**Payroll Management System**

A Minor Project

**Submitted in Partial fulfillment of the requirements for the degree of Bachelor of Engineering in Information Technology**

Submitted to



Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal (M.P.)

Submitted By:

**Lucky Jain [0126IT131052]**

**Piyush Malviya [0126IT131064]**

**Smita Rai [0126IT131089]**

Under the Guidance of

**Mr. Avinash Sharma**

Assistant Professor, IT

**Department of Information Technology**



**ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL**

**Approved by AICTE New Delhi & Govt. of M.P.**

**Affiliated to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal (M.P.)**

**Session: July-December,2016**

|  |  |
| --- | --- |
| H:\mono.jpg | **ORIENTAL COLLEGE OF TECHNOLOGY, BHOPAL**  **Approved by AICTE New Delhi & Govt. of M.P. & Affiliated to Rajiv Gandhi Proudyogiki Vishwavidhyalaya Bhopal (M.P.)**  **DEPARTMENT OF INFORMATION TECHNOLOGY** |

**CERTIFICATE**



This is to certify that the work embodied in this Minor Project entitled as **“Payroll Management System”** has been satisfactorily completed by Lucky Jain(0126IT131052), Piyush Malviya (0126IT131064), Smita Rai (0126IT131089) students of B.E, IV year, VII Sem.

It is a bonafide piece of work, carried out under my guidance in the **Information Technology, Oriental College of Technology, Bhopal** for the partial fulfillment of the **Bachelor of Engineering in** **Information Technology** degree during the academic session July-December, 2016.

|  |  |
| --- | --- |
| **Mr. Avinash Sharma**  Assistant Professor  Department, IT  OCT, Bhopal  (Project Guide) | **Prof. Roopali Soni**  Head of Department  Department of IT  OCT, Bhopal |

**CANDIDATE DECLARATION**

I hereby declare that the Minor Project work presented in the report entitled as **“Payroll Management System”** submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology of Oriental College of Technology is an authentic record of my own work carried out at Oriental College of Technology, Bhopal.

I have not submitted the part and partial of this report for the award of any other degree or diploma.

Date: Lucky Jain (0126IT131052)

Piyush Malviya (0126IT131064)

Smita Rai (0126IT131089)

This is to certify that the above statement made by the candidate/s is correct to the best the best of my knowledge.

|  |  |  |
| --- | --- | --- |
|  |  | **Prof. Roopali Soni**  Head of Department  Department of IT  OCT, Bhopal |

**Countersigned by**

Dr. N.K.Agrawal

Director

OCT, Bhopal

**ACKNOWLEDGMENT**

I am heartily thankful to the **Oriental College of Technology** for providing me all the facilities and infrastructure to take my work to the final stage.

It is the constant supervision, moral support and proper guidance of our respected Director **Dr**. **N.K.Agrawal**  who motivated throughout the work.

I express deep sense of gratitude and respect to my learned guide **Mr. Avinash Sharma** Assistant Professor in the Department of Information Technology during all phases of my work. Without him enthusiasm and encouragement this dissertation would not have been completed. Her valuable knowledge and innovative ideas helped me to take the work to the final stage. She has timely suggested actions and procedures to follow for which I am really grateful and thankful to her.

I express my gratefulness to **Prof. Roopali Soni**, Head Information Technology Department for providing all the facilities available in the department for her continuous support, advice, and encouragement during this work and also help to extend my knowledge and proper guidelines.

I am thankful to all the faculty members and other non teaching staff of Information Technology Department for their cooperation.

Constant help, moral and financial support of my loving parents motivated me to complete the work. I express my heartily thanks to my all family members for their co-operation.

I really admire the fond support of my class-mates for their co-operation and constant help. It gives immense pleasure to acknowledge the encouragement and support extended by them. Last but not the least I am extremely thankful to all who have directly or indirectly helped me for the completion of my work.

**Lucky Jain (0126IT131052)**

**Piyush Malviya (0126IT131064)**

**Smita Rai (0126IT131089)**

**ABSTRACT**

The Payroll Management covers nearly everything needed for gathering Payroll system requirements. Payroll system is the heart of any Human Resource System of an organization. The solution has to take care of the calculation of salary as per rules of the company, income tax calculation and various deductions to be done from the salary including statutory deductions like tds (tax deducted at source) ,professional tax and provident fund deductions. It has to generate pay-slip. The purpose of this study was to develop a payroll management system to assist in the management of payroll which ease the process of doing this job than earlier pen and paper based management.Nowadays this kind of application is very essential for any small or medium sized organization. An employer, regardless of the number of workers they employ, must maintain all records pertaining to payment system digitally.This application will help to calculate the employees’salary and other relevant calculations automatically. This helps to administer of the financial record of employees' salaries, wages, bonuses, net pay, increment, conveyance, loan and deductions. The main exiting feature of our application is that it will calculatethe monthly salary automatically.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No** | **Contents** | **Page No.** |
|  |  |  |
|  | **List of Figures** | I |
|  | **List of Tables** | II |
|  | **List Symbols and of Abbreviations** | III |
| **1** | **Introduction** | **1-4** |
|  | 1.1 Introduction | 2 |
|  | 1.2 Need of PMS | 2-3 |
|  | 1.3 Advantages and Disadvantages of PMS | 3-4 |
|  |  |  |
| **2** | **Software Requirement Specification** | **5-9** |
|  | 2.1 Purpose | 6 |
|  | 2.2 Scope | 6 |
|  | 2.3 Abbreviations | 6 |
|  | 2.4 Feasibility Study | 6-7 |
|  | 2.5 Hardware Requirement/ Software Requirement | 7-8 |
|  | 2.6 Software Process Model Used | 8-9 |
|  |  |  |
| **3** | **System Documentation** | **10-14** |
|  | 3.1 Data Flow Diagram | 11-12 |
|  | 3.2 System Flow Chart | 13 |
|  | 3.3 Entity Relationship Diagram | 14 |
|  |  |  |
| **4** | **User Manual** | **15-25** |
|  | 4.1 Introduction and Guidelines | 16 |
|  | 4.2 Screen Layouts and Description | 17-25 |
|  |  |  |
| **5** | **Testing and Test Cases** | **26-28** |
| **6** | **Limitations** | **29-30** |
| **7** | **Conclusion and Future Work** | **31-32** |
| **8** | **Bibliography** | **33-34** |
| **9** | **References** | **35-36** |
| **10** | **Appendix** | **37-100** |

**LIST OF FIGURES**



|  |  |  |
| --- | --- | --- |
| Figure 2.1 | Software Process Model Used | 9 |
|  |  |  |
| Figure 4.1 | Shows opening window of software | 17 |

**I**

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Table 5.1 | Department | 71 |
| Table 5.2 | Employee | 72 |

Table 5.3 Annual\_salary 75

Table 5.4 Deduction 76

Table 5.5 Attendance 77

Table 5.6 Address 78

**II**

**LIST OF SYMBOLS AND ABBREVIATIONS**



|  |  |
| --- | --- |
| PMS | Payroll Management System |
| SQL | Structured Query Language |
| PLSQL | Procedural Structured Query Language |
| ER | Entity Relationship Diagram |
| DFD | Data Flow Diagram |
| IDE | Integrated Development Environment |

**III**

1. **INTRODUCTION**
   1. **INTRODUCTION**

It is understood that we are tired of managing thousand of odd papers, pay slips, payroll reports, and salary details and so on. Imagine that we have a payroll processing system which will generate our pay slips and payroll reports within seconds. We can help others automated your payroll system by developing a customized payroll application that suits your specific requirements.

Thus Payroll Management System is used for calculation of salary as per rules of the company.A payroll system involves everything that has to do with the payment of employees and the filling of employment taxes ,keeping track of hours ,calculating salary.

Few of the additional components are

* Basic Pay
* HRA(House Rent Allowance)
* Conveyance allowance
* Leave encashment
* Medical allowance
* LTA(Leave Travel Allowance)
* Monthly Bonus

Few of the deductions are

TDS(Tax deduced at source)

Provident Fund

Professional Tax

* 1. **NEED OF PMS**

As your organization grows, the demands on your operations ensure you hire human resources and pay them salaries based on their responsibilities and deliverables. Salaries may be fixed or based on performance or a .As the number of variables grows each time you are processing salaries, the chances of errors creeping in increase. You need to keep a track of all your employee details, their attendance, Overtime, performance linked incentives, leaves, expense reimbursements as well as various deductions mandated by labour and income tax laws such as PF, ESI, PT, and Income Tax deductions.

Your salary calculations should be efficient enough to provision for any investments that your employees make and which entitle them to various tax exemptions provided by the GovernmentThe entire process becomes extremely cumbersome and any errors have serious implications in terms of interest and penalties for delayed statutory compliances as well low employee morale.

Eventually every organization outgrows Excel. In fact, it’s never too early to move to an automated payroll system.

* 1. **ADVANTAGES AND DISADVANTAGES OF PMS**

**ADVANTAGES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ease of Use** — With Advantage, your payroll responsibility is reduced to a simple function of communication each pay period. From pay checks to tax payments to the preparation of quarterly and annual tax returns, your time investment remains minimal, allowing you to focus on what's most important...managing yourbusiness.  **Accuracy** — Keeping track of the ever-changing tax regulations can be a nightmare. Why bother? With Advantage, you have the comfort of knowing your payroll is being expertly handled. We guarantee all federal, state and local taxes will be filed and paid accurately and on time...or WE pay the penalties and interest!  **Confidentiality** — Where earnings are concerned, a breach in confidentiality can be devastating for any business. With Advantage, your payroll information is kept strictly confidential, and can be accessed only by those authorized to do so,keeping  your business safe from any potential disasters.  **Flexibility** — At Advantage, we offer a variety of payroll input options, allowing you to choose what's most convenient for you — phone, fax, or Web. We also provide comprehensive, easy-to-read management reports. And our system features extensive earnings and deductions categories, so we're flexible enough for virtually any business. Most importantly, we're prepared to grow with your company, offering a wide scope of payroll services such as direct deposit, pretax health insurance, workers' compensation premium payment, and other services that can provide great value to you and your employees.  **DISADVANTAGES**  **Constant Archiving** – Mountains of information are collected with each payroll cycle, and most payroll software programs are designed to keep up. But, that information needs to be archived continuously. And, this daily process needs the help of a human hand. In a busy operation, setting aside this particular slice of time in a workday can become a slight hassle.  **Limited Access** – In most instances, payroll software is loaded onto one computer and that data can only be accessed from that machine. This can be a hassle, especially if the payroll processing computer goes off the rails. The result could be that whole payroll process goes with it, which can cause headaches throughout the business.  **Added Weight to Overhead** – Implementing payroll software may also call for a full-time employee to take on the responsibilities of doing payroll in-house. With an added employee comes the cost of an additional salary and benefits. Also to be considered is the cost of technical support when the inevitable software glitches occur.  **Risk of Underwithholding** – The Internal Revenue Service notes that some payroll software systems are unable to distinguish additional voluntary withholding amounts from regular withholding when calculating catch-up withholding for the current tax year. This kind of glitch does not apply to all payroll software programs, but if not identified or accounted for, a business can be fined for underwitholding. | |  |  | | --- | --- | |  |  | |
|  | |

**2.SOFTWARE REQUIREMENT**

**SPECIFICATION**

* 1. **Purpose**

Main aim of developing Payroll ManagementSystem is to provide an easy way not only to automate all functionalities involved managing leaves and Payroll for the employees of Company, but also to provide full functional reports to management of Company with the details about usage of leave facility.

We are committed to bring the best way of management in the various

forms of **PMS.**We understand that **PMS**in not a product to be sold, it is a tool

to manage the inner operation of Company related to employee leave and Payroll.

* 1. **Scope**

This Application works in Multiple PC’s installed on multiple Computers but sharing same database by which users of different department can use it sitting at different locations simultaneously.

But in future we can make the Application where the database will be hosted in order to manage the all departments which will be located in different places and by keeping domain of Application as Online.

* 1. **Definitions, Acronyms, Abbreviations**
* PMS  Payroll Management System
* SQLStructured Query Language
* PLSQL Procedural Structured Query Language
* CFD Context Flow Diagram
* DFD Data Flow Diagram
* IDE Integrated Development Environment
  1. **Feasibility Study**

The overall scope of the feasibility study was to provide sufficient information to allow a decision to be made as to whether the payroll management system project should proceed and so, its relative priority in the context of the other existing payroll management system.

The feasibility study of this project had undergone through various steps which as describe as under:

* Identify the origin of the information at different level.
* Identify the expectation of user from computerized system.
* Analyze the drawback of existing system
  1. **Hardware And Software Requirement**
* **Hardware interfaces**
* Memory minimum of 1GB RAM
* Hard disk of 40 GB
* Monitor
* Mouse
* Keyboard
* **Software interfaces**
* Operating System                               Windows 7
* Front End                                           HTML
* Backend                                              ORACLE 10g(SQL,PL/SQL)

Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is an [object-relational database management system](https://en.wikipedia.org/wiki/Object-relational_database_management_system) produced and marketed by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation).

**SQL** is a [special-purpose programming language](https://en.wikipedia.org/wiki/Special-purpose_programming_language) designed for managing data held in a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS), or for stream processing in a [relational data stream management system](https://en.wikipedia.org/wiki/Relational_data_stream_management_system) (RDSMS).

Originally based upon [relational algebra](https://en.wikipedia.org/wiki/Relational_algebra) and [tuple relational calculus](https://en.wikipedia.org/wiki/Tuple_relational_calculus), SQL consists of a [data definition language](https://en.wikipedia.org/wiki/Data_definition_language), [data manipulation language](https://en.wikipedia.org/wiki/Data_manipulation_language), and [Data Control Language](https://en.wikipedia.org/wiki/Data_Control_Language). The scope of SQL includes data insert, query, update and delete, [schema](https://en.wikipedia.org/wiki/Database_schema) creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a [declarative language](https://en.wikipedia.org/wiki/Declarative_programming) ([4GL](https://en.wikipedia.org/wiki/4GL)), it also includes [procedural](https://en.wikipedia.org/wiki/Procedural_programming) elements.

**PL/SQL** (**Procedural Language/Structured Query Language**) is Oracle Corporation's procedural extension for SQL and the Oracle relational database. PL/SQL is available in Oracle Database (since version 7), TimesTen in-memory database (since version 11.2.1), and IBM DB2 (since version 9.7).[[1]](https://en.wikipedia.org/wiki/PL/SQL#cite_note-1) Oracle Corporation usually extends PL/SQL functionality with each successive release of the Oracle Database.

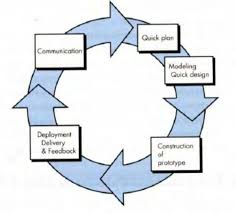
PL/SQL includes procedural language elements such as conditions and [loops](https://en.wikipedia.org/wiki/Program_loop). It allows declaration of constants and [variables](https://en.wikipedia.org/wiki/Variable_(programming)), procedures and functions, types and variables of those types, and triggers. It can handle exceptions (runtime errors). [Arrays](https://en.wikipedia.org/wiki/Array_data_type) are supported involving the use of PL/SQL collections. Implementations from version 8 of Oracle Database onwards have included features associated with object-orientation. One can create PL/SQL units such as procedures, functions, packages, types, and triggers, which are stored in the database for reuse by applications that use any of the Oracle Database programmatic interfaces.

**2.6 Software Process Model Used**

* Prototype is a working model of software with some limited functionality.
* The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
* Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
* It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

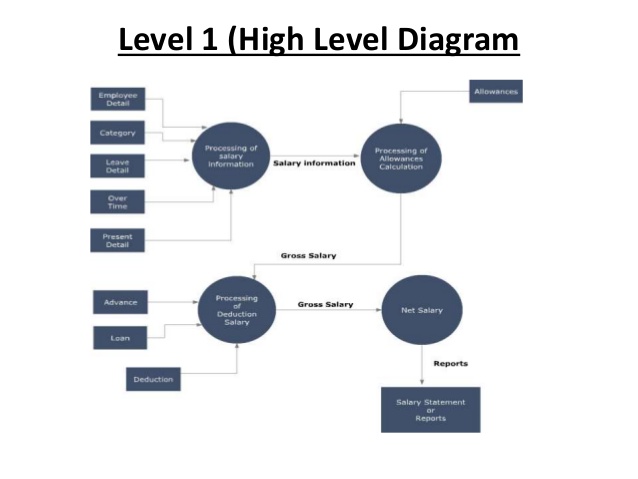
Following is the stepwise approach to design a software prototype:

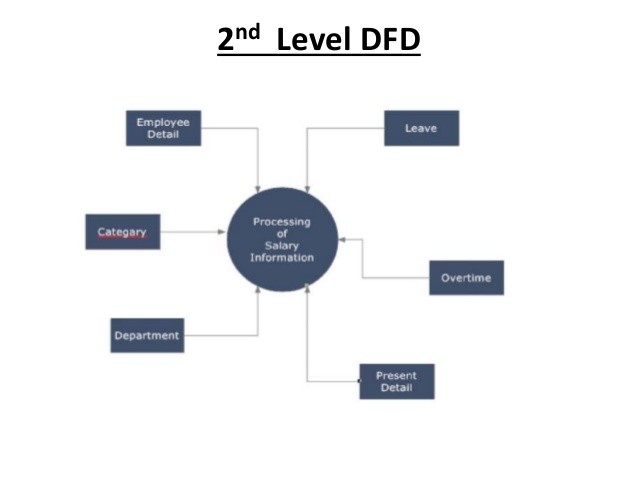
* **Basic Requirement Identification:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.
* **Developing the initial Prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.
* **Review of the Prototype:**The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.
* **Revise and enhance the Prototype:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like , time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.



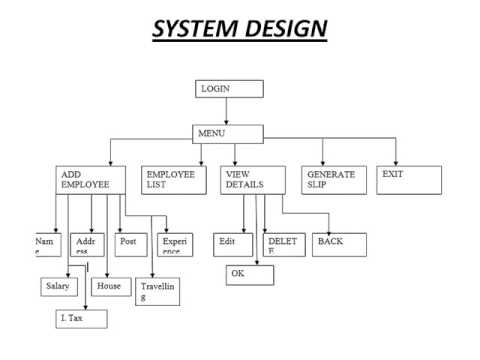
**3.SYSTEM DOCUMENTATION**

**3.1 Data Flow Diagram**

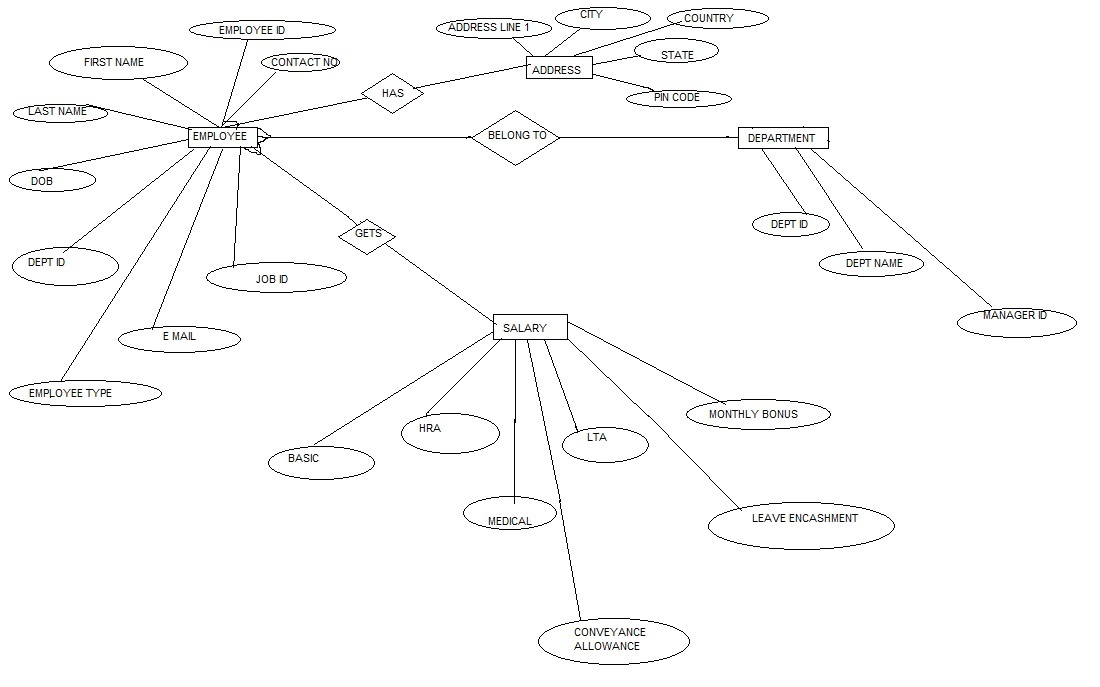
****

****

**3.2 System Flow Chart**

****

**3.3 E-R(Entity Relationship ) DIAGRAM**



**4.USER MANUAL**

**4.1 Introduction And Guidelines**

**1** **– Introduction**

This guide is designed to be used when you wish to add your own data into Payroll Manager.

This guide covers many day-to-day payroll procedures.

**2 – Create your own data file**

Payroll Manager stores data for each tax year in a separate data file.When you are ready to enter data for your owncompany you will need to create a new data file.Click File - New - Create a new blank file from the main menu at the top of the screen.Input your Employer Name (e.g. ABC Ltd) and select the Tax Year. You will then be prompted tosave your new file and to set the Pay Dates.

**3 – Enter the Employer details**

Click Employer - Employer Details fromthe menu .Enter the Employer Name and address information

**4 – Add your employees**

Click Employees - Add new Employee fromthe menu.You will be taken through a series of screenswhere the personal, work and payment details ofeach employee can be entered. The screens thatappear depend on the answers that you giveWhen you have finished adding your employees,click Employees - Employee Details to view oramend the information.

**5– Adding or editing pay on the Pay Details screen**

Click Pay - Pay Details from the menu (or clickthe button).If the regular pay amounts were not entered whencreating the employee, you should now enter thisinformation on the Pay Details screen. If theemployee is paid a standard, regular amount thenenter this figure in the Basic Pay column. Other additions and deductions can be entered by clicking on the tabsalong the top of the PayDetails screen.

**6– Producing Payslips**

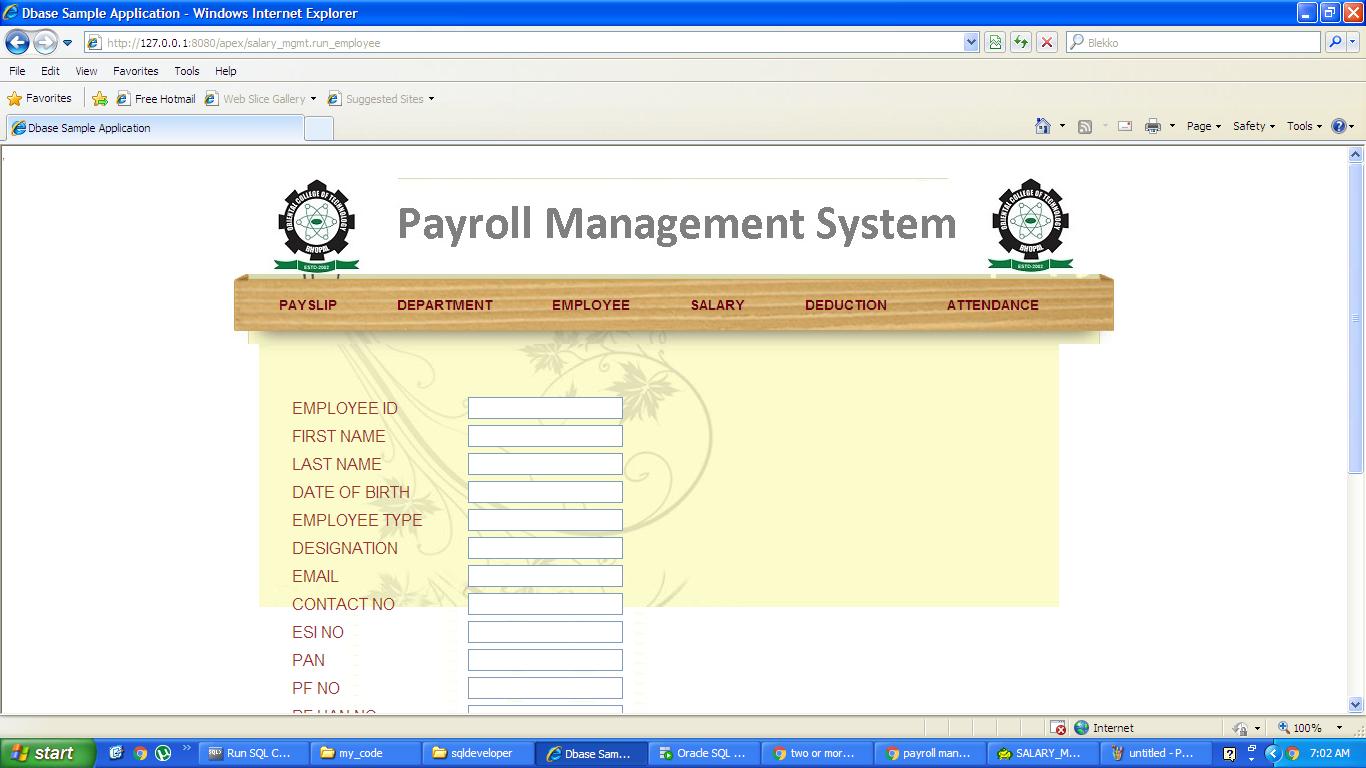
Click Pay - Employees Payslip from themenu (or click the button).Select the employee(s) you wish toproduce a payslip for

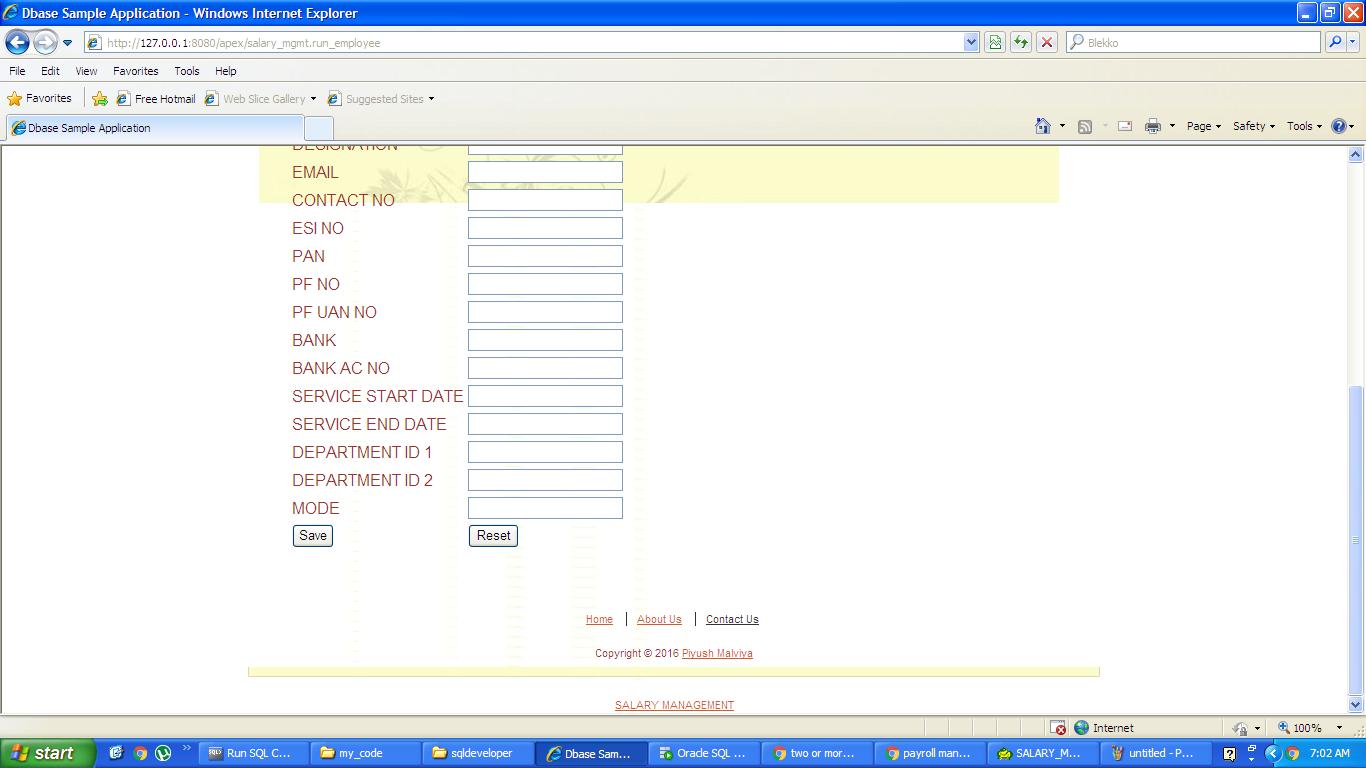
**4.2 Screen Layouts And Description**

****

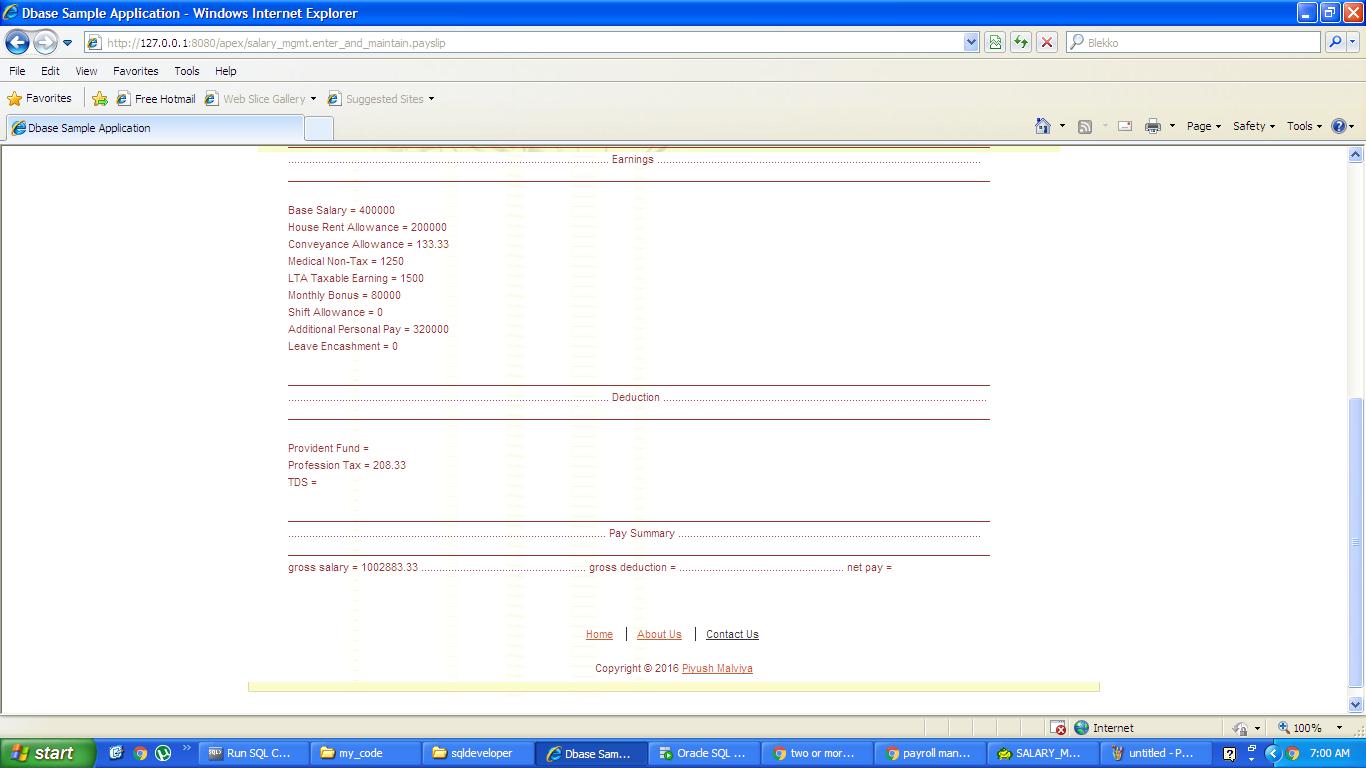
****

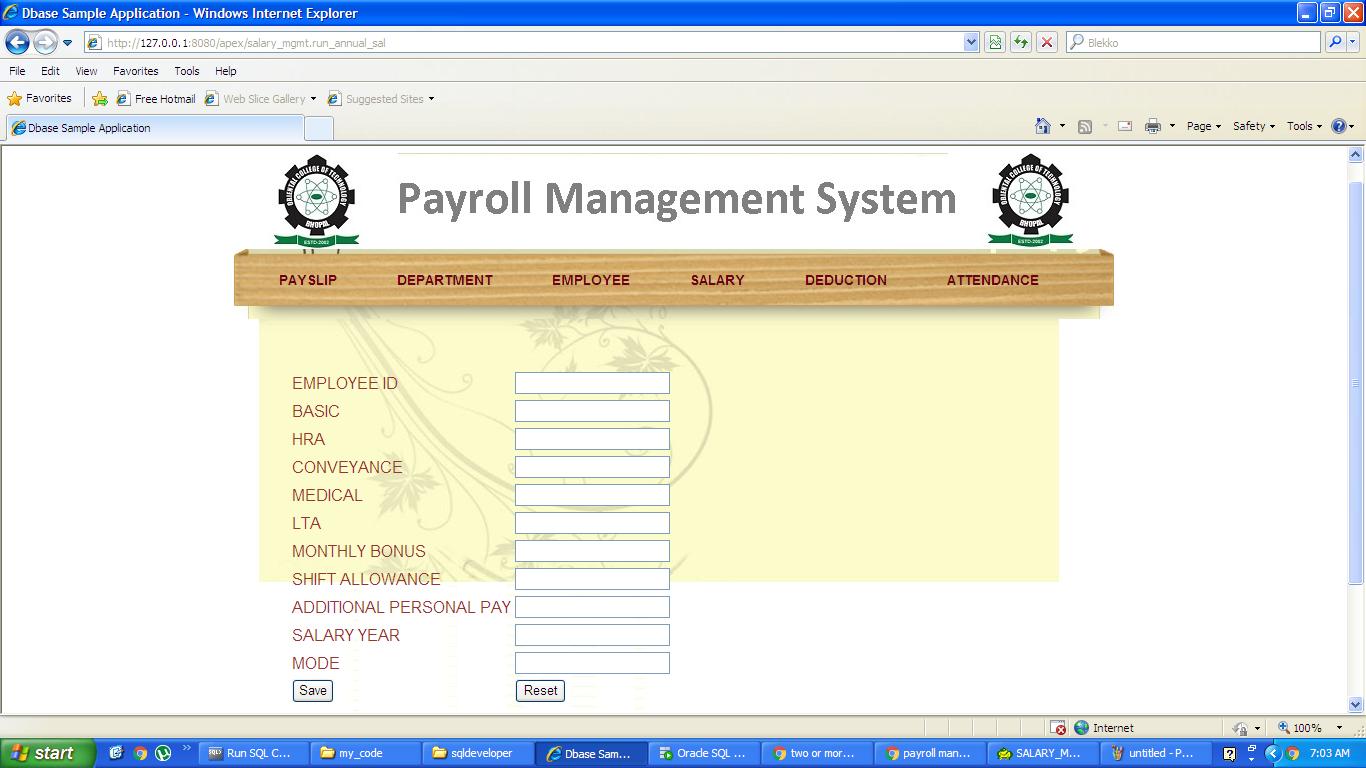
****

****

****

****

****

****

****

**5.TESTING AND TEST CASES**

**Test cases for Payroll System**

* Check if the ID field is system generated or requires manual input.
* Check if the ID field auto increments when next or previous button is pressed.
* Check if the ID field points to the correct employee ID.
* Check if the ID field only allows numeric input when under modification.
* Check if the ID field is editable or not if edit button is pressed.
* Check if the Employee Name field accepts text data only.
* Check the minimum required length for the Employee Name.
* Check the maximum required length for the Employee Name.
* Check if the Employee Name from the previous records points to the valid name for the employee.
* Check if the Employee Name accepts both uppercase and lowercase input.
* Check if the Employee Name does not accept any form of specialcharacter.
* Check if the DEPT field accept the department present within the company.
* Check if the DEPT field requires manual addon or if it gets auto generated.
* Check DEPT field for the minimum length of the input.
* Check DEPT field for the maximum length of the input.
* Check “No of days” field only accepts numeric data.
* Check ‘No of days” field for the maximum available input. (Ideally it should restricts for 31 days per month).
* Check Rate per day field for minimum length of the input.
* Check Rate per day field for the maximum length of the input.
* Check the salary field to verify if it is editable or not.
* Check the salary field for the calculation (no of days \* per day amount).
* Check the salary field for the live input or if it requires pressing “computer” button.
* Check if the add button clears the form for the fresh input.
* Check if the add button does not open existing record.
* Check if the save button saves the data.
* Verify if the save button saves the valid data.
* Check if the update button allows the modified data to be added into the database.
* Check if the update button does not retain the old information in the database.
* Check if the delete button removes the data from the database.
* Check if the deleted data using delete button does not remain in the database.
* Check if the first button click points the form to the first record in the database.
* Check if the last button click points the form to the last record in the database.
* Check if the next button click points the form to the next available record in the database.
* Check if the previous button click points the form to the previous record from the existing open record in the database.
* Check if the exit button click results in closure of the program.
* Check if the close window button on the program exits the program.
* Check if the minimize window button for the program minimizes the program to the taskbar.
* Check if the maximize window button for the program maximizes the program to the desktop

**6.LIMITATIONS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant Archiving** – Mountains of information are collected with each payroll cycle, and most payroll software programs are designed to keep up. But, that information needs to be archived continuously. And, this daily process needs the help of a human hand. In a busy operation, setting aside this particular slice of time in a workday can become a slight hassle.  **Limited Access** – In most instances, payroll software is loaded onto one computer and that data can only be accessed from that machine. This can be a hassle, especially if the payroll processing computer goes off the rails. The result could be that whole payroll process goes with it, which can cause headaches throughout the business.  **Added Weight to Overhead** – Implementing payroll software may also call for a full-time employee to take on the responsibilities of doing payroll in-house. With an added employee comes the cost of an additional salary and benefits. Also to be considered is the cost of technical support when the inevitable software glitches occur.  **Risk of Underwithholding** – The Internal Revenue Service notes that some payroll software systems are unable to distinguish additional voluntary withholding amounts from regular withholding when calculating catch-up withholding for the current tax year. This kind of glitch does not apply to all payroll software programs, but if not identified or accounted for, a business can be fined for underwitholding. | |  |  | | --- | --- | |  |  | |
|  | |

**7.CONCLUSION AND FUTURE WORK**

Payroll can also be used to provide valuable feedback for you business. The most obvious feedback that payroll can show a business is whether or not the business is making a profit. A precise payroll could determine that a business is losing more money then it’s bringing in because it is spending too much on paying employees. If that’s the scenario, the business may have to consider laying off some of it’s employees so that it can become more profitable.

Every business owner must have an accurate and consistent payroll so that they can pay employees correctly, withhold the right amount of taxes, and have a better understanding of their cash flow.

**8.BIBLIOGRAPHY**

* PL/SQL Packages and Types Reference
* Oracle Database 10g The Complete Reference

**9.REFERENCES**

* [www.safaribooksonline.com](http://www.safaribooksonline.com)
* [www.relationaldbdesign.com](http://www.relationaldbdesign.com)
* [www.google.com](http://www.google.com)

**10.APPENDIX**

**enter\_and\_maintain\_annual\_salary**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_annual\_sal(

p\_employee\_id IN NUMBER,

p\_basic IN NUMBER,

p\_hra IN NUMBER, --50% of basic

p\_conveyance IN NUMBER, --Rs.1500\*12 per year

p\_medical IN NUMBER, --15k per year

p\_lta IN NUMBER, --20k per year

p\_monthly\_bonus IN NUMBER, --20% of basic

p\_shift\_allowance IN NUMBER, --Rs.0

p\_additional\_personal\_pay IN NUMBER, --80% of basic

p\_salary\_year IN NUMBER,

p\_mode IN VARCHAR2)

AS

v\_old\_basic NUMBER(10);

v\_old\_hra NUMBER(10,2); --50% of basic

v\_old\_conveyance NUMBER(10,2); --Rs.1500\*12 per year

v\_old\_medical NUMBER(10,2); --15k per year

v\_old\_lta NUMBER(10,2); --20k per year

v\_old\_monthly\_bonus NUMBER(10,2); --20% of basic

v\_old\_shift\_allowance NUMBER(10,2); --Rs.0

v\_old\_additional\_personal\_pay NUMBER(10,2); --80% of basic

v\_employee\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

v\_status :=0;

SELECT COUNT(\*)

INTO v\_employee\_count

FROM employee

WHERE employee\_id =p\_employee\_id;

IF v\_employee\_count=1 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

IF lower(p\_mode)='insert' THEN

INSERT

INTO annual\_salary

(

employee\_id,

basic,

hra,

conveyance,

medical,

lta,

monthly\_bonus,

shift\_allowance,

additional\_personal\_pay

)

VALUES

(

employee\_id\_seq.nextval,

p\_basic,

p\_hra, --50% of basic

p\_conveyance, --Rs.1500\*12 per year

p\_medical, --15k per year

p\_lta, --20k per year

p\_monthly\_bonus, --20% of basic

p\_shift\_allowance, --Rs.0

p\_additional\_personal\_pay --80% of basic

);

HTP.P('Record added successfully');

ELSIF lower(p\_mode)='update' THEN

SELECT basic,

hra,

conveyance,

medical,

lta,

monthly\_bonus,

shift\_allowance,

additional\_personal\_pay

INTO v\_old\_basic,

v\_old\_hra, --50% of basic

v\_old\_conveyance, --Rs.1500\*12 per year

v\_old\_medical, --15k per year

v\_old\_lta, --20k per year

v\_old\_monthly\_bonus, --20% of basic

v\_old\_shift\_allowance, --Rs.0

v\_old\_additional\_personal\_pay --80% of basic

FROM annual\_salary

WHERE employee\_id= p\_employee\_id

AND salary\_year =p\_salary\_year;

IF SQL%NOTFOUND --why to use it when exception block is present

THEN

HTP.P('invalid employee id and/or salary year');

ELSE

UPDATE annual\_salary

SET basic =NVL(p\_basic,v\_old\_basic),

hra =NVL(p\_hra,v\_old\_hra),

conveyance =NVL(p\_conveyance,v\_old\_conveyance),

medical =NVL(p\_medical,v\_old\_medical),

lta =NVL(p\_lta,v\_old\_lta),

monthly\_bonus =NVL(p\_monthly\_bonus,v\_old\_monthly\_bonus),

shift\_allowance =NVL(p\_shift\_allowance,v\_old\_shift\_allowance),

additional\_personal\_pay=NVL(p\_additional\_personal\_pay,v\_old\_additional\_personal\_pay)

WHERE employee\_id = p\_employee\_id

AND salary\_year =p\_salary\_year;

enter\_and\_maintain\_tax(p\_employee\_id,p\_salary\_year);

HTP.P('Record Updated');

END IF;

ELSE

HTP.P('invalid mode') ;

END IF;

ELSE

HTP.P('no such employee is present as given in employee\_id');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_annual\_salary TO PUBLIC;

show error;

**enter\_and\_maintain\_attendance**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_attendance(

p\_employee\_id IN NUMBER,

p\_day IN NUMBER,

p\_month IN NUMBER,

p\_year IN NUMBER,

p\_status IN VARCHAR2,

p\_mode IN VARCHAR2)

AS

v\_old\_status VARCHAR2(10);

v\_employee\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

v\_status :=0;

SELECT COUNT(\*)

INTO v\_employee\_count

FROM employee

WHERE employee\_id =p\_employee\_id;

IF v\_employee\_count=1 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

IF lower(p\_mode)='insert' THEN

INSERT

INTO attendance

(

employee\_id,

status

)

VALUES

(

p\_employee\_id,

p\_status

);

enter\_and\_maintain\_leaves(p\_employee\_id);

HTP.P('Record added successfully');

ELSIF lower(p\_mode)='update' THEN

SELECT p\_status

INTO v\_old\_status

FROM attendance

WHERE employee\_id= p\_employee\_id

AND day=p\_day

AND month=p\_month

AND year =p\_year;

IF SQL%NOTFOUND --why to use it when exception block is present

THEN

HTP.P('invalid employee id and/or date');

ELSE

UPDATE attendance

SET status=NVL(p\_status,v\_old\_status)

WHERE employee\_id= p\_employee\_id

AND day=p\_day

AND month=p\_month

AND year =p\_year;

enter\_and\_maintain\_leaves(p\_employee\_id);

HTP.P('Record Updated');

END IF;

ELSE

HTP.P('invalid mode') ;

END IF;

ELSE

HTP.P('no such employee is present as given');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_attendance TO PUBLIC;

show error;

**enter\_and\_maintain\_deduction**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_deduction(

p\_employee\_id IN NUMBER,

p\_tax\_deduction IN NUMBER,

p\_salary\_year IN NUMBER,

p\_mode IN VARCHAR2)

AS

v\_old\_employee\_id NUMBER(10);

v\_old\_tax\_deduction NUMBER(10,2);

v\_old\_salary\_year NUMBER(4);

v\_empsal\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

v\_status :=0;

SELECT COUNT(\*)

INTO v\_empsal\_count

FROM annual\_salary

WHERE employee\_id =p\_employee\_id

AND salary\_year=p\_salary\_year;

IF v\_empsal\_count=1 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

IF lower(p\_mode)='insert' THEN

INSERT

INTO deduction

(

employee\_id,

tax\_deduction

)

VALUES

(

p\_employee\_id,

p\_tax\_deduction

);

enter\_and\_maintain\_tax(p\_employee\_id,p\_salary\_year);

HTP.P('Record added successfully');

ELSIF lower(p\_mode)='update' THEN

SELECT tax\_deduction

INTO v\_old\_tax\_deduction

FROM deduction

WHERE employee\_id= p\_employee\_id

AND salary\_year =p\_salary\_year;

IF SQL%NOTFOUND --why to use it when exception block is present

THEN

HTP.P('invalid employee id and/or salary year');

ELSE

UPDATE deduction

SET tax\_deduction=NVL(p\_tax\_deduction,v\_old\_tax\_deduction)

WHERE employee\_id = p\_employee\_id

AND salary\_year =p\_salary\_year;

enter\_and\_maintain\_tax(p\_employee\_id,p\_salary\_year);

HTP.P('Record Updated');

END IF;

ELSE

HTP.P('invalid mode') ;

END IF;

ELSE

HTP.P('no such employee and/or salary is present as given');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_deduction TO PUBLIC;

show error;

**enter\_and\_maintain\_department**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_department(

p\_department\_id IN NUMBER,--It is provided only at the time of update not at time of insert

p\_department\_name IN VARCHAR2,

p\_department\_head\_id IN NUMBER,

p\_mode IN VARCHAR2 )

AS

v\_old\_department\_name VARCHAR2(100);

v\_old\_department\_head\_id NUMBER(10);

v\_employee\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

v\_status :=0;

IF p\_department\_head\_id IS NOT NULL THEN

SELECT COUNT(\*)

INTO v\_employee\_count

FROM employee

WHERE employee\_id =p\_department\_head\_id;

IF v\_employee\_count=1 THEN

v\_status :=1;

END IF;

ELSE

v\_status:=1;

END IF;

IF v\_status =1 THEN

IF lower(p\_mode)='insert' THEN

INSERT

INTO department

(

department\_id ,

department\_name ,

department\_head\_id

)

VALUES

(

department\_id\_seq.nextval ,

p\_department\_name ,

p\_department\_head\_id

);

HTP.P('Record added successfully');

ELSIF lower(p\_mode)='update' THEN

SELECT department\_name ,

department\_head\_id

INTO v\_old\_department\_name ,

v\_old\_department\_head\_id

FROM department

WHERE department\_id= p\_department\_id;

IF SQL%NOTFOUND --It can also be implemented using another count for department.

THEN

HTP.P('invalid department id');

ELSE

UPDATE department

SET department\_name = NVL (p\_department\_name,v\_old\_department\_name) ,

department\_head\_id =NVL(p\_department\_head\_id,v\_old\_department\_head\_id)

WHERE department\_id = p\_department\_id;

HTP.P('Record Updated');

END IF;

ELSE

HTP.P('invalid mode') ;

END IF;

ELSE

HTP.P('no such employee is present as given in department\_head\_id');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_department TO PUBLIC;

show error;

**enter\_and\_maintain\_employee**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_employee(

p\_employee\_id IN NUMBER,--It is provided only at the time of update not at time of insert

p\_first\_name IN VARCHAR2,

p\_last\_name IN VARCHAR2,

p\_date\_of\_birth IN DATE,

p\_employee\_type IN VARCHAR2,

p\_designation IN VARCHAR2,

p\_email IN VARCHAR2,

p\_contact\_no IN NUMBER,

p\_ESI\_no IN VARCHAR2,

p\_PAN IN VARCHAR2,

p\_PF\_no IN VARCHAR2,

p\_PF\_UAN\_no IN NUMBER,

p\_bank IN VARCHAR2,

p\_bank\_ac\_no IN NUMBER,

p\_service\_start\_date IN DATE,

p\_service\_end\_date IN DATE,

p\_department\_id\_1 IN OUT NUMBER,

p\_department\_id\_2 IN OUT NUMBER,

p\_mode IN VARCHAR2)

AS

v\_old\_first\_name VARCHAR2(100);

v\_old\_last\_name VARCHAR2(100);

v\_old\_date\_of\_birth DATE;

v\_old\_employee\_type VARCHAR2(100);

v\_old\_designation VARCHAR2(100);

v\_old\_email VARCHAR2(100);

v\_old\_contact\_no NUMBER(10);

v\_old\_ESI\_no VARCHAR2(100);

v\_esi NUMBER(10,2);

v\_old\_PAN VARCHAR2(100);

v\_old\_PF\_no VARCHAR2(100);

v\_old\_PF\_UAN\_no NUMBER(12);

v\_old\_bank VARCHAR2(100);

v\_old\_bank\_ac\_no NUMBER(16);

v\_old\_service\_start\_date DATE;

v\_old\_service\_end\_date DATE;

v\_old\_department\_id\_1 NUMBER(10);

v\_old\_department\_id\_2 NUMBER(10);

v\_department\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

v\_status :=0;

IF p\_department\_id\_1 IS NOT NULL OR p\_department\_id\_2 IS NOT NULL THEN

p\_department\_id\_1 :=NVL(p\_department\_id\_1,p\_department\_id\_2);

p\_department\_id\_2 :=NVL(p\_department\_id\_2,p\_department\_id\_1);

IF p\_department\_id\_1 IS NOT NULL AND p\_department\_id\_2 IS NOT NULL THEN

SELECT COUNT(\*)

INTO v\_department\_count

FROM department

WHERE department\_id =p\_department\_id\_1

AND department\_id =p\_department\_id\_2;

IF v\_department\_count>0 THEN

v\_status :=1;

END IF;

END IF;

ELSE

v\_status:=1;

END IF;

IF v\_status =1 THEN

IF lower(p\_mode)='insert' THEN

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.nextval,

p\_first\_name,

p\_last\_name,

p\_date\_of\_birth,

p\_employee\_type,

p\_designation,

p\_email,

p\_contact\_no,

p\_PAN,

p\_PF\_no,

p\_PF\_UAN\_no,

p\_bank,

p\_bank\_ac\_no,

p\_service\_start\_date,

p\_service\_end\_date,

p\_department\_id\_1,

p\_department\_id\_2

);

SELECT esi

INTO v\_esi

FROM deduction

WHERE employee\_id=p\_employee\_id

AND salary\_year =extract(YEAR FROM sysdate);

IF SQL%NOTFOUND --why to use it when exception block is present

THEN

HTP.P('update deduction table for this employee first');

ELSE

IF v\_esi!=0 THEN

UPDATE employee SET esi\_no =p\_esi\_no WHERE employee\_id=p\_employee\_id;

ELSE

UPDATE employee SET esi\_no ='Not Eligible' WHERE employee\_id=p\_employee\_id;

END IF;

END IF;

HTP.P('Record added successfully');

ELSIF lower(p\_mode)='update' THEN

SELECT first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

ESI\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

INTO v\_old\_first\_name,

v\_old\_last\_name,

v\_old\_date\_of\_birth,

v\_old\_employee\_type,

v\_old\_designation,

v\_old\_email,

v\_old\_contact\_no,

v\_old\_ESI\_no,

v\_old\_PAN,

v\_old\_PF\_no,

v\_old\_PF\_UAN\_no,

v\_old\_bank,

v\_old\_bank\_ac\_no,

v\_old\_service\_start\_date,

v\_old\_service\_end\_date,

v\_old\_department\_id\_1,

v\_old\_department\_id\_2

FROM employee

WHERE employee\_id= p\_employee\_id;

IF SQL%NOTFOUND --why to use it when exception block is present

THEN

HTP.P('invalid employee id');

ELSE

UPDATE employee

SET first\_name =NVL(p\_first\_name,v\_old\_first\_name),

last\_name =NVL(p\_last\_name,v\_old\_last\_name),

date\_of\_birth =NVL(p\_date\_of\_birth,v\_old\_date\_of\_birth),

employee\_type =NVL(p\_employee\_type,v\_old\_employee\_type),

designation =NVL(p\_designation,v\_old\_designation),

email =NVL(p\_email,v\_old\_email),

contact\_no =NVL(p\_contact\_no,v\_old\_contact\_no),

PAN =NVL(p\_PAN,v\_old\_PAN),

PF\_no =NVL(p\_PF\_no,v\_old\_PF\_no),

PF\_UAN\_no =NVL(p\_PF\_UAN\_no,v\_old\_PF\_UAN\_no),

bank =NVL(p\_bank,v\_old\_bank),

bank\_ac\_no =NVL(p\_bank\_ac\_no,v\_old\_bank\_ac\_no),

service\_start\_date=NVL(p\_service\_start\_date,v\_old\_service\_start\_date),

service\_end\_date =NVL(p\_service\_end\_date,v\_old\_service\_end\_date),

department\_id\_1 =NVL(p\_department\_id\_1,v\_old\_department\_id\_1),

department\_id\_2 =NVL(p\_department\_id\_2,v\_old\_department\_id\_2)

WHERE employee\_id = p\_employee\_id;

SELECT esi

INTO v\_esi

FROM deduction

WHERE employee\_id=p\_employee\_id

AND salary\_year =extract(YEAR FROM sysdate);

IF v\_esi!=0 THEN

UPDATE employee

SET ESI\_no =NVL(p\_ESI\_no,v\_old\_ESI\_no)

WHERE employee\_id=p\_employee\_id;

ELSE

UPDATE employee SET esi\_no ='Not Eligible' WHERE employee\_id=p\_employee\_id;

END IF;

HTP.P('Record Updated');

END IF;

ELSE

HTP.P('invalid mode') ;

END IF;

ELSE

HTP.P('no such department is present as given in department\_id\_1 or department\_id\_2');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_employee TO PUBLIC;

show error;

**enter\_and\_maintain\_leaves**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_leaves(

p\_employee\_id IN NUMBER)

AS

v\_leaves\_taken NUMBER(2);

v\_monthly\_leaves NUMBER(2);

v\_remaining\_paid\_leaves NUMBER(2);

v\_previous\_paid\_leaves NUMBER(10);

v\_service\_start\_date DATE;

v\_employee\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

SELECT COUNT(\*)

INTO v\_leaves\_taken

FROM attendance

WHERE employee\_id=p\_employee\_id

AND lower(status)='absent'

AND YEAR =extract(YEAR FROM sysdate);

SELECT COUNT(\*)

INTO v\_monthly\_leaves

FROM attendance

WHERE employee\_id=p\_employee\_id

AND lower(status)='absent'

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

SELECT service\_start\_date

INTO v\_service\_start\_date

FROM employee

WHERE employee\_id=p\_employee\_id;

SELECT sum(remaining\_paid\_leaves)

INTO v\_previous\_paid\_leaves

FROM attendance

WHERE employee\_id=p\_employee\_id

AND year BETWEEN extract(YEAR FROM v\_service\_start\_date) AND (extract(YEAR FROM sysdate)-1)

AND remaining\_paid\_leaves!=0;

v\_status :=0;

SELECT COUNT(\*)

INTO v\_employee\_count

FROM employee

WHERE employee\_id =p\_employee\_id;

IF v\_employee\_count=1 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

IF v\_leaves\_taken<=10 THEN

UPDATE attendance

SET remaining\_unpaid\_leaves=(10-v\_leaves\_taken),

remaining\_paid\_leaves =15

WHERE employee\_id =p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

ELSE

UPDATE attendance

SET remaining\_unpaid\_leaves=0

WHERE employee\_id =p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

END IF;

IF v\_leaves\_taken>=10 AND v\_leaves\_taken<=25 THEN

UPDATE attendance

SET remaining\_paid\_leaves=(25-v\_leaves\_taken)

WHERE employee\_id =p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

ELSIF v\_leaves\_taken >25 THEN

UPDATE attendance

SET remaining\_paid\_leaves=0

WHERE employee\_id =p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

END IF;

SELECT remaining\_paid\_leaves

INTO v\_remaining\_paid\_leaves

FROM attendance

WHERE employee\_id =p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

IF v\_remaining\_paid\_leaves =0 THEN

IF (v\_leaves\_taken-v\_monthly\_leaves)<=25 THEN

UPDATE attendance

SET loss\_of\_pay =(v\_leaves\_taken-25)\*1000

WHERE employee\_id=p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

ELSE

UPDATE attendance

SET loss\_of\_pay =v\_monthly\_leaves\*1000

WHERE employee\_id=p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

END IF;

ELSE

UPDATE attendance

SET loss\_of\_pay =0

WHERE employee\_id=p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

END IF;

UPDATE attendance

SET leave\_encashable=v\_previous\_paid\_leaves\*1000 --leave encashment is entitled to be provided for previous complete years only

WHERE employee\_id =p\_employee\_id

AND YEAR =extract(YEAR FROM sysdate);

HTP.P('Record added successfully');

ELSE

HTP.P('no such employee is present as given');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_leaves TO PUBLIC;

show error;

**enter\_and\_maintain\_lencash**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_lencash(

p\_employee\_id IN NUMBER)--It is only for present year that's why no p\_salary\_year

AS

v\_service\_start\_date DATE;

v\_leave\_encashment NUMBER(10);

v\_employee\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

SELECT service\_start\_date

INTO v\_service\_start\_date

FROM employee

WHERE employee\_id=p\_employee\_id;

v\_status :=0;

SELECT COUNT(\*)

INTO v\_employee\_count

FROM employee

WHERE employee\_id =p\_employee\_id;

IF v\_employee\_count=1 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

SELECT leave\_encashable

INTO v\_leave\_encashment

FROM attendance

WHERE employee\_id =p\_employee\_id

AND DAY =extract(DAY FROM sysdate)

AND MONTH =extract(MONTH FROM sysdate)

AND YEAR =extract(YEAR FROM sysdate);

UPDATE annual\_salary

SET leave\_encashed=v\_leave\_encashment

WHERE employee\_id =p\_employee\_id

AND salary\_year =extract(YEAR FROM sysdate);

UPDATE attendance

SET remaining\_paid\_leaves=0

WHERE employee\_id =p\_employee\_id

AND YEAR BETWEEN extract(YEAR FROM v\_service\_start\_date) AND (extract(YEAR FROM sysdate)-1)

AND remaining\_paid\_leaves!=0;

HTP.P('Record added successfully');

ELSE

HTP.P('no such employee is present as given');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_leaves TO PUBLIC;

show error;

**enter\_and\_maintain\_tax**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE enter\_and\_maintain\_tax(

p\_employee\_id IN NUMBER,

p\_salary\_year IN NUMBER)

AS

v\_basic NUMBER(10);

v\_hra NUMBER(10,2); --50% of basic

v\_conveyance NUMBER(10,2); --Rs.1500\*12 per year

v\_medical NUMBER(10,2); --15k per year

v\_tds NUMBER(10,2);

v\_sum NUMBER(10,2);

v\_empsal\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

SELECT basic,

hra,

conveyance,

medical

INTO v\_basic,

v\_hra,

v\_conveyance,

v\_medical

FROM annual\_salary

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

SELECT basic+ hra+ conveyance+ medical+ lta+ monthly\_bonus+ shift\_allowance+ additional\_personal\_pay+leave\_encashed

INTO v\_sum

FROM annual\_salary

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

v\_status :=0;

SELECT COUNT(\*)

INTO v\_empsal\_count

FROM annual\_salary

WHERE employee\_id =p\_employee\_id

AND salary\_year =p\_salary\_year;

IF v\_empsal\_count =1 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

IF v\_sum <=15000\*12 THEN

UPDATE deduction

SET esi =0.0175\*v\_basic

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

ELSE

UPDATE deduction

SET esi =0

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

END IF;

UPDATE deduction

SET epf =0.12 \*v\_basic,

tax\_exemption = v\_hra+v\_conveyance+v\_medical,

tds =get\_tds(p\_employee\_id,p\_salary\_year,v\_sum)

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

ELSE

HTP.P('no such employee is present as given');

END IF;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_tax TO PUBLIC;

show error;

**generate\_pay\_slip**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE PROCEDURE generate\_pay\_slip(

p\_employee\_id IN NUMBER,

p\_date IN OUT DATE,

p\_lencash\_mode IN NUMBER)

AS

v\_first\_name VARCHAR2(100);

v\_last\_name VARCHAR2(100);

v\_employee\_type VARCHAR2(100);

v\_designation VARCHAR2(100);

v\_ESI\_no VARCHAR2(12);

v\_PAN VARCHAR2(10);

v\_PF\_no VARCHAR2(18);

v\_PF\_UAN\_no NUMBER(12);

v\_bank VARCHAR2(20);

v\_bank\_ac\_no NUMBER(16);

v\_service\_start\_date DATE;

v\_service\_end\_date DATE;

v\_department\_id\_1 NUMBER(10);

v\_department\_id\_2 NUMBER(10);

v\_department\_name\_1 VARCHAR2(100);

v\_department\_name\_2 VARCHAR2(100);

v\_basic NUMBER(10);

v\_hra NUMBER(10,2); --50% of basic

v\_conveyance NUMBER(10,2); --Rs.1500\*12 per year

v\_medical NUMBER(10,2); --15k per year

v\_lta NUMBER(10,2); --20k per year

v\_monthly\_bonus NUMBER(10,2); --20% of basic

v\_shift\_allowance NUMBER(10,2); --Rs.0

v\_additional\_personal\_pay NUMBER(10,2); --80% of basic

v\_leave\_encashed NUMBER(10);

v\_lencash\_days NUMBER(10);

v\_lpe NUMBER(1);

v\_sum NUMBER(10,2);

v\_sum\_new NUMBER(10,2);

v\_loss\_of\_pay NUMBER(10,2);

v\_esi NUMBER(10,2);

v\_epf NUMBER(10,2);

v\_profession\_tax NUMBER(10,2);

v\_tds NUMBER(10,2);

v\_tds\_new NUMBER(10,2);

v\_tds\_month NUMBER(10,2);

v\_gross\_earning NUMBER(10,2); --15k per year

v\_gross\_deduction NUMBER(10,2);

v\_net\_pay NUMBER(10,2);

v\_days\_payable NUMBER(2);

v\_calendar\_days NUMBER(2);

v\_empattend\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

page\_formatting('BEFORE');

p\_date :=add\_months(last\_day(p\_date),-1)+1;

v\_status :=0;

SELECT COUNT(\*)

INTO v\_empattend\_count

FROM attendance

WHERE employee\_id =p\_employee\_id

AND YEAR =extract(YEAR FROM p\_date)

AND MONTH = extract(MONTH FROM p\_date)-1;

IF v\_empattend\_count >0 THEN

v\_status :=1;

END IF;

IF v\_status =1 THEN

SELECT basic+ hra+ conveyance+ medical+ lta+ monthly\_bonus+ shift\_allowance+ additional\_personal\_pay

INTO v\_sum

FROM annual\_salary

WHERE employee\_id =p\_employee\_id

AND salary\_year =extract(YEAR FROM p\_date);

IF extract(YEAR FROM p\_date) =extract(YEAR FROM sysdate) THEN

IF lower(p\_lencash\_mode) ='yes' THEN

enter\_and\_maintain\_lencash(p\_employee\_id);

SELECT basic+ hra+ conveyance+ medical+ lta+ monthly\_bonus+ shift\_allowance+ additional\_personal\_pay+leave\_encashed

INTO v\_sum\_new

FROM annual\_salary

WHERE employee\_id =p\_employee\_id

AND salary\_year =extract(YEAR FROM p\_date);

UPDATE deduction

SET tds =get\_tds(p\_employee\_id,extract(YEAR FROM p\_date),v\_sum\_new)

WHERE employee\_id =p\_employee\_id

AND salary\_year =extract(YEAR FROM p\_date);

ELSIF lower(p\_lencash\_mode)='no' THEN

v\_sum\_new :=v\_sum;

ELSE

HTP.P('Invalid Input for p\_lencash\_mode');

END IF;

ELSE

SELECT leave\_encashed

INTO v\_leave\_encashed

FROM annual\_salary

WHERE employee\_id =p\_employee\_id

AND salary\_year =extract(YEAR FROM p\_date);

v\_lpe :=0;

IF v\_leave\_encashed!=0 THEN

SELECT COUNT(\*)

INTO v\_lpe

FROM

(SELECT \*

FROM attendance

WHERE employee\_id =p\_employee\_id

AND YEAR =extract(YEAR FROM p\_date)

AND leave\_encashable=0

AND MONTH =extract(MONTH FROM p\_date)+1

AND DAY =1

INTERSECT

SELECT \*

FROM attendance

WHERE employee\_id =p\_employee\_id

AND YEAR =extract(YEAR FROM p\_date)

AND leave\_encashable!=0

AND MONTH =extract(MONTH FROM p\_date)

AND DAY =1

);

END IF;

IF v\_lpe =1 THEN

v\_sum\_new:=v\_sum+v\_leave\_encashed;

ELSE

v\_sum\_new:=v\_sum;

END IF;

END IF;

v\_tds :=get\_tds(p\_employee\_id,extract(YEAR FROM p\_date),v\_sum);

v\_tds\_new :=get\_tds(p\_employee\_id,extract(YEAR FROM p\_date),v\_sum\_new)-v\_tds;

v\_tds\_month:=(v\_tds /12)+v\_tds\_new;

ELSE

HTP.P('no such employee is present in attendance table');

END IF;

SELECT first\_name,

last\_name,

employee\_type,

designation,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

INTO v\_first\_name,

v\_last\_name,

v\_employee\_type,

v\_designation,

v\_ESI\_no,

v\_PAN,

v\_PF\_no,

v\_PF\_UAN\_no,

v\_bank,

v\_bank\_ac\_no,

v\_service\_start\_date,

v\_service\_end\_date,

v\_department\_id\_1,

v\_department\_id\_2

FROM employee

WHERE employee\_id=p\_employee\_id;

SELECT department\_name

INTO v\_department\_name\_1

FROM department

WHERE department\_id=v\_department\_id\_1;

SELECT department\_name

INTO v\_department\_name\_2

FROM department

WHERE department\_id=v\_department\_id\_2;

SELECT basic,

hra,

conveyance,

medical,

lta,

monthly\_bonus,

shift\_allowance,

additional\_personal\_pay,

leave\_encashed

INTO v\_basic,

v\_hra,

v\_conveyance,

v\_medical,

v\_lta,

v\_monthly\_bonus,

v\_shift\_allowance,

v\_additional\_personal\_pay,

v\_leave\_encashed

FROM annual\_salary

WHERE employee\_id=p\_employee\_id

AND salary\_year =extract(YEAR FROM p\_date);

v\_lencash\_days :=v\_leave\_encashed/1000;

SELECT esi,

epf,

profession\_tax

INTO v\_esi,

v\_epf,

v\_profession\_tax

FROM deduction

WHERE employee\_id=p\_employee\_id

AND salary\_year =extract(YEAR FROM p\_date);

SELECT loss\_of\_pay

INTO v\_loss\_of\_pay

FROM attendance

WHERE employee\_id =p\_employee\_id

AND YEAR =extract(YEAR FROM p\_date)

AND MONTH =extract(MONTH FROM p\_date) -1

AND DAY =extract(DAY FROM last\_day(add\_months(p\_date,-1)));

v\_calendar\_days :=extract(DAY FROM last\_day(add\_months(p\_date,-1)));

v\_days\_payable :=v\_calendar\_days -(v\_loss\_of\_pay/1000);

v\_gross\_earning :=(v\_sum) /12+v\_leave\_encashed;

v\_gross\_deduction:=(v\_esi +v\_epf+v\_profession\_tax)/12+v\_tds\_month+v\_loss\_of\_pay;

v\_net\_pay :=v\_gross\_earning -v\_gross\_deduction;

COMMIT;

page\_formatting('AFTER');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN TOO\_MANY\_ROWS THEN

HTP.P(SQLCODE||' '||SQLERRM);

WHEN OTHERS THEN

HTP.P(SQLCODE||' '||SQLERRM);

END;

/

GRANT EXECUTE ON salary\_mgmt.enter\_and\_maintain\_tax TO PUBLIC;

show error;

**get\_tds**

CONNECT salary\_mgmt/dbase@xe

CREATE OR REPLACE FUNCTION get\_tds(

p\_employee\_id IN NUMBER,

p\_salary\_year IN NUMBER,

p\_sum IN NUMBER)

RETURN NUMBER

AS

v\_service\_end\_date DATE;

v\_te NUMBER(10,2);

v\_td NUMBER(10,2);

v\_ti NUMBER(10,2);

v\_tp NUMBER(10,2);

v\_tds NUMBER(10,2);

v\_empsal\_count NUMBER(10); --need to use count for all foreign keys to check integrity constraint

v\_status NUMBER(10);

BEGIN

v\_status :=0;

SELECT COUNT(\*)

INTO v\_empsal\_count

FROM annual\_salary

WHERE employee\_id =p\_employee\_id

AND salary\_year =p\_salary\_year;

IF v\_empsal\_count =1 THEN

v\_status :=1;

END IF;

IF v\_status=1 THEN

SELECT hra+conveyance+medical

INTO v\_te

FROM annual\_salary

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

SELECT tax\_deduction

INTO v\_td

FROM deduction

WHERE employee\_id=p\_employee\_id

AND salary\_year =p\_salary\_year;

v\_ti :=p\_sum-v\_te-v\_td;

IF v\_ti <=250000 THEN

v\_tp :=0;

ELSIF v\_ti <=500000 AND v\_ti>=250001 THEN

v\_tp :=0.1\*(v\_ti-250000);

ELSIF v\_ti <=1000000 AND v\_ti>=500001 THEN

v\_tp :=0.2\*(v\_ti-500000)+25000;

ELSIF v\_ti >=1000001 THEN

v\_tp :=0.3\*(v\_ti-1000000)+125000;

END IF;

IF (p\_sum -v\_te)>=10000000 THEN

v\_tds :=1.15\*v\_tp;

ELSE

v\_tds:=1.03\*v\_tp;

END IF;

RETURN v\_tds;

ELSE

RETURN NULL;

END IF;

EXCEPTION

WHEN OTHERS THEN

HTP.P(SQLCODE ||' '|| SQLERRM);

RETURN NULL;

END;

/

GRANT EXECUTE ON salary\_mgmt.get\_tds TO PUBLIC;

show error;

**install**

--set define off;

--add not null constraint if possible

CONNECT salary\_mgmt/dbase@XE;

---DROP SEQUENCE

DROP SEQUENCE employee\_id\_seq;

DROP SEQUENCE department\_id\_seq;

---DROP TABLE

DROP TABLE address;

DROP TABLE attendance;

DROP TABLE deduction;

DROP TABLE annual\_salary;

ALTER TABLE department

DROP CONSTRAINT fk\_emp;

DROP TABLE employee;

DROP TABLE department;

---SEQUENCE CREATION

CREATE SEQUENCE department\_id\_seq START WITH 101 INCREMENT BY 10 NOCACHE NOCYCLE MAXVALUE 1001;

CREATE SEQUENCE employee\_id\_seq START WITH 1001 INCREMENT BY 1 NOCACHE NOCYCLE MAXVALUE 99999;

--- TABLE CREATION

CREATE TABLE department

(

department\_id NUMBER(10) CONSTRAINT dept\_pk PRIMARY KEY,

department\_name VARCHAR2(100),

creation\_date DATE DEFAULT sysdate,

created\_by VARCHAR2(100) DEFAULT USER,

last\_updated\_date DATE DEFAULT sysdate,

last\_updated\_by VARCHAR2(100) DEFAULT USER

);

CREATE TABLE employee

(

employee\_id NUMBER(10) CONSTRAINT emp\_pk PRIMARY KEY,

first\_name VARCHAR2(100),

last\_name VARCHAR2(100),

date\_of\_birth DATE,

employee\_type VARCHAR2(100),

designation VARCHAR2(100),

email VARCHAR2(100),

contact\_no NUMBER(10),

ESI\_no VARCHAR2(12),

PAN VARCHAR2(10),

PF\_no VARCHAR2(18),

PF\_UAN\_no NUMBER(12),

bank VARCHAR2(20),

bank\_ac\_no NUMBER(16),

service\_start\_date DATE,

service\_end\_date DATE,

creation\_date DATE DEFAULT sysdate,

created\_by VARCHAR2(100) DEFAULT USER,

last\_updated\_date DATE DEFAULT sysdate,

last\_updated\_by VARCHAR2(100) DEFAULT USER,

department\_id\_1 CONSTRAINT fk\_dept\_1 REFERENCES department(department\_id),

department\_id\_2 CONSTRAINT fk\_dept\_2 REFERENCES department(department\_id)

);

ALTER TABLE department ADD department\_head\_id CONSTRAINT fk\_emp REFERENCES employee(employee\_id);

CREATE TABLE annual\_salary

(

employee\_id,

basic NUMBER(10),

hra NUMBER(10,2), --50% of basic

conveyance NUMBER(10,2), --Rs.1500\*12 per year

medical NUMBER(10,2), --15k per year

lta NUMBER(10,2), --20k per year

monthly\_bonus NUMBER(10,2), --20% of basic

shift\_allowance NUMBER(10,2), --Rs.0

additional\_personal\_pay NUMBER(10,2), --80% of basic

leave\_encashed NUMBER(10) DEFAULT 0,

salary\_year NUMBER(4) DEFAULT extract(YEAR FROM sysdate),

creation\_date DATE DEFAULT sysdate,

created\_by VARCHAR2(100) DEFAULT USER,

last\_updated\_date DATE DEFAULT sysdate,

last\_updated\_by VARCHAR2(100) DEFAULT USER,

CONSTRAINT annual\_pk PRIMARY KEY(employee\_id,salary\_year),

CONSTRAINT fk\_emp1 FOREIGN KEY(employee\_id) REFERENCES employee(employee\_id)

);

CREATE TABLE deduction

(

employee\_id,

esi NUMBER(10,2),

epf NUMBER(10,2),

profession\_tax NUMBER(10,2) DEFAULT 2500,

tax\_exemption NUMBER(10,2),

tax\_deduction NUMBER(10,2) DEFAULT 0,

tds NUMBER(10,2),

salary\_year NUMBER(4) DEFAULT extract(YEAR FROM sysdate),

creation\_date DATE DEFAULT sysdate,

created\_by VARCHAR2(100) DEFAULT USER,

last\_updated\_date DATE DEFAULT sysdate,

last\_updated\_by VARCHAR2(100) DEFAULT USER,

CONSTRAINT deduct\_pk PRIMARY KEY(employee\_id,salary\_year),

CONSTRAINT fk\_sal1 FOREIGN KEY(employee\_id,salary\_year) REFERENCES annual\_salary(employee\_id,salary\_year)

);

CREATE TABLE attendance

(

employee\_id,

DAY NUMBER(2) DEFAULT extract(DAY FROM sysdate),

MONTH NUMBER(2) DEFAULT extract(MONTH FROM sysdate),

YEAR NUMBER(4) DEFAULT extract(YEAR FROM sysdate),

status VARCHAR2(10) NOT NULL,

allotted\_leaves NUMBER(2) DEFAULT 25,

allotted\_unpaid\_leaves NUMBER(2) DEFAULT 15,

allotted\_paid\_leaves NUMBER(2) DEFAULT 10,

remaining\_unpaid\_leaves NUMBER(2) DEFAULT 15,

remaining\_paid\_leaves NUMBER(2) DEFAULT 10,

loss\_of\_pay NUMBER(10,2) DEFAULT 0,

leave\_encashable NUMBER(10),

CONSTRAINT attend\_pk PRIMARY KEY(employee\_id,DAY,MONTH,YEAR),

CONSTRAINT fk\_emp2 FOREIGN KEY(employee\_id) REFERENCES employee(employee\_id)

);

CREATE TABLE address

(

employee\_id,

address\_1 VARCHAR2(100),

address\_2 VARCHAR2(100),

address\_3 VARCHAR2(100),

city VARCHAR2(100),

pincode NUMBER(6),

state VARCHAR2(100),

country VARCHAR2(100),

address\_type VARCHAR2(100),

creation\_date DATE DEFAULT sysdate,

created\_by VARCHAR2(100) DEFAULT USER,

last\_updated\_date DATE DEFAULT sysdate,

last\_updated\_by VARCHAR2(100) DEFAULT USER,

CONSTRAINT address\_pk PRIMARY KEY(employee\_id,address\_type),

CONSTRAINT fk\_emp3 FOREIGN KEY(employee\_id) REFERENCES employee(employee\_id)

);

--Department table insertion

INSERT

INTO department

(

department\_id,

department\_name,

department\_head\_id

)

VALUES

(

department\_id\_seq.NEXTVAL,

'Directors',

NULL

);

INSERT

INTO department

(

department\_id,

department\_name,

department\_head\_id

)

VALUES

(

department\_id\_seq.NEXTVAL,

'Finance',

NULL

);

INSERT

INTO department

(

department\_id,

department\_name,

department\_head\_id

)

VALUES

(

department\_id\_seq.NEXTVAL,

'Human Resources',

NULL

);

INSERT

INTO department

(

department\_id,

department\_name,

department\_head\_id

)

VALUES

(

department\_id\_seq.NEXTVAL,

'Technical',

NULL

);

INSERT

INTO department

(

department\_id,

department\_name,

department\_head\_id

)

VALUES

(

department\_id\_seq.NEXTVAL,

'Marketting',

NULL

);

--employee table insertion

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Cyrus',

'Mistry',

'04-JUL-68',

'permanent',

'CHAIRMAN',

'cyrusmistry@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

'01-MAR-93',

'04-JUL-28',

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Natarajan',

'Chandrasekaran',

'14-JUN-65',

'permanent',

'CEO',

'nc@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

'11-APR-94',

'14-JUN-25',

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Aarthi',

'Subramanian',

'24-APR-62',

'permanent',

'Non-Independent Director',

'as@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

'04-JUL-2028',

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Aman',

'Mehta',

'04-SEP-68',

'permanent',

'Independent Director',

'am@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

add\_months(TRUNC(sysdate),60),

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Venkatraman',

'Thyagarajan',

'14-NOV-62',

'permanent',

'Independent Director',

'vt@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

add\_months(TRUNC(sysdate),60),

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Prof. Clayton M',

'Christensen',

'07-JUL-70',

'permanent',

'Independent Director',

'pcmc@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

add\_months(TRUNC(sysdate),60),

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Dr. Ron',

'Sommer',

'23-MAR-72',

'permanent',

'Independent Director',

'drs@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

add\_months(TRUNC(sysdate),60),

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Dr. Vijay',

'Kelkar',

'17-JAN-73',

'permanent',

'Independent Director',

'dvk@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

add\_months(TRUNC(sysdate),60),

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Ishaat',

'Hussain',

'04-DEC-69',

'permanent',

'Non-Independent Director',

'ih@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

'04-DEC-29',

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'O.P.',

'Bhatt',

'07-MAY-71',

'permanent',

'Independent Director',

'opb@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

TRUNC(sysdate),

add\_months(TRUNC(sysdate),60),

101,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Rajesh',

'Gopinathan',

'05-FEB-74',

'permanent',

'CFO and Director',

'rg@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

'05-FEB-34',

'04-JUL-28',

111,101

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Ajoyendra',

'Mukherjee',

'12-JUL-66',

'permanent',

'COO',

'ajm@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

'01-JUN-93',

'12-JUL-26',

121,121

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'K Ananth',

'Krishnan',

'09-AUG-65',

'permanent',

'CTO',

'kak@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

'01-OCT-94',

'09-AUG-25',

131,131

);

INSERT

INTO employee

(

employee\_id,

first\_name,

last\_name,

date\_of\_birth,

employee\_type,

designation,

email,

contact\_no,

esi\_no,

PAN,

PF\_no,

PF\_UAN\_no,

bank,

bank\_ac\_no,

service\_start\_date,

service\_end\_date,

department\_id\_1,

department\_id\_2

)

VALUES

(

employee\_id\_seq.NEXTVAL,

'Raja',

'Banerji',

'11-JUL-66',

'permanent',

'CMO',

'rb@tcs.com',

'9425371289',

'Not Eligible',

'BFFPM0532J',

'MP/CYR/34223/76232',

'100286405598',

'SBI',

'0742001684975213',

'11-NOV-92',

'11-JUL-26',

141,141

);

--department\_head\_id insertion

UPDATE department

SET department\_head\_id=1001

WHERE department\_id =101;

UPDATE department SET department\_head\_id=1011 WHERE department\_id=111;

UPDATE department SET department\_head\_id=1012 WHERE department\_id=121;

UPDATE department SET department\_head\_id=1013 WHERE department\_id=131;

UPDATE department SET department\_head\_id=1014 WHERE department\_id=141;

--annual\_salary table insertion

INSERT

INTO annual\_salary

(

employee\_id,

basic

)

VALUES

(

1001,400000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1002,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1003,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1004,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1005,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1006,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1007,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1008,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1009,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1010,320000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1011,240000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1012,240000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1013,240000\*12

);

INSERT INTO annual\_salary

( employee\_id, basic

) VALUES

( 1014,240000\*12

);

UPDATE annual\_salary

SET hra =0.5\*basic,

conveyance =1600,

medical =15000,

lta =18000,

monthly\_bonus =0.2\*basic,

shift\_allowance =0,

additional\_personal\_pay=0.8\*basic

WHERE employee\_id <1015;

--deduction table insertion

INSERT INTO deduction

( employee\_id

) VALUES

( 1001

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1002

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1003

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1004

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1005

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1006

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1007

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1008

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1009

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1010

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1011

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1012

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1013

);

INSERT INTO deduction

( employee\_id

) VALUES

( 1014

);

--attendance table insertion

INSERT

INTO attendance

(

employee\_id,

status

)

VALUES

(

1001,

'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1002,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1003,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1004,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1005,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1006,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1007,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1008,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1009,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1010,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1011,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1012,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1013,'PRESENT'

);

INSERT INTO attendance

( employee\_id, status

) VALUES

( 1014,'PRESENT'

);

COMMIT;