LIST OF EXPERIMENTS

1. Practice session: Students should be allowed to choose appropriate DBMS software, install it, configure it and start working on it. Create sample tables, execute some queries, use SQLPLUS features, use PL/SQL features like cursors on sample database. Students should be permitted to practice appropriate User interface creation tool and Report generation tool.

2. A college consists of number of employees working in different departments. In this context, create two tables' **employee** and **department**. Employee consists of columns empno, empname, basic, hra, da, deductions, gross, net, date-of-birth. The calculation of hra,da are as per the rules of the college. Initially only empno, empname, basic have valid values. Other values are to be computed and updated later. Department contains deptno, deptname, and description columns. Deptno is the primary key in department table and referential integrity constraint exists between employee and department tables. Perform the following operations on the database:

1. Create tables department and employee with required constraints.
2. Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command
3. Basic column should not be null
4. Add constraint that basic should not be less than 5000.
5. Calculate hra,da,gross and net by using PL/SQL program.
6. Whenever salary is updated and its value becomes less than 5000 a trigger has to be raised preventing the operation.
7. The assertions are: hra should not be less than 10% of basic and da should not be less than 50% of basic.
8. The percentage of hra and da are to be stored separately.
9. When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.
10. Empno should be unique and has to be generated automatically.
11. If the employee is going to retire in a particular month, automatically a message has to be generated.
12. The default value for date-of-birth is 1 jan, 1970.
13. When the employees called daily-wagers are to be added the constraint that salary should be greater than or equal to 5000 should be dropped.

14. Display the information of the employees and departments with description of the fields.
15. Display the average salary of all the departments.
16. Display the average salary department wise.
17. Display the maximum salary of each department and also all departments put together.
18. Commit the changes whenever required and rollback if necessary.
19. Use substitution variables to insert values repeatedly.
20. Assume some of the employees have given wrong information about date-of-birth. Update the corresponding tables to change the value.
21. Find the employees whose salary is between 5000 and 10000 but not exactly 7500.
22. Find the employees whose name contains 'en'.
23. Try to delete a particular deptno. What happens if there are employees in it and if there are no employees.
24. Create alias for columns and use them in queries.
25. List the employees according to ascending order of salary.
26. List the employees according to ascending order of salary in each department.
27. Use '&&' wherever necessary
28. Amount 6000 has to be deducted as CM relief fund in a particular month which has to be accepted as input from the user. Whenever the salary becomes negative it has to be maintained as 1000 and the deduction amount for those employees is reduced appropriately.
29. The retirement age is 60 years. Display the retirement day of all the employees.
30. If salary of all the employees is increased by 10% every year, what is the salary of all the employees at retirement time.
31. Find the employees who are born in leap year.
32. Find the employees who are born on feb 29.
33. Find the departments where the salary of at-least one employee is more than 20000.
34. Find the departments where the salary of all the employees is less than 20000.
35. On first January of every year a bonus of 10% has to be given to all the employees. The amount has to be deducted equally in the next 5 months. Write procedures for it.
36. As a designer identify the views that may have to be supported and create views.
37. As a designer identify the PL/SQL procedures necessary and create them using cursors.

38. Use appropriate Visual programming tools like oracle forms and reports, visual basic etc to create user interface screens and generate reports.

Note: As a designer identifies other operations that may be required and add to the above list. The above operations are not in order. Order them appropriately. Use SQL or PL/SQL depending on the requirement.

3. Students may be divided into batches and the following experiments may be given to them to better understand the DBMS concepts. Students should gather the required information, draw ER diagrams, map them to tables, normalize, create tables, triggers, procedures, execute queries, create user interfaces, and generate reports.

- Student information system
- APSRTC reservation system
- Hostel management
- Library management
- Indian Railways reservation
- Super market management
- Postal system
- Banking system
- Courier system
- Publishing house system

1. Practice session: Students should be allowed to choose appropriate DBMS software, install it, configure it and start working on it. Create sample tables, execute some queries, use SQLPLUS features, use PL/SQL features like cursors on sample database. Students should be permitted to practice appropriate User interface creation tool and Report generation tool.

oracle database 10g express edition installation

Welcome to Oracle Database 10g Express Edition (Oracle Database XE)! This tutorial gets you quickly up and running using Oracle Database XE by creating a simple application. This guide covers the following topics:

■Logging in as the Database Administrator
■Unlocking the Sample User Account
■Logging in as the Sample User Account
■Creating a Simple Application
■Running Your New Application
■Using the Oracle Database XE Menus

1.Logging in as the Database Administrator:

The first thing you need to do is to log in as the Oracle Database XE Administrator. Follow these steps:

1.Open the Database Home Page login window:

On Windows, from the Start menu, select **Programs**(or All Programs),then Oracle Database 10g Express Edition, and then **Go To Database Home Page**.

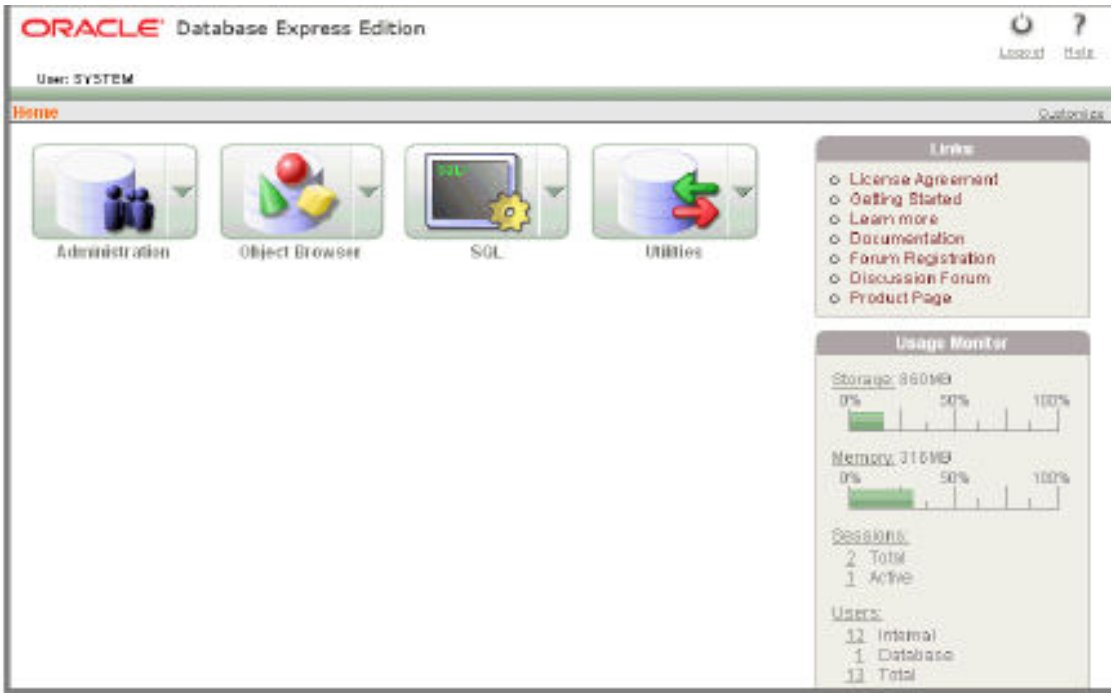
■On Linux, click the Application menu (on Gnome) or the K menu (on KDE), then point to Oracle Database 10g Express Edition, and then Go To Database Home Page.

2.At the Database Home Page login window, enter the following information:

■Username: Enter system for the user name.
■Password: Enter the password that was specified when Oracle Database XE was installed.

3.Click Login

The Oracle Database XE home page appears.



2.Unlocking the Sample User Account

To create your application, you need to log in as a database user. Oracle Database XE comes with a sample database user called HR. This user owns a number of database tables in a sample schema that can be used to create applications for a fictional Human Resources department. However, for security reasons, this user's account is locked. You need to unlock this account before you can build a sample application.

To unlock the sample user account:

1. Make sure you are still logged on as the database administrator, as described in the previous section.
2. Click the Administration icon, and then click Database Users.
3. Click the HR schema icon to display the user information for HR.

4. Under Manage Database User, enter the following settings:

■ Password and Confirm Password: Enter hr for the password.

■ Account Status: Select Unlocked.

■ Roles: Ensure that both CONNECT and RESOURCE are enabled.

5. Click Alter User.

Now you are ready to create your first application.

3. Logging in as the Sample User Account

To log in as the sample user account:

1. Log out from the database administrator account by clicking Logout in the upper right corner of the Database Home Page.

2. In the window, click Login.

3. In the Login window, enter hr for both the user name and password.

4. Click Login.

The Database Home Page appears.

4. Creating a Simple Application

Creating an application is an easy way to view and edit your database data. You create this application based on the EMPLOYEES table, which is part of the HR schema.

To create an application based on the EMPLOYEES table:

1. On the Database Home Page, click the Application Builder icon.

2. Click the Create button.

3. Under Create Application, select Create Application and click Next.

4. Under Create Application:

a. Name: Enter MyApp.

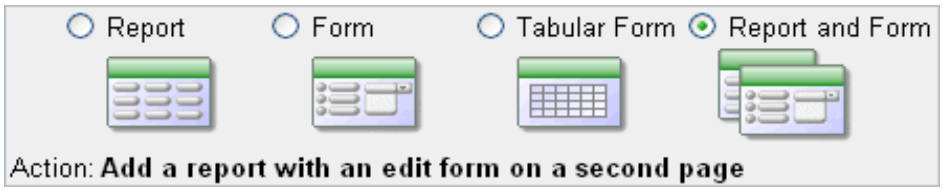
b. Accept the remaining defaults.

c. Click Next.

Next, add pages to your application.

5. Under Add Page:

a. For Select Page Type, select Report and Form.

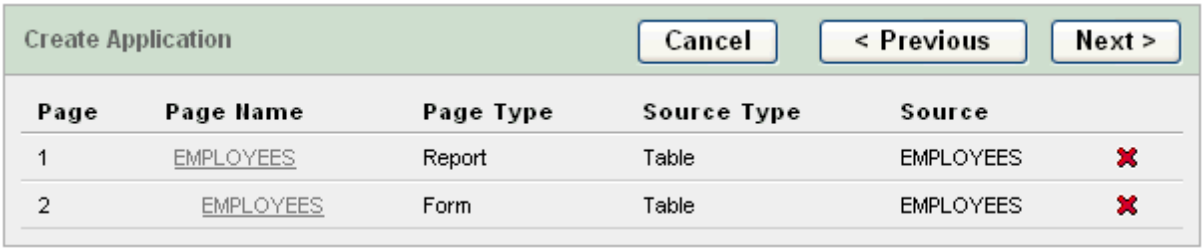


Notice that Action describes the type of page you are adding.

b. Next to the Table Name field, click the up arrow, and then select EMPLOYEES from the Search Dialog window.

c. Click Add Page.

Two new pages display at the top of the page, under Create Application.



d. Click Next

6. On the Tabs panel, accept the default (One Level of Tabs) and click Next.

7. On the Shared Components panel, accept the default (No) and click Next.

This option enables you to import shared components from another application. Shared components are common elements that can display or be applied on any page within an application.

8. For Authentication Scheme, Language, and User Language Preference Derived From, accept the defaults and click Next.

9. For the theme, select Theme 2 click Next. Themes are collections of templates that you can use to define the layout and style of an entire application.

10. Confirm your selections. To return to a previous wizard page, click Previous.

To accept your selections, click Create.

After you click Create, the following message displays at the top of the page:

Application created successfully.

5.Running Your New Application:

To run your application:

1. Click the Run Application icon.

In the log in page, enter hr for both the User Name and Password.

Your application appears, showing the EMPLOYEES table.

3. Explore your application.

You can query the EMPLOYEES table, if you want. To manage the application, use the Developer toolbar at the bottom on the page.

The Developer toolbar offers a quick way to edit the current page, create a new page, control, or component, view session state, or toggle debugging or edit links on and off.

4. To exit your application and return to Application Builder, click Edit Page 1 on the Developer toolbar.

5. To return to the Database Home Page, select the Home breadcrumb at the top of the page.

Congratulations! You have just created your first application using Oracle Database XE.

6.Using the Oracle Database XE Menus:

You can use the Oracle Database XE menus to perform basic functions with Oracle Database XE. To see the menus, do the following:

■ On Windows, from the Start menu, select Programs (or All Programs) and then Oracle Database 10g Express Edition.

■ On Linux, click the Application menu (on Gnome) or the K menu (on KDE) and then point to Oracle Database 10g Express Edition.

The following menu items are available:

■ Get Help: Displays the following selections:

- Go To Online Forum: Displays the online forum for discussions about Oracle Database XE.
- Read Documentation: Displays the Oracle Database XE documentation library on the Internet.
- Read Online Help: Displays the Oracle Database XE online help. This help is only available if the database is started.
- Register For Online Forum: Allows you to register for the Oracle Database XE online forum.

■ Backup Database: In NOARCHIVELOG mode (the default), shuts down the database, backs it up, and then restarts it. In ARCHIVELOG mode, performs an online backup of the database. For more information on backups, refer to Oracle Database Express Edition 2 Day DBA.

■ Get Started: Link to this tutorial.

■ Go To Database Home Page: Displays the Oracle Database XE Home Page in your default browser. "Logging in as the Database Administrator" on page 1 explains how to log into this home page as a database administrator.

■ Restore Database: Shuts down and then restores the database to the most recent backup. For more information on restoring a database, refer to Oracle Database Express Edition 2 Day DBA.

■ Run SQL Command Line: Starts the SQL Command Line utility for Oracle Database XE. To connect to the database, issue the following command at the **SQL prompt that appears**:

connect username/password

where username is the user name, such as sys, system, or another account name, and password is the password that was assigned when Oracle Database XE was installed. The get help, you can enter the command help at the SQL prompt, once you have connected to the database.

■ Start Database: Starts Oracle Database XE. By default, the database is started for you after installation and every time your computer is restarted. However, if you think the database is not running you can use this menu item to start it.

■ Stop Database: Stops Oracle Database XE.

CREATE SAMPLE TABLES, EXECUTE SOME SQL QUERIES

For create table:

1) Table name 2) Column name 3) Data type

Table restriction:

•Table name must be unique in database schema.

•The table name should begin with a letter and can be 1-30 characters long.

•Maximum 1000 columns

•Name can contain: A-Z, 0-9.

TO CREATE TABLE:

SYNTAX: create table<table name> (<column1><Data type>,<column2><Data type>,….);

SQL>create table emp(empno number(10),ename varchar2(10), job varchar2(10), mgr number(10),hiredate date,sal number(10),comm number(10), deptno number(10)) ;

Table created.

In sql*plus, we use **clscr** to clear screen.

DESCRIBE: Command will give us with what columns we created table and their data type.

SQL> DESC EMP

Name	Null?	Type
EMPNO		NUMBER (10)
ENAME		VARCHAR2 (10)
JOB		VARCHAR2 (10)
MGR		NUMBER (10)
HIREDATE		DATE
SAL		NUMBER (10)
COMM		NUMBER (10)
DEPTNO		NUMBER(10)

INSERTING VALUES IN TO TABLE:

SYNTAX: INSERT into<table name> [list of columns] values (list of values);

SQL> INSERT INTO EMP VALUES (100,'PAVAN','CHAIRMAN', NULL,'01-JAN-2005', 30000, 10000, 10);

1 Row Inserted.

INSERTING DATA INTO REQURIED COLUMNS:

SQL> INSERT INTO EMP (EMPNO, ENAME, JOB, DEPTNO) VALUES (100,'PAVAN','CHAIRMAN', 10);

SELECTING VALUES FROM A TABLE:

SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO FROM EMP;

HERE INSTEAD OF TYPING ALL THE COLUMN NAMES WE TYPE "*"

SQL> SELECT * FROM EMP;

SELECTING WITH WHERE CLAUSE:

Whenever we select using where clause we get particular information depends on the column you specify in where clause.

SQL> SELECT * FROM EMP WHERE EMPNO=100;

UPDATE STATEMENT:

While updating a table if you don't give where clause whole table will be updated.

SQL> UPDATE EMP SET SAL= SAL+1000;

UPDATE TABLE USING WHERE CLAUSE:

Whenever we give where clause in updating only that column corresponding rows will be updated.

SQL> UPDATE EMP SET SAL= SAL+1000 WHERE EMPNO=100;

DELETING DATA FROM A TABLE:

If you want to delete info from a table, here if you won't specify the where clause whole table info will be deleted.

SQL> DELETE FROM EMPLOYEE_INFO WHERE EMPNO=102;

2. **A college consists of number of employees working in different departments. In this context, create two tables' employee and department. Employee consists of columns empno, empname, basic, hra, da, deductions, gross, net, date-of-birth. The calculation of hra,da are as per the rules of the college. Initially only empno, empname, basic have valid values. Other values are to be computed and updated later. Department contains deptno, deptname, and description columns. Deptno is the primary key in department table and**

referential integrity constraint exists between employee and department tables. Perform the following operations on the database:

1. **Create tables department and employee with required constraints.**

   QUERY:

   create table emp (empno number(10) primary key, ename varchar2(10), job varchar2(10), mgr number(10),hiredate date, sal number(10),comm number(10),deptno number(10));

   EXPECTED OUTPUT:

   Table Created

   OUTPUT:

   QUERY:

   Create table dept(deptno number(10) primary key, dname varchar2(20), description varchar2(40));

   EXPECTED OUTPUT:

   Table Created

   OUTPUT:

2. **Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command**

   QUERY

   Alter table emp (add basic number(10),hra number(10),da number(10));

   EXPECTED OUTPUT:

   Table Altered

   OUTPUT:

3. **Basic column should not be null**

   QUERY:

Create table employee (empno number(10) not null, basic number(10) not null, ename varchar2(10));

Insert into employee values (null,null,'kumar');

Note: Above we kept not null for both empno, basic and we tried to insert null values in to both columns but it has not accepted.

EXPECTED OUTPUT:

Cannot insert null value in basic column(not null constraint violated)

OUTPUT:

4. **Add constraint that basic should not be less than 5000.**

QUERY:

alter table employee add check (basic>5000)

EXPECTED OUTPUT:

Table Altered

OUTPUT:

5. **Calculate hra,da, gross and net by using PL/SQL program.**

BASIC	HRA	DA
15000	12%	8%
12000	10%	6%
9000	7%	4%
OTHERS	5%	200/-

PL/SQL PROGRAM

```
DECLARE
BASIC NUMER
HRA NUMBER;
DA NUMBER;
NET NUMBER;
BEGIN
BASIC := &BASIC_SALARY;
```

```
IF BASIC > 15000 THEN
HRA := BASIC * .12;
DA := BASIC * .08;
ELSIF BASIC > 12000 THEN
HRA := BASIC * .1;
DA := BASIC * .06;
ELSIF BASIC > 9000 THEN
HRA := BASIC * ..07;
DA := BASIC * .04;
ELSE
HRA := BASIC * .05;
DA := BASIC * 200;
END IF;
NET := BASIC + HRA + DA;
DBMS_OUTPUT.PUT_LINE (‘BASIC: ‘ || BASIC);
DBMS_OUTPUT.PUT_LINE (‘HRA: ‘ || HRA);
DBMS_OUTPUT.PUT_LINE (‘DA: ‘ || DA);
DBMS_OUTPUT.PUT_LINE (‘NET: ‘ || NET);
END;
```

EXPECTED OUTPUT:

pl/sql procedure successfully completed

OUTPUT:

6. **Whenever salary is updated and its value becomes less than 5000 a trigger has to be raised preventing the operation.**

PL/SQL PROGRAM

```
Create or replace trigger sal
Before insert or update on emp11
for each row
begin
if :new.sal<5000 then
 raise_application_error(-20456,'the 5000 below sal not accepted');
end if;
 end;
```

EXPECTED OUTPUT:

pl/sql procedure successfully completed

OUTPUT:

7. **The assertions are: hra should not be less than 10% of basic and da should not be less than 50% of basic.**

QUERY:

```
Alter table employee add check (hra>sal*0.10);
Alter table employee add check (da>sal*0.45);
```

EXPECTED OUTPUT:

Table altered

OUTPUT:

8. **When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.**

PL/SQL PROGRAM

```
Create or replace trigger da1
Before insert or update on employee
for each row
begin
if :new.da>sal then
 raise_application_error (-20456,'the da is not accepted, so the da is merge to basic');
end if;
 end;
```

EXPECTED OUTPUT:

Trigger Created Successfully.

OUTPUT:

9. **Empno should be unique and has to be generated automatically.**

QUERY:

```
Create or replace sequence s1
Increment by 1
Start with 501
End with 590
No cache
No cycle;
Insert into employee values (s1.nextval,'siva', 9000, 900, 4500);
Insert into employee values (s1.nextval,'siva', 9000, 900, 4500);
```

EXPECTED OUTPUT:

Sequence created.

OUTPUT:

10. **If the employee is going to retire in a particular month, automatically a message has to be generated.**

PL/SQL PROGRAM

```
DECLARE
 v_dob DATE:='&v_dob';
 v_your_age NUMBER(3,1);
```

```
Emp_name varchar2(10);
BEGIN
v_your_age:= TRUNC (MONTHS_BETWEEN(SYSDATE, v_dob))/12;
DBMS_OUTPUT.PUT_LINE ('Your age is ' || v_your_age);
IF v_your_age>65
THEN
DBMS_OUTPUT.PUT_LINE('YOU ARE GOING TO RETAIRE IN THIS MONTH');
END IF;
END;
```

EXPECTED OUTPUT:

pl/sql procedure successfully completed

OUTPUT:

11. **The default value for date-of-birth is 1 jan, 1970.**

QUERY:

Alter table employee modify date_of_birth DEFAULT '01-jan-70';

EXPECTED OUTPUT:

Table Altered

OUTPUT:

12. **When the employees called daily-wagers are to be added the constraint that salary should be greater than or equal to 5000 should be dropped.**

QUERY:

Alter table employee Add daily wagers number (10);

Alter table employee add check (daily wagers>=5000);

EXPECTED OUTPUT:

Table Altered

OUTPUT:

13. Display the information of the employees and departments with description of the fields.

QUERY:

```
desc emp;
desc dept;
```

EXPECTED OUTPUT:

name	null?	type
empno		number(10)
ename		varchar2(10)
job		varchar2(10)
mgr		number(10)
hiredate		date
sal		number(10)
comm		number(10)
deptno		number(10)

OUTPUT:

14. Display the average salary of all the departments.

QUERY:

Select avg(sal) from emp;

EXPECTED OUTPUT:

Sal

2500

OUTPUT:

15. Display the average salary department wise.

QUERY:

Select deptno,avg(sal) from emp group by deptno;

EXPECTED OUTPUT:

DEPT	AVGSALARY
cse	200
eee	300
ece	600

OUTPUT:

16. Display the maximum salary of each department and also all departments put together.

QUERY:

Select deptno,max(sal) from emp
group by deptno;

EXPECTED OUTPUT:

DEPT	MAXSALARY
cse	300
eee	500
ece	600

OUTPUT:

17. **Commit the changes whenever required and rollback if necessary.**

QUERY:

select * from emp;

EXPECTED OUTPUT:

empno	ename	job	mgr	hiredate	sal	comm	deptno
100	Pavan	Chairman	10	01-Jan-05	32000	10000	10
101	Gayatri	P.A	100	01-Jan-05	12000	1000	10

Delete from emp;
EXPECTED OUTPUT:
2 rows deleted.

Select * from emp;
EXPECTED OUTPUT:
No rows selected

roll back;
EXPECTED OUTPUT:
Rollback complete.

select * from emp;
EXPECTED OUTPUT:

empno	ename	job	mgr	hiredate	sal	comm	deptno
100	Pavan	Chairman	10	01-Jan-05	32000	10000	10
101	Gayatri	P.A	100	01-Jan-05	12000	1000	10

delete from emp;
EXPECTED OUTPUT:
3 rows deleted.

commit;
EXPECTED OUTPUT:
Commit complete.

roll back;
EXPECTED OUTPUT:
Rollback complete.

select * from emp;
EXPECTED OUTPUT:
No rows selected

18. Use substitution variables to insert values repeatedly.

QUERY:

CREATE TABLE departments (department_id number(10) not null, department_name varchar2(50) not null);

INSERT INTO department VALUES(&department_id,'&department_name');

EXPECTED OUTPUT:

1 row inserted

OUTPUT:

19. Assume some of the employees have given wrong information about date-of-birth. Update the corresponding tables to change the value.

QUERY:

Update emp set date-of-birth='04-jan-14' Where empno =1001;

EXPECTED OUTPUT:

Table Updated

OUTPUT:

20. **Find the employees whose salary is between 5000 and 10000 but not exactly 7500**.

QUERY:

select empno,sal from emp where sal between 5000 and 10000 and not sal=7500;

EXPECTED OUTPUT:

Sal

7000

OUTPUT:

21. Find the employees whose name contains 'en'.

QUERY:

Select empno,ename from emp Where ename like '%en%';

EXPECTED OUTPUT:

Ename
----
meena

OUTPUT:

22. Try to delete a particular deptno. What happens if there are employees in it and if there are no employees.

QUERY:

a) delete from emp where deptno=10;

EXPECTED OUTPUT:

1 row deleted.

OUTPUT:

b) delete from emp where deptno=10;

EXPECTED OUTPUT:

0 rows deleted.

OUTPUT:

23. Create alias for columns and use them in queries.

QUERY:

select empno **as** emplyeenumber,ename employeename from emp;

EXPECTED OUTPUT:

emplyeenumber	employeename
100	PAVAN
100	PAVAN

OUTPUT:

24. List the employees according to ascending order of salary.

QUERY:

a)Select * from emp
Order by sal;

b)Select * from emp
Order by sal asc;

EXPECTED OUTPUT:

Eno	ename	sal
123	xyz	1000
124	abc	1200

OUTPUT:

25. List the employees according to ascending order of salary in each department.

QUERY:

select empno,sal,deptno from emp12 order by deptno,sal;

EXPECTED OUTPUT:

Empno sal deptno

---

123 1000 10

124 1200 11

OUTPUT:

26. Use '&&' wherever necessary

QUERY:

Insert into emp values(&empno,'&ename'**,&&sal**,&basic,&hra, &da,&pf,&gross,' &date');

Note: The sal column using **double amp cent(&&)** so the data inserting time sal is asked one time only for first row.the remining columns automatically is taken the same data.

EXPECTED OUTPUT:

1 row inserted

OUTPUT:

27. Amount 6000 has to be deducted as CM relief fund in a particular month which has to be accepted as input from the user. Whenever the salary becomes negative it has to be maintained as 1000 and the deduction amount for those employees is reduced appropriately.

PL/SQL program

```
Create table kcb_acc_tab (accno number(10),acctype varchar2(20),bal number(10));
```

Table created.

```
create table kcb_tran_tab(sno number(10),accno number(10),ttype varchar2(20), date_of_tran
date, amt number(10));
```

Table created.

```
create sequence s1
 increment by 1
start with 1
Maxvalue 100
 Nocycle
 Nocache;
```

Sequence created.

```
Create or replace procedure upd_bal (paccno in kcb_acc_tab.accno%type,pttype in
kcb_tran_tab.ttype%type,pamt in kab_tran_tab.amt%type)
Is
cbal kcb_acc_tab.bal%type;
 vacctype kcb_acc_tab.acctype%type;
Begin
Select acctype,bal into vacctype,cbal from kcb_acc_tab where accno=paccno;
If pttype='D' then
cbal:=cbal+pamt;
Else if pttype='w' then
cbal:=cbal-pamt;
End if;
If  vacctype ='S' and cbal<1000 then
Raise_application_error(-20345,'the bal is too low. so no transaction');
Else if vacctype ='C' and cbal<1000 then
```

Raise_application_error(-20346,'the bal is too low. so no transaction');

End if;

Update kcb_acc_tab set bal=cbal where accno=paccno;

Insert into kcb_tran_tab values(s1.nextval,paccno,'D',sysdate,pamt);

Commit;

exception

WHEN NO_DATA_FOUND THEN

Dbms_output.put_line(paccno‖ 'is not exist');

End upd_bal ;

EXPECTED OUTPUT:

Exec(1001,D,6000);

OUTPUT

28. The retirement age is 60 years. Display the retirement day of all the employees.

Note: In this I am considering 360 as month. So 360/12=30 years (it is retirement age)

QUERY:

Select to char (add months (date_of_birth, 720),'dd-mon-yyyy') "next month" from emp;

EXPECTED OUTPUT:

Next month

20-dec-2042
21-dec-2042
22-dec-2042

select * from emp;

EMPNO	ENAME	DATE_OF_BIRTH
1001	Anil	20-Dec-72
1002	Siva	21-Dec-72
1003	Gowtham	22-Dec-72

OUTPUT:

29. If salary of all the employees is increased by 10% every year, what is the salary of all the employees at retirement time.

QUERY:

select months_between ('19-oct-2012','03-jul-1988')/12-60 "retirement year" from dual;

EXPECTED OUTPUT:

RETIREMENT YEAR

-35.706989

Select 20000+ ((20000*0.10)*35) from dual

OUTPUT:

30. Find the employees who are born in leap year.

QUERY:

select mod(2016,4) from dual;

Note: MOD(2016,4) u get the answer is 0 it is leap year but u r not get the answer <>0 not leap year

EXPECTED OUTPUT:

Mod

0

OUTPUT:

31. Find the employees who are born on feb 29.

QUERY:

Select * from emp Where date_of_birth like '%feb-29%';

EXPECTED OUTPUT:

Empno ename dob
---
123 xxxx 29-FEB-1994

OUTPUT:

32. Find the departments where the salary of at-least one employee is more than 20000.

QUERY:

Select deptno from emp where sal>20000;

EXPECTED OUTPUT:

Deptno

10
12

OUTPUT:

33. Find the departments where the salary of all the employees is less than 20000.

QUERY:

Select deptno from emp where sal<20000;

EXPECTED OUTPUT:

Deptno

15
17

OUTPUT:

34. As a designer identify the views that may have to be supported and create views.

QUERY:

create view emvup10 **as** select * from emp where deptno=10

EXPECTED OUTPUT:

View created

select * from emvup10;

empno ename
----
123 xxxx
124 abcd

insert into empvu10 values(126,'xxxx');

select * from empvu10;

empno ename
----
123 xxxx
124 abcd
126 xxxx

OUTPUT:

35. As a designer identify the PL/SQL procedures necessary and create them using cursors.

PL/SQL program

DECLARE

```
CURSOR C1 IS SELECT EMPNO,ENAME ,DEPTNO FROM EMP;
EMPNUM EMP.EMPNO%TYPE;
EMPNAME EMP.ENAME%TYPE;
DEPTNUM EMP.DEPTNO%TYPE;
BEGIN
OPEN C1;
LOOP
FETCH C1 INTO EMPNUM,EMPNAME,DEPTNUM;
if c1%notfound then
exit;
else
dbms_output.put_line(EMPNUM||' '||EMPNAME||' '||DEPTNUM);
end if;
END LOOP;
end;
```

EXPECTED OUTPUT:

PL/SQL Procedure successfully Created

OUTPUT:

3. Students may be divided into batches and the following experiments may be given to them to better understand the DBMS concepts. Students should gather the required information, draw ER diagrams, map them to tables, normalize, create tables, triggers, procedures, execute queries, create user interfaces, and generate reports.

- Student information system
- APSRTC reservation system
- Hostel management
- Library management
- Indian Railways reservation
- Super market management

- Postal system
- Banking system
- Courier system
- Publishing house system

SAMPLE EXAMPLES

LIBRARY MANAGEMENT SYSTEM

The Language Management System implements Oracle as the Backend and thus the database schemas defined and modified through Oracle SQL. The Language Management System consists of two tables to store all the information . The tables are Books and Patron. The schemas are as follows.

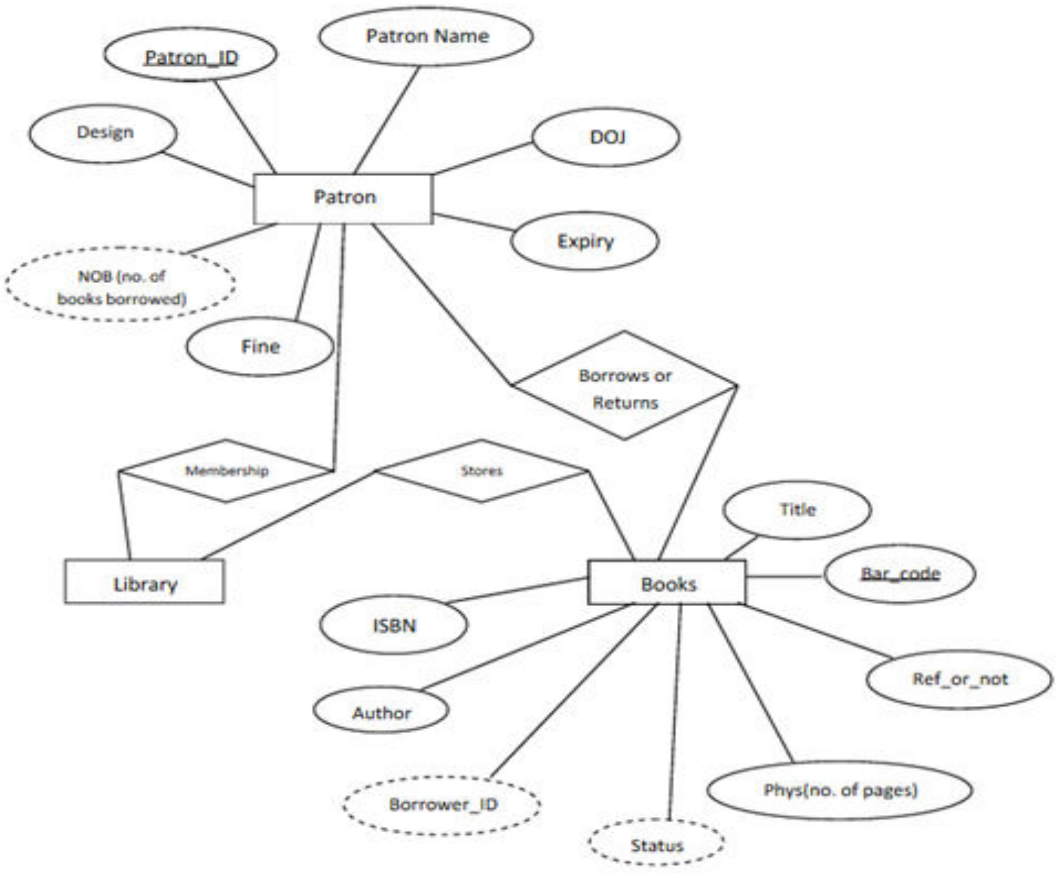
PATRON TABLE:

Name	Null?	Type
Patron_id	Not null	NUMBER(10)
Name		VARCHAR2(20)
Design		VARCHAR2(20)
DOJ		DATE
Expiry		DATE
NOB		NUMBER(1)
Fine		NUMBER(4)

BOOK TABLE:

Name	Null?	Type
BAR_CODE		NUMBER(6)
TITLE		VARCHAR2(20)
AUTHOR		VARCHAR2(20)
ISBN		NUMBER(10)
STATUS		VARCHAR2(10)
REF_OR_NOT		VARCHAR2(10)
BORROWER_ID		NUMBER(10)
PHYS		NUMBER(4)

ER DIAGRAM:



SYNTAX FOR CREATING A TABLE:

Create table tablename(columnname1 datatype,columnname2 datatype,……..);

CREATING A TABLE PATRON:

Create table patron(patron_id number(10) primary key,name varchar2(20),design varchar2(20),DOJ date,expiry date,NOB number(1),fine number(4));

CREATING A TABLE BOOKS:

SQL>Create table books(BAR_CODE number(6),title varchar2(20),author varchar2(20),ISBN number(10),status varchar2(10),REF_OR_NOT varchar2(10),borrower_id number(10),phys number(4));

SYNTAX FOR INSERT:

Insert into tablename values(values list);

INSERTING VALUES IN PATRON:

SQL>Insert into patron values(&patron_id,'&name','&design',&DOJ,&expiry,&NOB,&fine);

Values are inserted.

APSRTC RESERVATION SYSTEM

Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

The Following are the entities and its attributes

Bus:

Bus_No: varchar (10) **(primary key)**

Source: varchar (20)

Destination: varchar (20)

Passenger:

PNR_No : Number(9) **(primary key)**

Ticket_No : Number(9)

Name : varchar(15)

Age : integer(4)

Sex : char(10) ; Male/Female

PPNO : varchar(15)

Reservation:

PNR_No : number(9) **(foreign key)**

Journey_date : date

No_of_seats : integer(8)

Address : varchar(50)

Contact_No : Number(9)

Status : Char(2)

Cancellation :

PNR_No : number(9)**(foreign key)**

Journey_date : date

No_of_seats : integer(8)

Address : varchar(50)

Contact_No : Number(9)

Status : Char(2)

Ticket:

Ticket_No : number(9)**(primary key)**

Journey_date : date

Age : int(4)

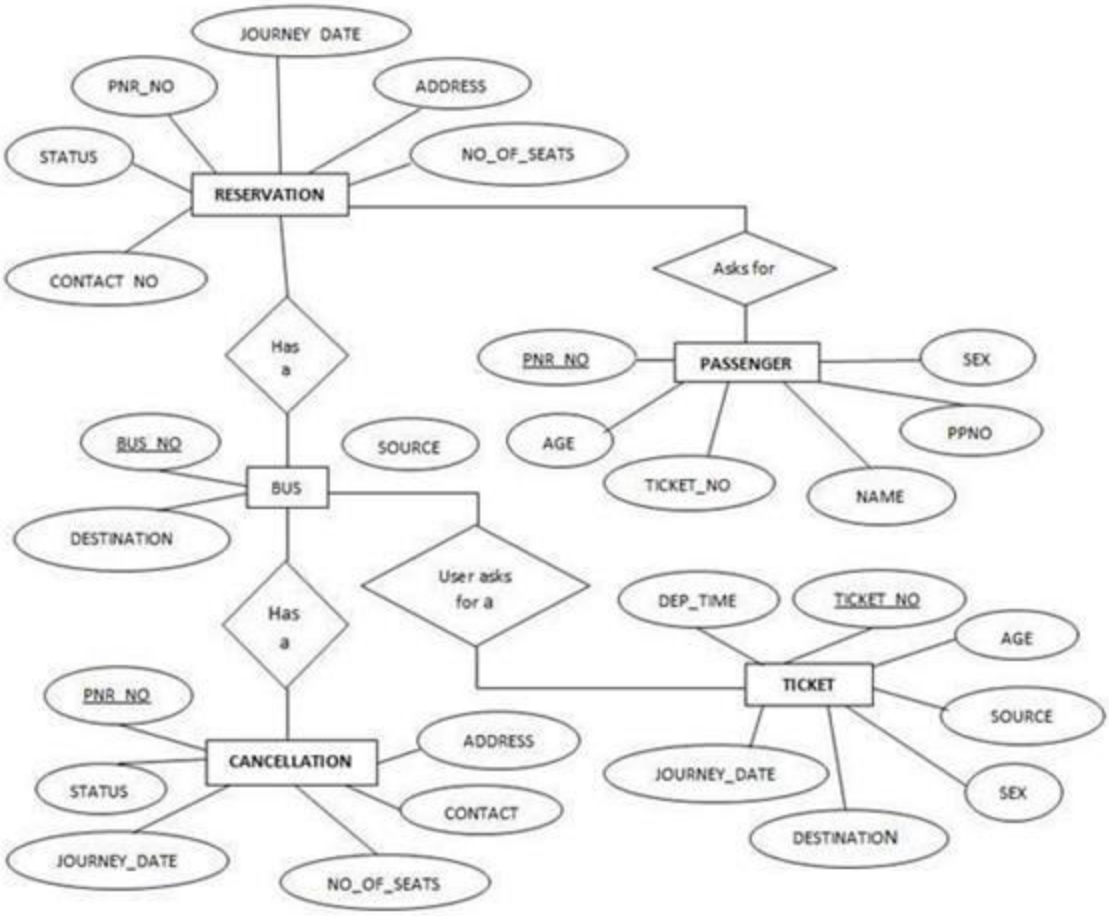
Sex : Char(10)

Source : varchar

Destination :varchar

Dep_time : varchar

E-R Diagram


JOURNEY DATE
PNR_NO
ADDRESS
STATUS
NO_OF_SEATS
RESERVATION
CONTACT NO
Asks for
Has a
PNR_NO
PASSENGER
SEX
AGE
PPNO
TICKET_NO
NAME
BUS_NO
SOURCE
BUS
DESTINATION
User asks for a
Has a
DEP_TIME
TICKET_NO
AGE
TICKET
PNR_NO
SOURCE
CANCELLATION
ADDRESS
STATUS
CONTACT
JOURNEY_DATE
SEX
JOURNEY_DATE
DESTINATION
NO_OF_SEATS