

Salary Management System

Mini Project Report -Database Lab (DSE 2260)

Department of Data Science & Computer Applications



B. Tech Data Science

4th Semester – Batch: B3

Submitted By

Apoorv Singhai	200968026
K. Sudhesh Chowdhary	200968027
Arnav Gupta	200968030
Neeil Nandal	200968019

Mentored By

Vinayak M
Assistant Professor-Senior
DSCA, MIT

Archana H/ Shameem
Assistant Professor-Senior
DSCA, MIT



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

Date: 14/5/2022

CERTIFICATE

This is to certify that the K. Sudhesh Chowdhary(200968027), Arnav Gupta(200968030), Neeil Nandal(200968019), Apoorv Singhai(200968026), have successfully executed a mini project titled “**Salary Management System**” rightly brining fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2262 & DSE), thereby resulting in the culmination of this project.

Vinayak M
Assistant Professor-Senior
DSCA, MIT

Archana H / Shameem
Assistant Professor-Senior
DSCA, MIT

ABSTRACT

Salary Management System is aimed at efficient management of employee information, emoluments, expenses, net pay-outs, calculation salary based on workdays and pay salary etc. All of these are managed through a database. This project is terminal project, which implemented using PL/SQL and oracle database. The database is divided based on different conditions which are known as fragments and these fragments are kept at different locations which has Database Management System to deal with the data. The idea of dividing/fragmenting the data makes the system reliable, fast with better response. In case of database failures, the system remains functional though it may reduce performance.

Contents

1. Introduction	5
2. Synopsis	6
2.1 Proposed System	6
2.2 Objectives	6
3. Functional Requirements	7
4. Detailed Design	
4.1 ER Diagram	10
4.2 Schema Diagram	11
4.3 Data Dictionary	12
4.4 Relational Model Implementation	13
4.5 Insert	14
4.6 Packages	15
4.7 Triggers	17
4.8 Procedures	20
4.9 Functions	21
5. Conclusion	30

INTRODUCTION

The proposed project “Salary 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. Companies, it may be public or private increasing to fast with population growth. And in every company, there is a must essential thing, a system which manage employee information and their payments. So having a well-organized Salary Management System is a market demand. So, we tried to implement a smaller conceptual version of Salary Management System. And as there are some large companies has distributed database system, so we focus on distributed database system in small manner so that system can perform fast and efficient ways.

Synopsis

2.1 Proposed System

Salary Management System is aimed at efficient management of employee information, emoluments, expenses, net pay-outs, calculation salary based on workdays and pay salary etc. Having such a Management system is well in demand. There are three types of user for our project – Admin, Accountant, Employees.

2.2 Objectives

- To view employee details
- To handle funds of an employee
- To add details of employees or salaries
- To carry out transactions

Functional Requirements

1. Admin (Employer or HR)
 - Add new employee
 - Assign salary
 - Change employee Post
2. Accountant
 - Add Funds
 - Pay Salary
 - Handle funds
3. Employees
 - View payments

3.1 User Registering/Login module

3.1.1 New User Registration

User registers by giving appropriate details of oneself, which can be submitted and verified to the concerned authorities.

3.1.2 Login.

INPUT	username, password
OUTPUT	If correct details are entered Login is successful Else Login not successful, retry logging in

3.1.3 Forgot password

If existing user name is not bale to login, forgot password can be used to reset password.

INPUT	Prompt user to enter username, Phone
Processing	If username and corresponding phone exist in the data storage Send OTP to Phone. Prompt the user to enter OTP If OTP matching Prompt user to change password according to criteria. Else OTP not matching. Else User name and corresponding Phone not existing in the storage
OUTPUT	Password successfully changed / User name, phone not matching

3.1.4 Employee Module

- ViewDetails()

Usage : Show employee his/her details with payment history

Parameter: EmployeeID

Output : Show details of given employee ID

3.1.5 Accountant Module

- PaySalary()

Usage : Pay Employees Salaries by a function GenerateSalary() and a procedure TransectSalary() call

Parameter: Employee ID, Month of Giving salary

Output : Pay salary of given EID and month

Procedures

- AddFund()
Usage : Increase fund
Parameter: Amount
Output : Increase fund with given amount
- AddLeave()
Usage : Add leaves of Employees of specific month
Parameter: EID, S_month, L_days
Output : Add leaves of given Employee ID, Month
- TransectSalary()
Usage : It's the transection procedure for paymets use
functionChechValid() and procedure UpdateFund() to complete
Parameter: EID, amount, month
Output : Transection for payments
- UpdateFund()

Usage : After transection the number of payments will be reduce by this
procedure
Parameter: Amount
Output: Updated fund after transection

Functions

- GenerateSalary()
Usage : Generate salary calculate with leaves of given month
Parameter: EID, month
Return: Generated salary
- CheckValid()
Usage: This says either its payable or not
Parameter: EID, month
Return: 1 for valid 2 for invalid

3.1.5 Administrator Module

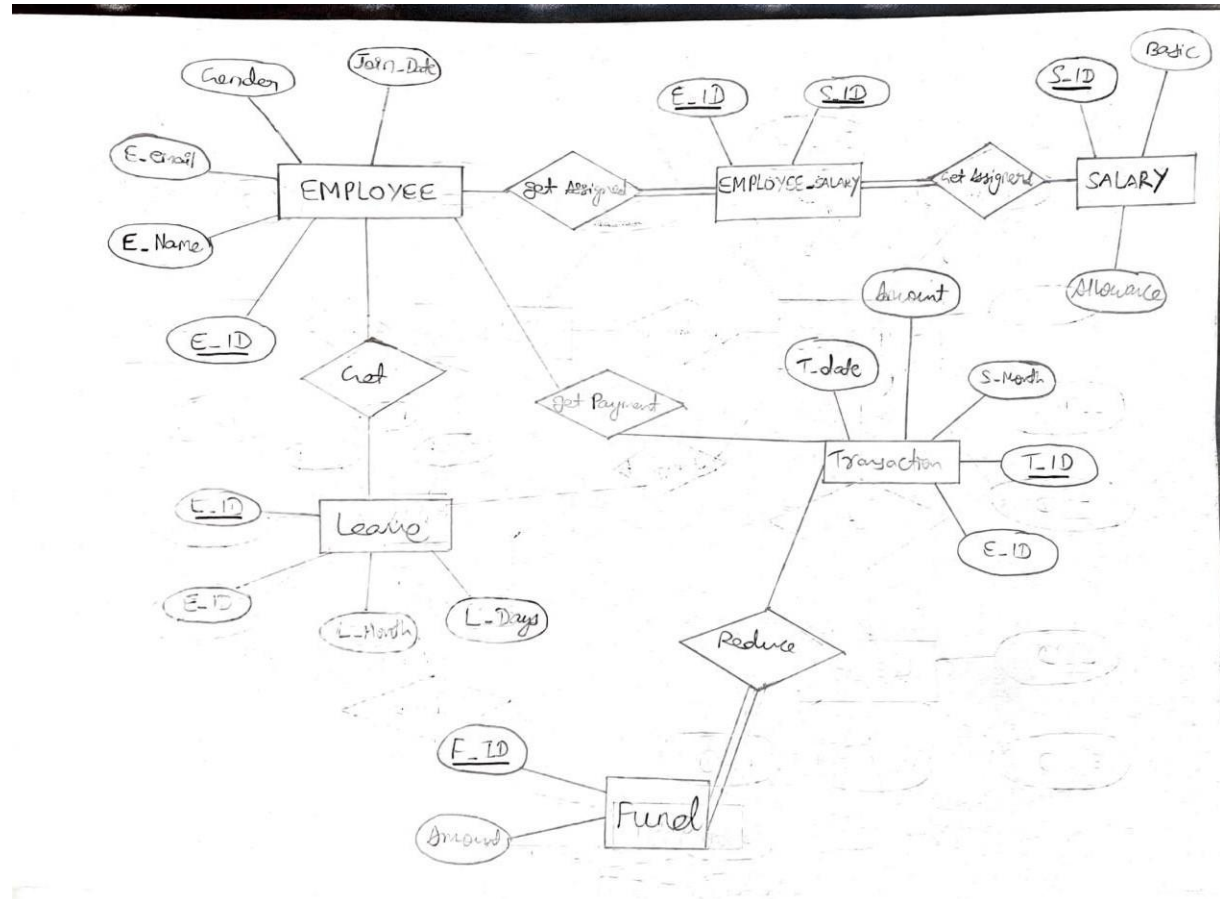
- **AddEmployee()**
Usage: Add new employee and assign his salary
Parameter: Name, gender, email, joiningDate, SID
Output: Add the person as employee and assigned his salary
- **ChangeEmpPost()**
Usage: Employee promote or demote
Parameter: EID, SID
Output: Changed Pay Scale with given EID, SID

3.1.5 Triggers

- AddLeaves: Triggered if data insert into Leave table.
- ChangeEmpSalary: Triggered if data update in Employee_salary Table and there is an audit table to track changes.
- AddEmployee: Triggered if data insert into Employee table.
- AddEmpSalary: Triggered if new employee get assigned salary.
- Transect: Triggered when transection occurs.
- UpdateFund: Triggered id data update in Fund Table and there is an audit table to track changes.

Detailed Design

4.1 ER Diagram



4.2 Schema Diagram

- Employee (EID, EName, Gender, Email, JoinDate)
- Salary (SID, Basic, Allowance)
- Employee_Salary (EID(FK), SID(FK),)
- Leave (LID, EID(FK), L_month, L_days)
- Transection (TID, EID(FK), Amount, T_Date, S_month)
- Fund (FID, Fund_amount)

Employee_Salary(EID,SID) references Employee(EID) and Salary (SID)

Leave(EID) references Employee(EID)

Transection(EID) references Employee(EID)

4.3 Data Dictionary

EMPLOYEE

Column	Data type (size)	Constraint	Constraint Name
eid	int	Primary Key	
ename	varchar2(20)		
gender	varchar2(5)	check gender in 'M','F','Male','Female'	
email	varchar2(20)	like '% @ %'	
join_date	varchar2(20)		

SALARY

Column	Data type (size)	Constraint	Constraint Name
sid	int	Primary Key	
basic	int		
allowance	int		

EMPLOYEE_SALARY

Column	Data type (size)	Constraint	Constraint Name
eid	int	FOREIGN KEY referencing EMPLOYEE	
sid	int	FOREIGN KEY referencing SALARY	

TRANSECTION

Column	Data type (size)	Constraint	Constraint Name
tid	int	PRIMARY KEY	
eid	int	FOREIGN KEY referencing EMPLOYEE	
ammount	int		
t_date	date		
s_month	varchar2(15)		

FUND

Column	Data type (size)	Constraint	Constraint Name
fid	int	Primary Key	
fund_amnt	int		

TABLE FUND_Audit

Column	Data type (size)	Constraint	Constraint Name
amnt_new	int		
amnt_old	int		
Update_date	varchar2(30)		

EMPLOYEE_SALARY_Audit

Column	Data type (size)	Constraint	Constraint Name
new_sid	int		
old_sid	int		
Changing_date	varchar2(30)		

4.4 Relational Model Implementation

```
CREATE TABLE EMPLOYEE (  
    eid int,ename varchar2(20),  
    gender varchar2(5) check (gender in('M','F','Male','Female')),  
    email varchar2(25) check (email like '%@%'),  
    join_date varchar2(20) ,  
    PRIMARY KEY(eid));
```

```
CREATE TABLE SALARY(  
    sid int,  
    basic int,  
    allowance int,  
    PRIMARY KEY(sid));
```

```
CREATE TABLE EMPLOYEE_SALARY(  
    eid int,  
    sid int,  
    FOREIGN KEY(eid) REFERENCES EMPLOYEE(eid),  
    FOREIGN KEY(sid) REFERENCES SALARY(sid));
```

```
CREATE TABLE LEAVE(  
    lid int,  
    eid int,  
    l_month varchar2(15),  
    l_day int,  
    PRIMARY KEY(lid));
```

```
CREATE TABLE TRANSECTION (  
    tid int,  
    eid int,  
    ammount int,  
    t_date date,  
    s_month varchar2(15),  
    PRIMARY KEY(tid),  
    FOREIGN KEY(eid) REFERENCES EMPLOYEE(eid));
```

```
CREATE TABLE FUND (  
    fid int,  
    fund_amnt int,  
    PRIMARY KEY(fid));
```

```
CREATE TABLE FUND_Audit (  
    amnt_new int,  
    amnt_old int,  
    Update_date varchar2(30));
```

```
CREATE TABLE EMPLOYEE_SALARY_Audit(  

```

```
new_sid int,  
old_sid int,  
Changing_date varchar2(30));
```

4.5 Insert

```
insert into employee values (1, 'Sajid Abdullah', 'M', 'sajid@gamil.com',  
'1/1/2019');  
insert into employee values (2, 'Samia Zahan', 'F', 'samia@gamil.com', '1/1/2019');  
insert into employee values (3, 'Muna Saha', 'F', 'muna@gmail.com', '1/1/2019');  
insert into employee values (4, 'Robiul Hasan', 'M', 'nowshad@gamil.com',  
'1/1/2019');  
insert into employee values (5, 'Apoorva Mitai', 'M', 'apoorv@yahoo.com',  
'1/1/2019');
```

```
insert into salary values(1, 18000,5000);  
insert into salary values(2, 20000,5000);  
insert into salary values(3, 22000,6000);  
insert into salary values(4, 35000,6500);  
insert into salary values(5, 50000,7000);
```

```
insert into employee_salary values(1,1);  
insert into employee_salary values(2, 3);  
insert into employee_salary values(3,5);  
insert into employee_salary values(4,2);  
insert into employee_salary values(5,1);
```

```
insert into leave values(1,1, 'Jan/19', 3);  
insert into leave values(2,3, 'Jan/19', 4);  
insert into leave values(3,2, 'Jan/19', 5);  
insert into leave values(4,6, 'Jan/19', 3);  
insert into leave values(5,4, 'Jan/19', 1);
```

4.6 Packages

```
set serveroutput on;
```

```

create or replace package emp_proc_func as
function emp_name(emp_no in number) return varchar;
procedure prj_emp_name(prj_no in char);
procedure prj_date(pstart_date in date);
procedure work_force;
function emp_desig(idesig in varchar) return number;
end emp_proc_func;
/

```

```

create or replace package body emp_proc_func as

```

```

function emp_name(emp_no in number) return varchar is
ename emp.name%type;
reports emp.reports_to%type;
begin
select reports_to into reports from emp where empcode=emp_no;
select name into ename from emp where empcode=reports;
return(ename);
end emp_name;

```

```

procedure prj_emp_name(prj_no in char) is
cursor cemp is select name from emp natural join work_exp where prjid=prj_no;
begin
for c in cemp loop
dbms_output.put_line(c.name);
end loop;
end prj_emp_name;

```

```

procedure prj_date(pstart_date in date) is
cursor cprj is select prj_name from prj_details where start_date=pstart_date;
begin
for c in cprj loop
dbms_output.put_line(c.prj_name);
end loop;
end prj_date;

```

```

procedure work_force is
cursor cskill is select skillid, skillname from skill;
cursor cemp is select empcode, name, skillid, skill_experience from skill natural
join emp_skill join emp on emp_skill.empno=emp.empcode;
begin
for cs in cskill loop
dbms_output.put_line('Skill ID: '||cs.skillid);
dbms_output.put_line('Skill Name: '||cs.skillname);
for ce in cemp loop
if(ce.skillid=cs.skillid) then
dbms_output.put_line('Employee Code: '||ce.empcode);
dbms_output.put_line('Employee Name: '||ce.name);
dbms_output.put_line('Skill Experience: '||ce.skill_experience);
end if;
end loop;
end loop;
end work_force;

```

```

function emp_desig(idesig in varchar) return number is
no number(2);
begin
select count(empcode) into no from emp where designation=idesig;
return no;
end emp_desig;

end emp_proc_func;
/

```

4.7 Triggers

TriggerAddLeave.sql

```

SET SERVEROUTPUT ON;

CREATE OR REPLACE trigger EmpSalary_audit
BEFORE UPDATE ON EMPLOYEE_SALARY
FOR EACH ROW
DECLARE
    v_date varchar2(30);
begin
    select sysdate into v_date from dual;
    insert into EMPLOYEE_SALARY_Audit values
(:NEW.sid,:OLD.sid,v_date);
    dbms_output.put_line('Salary Change for The Employee');
    commit;

```

```
end;  
/
```

TriggerChangeEmpSalaryInsert.sql

```
SET SERVEROUTPUT ON;  
  CREATE OR REPLACE trigger EmpSalary_audit  
  BEFORE UPDATE ON EMPLOYEE_SALARY  
  FOR EACH ROW  
  DECLARE  
      v_date varchar2(30);  
  begin  
      select sysdate into v_date from dual;  
      insert into EMPLOYEE_SALARY_Audit values  
(:NEW.sid,:OLD.sid,v_date);  
      dbms_output.put_line('Salary Change for The Employee');  
      commit;  
  end;  
/
```

TriggerEmpSalaryInsert.sql

```
SET SERVEROUTPUT ON;  
  create or replace trigger Trig_EmpSalaryInsert  
  after insert on EMPLOYEE_SALARY  
  begin  
      dbms_output.put_line('Salary Assigned To the Employee');
```

```
end;  
/
```

TriggerEmployeeInsert.sql

```
SET SERVEROUTPUT ON;  
  
create or replace trigger Trig_EmployeeInsert  
after insert on Employee  
begin  
    dbms_output.put_line('One Employee Added');  
end;  
/
```

TriggerTransect.sql

```
SET SERVEROUTPUT ON;  
  
create or replace trigger Trig_Tansect  
after insert on transection  
begin  
    dbms_output.put_line('One Transection completed');  
end;  
/
```

TriggerUpdate.sql

```
SET SERVEROUTPUT ON;  
  
CREATE OR REPLACE trigger Fund_audit  
BEFORE UPDATE ON FUND  
FOR EACH ROW
```

```

DECLARE
    v_date varchar2(30);
begin
    select sysdate into v_date from dual;
    insert into FUND_Audit values
(:NEW.fund_amnt,:OLD.fund_amnt,v_date);
    dbms_output.put_line('Fund Updated');
    commit;
end;
/

```

4.8 Procedures

ProcedureAddLeaves.sql

```

create or replace Procedure AddLeave(v_lid in number, v_eid in
number,v_l_month in varchar2,v_l_day in number)
is
begin
    insert into leave values(v_lid,v_eid, v_l_month, v_l_day);
    commit;
end AddLeave;
/

```

ProcedureChangeEmpSalary.sql

```

create or replace Procedure ChangeEmployeeSalary(v_eid in number,v_sid in
number)
is
Begin
    update EMPLOYEE_SALARY set sid=v_sid where eid=v_eid;

```

```

commit;
EXCEPTION
    When no_data_found then
        DBMS_OUTPUT.PUT_LINE('No Data Found');
    When others then
        DBMS_OUTPUT.PUT_LINE('Something wrong
happened');
end ChangeEmployeeSalary;
/

```

ProcedureUpdateFund.sql

```

create or replace Procedure Update_Fund(n1 in number)
    is
        v_fund_amnt fund.fund_amnt %TYPE;
Begin
    select fund_amnt into v_fund_amnt from fund where fid=1;
    v_fund_amnt:= v_fund_amnt-n1;
    update FUND set fund_amnt=v_fund_amnt where fid=1;
    commit;
EXCEPTION
    When no_data_found then
        DBMS_OUTPUT.PUT_LINE('No Data Found');
    When others then
        DBMS_OUTPUT.PUT_LINE('Something wrong
happened');
end Update_Fund;
/

```

4.9 Functions

AccountantAddFund.sql

```
SET SERVEROUTPUT ON;
```

```
SET VERIFY OFF;
```

```
DECLARE
```

```
X number:= &Fund_Amount_to_add;
```

```
v_fund_amnt number;
```

```
Begin
```

```
    select fund_amnt into v_fund_amnt from fund where fid=1;
```

```
    dbms_output.put_line('Old Fund: '||v_fund_amnt);
```

```
    v_fund_amnt:= v_fund_amnt+X;
```

```
    dbms_output.put_line('New Fund: '||v_fund_amnt);
```

```
    update FUND set fund_amnt=v_fund_amnt where fid=1;
```

```
    commit;
```

```
EXCEPTION
```

```
When no_data_found then
```

```
    DBMS_OUTPUT.PUT_LINE('No Data Found');
```

```
When others then
```

```
    DBMS_OUTPUT.PUT_LINE('Something wrong  
happened');
```

```
end;
```

```
/
```

AccountantAddLeave.sql

```
SET SERVEROUTPUT ON;
SET VERIFY OFF;
DECLARE
    v_lid LEAVE.lid %type;
    v_eid LEAVE.eid %type:=&EmployeeID;
    v_l_month LEAVE.l_month %type:='&Month';
    v_l_day LEAVE.l_day %type:=&No_of_Leaves;
BEGIN
    SELECT lid
    into v_lid
    FROM (select * from LEAVE ORDER BY lid DESC) leave1
    WHERE rownum <= 1 ORDER BY rownum DESC;
    v_lid:=v_lid+1;
    AddLeave(v_lid,v_eid,v_l_month,v_l_day);

    EXCEPTION
    When no_data_found then
        DBMS_OUTPUT.PUT_LINE('No Data Found');
    When others then
        DBMS_OUTPUT.PUT_LINE('Something wrong
happened');
    END;
/
```

AccountantPaySalary.sql

```
SET SERVEROUTPUT ON;
SET VERIFY OFF;
DECLARE
X number:= &Eid;
Y varchar2(15):= '&Month_of_Salary';
r number;
Begin
r := myPackage.Generate_Salary(X,Y);
DBMS_OUTPUT.PUT_LINE(r);
myPackage.Transect_Salary(X,r,Y);
commit;
end;
/
```

AdminAddEmployee.sql

```
SET VERIFY OFF;
DECLARE
v_eid EMPLOYEE.eid %TYPE;
v_ename EMPLOYEE.ename %TYPE := '&Name';
v_gender EMPLOYEE.gender %TYPE := '&Gender';
v_email EMPLOYEE.email %TYPE := '&Email';
v_joinDate EMPLOYEE.join_date %TYPE := '&JoinDate';
v_sid employee_salary.sid %TYPE := '&Salary';
BEGIN
SELECT eid
into v_eid
```

```

FROM (select * from EMPLOYEE ORDER BY eid DESC) Emp1
WHERE rownum <= 1 ORDER BY rownum DESC;
v_eid:=v_eid+1;
insert into employee values (v_eid, v_ename, v_gender,v_email,
v_joinDate);
insert into employee_salary values(v_eid,v_sid);
commit;
EXCEPTION
When no_data_found then
    DBMS_OUTPUT.PUT_LINE('No Data Found');
When others then
    DBMS_OUTPUT.PUT_LINE('Something wrong
happened');
END;
/

```

AdminEmpChangeSalary.sql

```
SET VERIFY OFF;

DECLARE

    v_eid EMPLOYEE.eid %TYPE:=&EmpolyeeID;
    v_sid employee_salary.sid%TYPE:=&NewSalaryId;

BEGIN

    ChangeEmployeeSalary(v_eid,v_sid);

END;

/
```

EmployeePayments.sql

```
SET SERVEROUTPUT ON;

SET VERIFY OFF;

clear screen;

DECLARE

    X number:= &Your_EmployeeID;

    --v_eid EMPLOYEE.eid %TYPE;

    v_ename EMPLOYEE.ename %TYPE;

    v_gender EMPLOYEE.gender %TYPE;

    v_email EMPLOYEE.email %TYPE;

    v_joinDate EMPLOYEE.join_date %TYPE;

    v_l_day leave.l_day%TYPE;

    v_basic salary.basic%TYPE;

    v_allowance salary.allowance%TYPE;

    v_sid employee_salary.sid%TYPE;

    n1 number;

    v_tid transection.tid %TYPE;

    v_eid transection.eid %TYPE;

    v_ammount transection.ammount %TYPE;
```

```

v_t_date transection.t_date %TYPE;

v_s_month transection.s_month %TYPE;

--s1 varchar2;

cursor payments_cur is
select tid, eid, s_month,t_date,ammount from
TRANSECTION@site_link where eid=X;

Begin

select ename,gender,email,join_date
into v_ename,v_gender,v_email,v_joinDate
from EMPLOYEE@site_link where eid=X;

select sid into v_sid from employee_salary@site_link where
eid=X;

select basic,allowance into v_basic,v_allowance from
salary@site_link where sid=v_sid;

dbms_output.put_line('_____');
dbms_output.put_line('Employee Details_____');
dbms_output.put_line('Employee ID   :'||X);
dbms_output.put_line('Name       : '||v_ename);
dbms_output.put_line('Gender    : '||v_gender);
dbms_output.put_line('Email     : '||v_email);
dbms_output.put_line('Joining Date : '||v_joinDate);
dbms_output.put_line('_____');
dbms_output.put_line('Salary Details_____');
dbms_output.put_line('Basic      : '||v_basic);
dbms_output.put_line('Allowance  : '||v_allowance);
n1:=v_basic+v_allowance;
dbms_output.put_line('Total Salary : '||n1);
dbms_output.put_line('_____');
dbms_output.put_line('Payments Details_____');

```

```

        dbms_output.put_line('TransectionID EmployeeID
MonthOfSalary Leaves PaymentDate Amount');
        open payments_cur;
        loop
            fetch payments_cur into
v_tid,v_eid,v_s_month,v_t_date,v_ammount;
            exit when payments_cur%notfound;
            select l_day into v_l_day from leave@site_link where
eid=X and l_month=v_s_month;
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_tid)||
'||TO_CHAR(v_eid)||'      '||TO_CHAR(v_s_month)||'
'||TO_CHAR(v_l_day)||'    '||TO_CHAR(v_t_date)||'
'||TO_CHAR(v_ammount));
            end loop;
        close payments_cur;

```

EXCEPTION

```

        WHEN no_data_found THEN
            dbms_output.put_line('THIS EMPLOYEE DOESNT EXIST!');
        WHEN others THEN
            dbms_output.put_line('ERROR!');
        end;
/

```

FunctionCheckValid.sql

```

create or replace function Check_Valid(n1 in number, s1 in varchar2)
    return number
is

    temp_eid transection.eid %TYPE:= n1;

```

```

temp_s_month transection.s_month %TYPE:=s1;
v_eid transection.eid %TYPE;
v_s_month transection.s_month %TYPE;
return_val number:=0;
cursor my_cur is
select eid, s_month from TRANSECTION;

BEGIN

OPEN my_cur;
    loop
        fetch my_cur into v_eid,v_s_month;
        exit when (my_cur%notfound or return_val=1) ;
        if((v_eid=temp_eid) and
(v_s_month=temp_s_month)) then
            return_val:=1;
        else
            return_val:=2;
        end if;
    end loop;
CLOSE my_cur;
return return_val;

EXCEPTION
When no_data_found then
    DBMS_OUTPUT.PUT_LINE('No Data Found');
When others then
    DBMS_OUTPUT.PUT_LINE('Something wrong
happened');
end Check_Valid;

```

/

5. Conclusion

Established an idea of how distributed database work in real life scenario.
Worked through the management system and discovered how the working takes place. We look forward to implement the project in future on a larger scale.