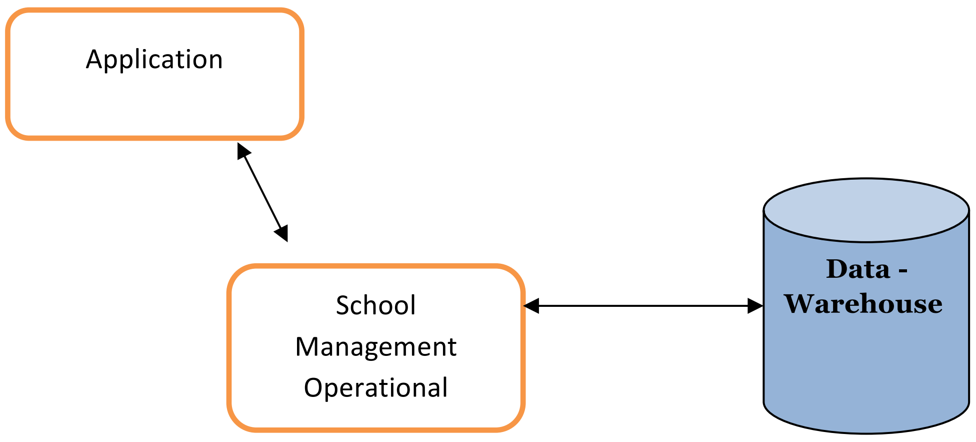
School ERP Solutions

Project Architecture



Project Main Modules:-

- Students Data Management
- Staff Management(teaching and non-teaching)
- Courses & Subjects management
- Student Attendance
- Staff attendance
- Exam and Result management
- Student Fees payment & Balance Tracking.
- Staff payroll
- Historical Data Maintenance.

An operational system is a system that supports the day to day activities of the school or any institution

Data Warehouse maintains historical data and is used for query , analysis and Decision Making. Basically it is used for advance reporting.

ER model for School Management System

Module 1:- Student Data Management.

STUDENT_Tbl

Column Name	Datatype	Remarks
Stud_id	Number	Auto generated & PK
Stud_name	Varchar2(100)	Not Null
Stud_address	Varchar2(1000)	
Stud_contact_no	Number	
Stud_Standard_id	Number	This is a FK to the course_id column in the courses_tbl.
Stud_div	Char	
Parent_name	Varchar2(100)	
P_contact_no	Number	
P_office_add	Varchar2(1000)	
P_Mobile_no	Number	
P_email_id	Varchar2(300)	

➔ Writing the Stored Procedures for Adding, Editing and Deleting data in Student table.

Adding :-

```
  create sequence stud_id_seq
        increment by 1
        start with 1
        minvalue 1
        nomaxvalue
        nocycle
        nocache;

Sequence created.

SQL> create or replace procedure add_proc(
    stud_name IN student_tbl.stud_name%type,
    stud_address IN student_tbl.stud_address%type,
    stud_contact_no IN student_tbl.stud_contact_no%type,
    stud_standard_id IN student_tbl.stud_standard_id%type,
    stud_div IN student_tbl.stud_div%type,
    parent_name IN student_tbl.parent_name%type,
    p_contact_no IN student_tbl.p_contact_no%type,
    p_office_add IN student_tbl.p_office_add%type,
    p_mobile_no IN student_tbl.p_mobile_no%type,
    p_email_id IN student_tbl.p_email_id%type)
    is
    Begin
    insert into student_tbl values(stud_id_seq.nextval,stud_name,stud_address,stud_contact_no,
stud_standard_id,stud_div,parent_name,p_contact_no,p_office_add,p_mobile_no,p_email_id);
  end;
  /

Procedure created.
```

➔ **Update/Edit :-**

```
SQL> create or replace procedure edit_proc(
        stud_id_pr IN student_tbl.stud_id%type,
        stud_name_pr IN student_tbl.stud_name%type,
            stud_address_pr IN student_tbl.stud_address%type,
            stud_contact_no_pr IN student_tbl.stud_contact_no%type,
            stud_standard_id_pr IN student_tbl.stud_standard_id%type,
            stud_div_pr IN student_tbl.stud_div%type,
            parent_name_pr IN student_tbl.parent_name%type,
           p_contact_no_pr IN student_tbl.p_contact_no%type,
          p_office_add_pr IN student_tbl.p_office_add%type,
          p_mobile_no_pr IN student_tbl.p_mobile_no%type,
          p_email_id_pr IN student_tbl.p_email_id%type)
          is
   abc exception;
   x char(10);
      Begin
   Select to_char(sysdate, 'dy') into x from dual;
   If x in ('sat','sun') then
   Raise abc;
   else
      update student_tbl set stud_name=stud_name_pr,
      stud_contact_no=stud_contact_no_pr,
      stud_address=stud_address_pr,
      stud_standard_id=stud_standard_id_pr,
      stud_div=stud_div_pr,
      parent_name=parent_name_pr,
      p_contact_no=P_contact_no_pr,
      p_office_add=p_office_add_pr,
      p_mobile_no=p_mobile_no_pr,
      p_email_id=p_email_id_pr
      WHERE stud_id=stud_id_pr;
   End if;
       Exception when abc then
   Dbms_output.put_line('Deletion not possible today');
      end;
   /

Procedure created.
```

➔ **Delete :-**

```
SQL> create or replace procedure delete_proc(
    stud_id_pr IN student_tbl.stud_id%type)
    is
    Begin
    delete student_tbl where stud_id=stud_id_pr;
    end;
    /

Procedure created.
```

➔ **Search Button :-**

```
SQL> create or replace procedure search_proc(
   stud_id_pr IN student_tbl.stud_id%type,
   stud_name_pr OUT student_tbl.stud_name%type,
   stud_address_pr OUT student_tbl.stud_address%type,
   stud_contact_no_pr OUT student_tbl.stud_contact_no%type,
   stud_standard_id_pr OUT student_tbl.stud_standard_id%type,
   stud_div_pr OUT student_tbl.stud_div%type,
   parent_name_pr OUT student_tbl.parent_name%type,
   p_contact_no_pr OUT student_tbl.p_contact_no%type,
   p_office_add_pr OUT student_tbl.p_office_add%type,
   p_mobile_no_pr OUT student_tbl.p_mobile_no%type,
   p_email_id_pr OUT student_tbl.p_email_id%type)
   is
   Begin
   select
stud_name,stud_address,stud_contact_no,stud_standard_id,stud_div,parent_name,p_contact_no,p_office_add,p_mobile_no,p_email_id into

stud_name_pr,stud_address_pr,stud_contact_no_pr,stud_standard_id_pr,stud_div_pr,parent_name_pr,p_contact_no_pr,p_office_add_pr,p_mobile_no_pr,p_email_id_pr
   from student_tbl
   where stud_id=stud_id_pr;
   end;
   /

Procedure created.
```

Execution :-

```
SQL> variable g_name varchar2(20);
SQL> variable g_address varchar2(20);
SQL> variable g_contact number;
SQL> variable g_standard number;
SQL> variable g_div char(10);
SQL> variable g_p_name varchar(20);
SQL> variable g_p_name varchar2(20);
SQL> variable g_p_contact number;
SQL> variable g_p_office varchar2(20);
SQL> variable g_p_mobile number;
SQL> variable g_p_email varchar2(20);
SQL> exec
search_proc(1,:g_name,:g_address,:g_contact,:g_standard,:g_div,:g_p_name,:g_p_contact,:g_p_office,:g_p_mobile,:g_p_email);

PL/SQL procedure successfully completed.

SQL> print g_name

G_NAME
--------------------------------------------------------------------------------
Rohan
SQL> print g_address
G_ADDRESS
--------------------------------------------------------------------------------
Gokul APT
```

➔ Creating STUDENT_BKP and STUDENT_HIST as exact replica of STUDENT Table.

```
SQL> create table student_bkp(
     stud_id number primary key,
       stud_name varchar2(100)not null,
        stud_address varchar2(1000),
        stud_contact_no number,
        stud_standard_id number,
        constraint student_courses_fk_bkp foreign key(stud_standard_id) references
courses_tbl(course_id),
        stud_div Char,
       Parent_name varchar2(100),
      P_contact_no number,
      P_office_add varchar2(1000),
      P_mobile_no number,
      P_email_id varchar2(300));

Table created.

SQL> create table student_hist(
     stud_id number primary key,
       stud_name varchar2(100)not null,
        stud_address varchar2(1000),
        stud_contact_no number,
        stud_standard_id number,
        constraint student_courses_fk_hist foreign key(stud_standard_id) references
courses_tbl(course_id),
        stud_div Char,
        Parent_name varchar2(100),
      P_contact_no number,
      P_office_add varchar2(1000),
      P_mobile_no number,
      P_email_id varchar2(300));

Table created.
```

➔ Whenever Data is added using the ADD button, it should be automatically populate the same in STUDENT_BKP table using Triggers.

```
SQL> create or replace trigger trg_bkp
after insert on student_tbl for each row
Begin
insert into student_bkp
values(:new.stud_id,:new.stud_name,:new.stud_address,:new.stud_contact_no,:new.stud_standard_id,:
new.stud_div,:new.parent_name,
 :new.p_contact_no,:new.p_office_add,:new.p_mobile_no,:new.p_email_id);
end;
/

Trigger created.
```

➔ Similarly on DELETE, it should be moved to the STUDENT_HIST table. Also DELETE should not be allowed if there is any pending Fees Balance.

```
SQL> create or replace trigger trg_hist
    before delete on student_tbl
    for each row
    Declare
    x  stud_fees_details.fees_balance%type;
    begin
    select fees_balance into X from stud_fees_details where stud_id=:old.stud_id;
    if x>0 then
    Raise_application_error(-20001,'deletion not allowed');
   else
   insert into student_hist
values(:old.stud_id,:old.stud_name,:old.stud_address,:old.stud_contact_no,:old.stud_standard_id,:old.s
tud_div,
:old.parent_name,:old.p_contact_no,:old.p_office_add,:old.p_mobile_no,:old.p_email_id);
   end if;
   end;
   /

Trigger created.
```

➔ In edit section implement user defined exception that prevents a student being updated from class N to class N+2 or more i.e. only valid update is from standard N to N+1.

```
create or replace procedure edit_proc(
            stud_id_pr IN student_tbl.stud_id%type,
            stud_name_pr IN student_tbl.stud_name%type,
                stud_address_pr IN student_tbl.stud_address%type,
                stud_contact_no_pr IN student_tbl.stud_contact_no%type,
                stud_standard_id_pr IN student_tbl.stud_standard_id%type,
                stud_div_pr IN student_tbl.stud_div%type,
                parent_name_pr IN student_tbl.parent_name%type,
               p_contact_no_pr IN student_tbl.p_contact_no%type,
             p_office_add_pr IN student_tbl.p_office_add%type,
             p_mobile_no_pr IN student_tbl.p_mobile_no%type,
             p_email_id_pr IN student_tbl.p_email_id%type)
             is
      abc exception;
      edit_standard exception;
      x char(10);
      y number(6);
         Begin
      Select to_char(sysdate, 'dy') into x from dual;
      If x in ('sat','sun') then
      Raise abc;
      Else
         Select stud_standard_id into x from student_tbl where stud_id=stud_id_pr;
         If(x+1=stud_standard_id_pr) then
         update student_tbl set stud_name=stud_name_pr,
         stud_contact_no=stud_contact_no_pr,
         stud_address=stud_address_pr,
         stud_standard_id=stud_standard_id_pr,
         stud_div=stud_div_pr,
         parent_name=parent_name_pr,
         p_contact_no=P_contact_no_pr,
         p_office_add=p_office_add_pr,
         p_mobile_no=p_mobile_no_pr,
         p_email_id=p_email_id_pr
         WHERE stud_id=stud_id_pr;
      End if;
      End if;
          Exception when abc then
      Dbms_output.put_line('Update not possible today');
          When edit_standard then
           Dbms_output.put_line('Standard can be updated by +1 only');
         end;
   /

Procedure created.
```

➔ Whenever a new student is added, an automatic insert should go in the student fees table with fees_paid as 0 and Total and Balance fees same as the course fees.

```
create or replace trigger trig_Q8
      after insert or update on student_tbl for each row
      declare
      x number;
      begin
      select course_fees into X from courses_tbl
      where course_id= :new.stud_standard_id;
      insert into stud_fees_details(stud_id,fees_paid,fees_balance,total_fees) values
(:new.stud_id,0,x,x);
      end;
     /

Trigger created.
```

Module 2:- Staff Data Management.

STAFF_Tbl

Column Name	Datatype	Remarks
staff_id	Number	Auto generated
staff_name	Varchar2(100)	Not Null
staff_address	Varchar2(1000)	
staff_contact_no	Number	
staff_email	Varchar2(100)	
staff_Designation	Varchar2(100)	
staff_Type	Char	
staff_Qualification	Varchar2(300)	

- ➔ Create a Data Entry Screen to ADD, EDIT, and DELETE Data from EMP table using Stored Procedures in Oracle.

```
SQL> create table staff_tbl(staff_id number(5),staff_name varchar(30) not null,staff_address
varchar(50),staff_contact_no number(20),staff_email varchar(20),staff_designation varchar(20),staff_type
varchar(20),staff_Qualification varchar(50));



Table created.
```

- ➔ Write Stored Procedures for Adding, Editing and Deleting data in EMP table.

```
SQL> create or replace procedure add_staff
    (staff_name_pr IN staff_tbl.staff_name%type,
    staff_address_pr IN staff_tbl.staff_address%type,
    staff_contact_no_pr IN staff_tbl.staff_contact_no%type,
    staff_email_pr IN staff_tbl.staff_email%type,
    staff_designation_pr IN staff_tbl.staff_designation%type,
    staff_type_pr IN staff_tbl.staff_type%type,
    staff_Qualification_pr IN staff_tbl.staff_Qualification%type)
    IS
   BEGIN
   insert into staff_tbl
values(staff_id_seq.nextval,staff_name_pr,staff_address_pr,staff_contact_no_pr,staff_email_pr,staff_desi
gnation_pr,staff_type_pr,staff_Qualification_pr);
   End;
   /
Procedure created.
```

➔ Implement a Search Button on STAFF ID.

```
SQL> create or replace procedure search_staff(
       staff_id_pr IN staff_tbl.staff_id%type,
       staff_name_pr OUT staff_tbl.staff_name%type,
       staff_address_pr OUT staff_tbl.staff_address%type,
       staff_contact_no_pr OUT staff_tbl.staff_contact_no%type,
       staff_email_pr OUT staff_tbl.staff_email%type,
       staff_designation_pr OUT staff_tbl.staff_designation%type,
       staff_type_pr OUT staff_tbl.staff_type%type,
       staff_qualification_pr OUT staff_tbl.staff_qualification%type)
      is
      Begin
      select
staff_name,staff_address,staff_contact_no,staff_email,staff_designation,staff_type,staff_qualificati
on into

staff_name_pr,staff_address_pr,staff_contact_no_pr,staff_email_pr,staff_designation_pr,staff_type
_pr,staff_qualification_pr
      from staff_tbl
      where staff_id=staff_id_pr;
      end;
      /

Procedure created.
```

➔ Whenever Data is added using the ADD button, it should be automatically populate the same in EMP_BKP table using Triggers.

```
SQL> create table staff_tbl_bkp(staff_id number(5),staff_name varchar(30) not null,staff_address
varchar(50),staff_contact_no number(20),staff_email varchar(20),staff_designation varchar(20),staff_type
varchar(20),staff_Qualification varchar(50));

Table created.

SQL> create or replace trigger trg_staff_bkp
   after insert on staff_tbl for each row
   Begin
   insert into staff_tbl_bkp
values(:new.staff_id,:new.staff_name,:new.staff_address,:new.staff_contact_no,:new.staff_email,:new.st
aff_designation,:new.staff_type,
    :new.staff_Qualification);
   end;
   /

Trigger created.
```

➔ Similarly on DELETE, it should be moved to the EMP_HIST table.

```
SQL> create table staff_tbl_hist(staff_id number(5),staff_name varchar(30) not null,staff_address
varchar(50),staff_contact_no number(20),staff_email varchar(20),staff_designation varchar(20),staff_type
varchar(20),staff_Qualification varchar(50));

Table created.

create or replace procedure staff_delete(
        staff_id_pr IN staff_tbl.staff_id%type)
        is
        Begin
        delete staff_tbl where staff_id=staff_id_pr;
        end;
        /

Procedure created.

SQL> create or replace trigger trg_staff_hist
    before delete on staff_tbl for each row
    Begin
    insert into staff_tbl_hist
values(:old.staff_id,:old.staff_name,:old.staff_address,:old.staff_contact_no,:old.staff_email,:old.staff_des
ignation,:old.staff_type,
     :old.staff_Qualification);
    end;
    /

Trigger created.
```

➔ Editing staff records

```
SQL> create or replace procedure edit_staff(
        staff_id_pr IN staff_tbl.staff_id%type,
        staff_name_pr IN staff_tbl.staff_name%type,
        staff_address_pr IN staff_tbl.staff_address%type,
        staff_contact_no_pr IN staff_tbl.staff_contact_no%type,
        staff_email_pr IN staff_tbl.staff_email%type,
        staff_designation_pr IN staff_tbl.staff_designation%type,
        staff_type_pr IN staff_tbl.staff_type%type,
        staff_qualification_pr IN staff_tbl.staff_qualification%type)
       IS
       BEGIN
       UPDATE staff_tbl set
       staff_name=staff_name_pr,
       staff_address=staff_address_pr,
       staff_contact_no=staff_contact_no_pr,
       staff_email=staff_email_pr,
       staff_designation=staff_designation_pr,
       staff_type=staff_type_pr,
       staff_qualification=staff_qualification_pr
       where
       staff_id=staff_id_pr;
       END;
       /
Procedure created.
```

➔ Implement sequences to generate emp_id.

```
SQL> create sequence staff_id_seq

1.        increment by 1
2.        start with 1
3.        nomaxvalue
4.        nocycle
5.       nocache;
6.   Sequence created.
```

Module 3:- Courses and Subject Management

Courses_tbl

Column Name	Datatype	Remarks
Course_id	Number	This can be standard for school , also for College it can be FYBA,FYBCOM
Course_name	Varchar2(100)	Not Null
Course_Type	Varchar2(100)	Primary,Secondary, Junior College
Course_Div	Char	This is the Division ..ie A, B,C
Course_Fees	Number	

Subjects_tbl

Column Name	Datatype	Remarks
Subject_id	Number	Auto generated
Sub_name	Varchar2(100)	Not Null
Sub_Max_marks	Number	
Sub_Passing_Marks	Number	
Pract_Marks	Number	
Pract_pass_marks	Number	

Courses_Sub_tbl *(Many to Many Relations)*

Column Name	Datatype	Remarks
Course_id	Number	
Subject_id	Number	
Eff_Start_date	Date	
Eff_End_date	Date	

➔ Inserting courses offered/available in the school.

```
SQL> create sequence subject_id_seq
    increment by 1
    start with 1
    minvalue 1
    nomaxvalue
    nocycle
    cache 20;

Sequence created.

SQL> create or replace procedure insert_course_proc(
   course_id_pr IN courses_tbl.course_id%type,
   course_name_pr IN courses_tbl.course_name%type,
   course_type_pr IN courses_tbl.course_type%type,
   course_fees_pr IN courses_tbl.course_fees%type)
   IS
   BEGIN
   insert into courses_tbl values(course_id_pr,course_name_pr,course_type_pr,course_fees_pr);
   end;
   /

Procedure created.


SQL> create or replace procedure delete_course_proc(course_id_pr IN courses_tbl.course_id%type)
    IS
    BEGIN
    delete from courses_tbl where course_id=course_id_pr;
    end;
    /

Procedure created.


SQL> create or replace procedure edit_course_proc(
    course_id_pr IN courses_tbl.course_id%type,
    course_name_pr IN courses_tbl.course_name%type,
    course_type_pr IN courses_tbl.course_type%type,
    course_fees_pr IN courses_tbl.course_fees%type)
    IS
    BEGIN
    update courses_tbl set course_id=course_id_pr,
   course_name=course_name_pr,
   course_type=course_type_pr,
   course_fees=course_fees_pr where course_id=course_id_pr;
   end;
   /
Procedure created.
```

Module 4:- Student Attendance

Stud_Attendance_Details

Column Name	Datatype	Remarks
Stud_id	Number	Auto generated
Stud_standard	Number	Columns added to avoid joins
Stud_div	Char	Columns added to avoid joins
Stud_attend	Char	P/A
Attendance_date	Date	Should be defaulted to Sysdate.

➔ Attendance Data entry should be done only for absent students, for other students data should be automatically inserted as present.(Use of Cursors or Use of Minus operator).

Submit Button

```
SQL> create or replace procedure stud_attendance_absent(
stud_id_pr IN stud_attendance_details.stud_id%type,
Stud_standard_id_pr IN stud_attendance_details.stud_standard%type,
Stud_div_pr IN stud_attendance_details.stud_div%type,
Stud_attend_pr IN stud_attendance_details.stud_attend%type,
Attendance_date_pr IN stud_attendance_details.attendance_date%type)
   is
   begin
   insert into stud_attendance_details
values(stud_id_pr,stud_standard_id_pr,stud_div_pr,stud_attend_pr,attendance_date_pr);
   exception when  OTHERS then
   update stud_attendance_details set stud_attend='A' where stud_id=stud_id_pr and
attendance_date=attendance_date_pr;
   end;
   /

Procedure created.

SQL> create or replace procedure trial_attend(
   stud_id_pr IN stud_attendance_details.stud_id%type,
   stud_standard_pr IN stud_attendance_details.stud_standard%type,
   stud_div_pr IN stud_attendance_details.stud_div%type,
   stud_attend_pr IN stud_attendance_details.stud_attend%type,
   attendance_date_pr IN stud_attendance_details.attendance_date%type)
   IS
   X NUMBER(5);
   BEGIN
  select 1 into X  from stud_attendance_details where stud_id=stud_id_pr AND
attendance_date=attendance_date_pr;
  if x=1 then
  UPDATE stud_attendance_details set stud_attend=stud_attend_pr where stud_id=stud_id_pr AND
attendance_date=attendance_date_pr;
  ElSE
  insert into stud_attendance_details
values(stud_id_pr,stud_standard_pr,stud_div_pr,stud_attend_pr,attendance_date_pr);
  END IF;
  END;
  /
Procedure created.
```

Trial Save Button (complicated Array concept)

```
SQL> Create or replace procedure stud_attendance_present(
    attendance_date_pr IN stud_attendance_details.attendance_date%type)
    is
    Type attendance_rec is record(stud_id number,stud_standard number,stud_div char);
    Type attendance_plsqltab IS TABLE OF attendance_rec INDEX BY BINARY_INTEGER;
    ar attendance_plsqltab;
    i binary_integer:=1;
    cursor c1 is
    select stud_id,stud_standard_id,stud_div  from student_tbl
       Minus
       select stud_id,stud_standard,stud_div from stud_attendance_details where
attendance_date=attendance_date_pr;
   Begin
   for X in c1
   LOOP
   ar(i).stud_id:=X.stud_id;
   ar(i).stud_standard:=X.stud_standard_id;
   ar(i).stud_div:=X.stud_div;
   Insert into stud_attendance_details values(ar(i).stud_id,ar(i).stud_standard,ar(i).stud_div,
'P',attendance_date_pr);
   End loop;
   End;
   /

Procedure created.

SQL> exec stud_attendance_present('1-Aug-2016');

PL/SQL procedure successfully completed.

SQL> select * from stud_attendance_details;

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        13           100 A A 23-JUL-16
        14           100 A   23-JUL-16
        15           100 B A 23-JUL-16
        16           100 A A 23-JUL-16
        17           100 A A 23-JUL-16
        19           100 A   26-JUL-16
        20           100 A   26-JUL-16
        21             2 A A 26-JUL-16
        22             2 A A 26-JUL-16
        25             2 B A 26-JUL-16
        26             2 B A 26-JUL-16

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        45           100 A   05-AUG-16
        46           100 A   05-AUG-16
        47             2 A   05-AUG-16
        47             2 A A 01-AUG-16
        46           100 A A 01-AUG-16
```

```
         1                A P 01-AUG-16
         2            100 A P 01-AUG-16
         3            100 A P 01-AUG-16
        10            100 A P 01-AUG-16
        23            100 A P 01-AUG-16
        24            100 A P 01-AUG-16

22 rows selected.
```

<u>Save Button (Using CURSOR)</u>

Final Save Button (Using ONLY CURSOR)

```
SQL> Create or replace procedure stud_attendance_present(
    attendance_date_pr IN stud_attendance_details.attendance_date%type)
    is
    cursor c1 is
    select stud_id,stud_standard_id,stud_div  from student_tbl
       Minus
       select stud_id,stud_standard,stud_div from stud_attendance_details where
attendance_date=attendance_date_pr;
   Begin
   for X in c1
   LOOP
   Insert into stud_attendance_details values(X.stud_id,X.stud_standard_id,X.stud_div,
'P',attendance_date_pr);
   End loop;
   End;
   /

Procedure created.
```

<u>Execution :-</u>

```
SQL> exec stud_attendance_absent(45,100,'A','A','2-Aug-2016');

PL/SQL procedure successfully completed.

SQL> select * from stud_attendance_details;

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        13           100 A A 23-JUL-16
        14           100 A   23-JUL-16
        15           100 B A 23-JUL-16
        16           100 A A 23-JUL-16
        17           100 A A 23-JUL-16
        19           100 A   26-JUL-16
        20           100 A   26-JUL-16
        21             2 A A 26-JUL-16
        22             2 A A 26-JUL-16
        25             2 B A 26-JUL-16
        26             2 B A 26-JUL-16

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        45           100 A   05-AUG-16
        46           100 A   05-AUG-16
        47             2 A   05-AUG-16
        47             2 A A 01-AUG-16
        46           100 A A 01-AUG-16
         1               A P 01-AUG-16
         2           100 A P 01-AUG-16
         3           100 A P 01-AUG-16
        10           100 A P 01-AUG-16
        23           100 A P 01-AUG-16
        24           100 A P 01-AUG-16

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        45           100 A A 02-AUG-16

23 rows selected.

SQL> exec stud_attendance_present('2-Aug-2016');

PL/SQL procedure successfully completed.

SQL> select * from stud_attendance_details;

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        13           100 A A 23-JUL-16
        14           100 A   23-JUL-16
        15           100 B A 23-JUL-16
        16           100 A A 23-JUL-16
        17           100 A A 23-JUL-16
        19           100 A   26-JUL-16
        20           100 A   26-JUL-16
        21             2 A A 26-JUL-16
        22             2 A A 26-JUL-16
        25             2 B A 26-JUL-16
        26             2 B A 26-JUL-16

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        45           100 A   05-AUG-16
        46           100 A   05-AUG-16
        47             2 A   05-AUG-16
        47             2 A A 01-AUG-16
        46           100 A A 01-AUG-16
         1               A P 01-AUG-16
         2           100 A P 01-AUG-16
         3           100 A P 01-AUG-16
        10           100 A P 01-AUG-16
        23           100 A P 01-AUG-16
        24           100 A P 01-AUG-16

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        45           100 A A 02-AUG-16
         1               A P 02-AUG-16
         2         100 A P 02-AUG-16
         3         100 A P 02-AUG-16
        10         100 A P 02-AUG-16
        13         100 A P 02-AUG-16
        14         100 A P 02-AUG-16
        15         100 B P 02-AUG-16
        16         100 A P 02-AUG-16
        17         100 A P 02-AUG-16
        19         100 A P 02-AUG-16

   STUD_ID STUD_STANDARD S S ATTENDANC
---------- ------------- - - ---------
        20           100 A P 02-AUG-16
        21             2 A P 02-AUG-16
        22             2 A P 02-AUG-16
        23           100 A P 02-AUG-16
        24           100 A P 02-AUG-16
        25             2 B P 02-AUG-16
        26             2 B P 02-AUG-16
        46           100 A P 02-AUG-16
        47             2 A P 02-AUG-16

42 rows selected.
```

Think about Merge on Submit Button Since Auditing is required only for update attendance.

➔ Implement Auditing that tracks updates on the attendance and Leave tables.

```
SQL> create table audit_stud_attendance_details(user_nm varchar2(20),table_nm
varchar2(30),stud_id number,old_val_stud_attend char,new_val_stud_attend char,systime timestamp);
Table created.

SQL> create or replace trigger trig_update_stud_attend
    after update of stud_attend on stud_attendance_details for each row
    begin
    insert into audit_stud_attendance_details
values(user,'stud_attendance_details',:new.stud_id,:old.stud_attend,:new.stud_attend,systimestamp);
    end;
    /

Trigger created.
```

```
SQL> create table audit_leave_details(user_nm varchar2(30),table_nm varchar2(30),stud_id
number,old_l_srt_date date,new_l_srt_date date,old_l_end_date date,new_l_end_date date,systimme
timestamp);

Table created.

SQL> create or replace trigger trig_update_stud_leave
 after update of leave_st_date,leave_end_date on stud_leave_details for each row
 begin
 insert into audit_leave_details
values(user,'stud_leave_details',:new.stud_id,:old.leave_st_date,:new.leave_st_date,:old.leave_end_date,:n
ew.leave_end_date,systimestamp);
 end;
 /
Trigger created.
```

- ➔ Write a stored procedure that purges all the old data(More than 2 years old) from attendance table to history tables.

```
SQL> create or replace procedure proc_old_attendance
   is
   begin
   insert into stud_attendance_details_bkp select * from stud_attendance_details where sysdate-
730>attendance_date;
   end;
   /

Procedure created.
```

- ➔ Generate a Report that displays the details of all the students that have taken more than N leaves for particular month.

```
SQL> select * from stud_attendance_details a, stud_leave_details l
   where a.stud_id=l.stud_id
   AND no_of_days>5
   AND to_char(attendance_date,'month')='july';
```

Module 5:- Staff Attendance & Leave Tracking

staff_Attendance_Details

Column Name	Datatype	Remarks
staff_id	Number	
staff_attend	Char	P/A
Attend_date	Date	
Staff_Entry_Time	Date	
Staff_Exit_Time	Date	

Emp_Leave_Master

Column Name	Datatype	Remarks
Leave_id	Number	
Leave_Type	Varchar2(10)	

Emp_Leave_allotment

Column Name	Datatype	Remarks
Emp_id	Number	
Leave_id	Number	
Leave_Balance	Number	
Total_Leaves	Number	

Emp_Leave_details

Column Name	Datatype	Remarks
Emp_id	Number	
Leave_id	Number	
Start_date	Date	
End_date	Date	
No_of_Days	Number	
Leave_App_Date	Date	
Approved_Flag	Char	Y/N

Module 6:- Exam & Result Management

Exam_Master

Column Name	Datatype	Remarks
Exam_id	Number	
Exam_type	Varchar2(100)	

Stud_Exam_Details

Column Name	Datatype	Remarks
Stud_id	Number	FK
Stud_course_id	Number	FK—same as standard_id
Stud_div	Char	
Stud_Sub_id	Number	FK
Stud_Sub_Marks	Number	
Sub_total_Marks	Number	
Exam_id	Number	UT1, UT2,SM1,SM2.
Exam_Year	Char(4)	2012
Exam_Date	Date	

➔ Implement an audit log that tracks updates on the table.

```
SQL>create table audit_exam_master(user_nm varchar2(20),table_nm varchar2(30),old_exam_id
number,new_exam_id number,old_exam_type varchar2(100),new_exam_type varchar2(100),systime
timestamp);

Table created.

SQL> create or replace trigger trig_update_exam_master
        after update of exam_id,exam_type on exam_master for each row
        begin
        insert into audit_exam_master
values(user,'exam_master',:old.exam_id,:new.exam_id,:old.exam_type,:new.exam_type,systimestamp)
;
        end;
/
Trigger created.

SQL> create table audit_stud_exam_details(user_nm varchar2(20),table_nm
varchar2(30),old_stud_sub_marks number,new_stud_sub_marks  number,old_sub_total_marks
number,new_sub_total_marks number,old_exam_id number,new_exam_id number,old_exam_year
char(4),new_exam_year char(4),old_exam_date date,new_exam_date date,systime timestamp);

Table created.
```

```
create or replace trigger trig_update_stud_exam_details
        after update of stud_sub_marks,sub_total_marks,exam_id,exam_year,exam_date on
stud_exam_details for each row
        begin
        insert into audit_stud_exam_details
values(user,'stud_exam_details',:old.stud_sub_marks,:new.stud_sub_marks,:old.sub_total_marks,:new.
sub_total_marks,:old.exam_id,:new.exam_id,:old.exam_year,:new.exam_year,:old.exam_date,:new.ex
am_date,systimestamp);
        end;
        /
Trigger created.
```

➔ Inserting the exam details

```
SQL> create or replace procedure exam_insert(
    stud_id_pr IN stud_exam_details.stud_id%type,
    stud_course_id_pr IN stud_exam_details.stud_course_id%type,
    stud_div_pr IN stud_exam_details.stud_div%type,
    stud_sub_marks_pr IN stud_exam_details.stud_sub_marks%type,
    sub_total_marks_pr IN stud_exam_details.sub_total_marks%type,
    exam_id_pr IN stud_exam_details.exam_id%type,
    exam_year_pr IN stud_exam_details.exam_year%type,
    exam_date_pr IN stud_exam_details.exam_date%type)
   IS
   BEGIN
   insert into stud_exam_details
values(stud_id_pr,stud_course_id_pr,stud_div_pr,stud_sub_marks_pr,sub_total_marks_pr,exam_id
_pr,exam_year_pr,exam_date_pr);
   End;
   /

Procedure created.
```

- ➔ Generate a Report that displays the Student Mark sheet ,it has the Percentage, Overall Rank Div wise and Rank Standard wise, also Subject wise Rank at the Standard/Div level. Basically insert data into a table T1 from where front end can select.

```
SQL> select
stud_id,stud_course_id,stud_sub_id,stud_div,stud_sub_marks,sub_total_marks,exam_id,exam_date,((stud
_sub_marks/sub_total_marks)*100) "Percentage" from stud_exam_details;

   STUD_ID STUD_COURSE_ID STUD_SUB_ID S STUD_SUB_MARKS SUB_TOTAL_MARKS
EXAM_ID EXAM_DATE Percentage
---------- -------------- ----------- - -------------- --------------- ---------- --------- ----------
        13            100           1 A             50             100         91 21-JAN-12         50
        14            100           1 A             40             100         92 21-JAN-12         40
        15            100           1 A             60             100         93 21-JAN-12         60
        16            100           1 B             55             100         94 21-JAN-12         55
        17            100           1 B             45             100         94 21-JAN-12         45

SQL> select rank(45) within group (order by stud_sub_marks desc) "Rank of student" from
stud_exam_details;

Rank of student
---------------
              4

SQL> select rank(45) within group (order by stud_sub_marks desc) "Rank of student" from
stud_exam_details where stud_div='B';

Rank of student
---------------
              2
```

Final :➔ This query will give standard wise and its division wise total report based on students total marks.

```
SQL> SELECT stud_course_id, stud_div, stud_id, stud_sub_marks,
      RANK() OVER (PARTITION BY stud_course_id,stud_div
      ORDER BY stud_sub_marks DESC ) "Rank"
      FROM stud_exam_details WHERE stud_course_id = 100;

STUD_COURSE_ID S    STUD_ID STUD_SUB_MARKS       Rank
-------------- - ---------- -------------- ----------
           100 A         15             60          1
           100 A         13             50          2
           100 A         14             40          3
           100 B         16             55          1
           100 B         17             45          2

SQL>  SELECT stud_course_id, stud_div, stud_id, stud_sub_marks,sub_total_marks,
          RANK() OVER (PARTITION BY stud_course_id,stud_div
          ORDER BY stud_sub_marks DESC ) "Rank",
    ((stud_sub_marks/sub_total_marks)*100) "Percentage"
          FROM stud_exam_details WHERE stud_course_id = 2;
```

```
STUD_COURSE_ID S    STUD_ID STUD_SUB_MARKS SUB_TOTAL_MARKS       Rank Percentage
-------------- - ---------- -------------- --------------- ---------- ----------
          100 A        15            60            100         1         60
          100 A        13            50            100         2         50
          100 A        14            40            100         3         40
          100 B        16            55            100         1         55
          100 B        17            45            100         2         45

SQL> SELECT stud_course_id, stud_div, stud_id, stud_sub_marks,sub_total_marks,
             RANK() OVER (PARTITION BY stud_course_id
             ORDER BY stud_sub_marks DESC ) "Rank_Standardwise",
          RANK() OVER (PARTITION BY stud_course_id,stud_div
             ORDER BY stud_sub_marks DESC ) "Rank_Divisionwise",
       ((stud_sub_marks/sub_total_marks)*100) "Percentage"
             FROM stud_exam_details order by stud_course_id;

STUD_COURSE_ID S    STUD_ID STUD_SUB_MARKS SUB_TOTAL_MARKS       Rank Percentage
-------------- - ---------- -------------- --------------- ---------- ----------
            2 B        22            80            100         1         80
            2 B        25            70            100         2         70
            2 A        20            65            100         3         65
            2 A        21            63            100         4         63
            2 A        19            35            100         5         35
          100 A        15            60            100         1         60
          100 B        16            55            100         2         55
          100 A        13            50            100         3         50
          100 B        17            45            100         4         45
          100 A        14            40            100         5         40

10 rows selected.

SELECT stud_course_id, stud_div, stud_id, stud_sub_marks,sub_total_marks,
             RANK() OVER (PARTITION BY stud_course_id
             ORDER BY stud_sub_marks DESC ) "Rank_Standardwise",
          RANK() OVER (PARTITION BY stud_course_id,stud_div
             ORDER BY stud_sub_marks DESC ) "Rank_Divisionwise",
       ((stud_sub_marks/sub_total_marks)*100) "Percentage"
             FROM stud_exam_details order by stud_course_id;
```

Final Output:

```
SQL> SELECT stud_course_id, stud_div, stud_id, stud_sub_marks,sub_total_marks,
                RANK() OVER (PARTITION BY stud_course_id
                ORDER BY stud_sub_marks DESC) "Rank_Standardwise",
   RANK() OVER (PARTITION BY stud_course_id,stud_div
  ORDER BY stud_sub_marks DESC) "Rank_Divisionwise",
          ((stud_sub_marks/sub_total_marks)*100) "Percentage"
                FROM stud_exam_details order by stud_course_id;
o/p:-
STUD_COURSE_ID S    STUD_ID STUD_SUB_MARKS SUB_TOTAL_MARKS Rank_Standardwise
Rank_Divisionwise Percentage
-------------- - ---------- -------------- --------------- ----------------- ----------------- ----------
           2 B        22            80            100              1              1        80
           2 B        25            70            100              2              2        70
           2 A        20            65            100              3              1        65
           2 A        21            63            100              4              2        63
           2 A        19            35            100              5              3        35
         100 A        15            60            100              1              1        60
         100 B        16            55            100              2              1        55
         100 A        13            50            100              3              2        50
         100 B        17            45            100              4              2        45
         100 A        14            40            100              5              3        40

10 rows selected.
```

➔ Generate a report that displays the Top3 students for each standard

```
SQL> Select * from (
  2  SELECT stud_course_id,stud_id,stud_div , stud_sub_marks,sub_total_marks,
                    RANK() OVER (
                   ORDER BY stud_sub_marks DESC)  as Rank_Standardwise,
             ((stud_sub_marks/sub_total_marks)*100) "Percentage"
                   FROM stud_exam_details ) where Rank_Standardwise<4;

STUD_COURSE_ID    STUD_ID S STUD_SUB_MARKS SUB_TOTAL_MARKS
RANK_STANDARDWISE Percentage
-------------- ---------- - -------------- --------------- ----------------- ----------
           2        22 B            80            100              1         80
           2        25 B            70            100              2         70
           2        20 A            65            100              3         65
```

Note :- Not able to get the top 3 students from another standard

➔ Purge the data from exam details table to the history tables.(2 years old data).

```
SQL> create table stud_exam_details_hist (stud_id number,
    constraint student_exam_hist_fk foreign key(stud_id) references student_tbl(stud_id),
    stud_course_id number,
    constraint course_exam_hist_fk foreign key(stud_course_id) references courses_tbl(course_id),
    stud_div char,
    stud_sub_id  number,
    constraint subjects_exam_hist_fk foreign key(stud_sub_id) references subjects_tbl(subject_id),
    stud_sub_marks number,
   stud_total_marks number,
   exam_id number,
  exam_year char(4),
  exam_date date);

Table created.

SQL>create or replace trigger trig_stud_exam_details_hist
  after insert on stud_exam_details for each row
  begin
  if sysdate-730>:new.exam_date then
  insert into stud_exam_details_hist
values(:new.stud_id,:new.stud_course_id,:new.stud_div,:new.stud_sub_id,:new.stud_sub_marks,:new.
sub_total_marks,:new.exam_id,:new.exam_year,:new.exam_date);
end if;
    end;
 /
Trigger created.
```

Module 7:- Student Fees payment & Balance Tracking

Stud_Fees_Details

Column Name	Datatype	Remarks
Stud_id	Number	FK
Fees_Paid	Number	
Fees_balance	Number	
Total_Fees	Number	Should be populated from the Course_fees table.
Payment_date	Date	
Due_date	Date	

- ➔ Whenever a new student is added or students are moved to the next standard the fees details should be automatically populated with the total fees from the course_fees table.

Answer: - Feature already solved in module 1 Q8 (Otherwise getting 2 entries for one insert in stud_fees_details)

```
SQL> create or replace trigger trig_new_fees
   after insert or update of stud_standard_id on student_tbl for each row
   declare
   x courses_tbl.course_fees%type;
   begin
   select course_fees into X from courses_tbl
        where course_id= :new.stud_standard_id;
   insert into stud_fees_details(total_fees) values(x);
   end;
  /

Trigger created.
```

➔ When a part/full payment is done, the balance should be automatically updated.

```
Create or replace procedure update_fees_pay(
Stud_id_pr IN stud_fees_details.stud_id%type,
Fees_paid_pr IN stud_fees_details.fees_paid%type,
Payment_date_pr IN stud_fees_details.payment_date%type)
Is
X number;
Y number;
Begin
Select fees_balance into x from stud_fees_details where stud_id=stud_id_pr;
Update stud_fees_details set fees_paid=fees_paid_pr, payment_date=payment_date_pr,
fees_balance= (X-fees_paid_pr)
Where stud_id=stud_id_pr;
End;
/

SQL> Create or replace trigger trig_fees_balance
   After update of fees_paid on stud_fees_details for each row
   Declare
   X number;
   Begin
   Select total_fees into x from stud_fees_details where stud_id=:new.stud_id;
   Update stud_fees_details set fees_balance=x-(:new.fees_paid);
   End;
   /

Trigger created.
```

➔ Implement auditing that tracks updates on this table.

```
SQL> create table audit_stud_fees_details(user_nm varchar2(20),table_nm varchar2(30),new_stud_id
number,old_fees_paid number,new_fees_paid number,old_payment_date date,new_payment_date
date,systime timestamp);

Table created.

SQL>create or replace trigger trig_update_stud_fees
        after update of stud_id,fees_paid,payment_date on stud_fees_details for each row
        begin
        insert into audit_stud_fees_details
values(user,'Stud_fees_details',:new.stud_id,:old.fees_paid,:new.fees_paid,:old.payment_date,:new.pa
yment_date ,systimestamp);
        end;
/
Trigger created
```

➔ Whenever payment is made after the due date, a penalty of Rs 500/- should be applicable.

```
 Create or replace procedure update_fees_pay(
Stud_id_pr IN stud_fees_details.stud_id%type,
Fees_paid_pr IN stud_fees_details.fees_paid%type,
Payment_date_pr IN stud_fees_details.payment_date%type)
Is
X number;
Y date;
Pay_date_exceed exception;
Begin
Select fees_balance into x from stud_fees_details where stud_id=stud_id_pr;
Select due_date into Y from stud_fees_details where stud_id=stud_id_pr;
If payment_date_pr>Y then
        Raise pay_date_exceed;
End if;
Update stud_fees_details set fees_paid=fees_paid_pr, payment_date=payment_date_pr,
fees_balance= (X-fees_paid_pr)
Where stud_id=stud_id_pr;
Exception when pay_date_exceed then
Update stud_fees_details set total_fees=total_fees+500 where stud_id=stud_id_pr;
Dbms_output.put_line('Payment Date Exceeded Penalty Rs. 500/-');
End;
/

Procedure created.

SQL> Create or replace trigger trig_fees_penalty
    After insert or update of fees_paid on stud_fees_details for each row
    Declare
    X date;
    Y date;
    Z number;
    Begin
    Select payment_date into X from stud_fees_details where stud_id=:new.stud_id;
    Select due_date into Y from stud_fees_details where stud_id=:new.stud_id;
   If(X>Y) then
     Select total_fees into Z from stud_fees_details where stud_id=:new.stud_id;
   Update stud_fees_details set total_fees=Z+500;
   End if;
   End;
   /

Trigger created.
```

➔ Purge the old data into the hist table and implement error log.(**2 years old data**)

```
SQL> create table stud_fees_details_hist(stud_id number,
       constraint student_fees_fk_hist foreign key(stud_id) references student_tbl(stud_id),
       fees_paid number,
       fees_balance number,
       total_fees number,
       payment_date date,
       due_date date);

Table created.

SQL> create or replace trigger trig_stud_fees_details_hist
       after insert on stud_fees_details for each row
       begin
       if sysdate-730>:new.payment_date
then
insert into stud_fees_details_hist
values(:new.stud_id,:new.fees_paid,:new.fees_balance,:new.total_fees,:new.payment_date,:new.paym
ent_date);
end if;
       end;
    /

Trigger created.
```

➔ Generate a report that displays a list of students where there is part payment and payment is made after due date.

```
SQL> Select * from stud_fees_details
  2  MINUS
  3  select * from stud_fees_details
  4  where fees_balance=0 and
  5  due_date<payment_date;

   STUD_ID  FEES_PAID FEES_BALANCE TOTAL_FEES PAYMENT_D DUE_DATE
---------- ---------- ------------ ---------- --------- ---------
         1      10000         2000     120000 01-JAN-14 02-FEB-14
```

Module 8:- EMP Payroll

staff_Salary_details

Column Name	Datatype	Remarks
staff_id	Number	
Gross_Sal	Number	Basic+HRA+DA+TA
Net_sal	Number	Gross –PF-TAX-Extra
Pf_deduct_amt	Number	
Tax_deduct_amt	Number	
Extra_deduct_amt	Number	
Sal_date	Date	

- ➔ Creating a stored function that calculates the Gross Salary = BASIC+ 0.40%Basic +0.15% Basic +0.10% Basic.
- ➔ Create a stored function to calculate the TAX using the below logic.

Income range	Tax Percent
0-2 Lacs	0
2-3 Lacs	10
3-5 Lacs	20
>=5 Lacs	30

- ➔ PF is calculated as 12.5 % of the Gross Salary.
- ➔ Extra Deduction is calculated for any extra leaves taken i.e. more than the allotted. Deduction amt per day= Gross sal/No of working days in Month.
- ➔ Writing a stored procedure that will call the above functions and will insert the data in the above table on monthly basis.