**Salary Management**

A PROJECT REPORT

SUBMITTED IN COMPLETE FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

BACHELOR OF TECHNOLOGY (COE)



**SUBMITTED BY:**

**ARCHIT ASHRI (2K21/CO/89) Arjit Tyagi (2K21/CO/93)**

**UNDER THE SUPERVISION OF:**

**PROF. SHAILENDER KUMAR**

**DELHI TECHNOLOGICAL UNIVERSITY**

**(FORMERLY DELHI COLLEGE OF ENGINEERING)**

**BAWANA ROAD, DELHI – 110042**

**MAY 2023**

1

# CANDIDATE’S DECLARATION

I, ARCHIT ASHRI (2K21/CO/89) and ARJIT TYAGI (2K21/CO/93) student of Bachelor of Technology,

Hereby, declare that the dissertation titled “Salary Management” which is submitted by us to the Department of Computer Engineering, Delhi Technological

University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is original and is not copied from any source without proper citation.

This work has not previously formed the basis of any award of any degree, diploma associateship, fellowship or any other similar title or recognition.

Place: Delhi ARCHIT ASHRI

## (2K21/CO/89)

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING) BAWANA ROAD, DELHI – 110042

**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
2. Synopsis

[Proposed System](#_TOC_250008)

[Objectives](#_TOC_250007)

1. Functional Requirements
2. Detailed Design

[ER Diagram](#_TOC_250006)

[Schema Diagram](#_TOC_250005)

[Relational Model Implementation](#_TOC_250004)

Insert

Packages

Triggers

Procedures

Functions

1. [Conclusion](#_TOC_250000)

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 asthere 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

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.

### 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

#### User Registering/Login module

##### New User Registration

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

##### Login.

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

**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 |

##### Employee Module

* + - * ViewDetails()

Usage : Show employee his/her details with payment history Parameter: EmployeeID

Output : Show details of given employee ID

##### Accountant Module

* + - * PaySalary()

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

Procedures

Parameter: Employee ID, Month of Giving salary Output : Pay salary of given EID and month

* + - * 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 transaction the number of payments will be reduce by this procedure

Parameter: Amount

Output: Updated fund after transaction

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

##### 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

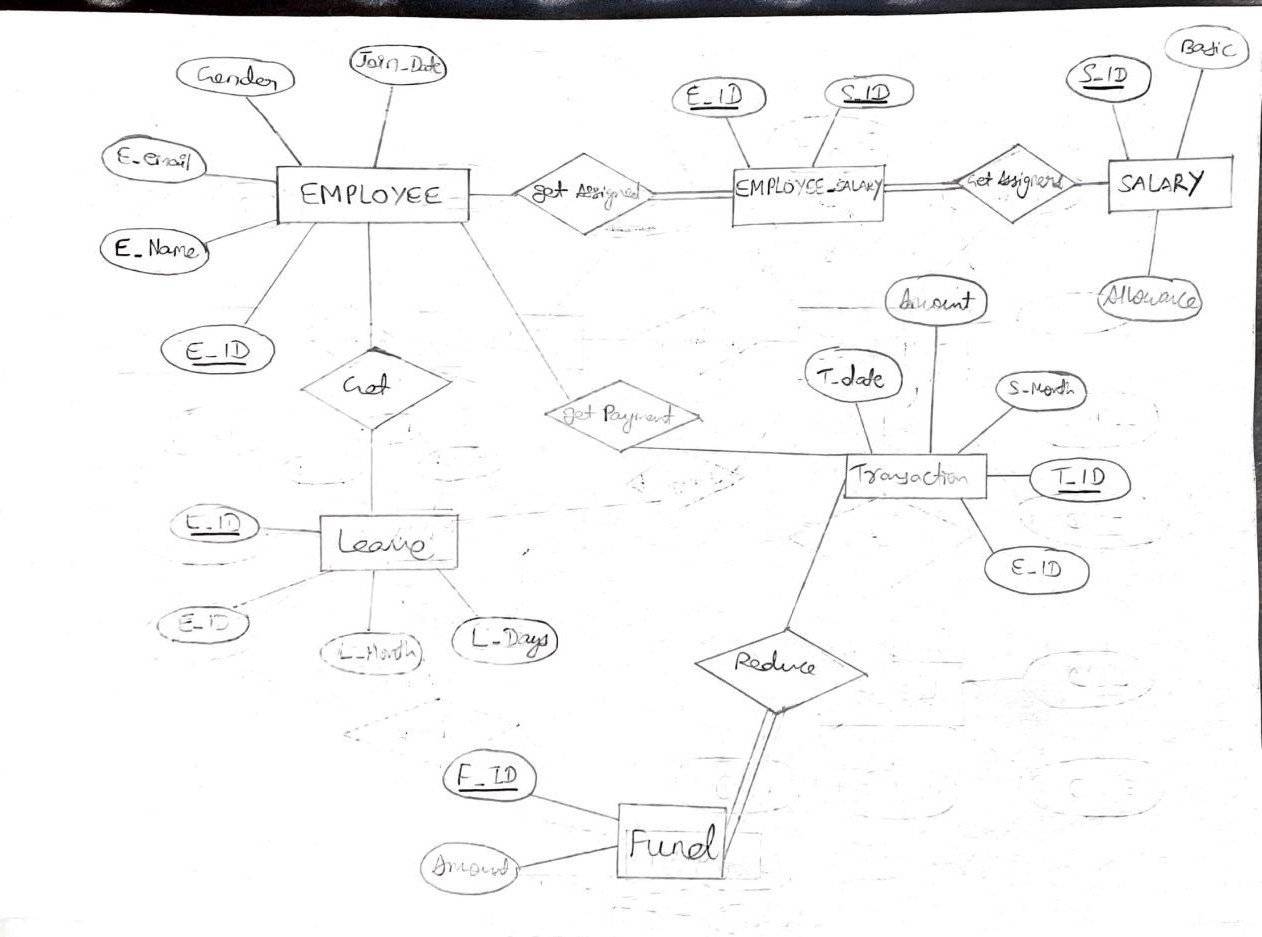
##### Triggers

* + - * 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

#### ER Diagram



#### 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)

Transaction(EID) references Employee(EID)

* 1. **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 |  |

**TRANSACTION**

|  |  |  |  |
| --- | --- | --- | --- |
| 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) |  |  |

#### 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));

## Insert

insert into employee values (1, 'Archit', 'M','archit@gmail.com', '1/1/2019');

insert into employee values (2, 'Saloni','F', 'saloni@gmail.com', '1/1/2019');

insert into employee values (3, 'Riya','F', 'riya@gmail.com', '1/1/2019');

insert into employee values (4, 'Arsh','M', 'arsh@gmail.com', '1/1/2019');

insert into employee values (5, ‘Arjit’,'M', 'arjit@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);

## 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;

###### /

## 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;

/

#### TriggerTransact.sql

###### SET SERVEROUTPUT ON;

create or replace trigger Trig\_Tansact after insert on transaction

begin

dbms\_output.put\_line('One Transaction 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;

/

## Procedures

#### ProcedureAddLeaves.spl

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;

/

## Functions

#### AccountantAddFund.sql

###### SET SERVEROUTPUT ON; SET VERIFY OFF;

DECLARE

X number:= &Fund\_Amount\_to\_add;

v\_fund\_amnt number; Begin

happened');

end;

/

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

**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);

happened');

###### END;

/

###### EXCEPTION

When no\_data\_found then DBMS\_OUTPUT.PUT\_LINE('No Data Found');

When others then

DBMS\_OUTPUT.PUT\_LINE('Something wrong

#### 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);

happened');

###### END;

/

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

#### 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

eid=X;

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

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;

happened');

###### EXCEPTION

When no\_data\_found then DBMS\_OUTPUT.PUT\_LINE('No Data Found');

When others then

DBMS\_OUTPUT.PUT\_LINE('Something wrong

end Check\_Valid;

**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.