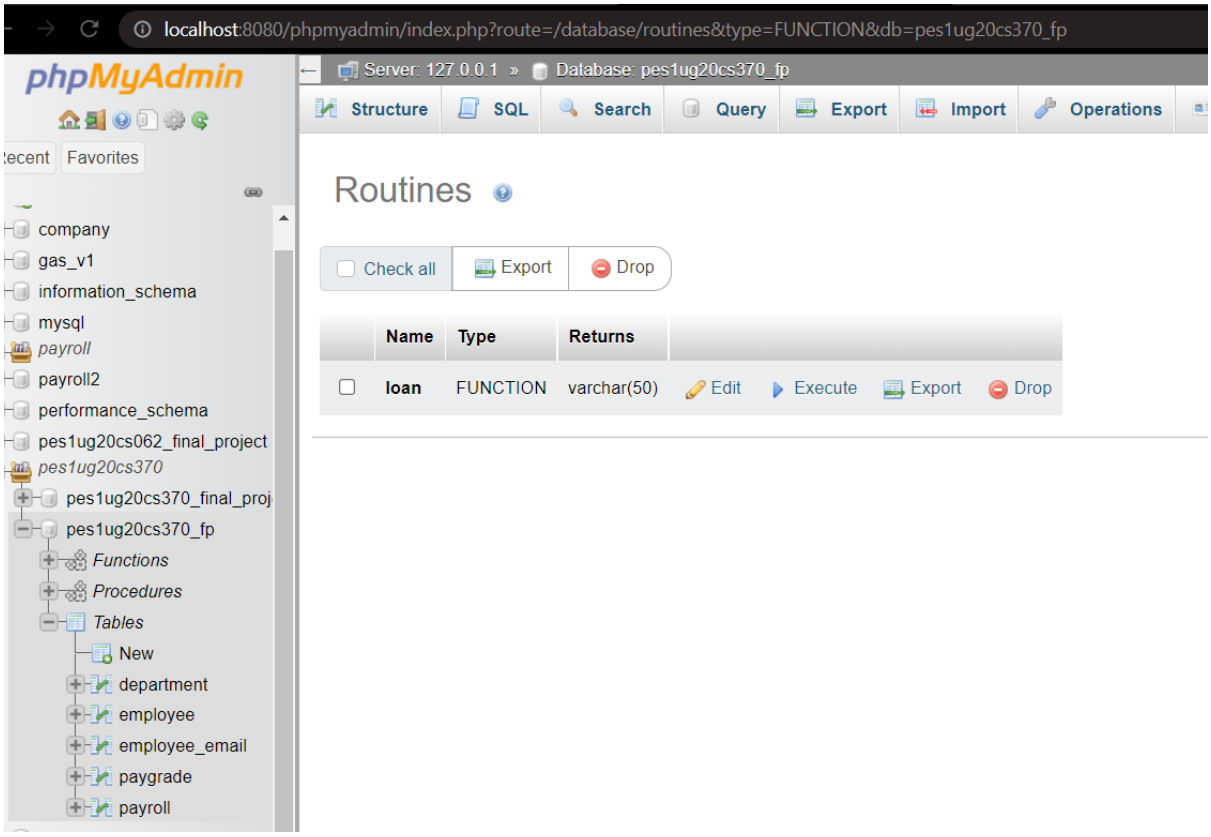
Procedure:

```
DELIMITER $$
CREATE PROCEDURE getemployee_admin(in Emp_id int)
    BEGIN
    select fname, lname
    from employee WHERE employee.Emp_id = Emp_id;
    end $$
delimiter ;
CALL getemployee_admin(1);
```



Function:

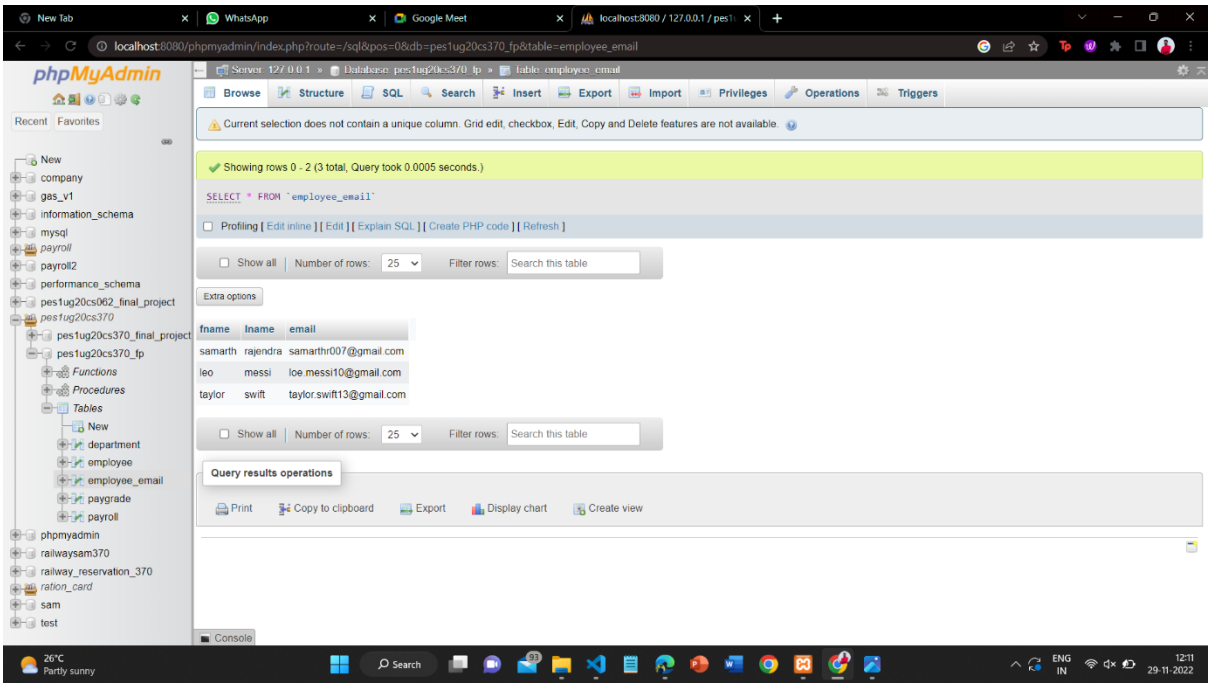
```
DELIMITER $$
CREATE FUNCTION `loan`(balance int) RETURNS varchar(50)
    DETERMINISTIC
BEGIN
DECLARE VALUE varchar(50);

IF balance<10000 then
set VALUE="no";

ELSE
set VALUE ="yes";

end if;

return value;
END$$
DELIMITER ;
```



Triggers and Cursors

Curser:

```
CREATE TABLE EMPLOYEE_EMAIL(
    fname varchar(20),
    lname varchar(20),
    email VARCHAR(50)
);

DELIMITER $$
CREATE PROCEDURE create_email ()
BEGIN
    DECLARE done INTEGER DEFAULT 0;
    DECLARE b_fname varchar(15) ;
    DECLARE b_lname varchar(15) ;
    DECLARE b_email varchar(40) ;
    DEClARE curemail CURSOR FOR SELECT fname, lname, email FROM employee;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN curEmail;

    LABLE: LOOP
    FETCH curemail INTO b_fname, b_lname, b_email;
    IF done = 1 THEN
    LEAVE LABLE;
    END IF;
    INSERT INTO EMPLOYEE_EMAIL VALUES(b_fname, b_lname, b_email);
    END LOOP;

    CLOSE curemail;
END$$

DELIMITER ;
CALL create_email();
```



PROCEDURE:

Display the information which was implemented using curser

	First_name	Last_name	Email
0	samarth	rajendra	samarthr007@gmail.com
1	leo	messi	loe.messi10@gmail.com
2	taylor	swift	taylor.swift13@gmail.com

1) To display the gross salary and net salary of employees

```
CREATE TABLE tax_table(
    fname varchar(20),
    tax varchar(20),
    loan varchar(20),
    Emp_net_sal varchar(20),
    final_sal VARCHAR(50)
);
DELIMITER $$
CREATE PROCEDURE tax_t()
BEGIN
    DECLARE done INTEGER DEFAULT 0;
    DECLARE b_fname varchar(15) ;
    DECLARE b_tax varchar(15) ;
    DECLARE b_loan varchar(40) ;
    DECLARE b_Emp_net_sal varchar(20) ;
    DECLARE b_final_sal varchar(20) ;
    DEClARE curetax CURSOR FOR select fname, Emp_net_sal * 0.30 as tax ,(loan-
(loan*0.1))as loan, Emp_net_sal, Emp_net_sal-(Emp_net_sal * 0.30)-(loan*0.1)
as final_sal
    FROM payroll JOIN Employee ON payroll.Emp_id = Employee.Emp_id
    where Employee.Emp_id=Empi_id;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN curetax;

    LABLE: LOOP
    FETCH curtax INTO b_fname, b_tax, b_loan,b_Emp_net_sal, b_final_sal;
    IF done = 1 THEN
    LEAVE LABLE;
    END IF;
    INSERT INTO tax_table VALUES(b_fname, b_tax, b_loan,b_Emp_net_sal,
b_final_sal);
    END LOOP;

    CLOSE curetax;
END$$

DELIMITER ;
CALL tax_t();
```

PAYROLL MANAGEMENT SYSTEM

PROCEDURE:

Display the information which was implemented using curser

Display the tax table and final salary of all the employees using curser

	First_name	tax	loan	Emp_net_sal	Gross_sal
0	samarth	150000	8100	500000	349100
1	leo	120000	4500	400000	279500
2	taylor	150000	8999.1	500000	349000.1
3	bhargavi	150000	0	500000	350000

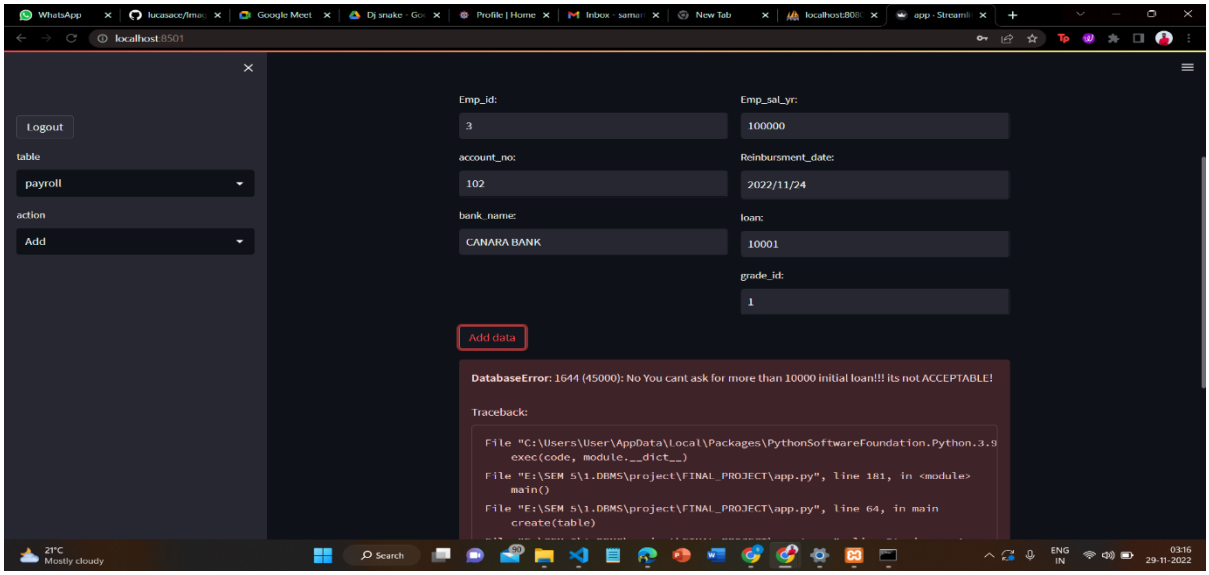
Only to show the emp salary table using join

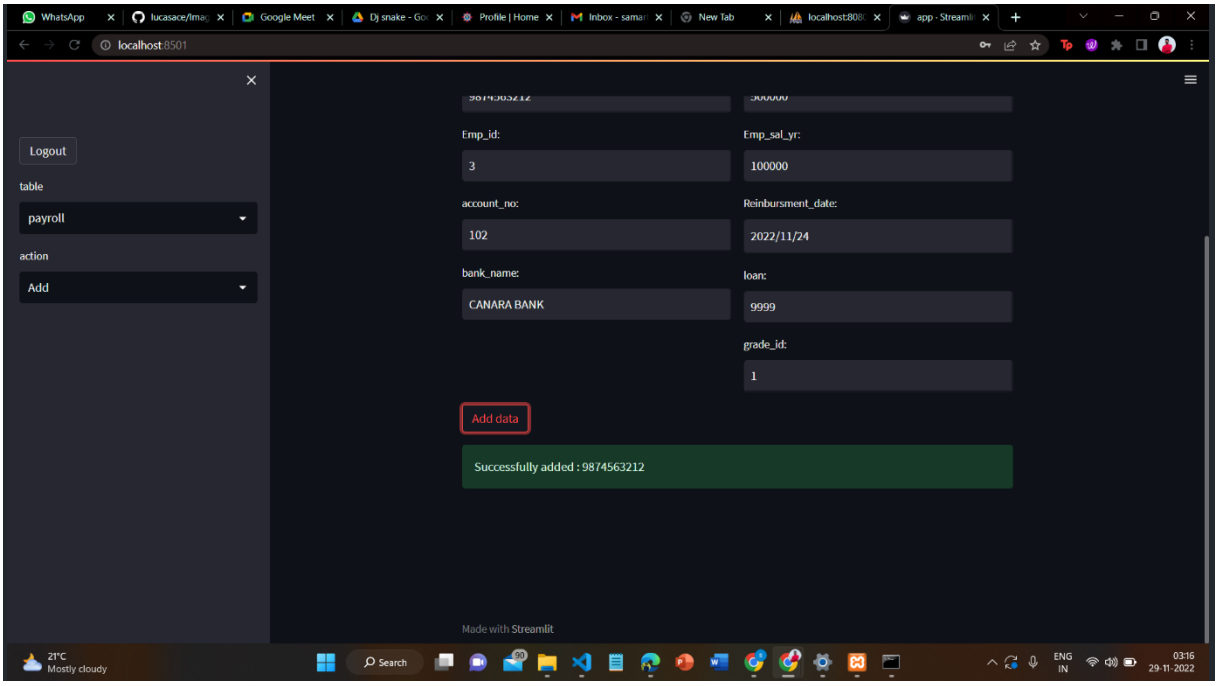
Display Join On payroll and Employee

	Emp_id	bank_name	transaction_id	city	loan	fname	Emp_Net_sal
0	1	CANARA BANK	9874563210	Davanagere	9000	samarth	500000
1	2	SBI	9874563211	ROBERTO	5000	leo	400000
2	3	CANARA BANK	9874563212	LA	9999	taylor	500000
3	4	Union	098765432	blore	0	bhargavi	500000

Trigger:

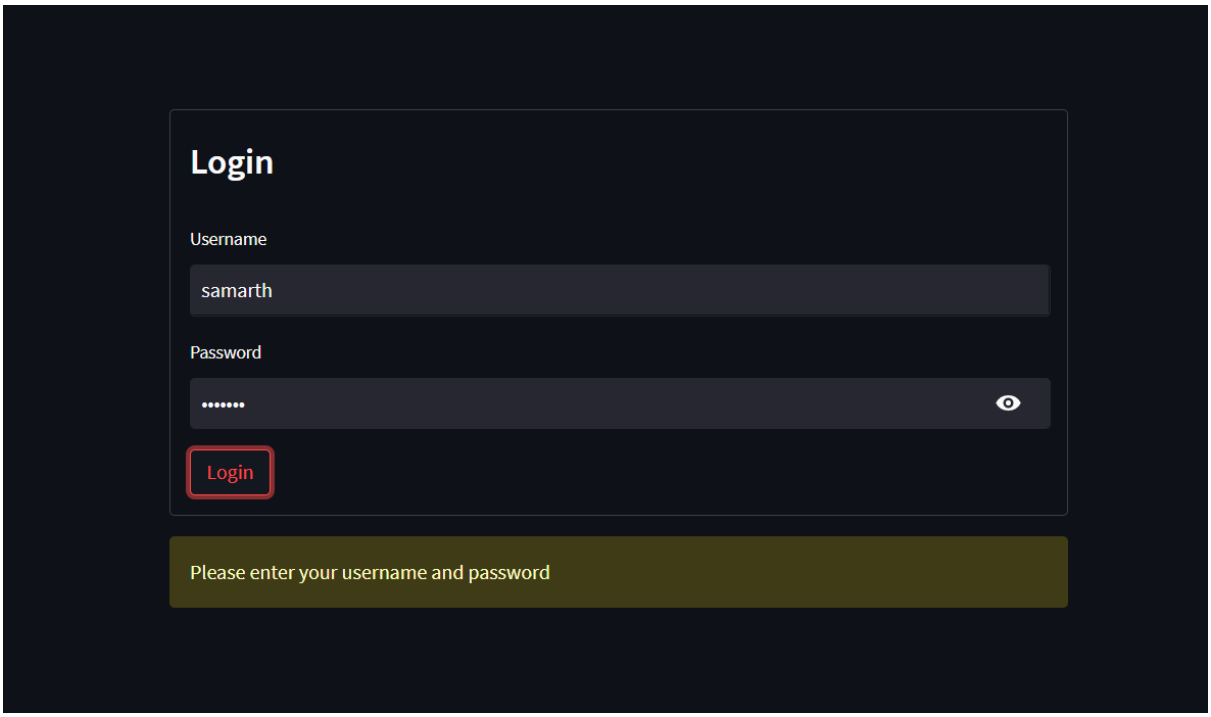
```
DELIMITER $$
CREATE TRIGGER loan_validation1
BEFORE INSERT
ON payroll
FOR EACH ROW
BEGIN
    DECLARE error_msg VARCHAR(300);
    SET error_msg = ("No You cant ask for more than 10000 initial loan!!! its
not ACCEPTABLE!");
    IF new.loan > 10000 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = error_msg;
    END IF;
END $$
DELIMITER ;
```




localhost:8501
Logout
table
payroll
action
Add
Emp_id:
3
Emp_sal_yr:
100000
account_no:
102
Reinbursment_date:
2022/11/24
bank_name:
CANARA BANK
loan:
9999
grade_id:
1
Add data
Successfully added : 9874563212
Made with Streamlit

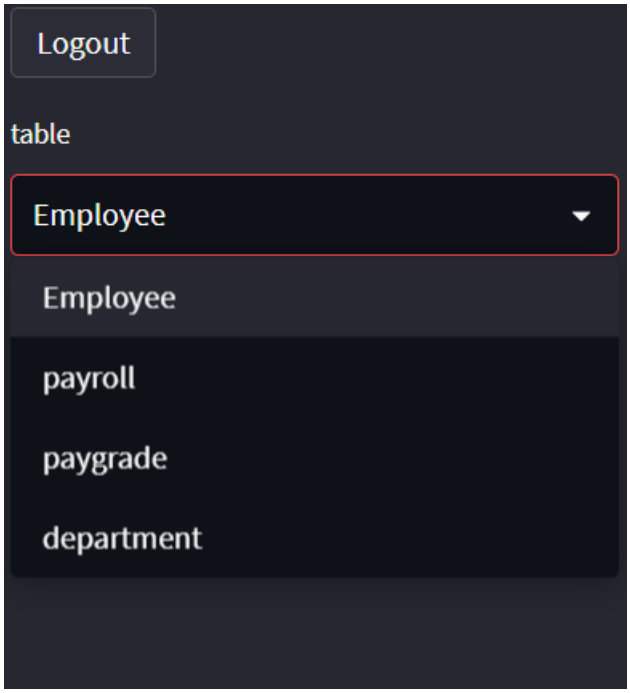
FRONT END

1) login page:

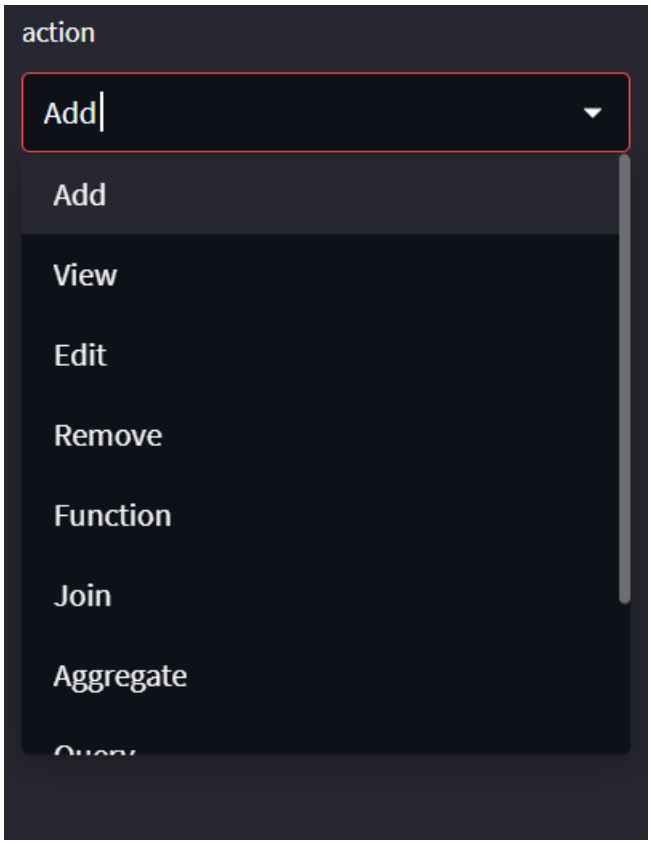


2)after logging in the page looks like this

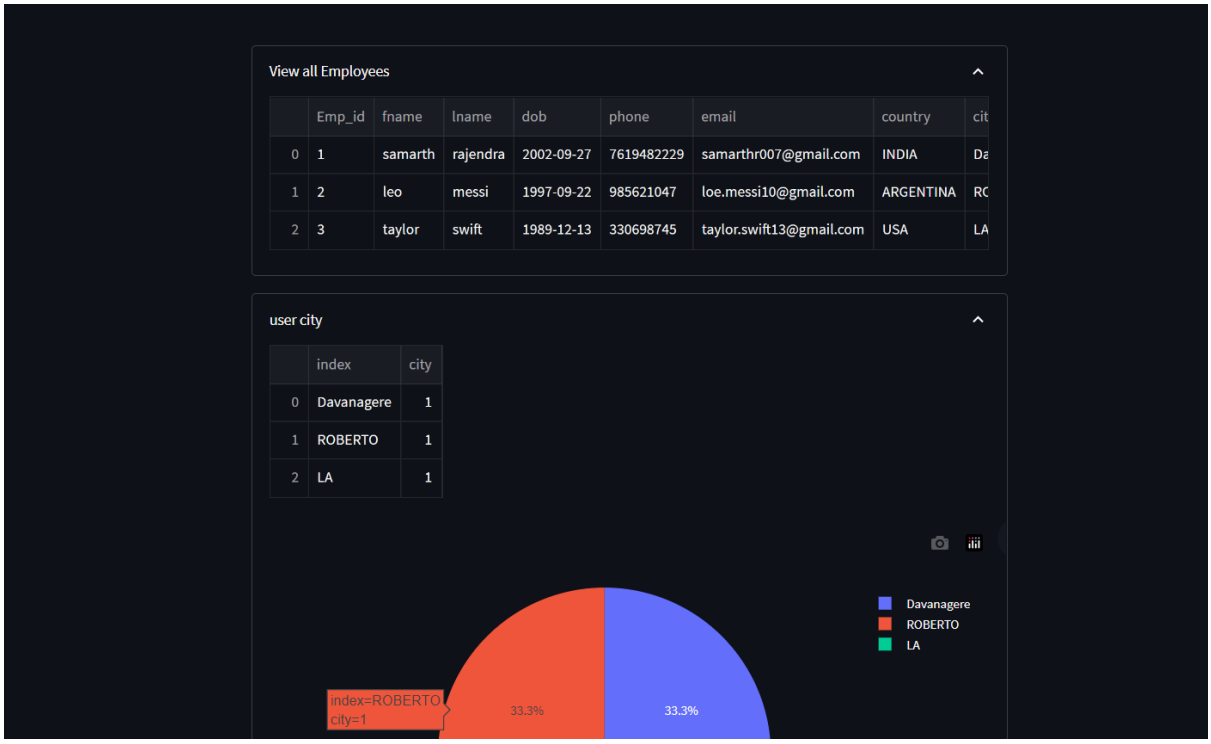
3)tables:



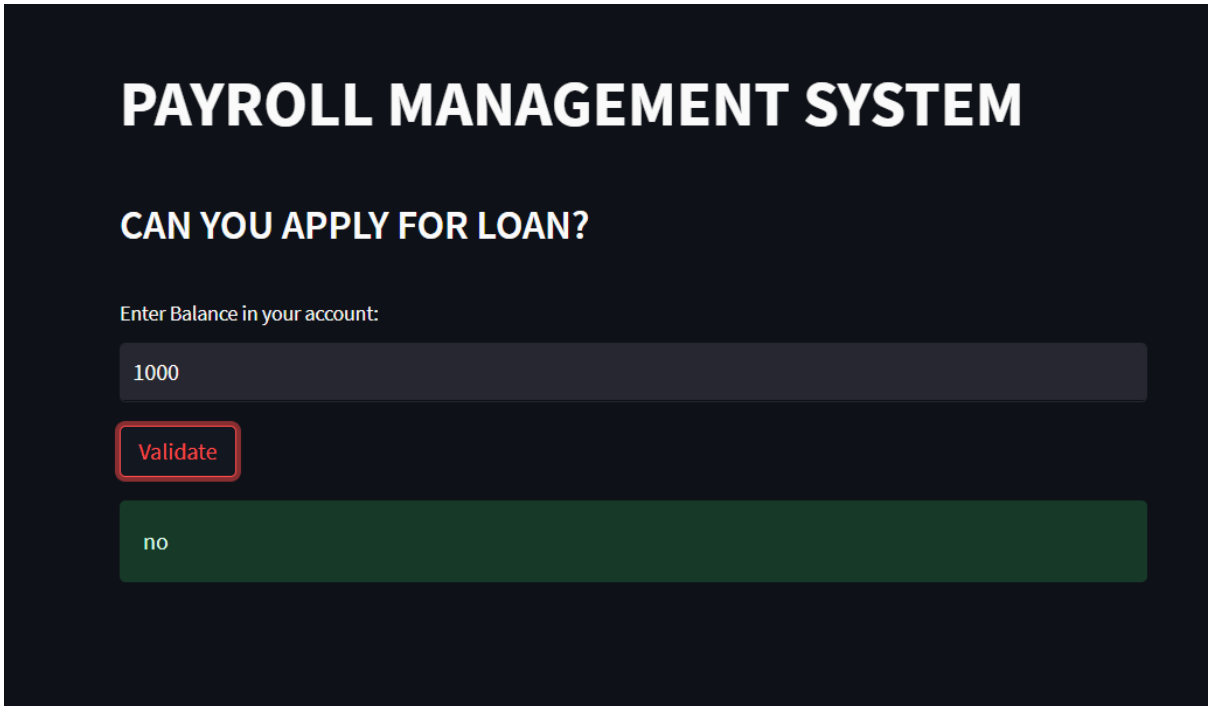
4) action:



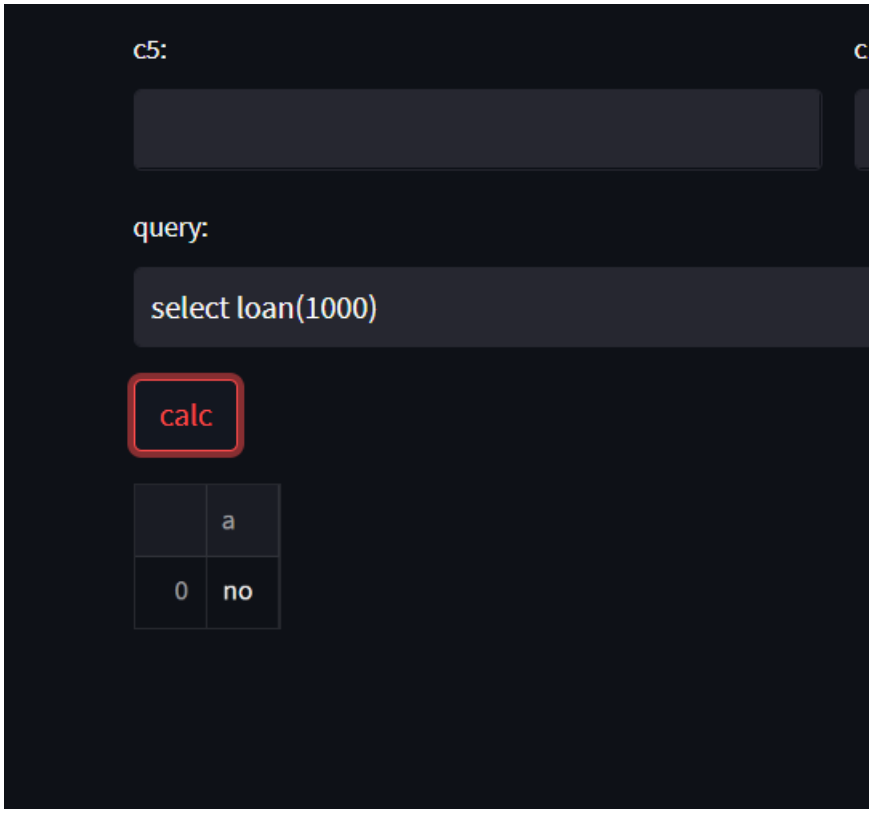
5)view with pie chart



6)function working



7) function loan working from backend in frontend



8)aggregate