**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**FACULTY OF TECHNOLOGY & ENGINEERING**

U & P U. Patel Department of Computer Engineering

**Subject Name:** Database Management System **Semester :** IVth

**Subject Code:** CE246 **Academic Year :** Dec-May 2018

Practical List

**Instructions:**

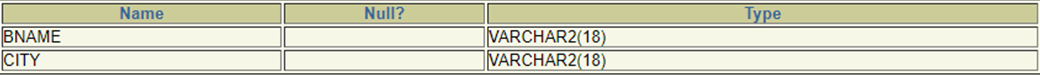
ISO Practical Format: Aim, Software/Hardware Required, Knowledge Required, Theory, Algorithm/Flow chart, Program, Input and Output, Questions and Answers / Case Study.

Note: Following Practical(s) are to be implemented on Oracle, DB2, Microsoft, NoSql and MongoDB

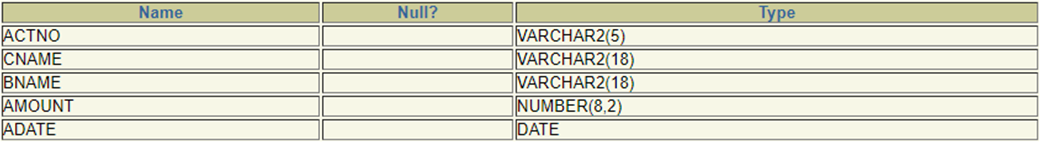
PRACTICAL 1: **To study DDL-create and DML-insert commands**.

(1) Describe deposit, branch.

describe branch16ce068;

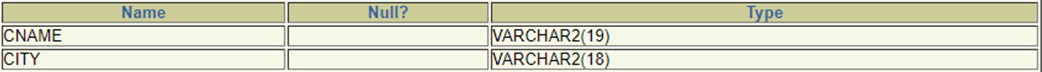


describe deposit16ce068

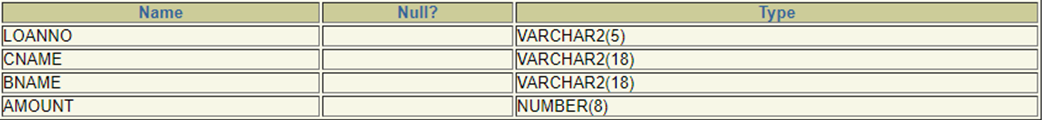


(2) Describe borrow, customers.

describe customers16ce068;

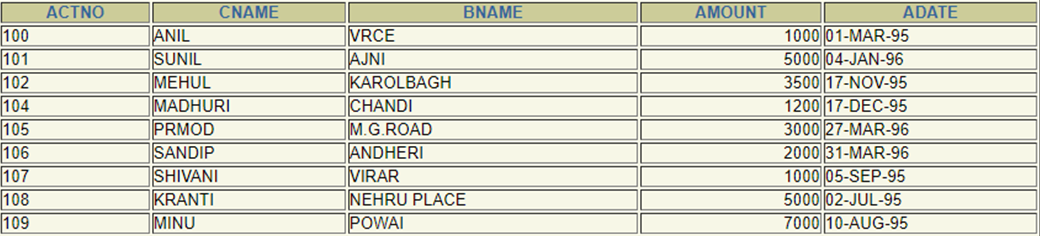


describe borrow16ce068;



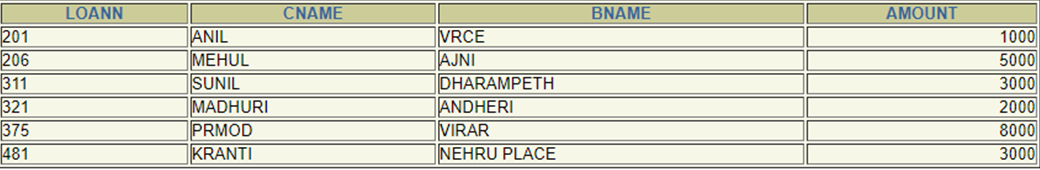
(3) List all data from table DEPOSIT.

Select \* from deposit16ce068



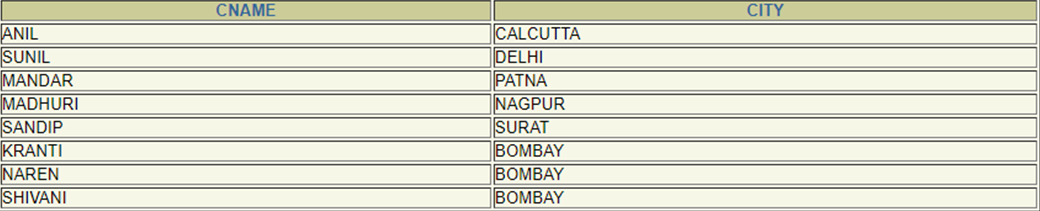
(4) List all data from table BORROW.

Select \* from borrow16ce068



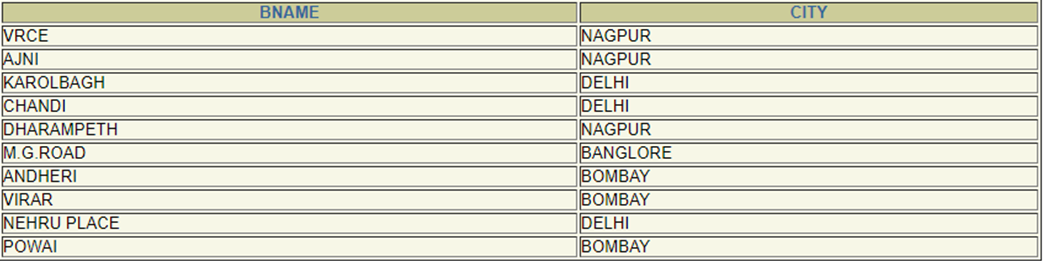
(5) List all data from table CUSTOMERS.

Select \* from customers16ce068



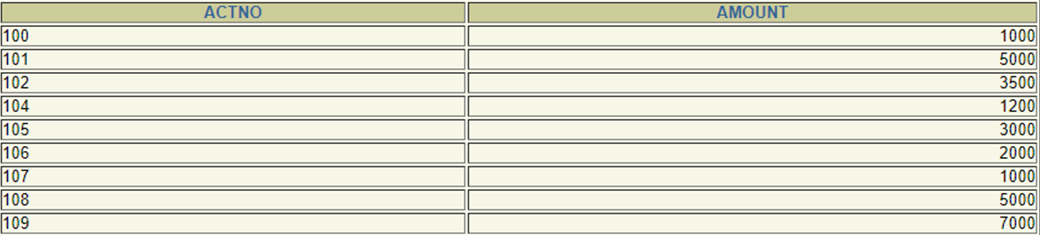
(6) List all data from table BRANCH.

Select \* from branch16ce068



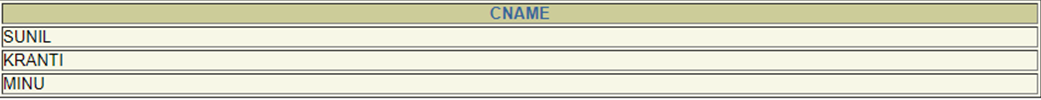
(7) Give account no and amount of depositors.

Select actno,amount from deposit16ce068



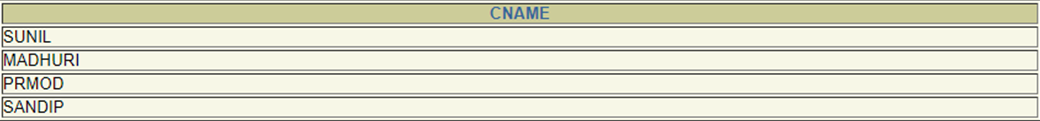
(8) Give name of depositors having amount greater than 4000.

select CNAME from deposit16ce068 where (amount>4000);



(9) Give name of customers who opened account after date '1-12-96'.

Select cname from deposit16ce068 where adate>’1-DEC-96’;



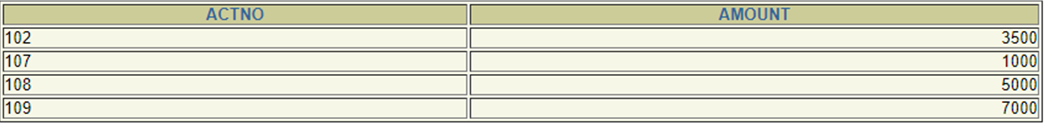
(10) Give name of city where branch karolbagh is located.

Select city from branch where bname=karolbagh;



(11) Give account no and amount of customer having account opened between date 1-12-96 and 1-6-96.

Select actno,amount from deposit16ce068 where adate<’1-DEC-95’ and adatw >’1-JUN-95’;



(12) Give names of depositors having account at VRCE.

Select cname from deposit16ce068 where bname=’VRCE’;

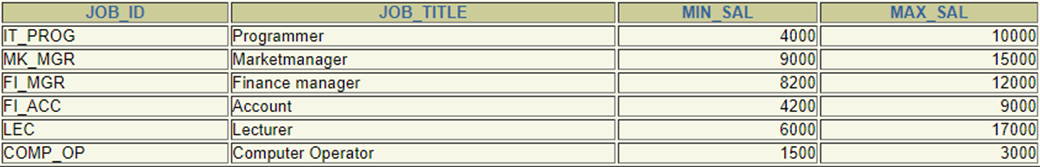


Conclusion: we learnt data insertion in sql language and retrival of it. Also seen different data types in sql.

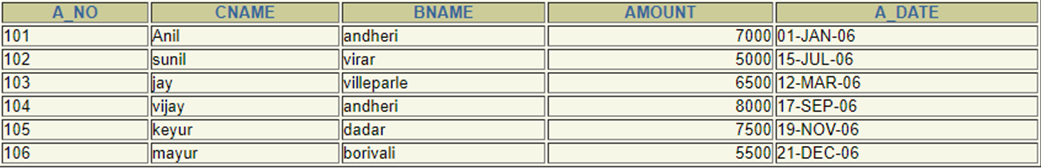
PRACTICAL 2:

(1) Retrieve all data from **employee, jobs and deposit.**

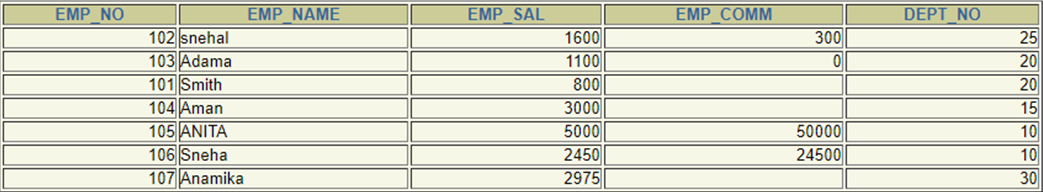
select \* from job;



select \* from deposit;

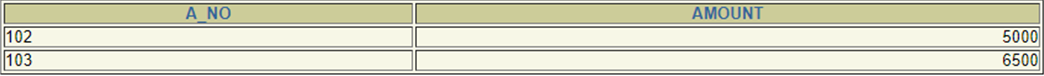


select \* from employee;



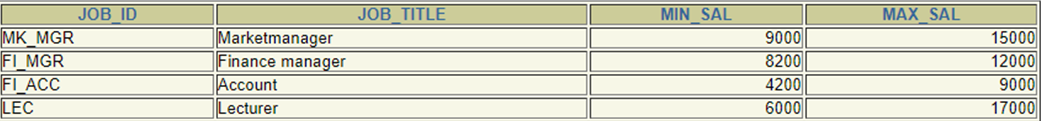
(2) Give details of account no. and deposited rupees of customers having account opened between dates **01-01-06 and 25-07-06**.

select a\_no,amount from deposit where a\_date>'2006-01-01' AND a\_date<'2006-07-25';



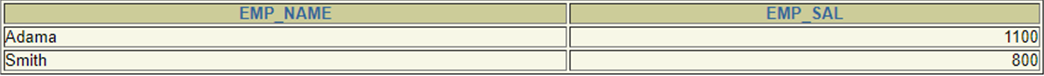
(3) Display all jobs with minimum salary is greater than 4000.

Select job\_title from job where min\_sal>4000;



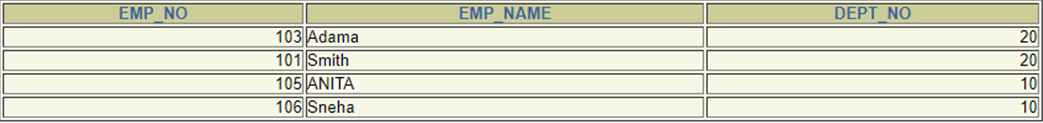
(4)Display name and salary of employee whose department no is 20. Give alias name to name of employee.

select emp\_name ename,emp\_sal from employee where dept\_no=20;



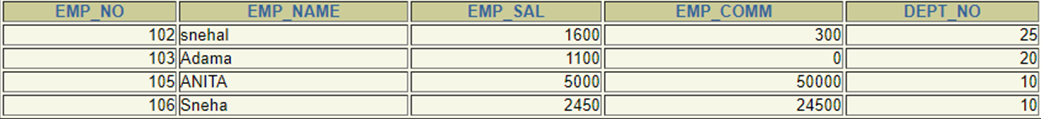
(5)Display employee no,name and department details of those employee whose department lies **in(10,20).**

select emp\_no,emp\_name,dept\_no from employee where dept\_no in(10,20);



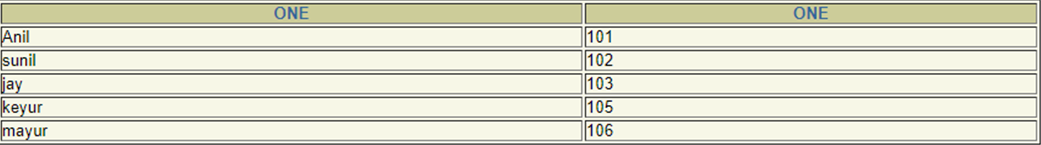
(6)Display the **non-null** values of employees.

select \* from employee where emp\_comm!='null';



(7)Display name of customer along with its account no**( both column should be** **displayed as one )** whose amount is not equal to 8000 Rs.

select a\_no as one,cname as one from deposit where amount!='8000';



(8)Display the content of job details with minimum salary either 2000 or 4000.

select job\_id,job\_title from job where min\_sal='2000' OR min\_sal='4000';

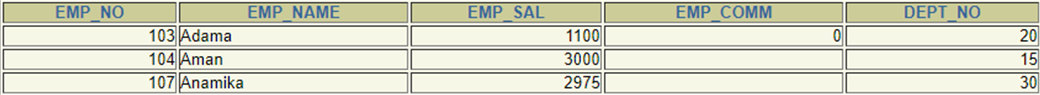


Conclusion: we have learnt ‘where’ keyword and,or,null,!=,=.

**To study various options of LIKE predicate**

(1) Display all employee whose name start with ‘A’ and third character is ‘ ‘a’.

select emp\_name from employee where emp\_name like 'A\_a%';



(2) Display name, number and salary of those employees whose name is 5 characters long and first three characters are ‘Ani’.

select emp\_no,emp\_name,emp\_sal from employee where emp\_name like 'ANI\_\_';



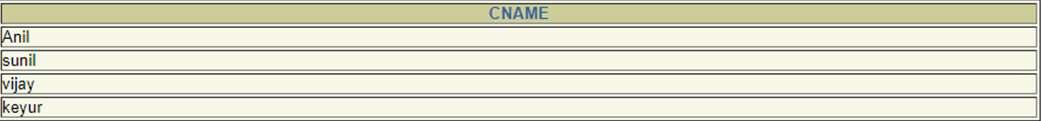
(3) Display all information of employee whose second character of name is either ‘M’ or ‘N’.

select \* from employee where emp\_name like '\_M%' OR emp\_name like '\_N%';



(4) Find the list of all customer name whose branch is in ‘andheri’ or ‘dadar’ or ‘virar’

select cname from deposit where bname='andheri' or bname='dadar' or bname='virar';



(5) Display the job name whose first three character in job id field is ‘FI\_’.

select job\_title from job where job\_id like 'FI\_%';



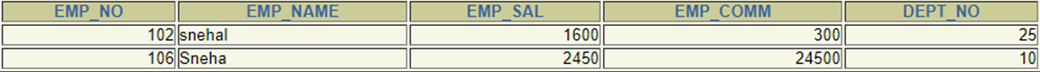
(6) Display the title/name of job whose last three character are ‘\_MGR’ and there maximum salary is greater than Rs 12000.

select job\_title from job where job\_id like '%MGR' AND max\_sal>12000;



(7) Display the non-null values of employees and also employee name second character should be ‘n’ and string should be 5 character long.

select \* from employee where emp\_comm!='null' AND emp\_name like '\_n\_\_\_';



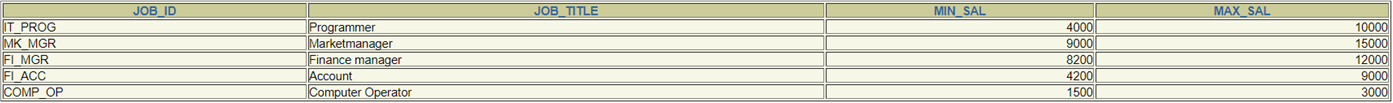
(8) Display the null values of employee and also employee name’s third character should be ‘a’.

select \* from employee where emp\_comm is NULL AND emp\_name like '\_\_a%';



(9) What will be output if you are giving LIKE predicate as ‘%\\_%’ ESCAPE ‘\’

select \* from job where job\_id like '%\\_%' ESCAPE '\' ;



Conclusion: we have learnt like predicate,%,\_,ESCAPE.

PRACTICAL 3

Aim: **To Perform various data manipulation commands, aggregate functions and sorting concept on all created tables.**

(1) List total deposit from deposit.

select sum(amount) from deposit;



(2) List total loan from karolbagh branch

select sum(amount) from branch,borrow where branch.bname='karolbaugh';



(3) Give maximum loan from branch vrce.

select max(amount) from borrow where bname='VRCE';



(4) Count total number of customers

select count(cname) from customers;



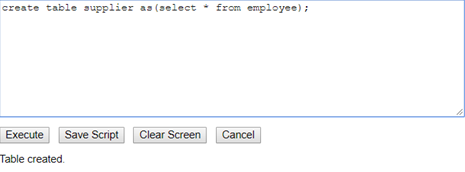
(5) Count total number of customer’s cities.

select count(city) from customer;



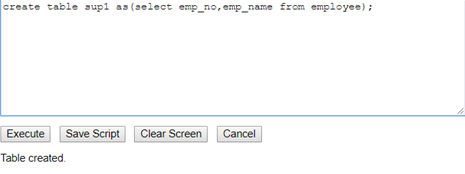
(6) Create table supplier from employee with all the columns.

create table supplier as(select \* from employee);



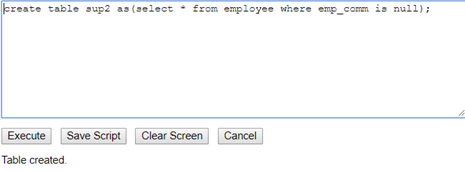
(7) Create table sup1 from employee with first two columns.

create table sup1 as(select emp\_no,emp\_name from employee);



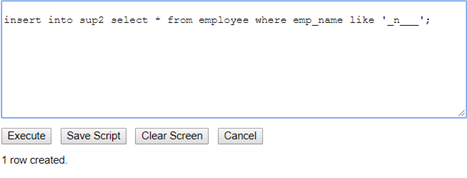
(8) Create table sup2 from employee with no data

create table sup2 as(select \* from employee where emp\_comm is null);



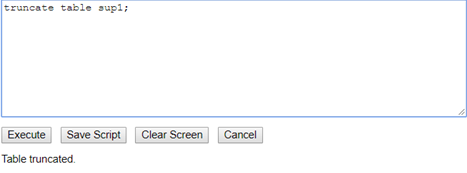
(9) Insert the data into sup2 from employee whose second character should be ‘n’ and string should be 5 characters long in employee name field.

insert into sup2 select \* from employee where emp\_name like '\_n\_\_\_';



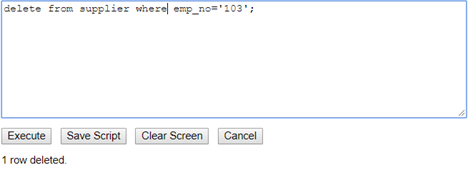
(10) Delete all the rows from sup1.

truncate table sup1;



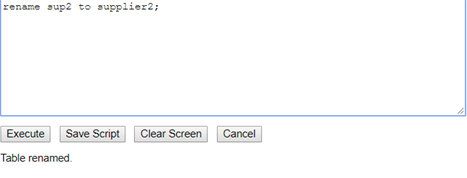
(11) Delete the detail of supplier whose sup\_no is 103.

delete from supplier where emp\_no='103';



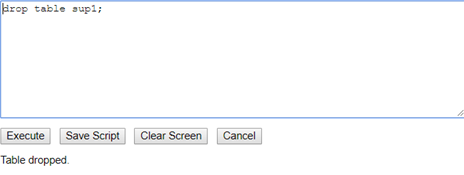
(12) Rename the table sup2.

rename sup2 to supplier2;



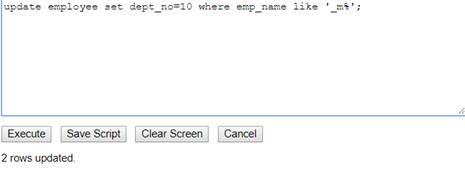
(13) Destroy table sup1 with all the data.

drop table sup1;



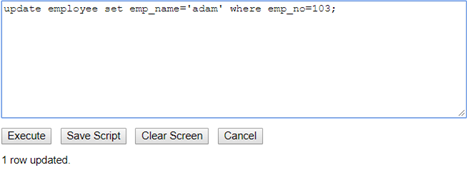
(14) Update the value dept\_no to 10 where second character of emp. name is ‘m’.

update employee set dept\_no=10 where emp\_name like '\_m%';



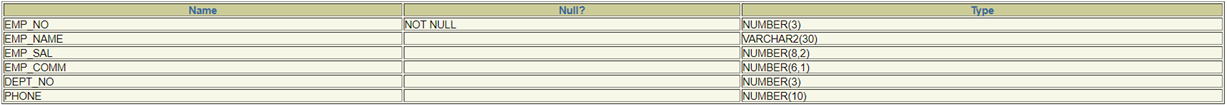
(15) Update the value of employee name whose employee number is 103.

update employee set emp\_name='adam' where emp\_no=103;



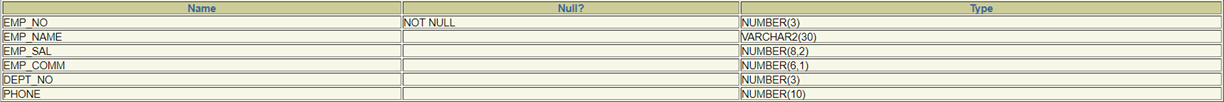
(16) Add one column phone to employee with size of column is 10.

alter table employee add phone number(10);desc employee2;



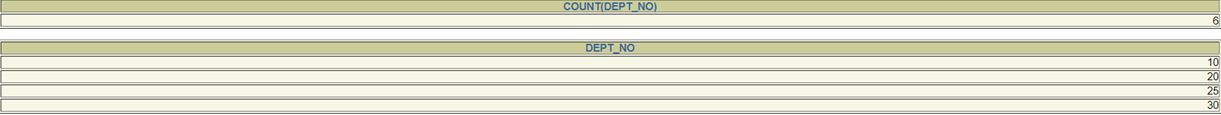
(17) Modify the column emp\_name to hold maximum of 30 characters.

alter table employee modify emp\_name varchar(30);desc empolyee;



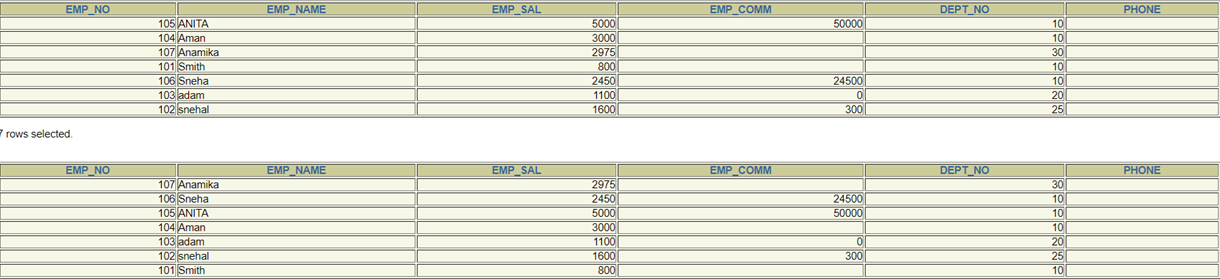
(18) Count the total no as well as distinct rows in dept\_no column with a condition of salary greater than 1000 of employee

select count(dept\_no) from employee2 where emp\_sal>1000;select distinct(dept\_no) from employee2 where emp\_sal>1000;



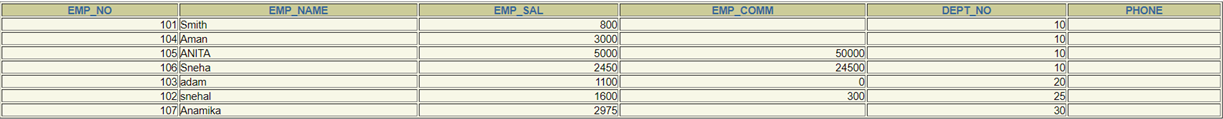
(19) Display the detail of all employees in ascending order, descending order of their name and no.

select \* from employee order by emp\_name asc;select \* from employee order by emp\_no desc;



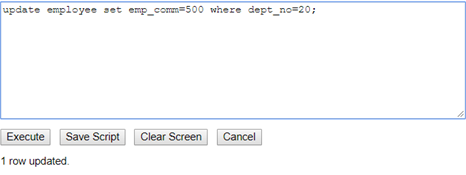
(20) Display the dept\_no in ascending order and accordingly display emp\_comm in descending order.

select \* from employee order by dept\_no asc,emp\_comm desc;



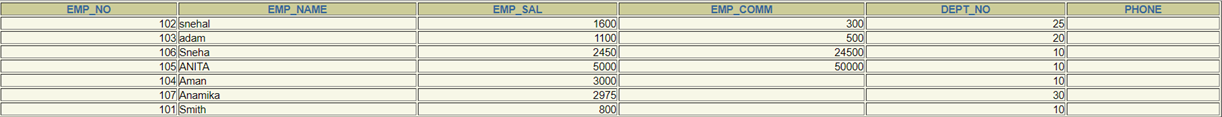
(21) Update the value of emp\_comm to 500 where dept\_no is 20.

update employee set emp\_comm=500 where dept\_no=20;



(22) Display the emp\_comm in ascending order with null value first and accordingly sort employee salary in descending order.

select \* from employee order by emp\_comm asc,emp\_sal desc;



(23) Display the emp\_comm in ascending order with null value last and accordingly sort emp\_no in descending order.

select \* from employee order by emp\_comm IS NULL,emp\_comm asc , emp\_no desc;



Conclusion: we have learnt sum,max,count,truncate,delete from,rename to,drop,update,alter add,alter modify,distinct,orderby.

PRACTICAL 4

Aim: **To study Single-row functions.**

(1) Write a query to display the current date. Label the column Date

SELECT TO\_CHAR(sysdate,'Day, ddth Month, yyyy')"DATE" FROM dual;



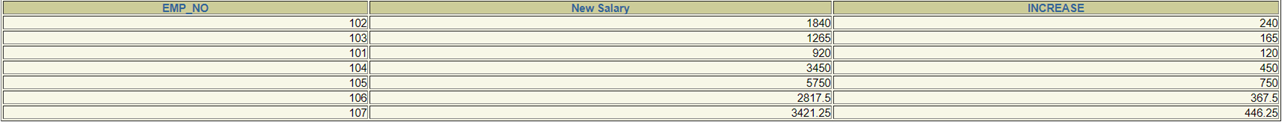
(2) For each employee, display the employee number, job, salary, and salary increased by 15% and expressed as a whole number. Label the column New Salary

select emp\_no,(emp\_sal+(emp\_sal\*0.15)) "New Salary" from employee;



(3) Modify your query no 4.(2) to add a column that subtracts the old salary from the new salary. Label the column Increase

select emp\_no,(emp\_sal+(emp\_sal\*0.15)) "New Salary",((emp\_sal+(emp\_sal\*0.15))-emp\_sal) "INCREASE" from employee;



(4) Write a query that displays the employee’s names with the first letter capitalized and all other letters lowercase, and the length of the names, for all employees whose name starts with J, A, or M. Give each column an appropriate label. Sort the results by the employees’ last names.

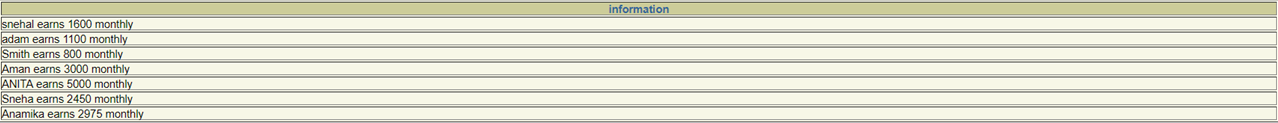
select emp\_no,initcap(emp\_name),emp\_sal,emp\_comm,dept\_no,phone from employee where emp\_name like 'J%' or emp\_name like 'A%' or emp\_name like 'M%' order by emp\_name ;



(5) Write a query that produces the following for each employee:

<employee last name> earns <salary> monthly

select concat(CONCAT (emp\_name,' earns '),concat(emp\_sal,' monthly ')) "information" from employee;



(6) Display the name, hire date, number of months employed and day of the week on which the employee has started. Order the results by the day of the week starting with Monday.

select emp\_name,hire\_date,round(datediff(sysdate(),hire\_date)/30) "Month difference between current date and hired\_date",dayname(hire\_date) from prac4no6 order by weekday(hire\_date) asc;

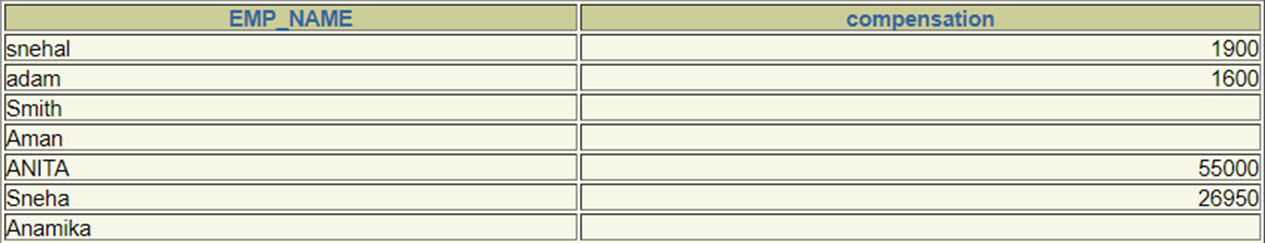


(7) Display the hiredate of emp in a format that appears as Seventh of June 1994 12:00:00 AM.

select concat(date\_format(hire\_date,'%D')," of ",date\_format(hire\_date,'%M %Y %r')) from prac4no6;

(8) Write a query to calculate the annual compensation of all employees (sal+comm.).

select emp\_name,emp\_sal+emp\_comm "compensation" from employee;



Practical-5

Aim: **Displaying data from Multiple Tables (join)**

(1) Give details of customers ANIL.

select LOANNO,CNAME,BNAME,borrow.AMOUNT "LOAN AMOUNT",customers.cname,city,a\_no,bname,amount,a\_date from borrow join customers on borrow.cname=customers.cname join deposit on customers.cname=deposit.cname;



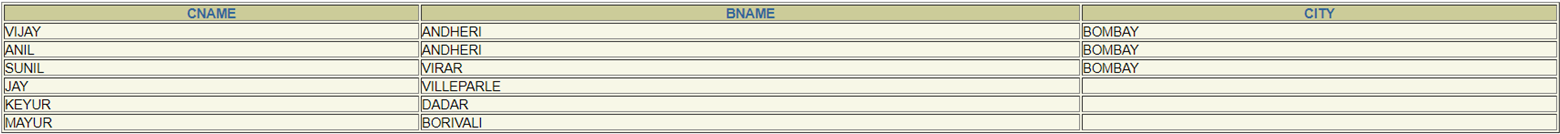
(2) Give name of customer who are borrowers and depositors and having living city Nagpur

(select a\_no,loanno,borrow.cname,deposit.bname,deposit.amount "Deposit",borrow.amount "Loan",a\_date from borrow left join(deposit) on borrow.cname=deposit.cname left join(customers) on borrow.cname=customers.cname where customers.city='NAGPUR');



(3) Give city as their city name of customers having same living branch.

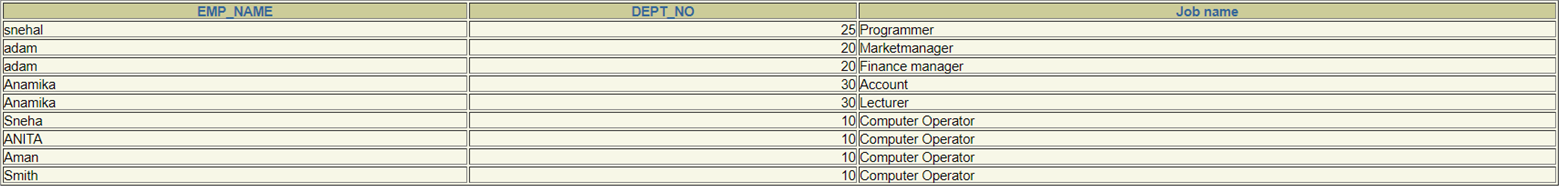
select deposit.cname,deposit.bname,branch.city from deposit left join(branch) on deposit.bname=branch.bname;



(4) Write a query to display the last name, department number, and department name for

all employees.

select employee.emp\_name,employee.dept\_no,jobwithdeptno.job\_title "Job name" from employee left join(job) on employee.dept\_no=job.dept\_no;



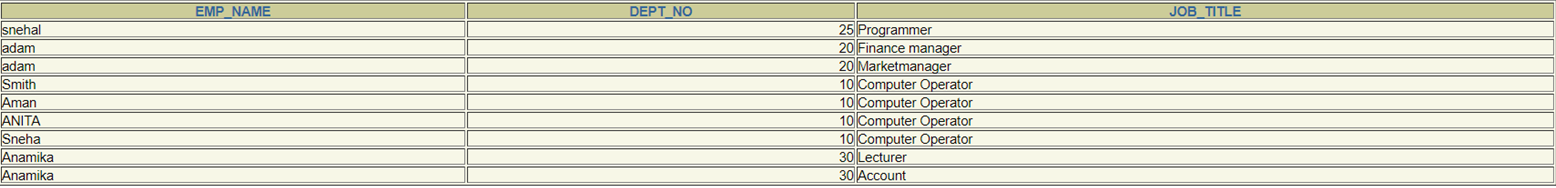
(5) Create a unique listing of all jobs that are in department 30. Include the location of the department in the output

select employee.emp\_name,employee.dept\_no,job.job\_title "Job name",location from employee left join(job) on employee.dept\_no=job.dept\_no where employee.dept\_no=30 and location='NEW YORK';



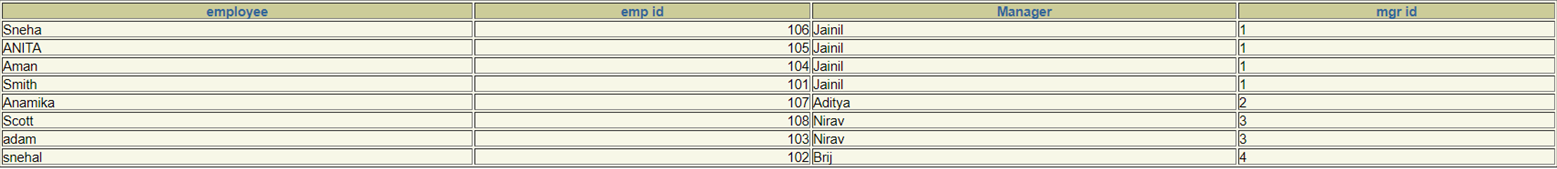
(6) Write a query to display the employee name, department number, and department name for all employees who work in NEW YORK.

select emp\_name,job.dept\_no,job\_title from employee left join(job) on employee.dept\_no=job.dept\_no where location='NEW YORK';



(7) Display the employee last name and employee number along with their manager’s last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.

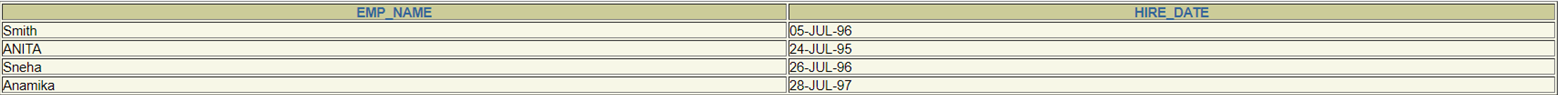
select emp\_name "employee",emp\_no "emp id",manager.manager\_name "Manager",manager\_no "mgr id" from employee left join(manager) on manager.dept\_no=employee.dept\_no;



(8) Create a query to display the name and hire date of any employee hired after employee SCOTT.

insert into employee values('108','Scott','1500','4000','20',NULL,'01-OCT-94');

select emp\_name,hire\_date from employee where hire\_date>(select hire\_date from employee where emp\_name='Scott');



Conclusion: learnt join,inner join,left join,right join,on,having keywords.

PRACTICAL 6:

**To apply the concept of Aggregating Data using Group functions.**

(1) List total deposit of customer having account date after 1-jan-96.

select sum(amount) from deposit where A\_DATE>'1-jan-96';



(2) List total deposit of customers living in city Nagpur.

select sum(amount) from (select customers.cname,customers.city,amount from customers inner join deposit on deposit.cname=customers.cname) group by city having city='Nagpur';



(3) List maximum deposit of customers living in bombay.

select max(amount) from (select customers.cname,customers.city,amount from customers inner join deposit on deposit.cname=customers.cname) group by city having city='bombay';



(4) Display the highest, lowest, sum, and average salary of all employees. Label the columns

Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.

select max(emp\_sal) "Maximum",min(emp\_sal) "Minimum",sum(emp\_sal) "Sum", round(avg(emp\_sal)) "Average" from employee;



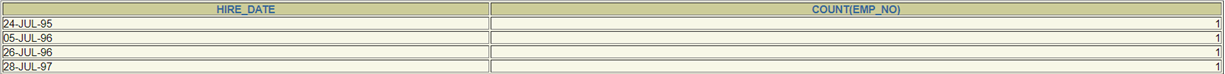
(5) Write a query that displays the difference between the highest and lowest salaries. Label the column DIFFERENCE.

select max(emp\_sal)- min(emp\_sal) "Difference" from employee;



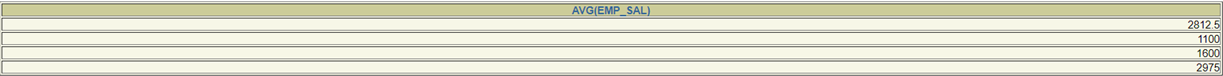
(6) Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998

select hire\_date,count(emp\_no) from employee group by hire\_date having hire\_date>'1-JAN-95' and hire\_date<'1-JAN-98';



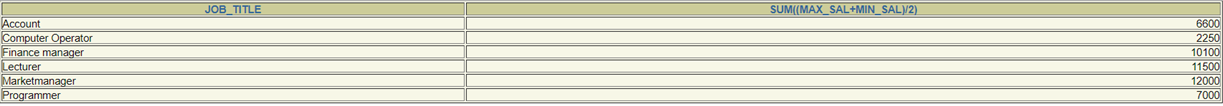
(7) Find the average salaries for each department without displaying the respective department numbers.

select avg(emp\_sal) from employee group by dept\_no;



(8) Write a query to display the total salary being paid to each job title, within each department.

select job\_title,sum((max\_sal+min\_sal)/2) from job group by job\_title;



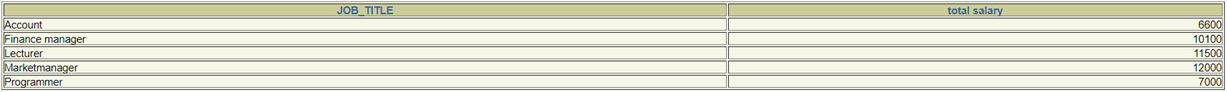
(9) Find the average salaries > 2000 for each department without displaying the respective department numbers.

select avg(emp\_sal) from employee group by dept\_no having avg(emp\_sal)>2000;



(10) Display the job and total salary for each job with a total salary amount exceeding 3000, in which excludes president and sorts the list by the total salary.

select job\_title, sum((max\_sal+min\_sal)/2) "total salary" from job group by job\_title having job\_title!='president' and sum((max\_sal+min\_sal)/2)>3000;



(11) List the branches having sum of deposit more than 5000 and located in city bombay.

select bname from (select bname,customers.city,amount from customers left join deposit on deposit.cname=customers.cname) group by bname having sum(amount)>3000 and city='bombay';



Conclusion: .learnt to combine two queries.

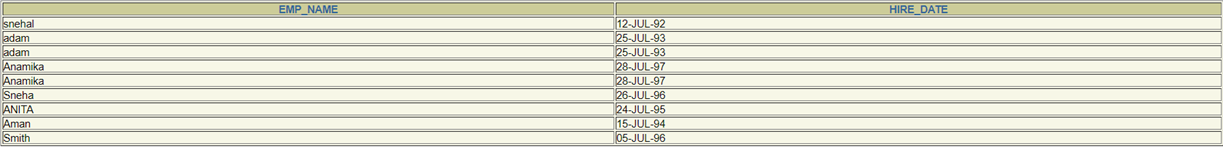
PRACTICAL 7:

Aim: **To solve queries using the concept of sub query.**

1) Write a query to display the last name and hire date of any employee in the same

department as SCOTT. Exclude SCOTT

select emp\_name,hire\_date from employee left join job on employee.dept\_no=job.dept\_no where emp\_name!='Scott';



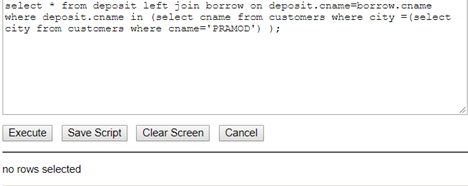
2) Give name of customers who are depositors having same branch city of mr. sunil.

select customers.cname from customers left join deposit on customers.cname=deposit.cname where bname in (select bname from deposit where cname='SUNIL');



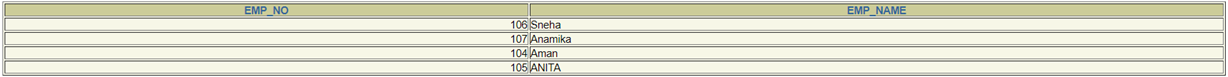
3) Give deposit details and loan details of customer in same city where pramod is living.

select \* from deposit left join borrow on deposit.cname=borrow.cname where deposit.cname in (select cname from customers where city =(select city from customers where cname='PRAMOD') );



4) Create a query to display the employee numbers and last names of all employees who earn more than the average salary. Sort the results in ascending order of salary.

select emp\_no,emp\_name from employee where emp\_sal>(select avg(emp\_sal) from employee) order by emp\_sal asc;



5) Give names of depositors having same living city as mr. anil and having deposit amount greater than 2000

select \* from deposit where cname in(select cname from customers where city in(select city from customers where cname='ANIL')) and amount>2000;



6) Display the last name and salary of every employee who reports to ford.

select emp\_name, emp\_sal from employee where dept\_no=(select dept\_no from manager where manager\_name='ford');



7) Display the department number, name, and job for every employee in the Accounting department.

select employee.dept\_no,emp\_name,job\_title,job\_id from employee left join(job) on employee.dept\_no=job.dept\_no where job.job\_title like '%manager%';



8) List the name of branch having highest number of depositors.

SELECT bname, COUNT(bname) "count" FROM deposit WHERE ROWNUM <= 1 group by bname order by count(bname) desc ;



9) Give the name of cities where in which the maximum numbers of branches are located.

select city,count(city) from branch WHERE ROWNUM <= 1 group by city order by count(city) desc ;



10) Give name of customers living in same city where maximum depositors are located.

select \* from deposit where cname in(select cname from customers where city=(select city from customers WHERE ROWNUM <= 1 group by city ));



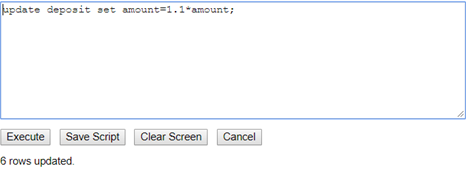
Conclusion:learnt subquery in sql.

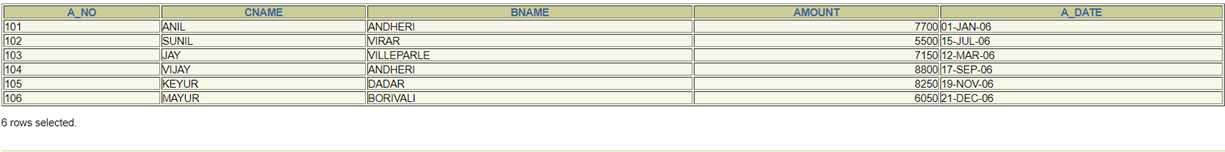
PRACTICAL 8:

Aim: **Manipulating Data**

(1) Give 10% interest to all depositors.

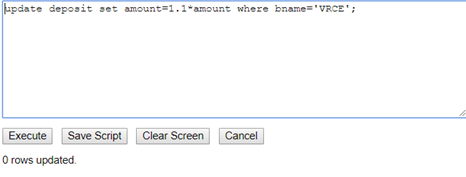
update deposit set amount=1.1\*amount;





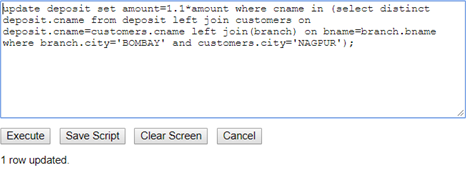
(2) Give 10% interest to all depositors having branch vrce

update deposit set amount=1.1\*amount where bname='VRCE';



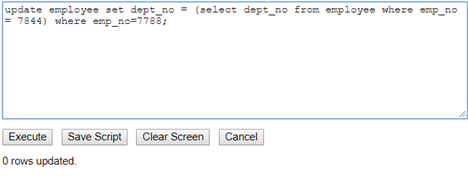
(3) Give 10% interest to all depositors living in nagpur and having branch city bombay.

update deposit set amount=1.1\*amount where cname in (select distinct deposit.cname from deposit left join customers on deposit.cname=customers.cname left join(branch) on bname=branch.bname where branch.city='BOMBAY' and customers.city='NAGPUR');



(4) Write a query which changes the department number of all employees with empno 7788’s job to employee 7844’current department number.

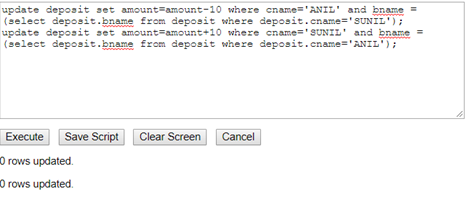
update employee set dept\_no = (select dept\_no from employee where emp\_no = 7844) where emp\_no=7788;



(5) Transfer 10 Rs from account of anil to sunil if both are having same branch.

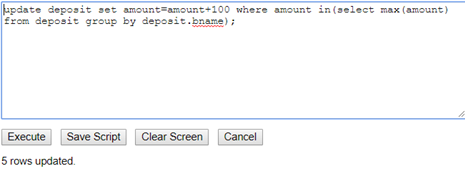
update deposit set amount=amount-10 where cname='ANIL' and bname = (select deposit.bname from deposit where deposit.cname='SUNIL');

update deposit set amount=amount+10 where cname='SUNIL' and bname = (select deposit.bname from deposit where deposit.cname='ANIL');



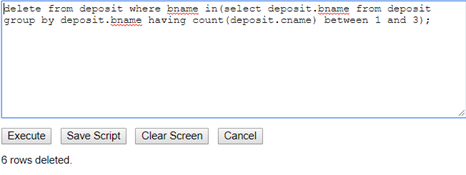
(6) Give 100 Rs more to all depositors if they are maximum depositors in their respective branch.

update deposit set amount=amount+100 where amount in(select max(amount) from deposit group by deposit.bname);



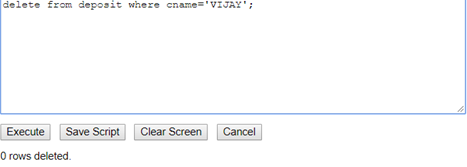
(7) Delete depositors of branches having number of customers between 1 to 3.

delete from deposit where bname in(select deposit.bname from deposit group by deposit.bname having count(deposit.cname) between 1 and 3);



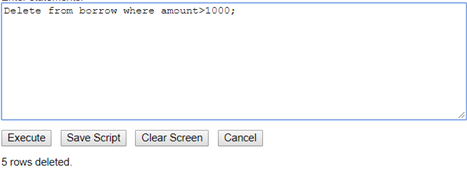
(8) Delete deposit of vijay.

delete from deposit where cname='VIJAY';



(9) Delete borrower of branches having average loan less than 1000.

Delete from borrow where amount>1000;



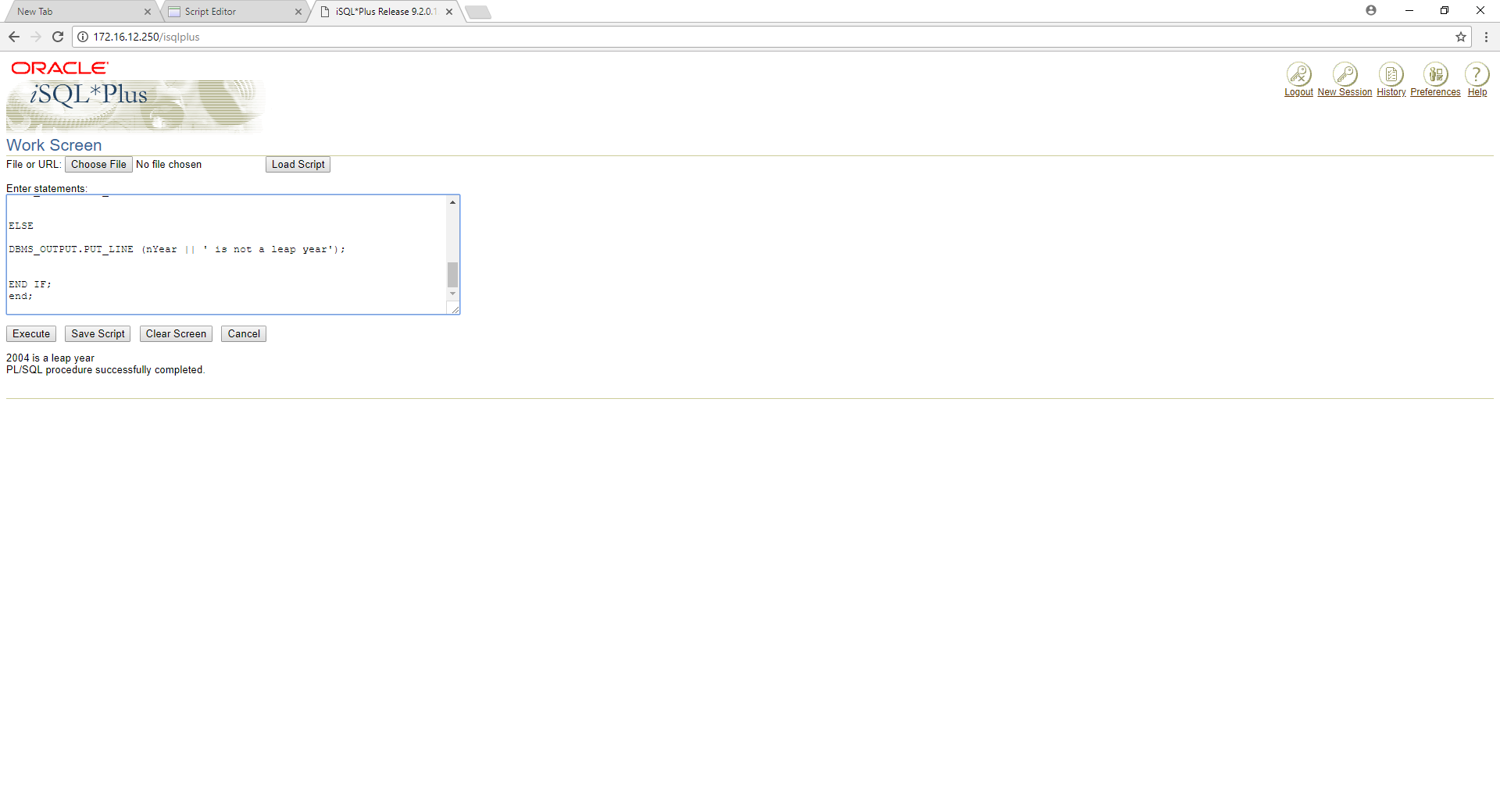
Conclidion: learnt data manipulation and updating contents of table using update table set query

**PRACTICAL 9:**

**To perform basic PL/SQL blocks**

Write a PL-SQL block for checking weather a given year is a Leap year or not

declare  
    nyear NUMBER(4,0);  
V\_REMAINDER1 NUMBER(5,2);  
V\_REMAINDER2 NUMBER(5,2);  
V\_REMAINDER3 NUMBER(5,2);  
  
BEGIN  
    nyear := 2004;  
V\_REMAINDER1 := MOD(nYear,4);  
V\_REMAINDER2 := MOD(nYear,100);  
V\_REMAINDER3 := MOD(nYear,400);  
IF ((V\_REMAINDER1 = 0 AND V\_REMAINDER2 <> 0 ) OR V\_REMAINDER3 = 0) THEN  
DBMS\_OUTPUT.PUT\_LINE(nYear || ' is a leap year');  
ELSE  
DBMS\_OUTPUT.PUT\_LINE (nYear || ' is not a leap year');  
END IF;  
end;



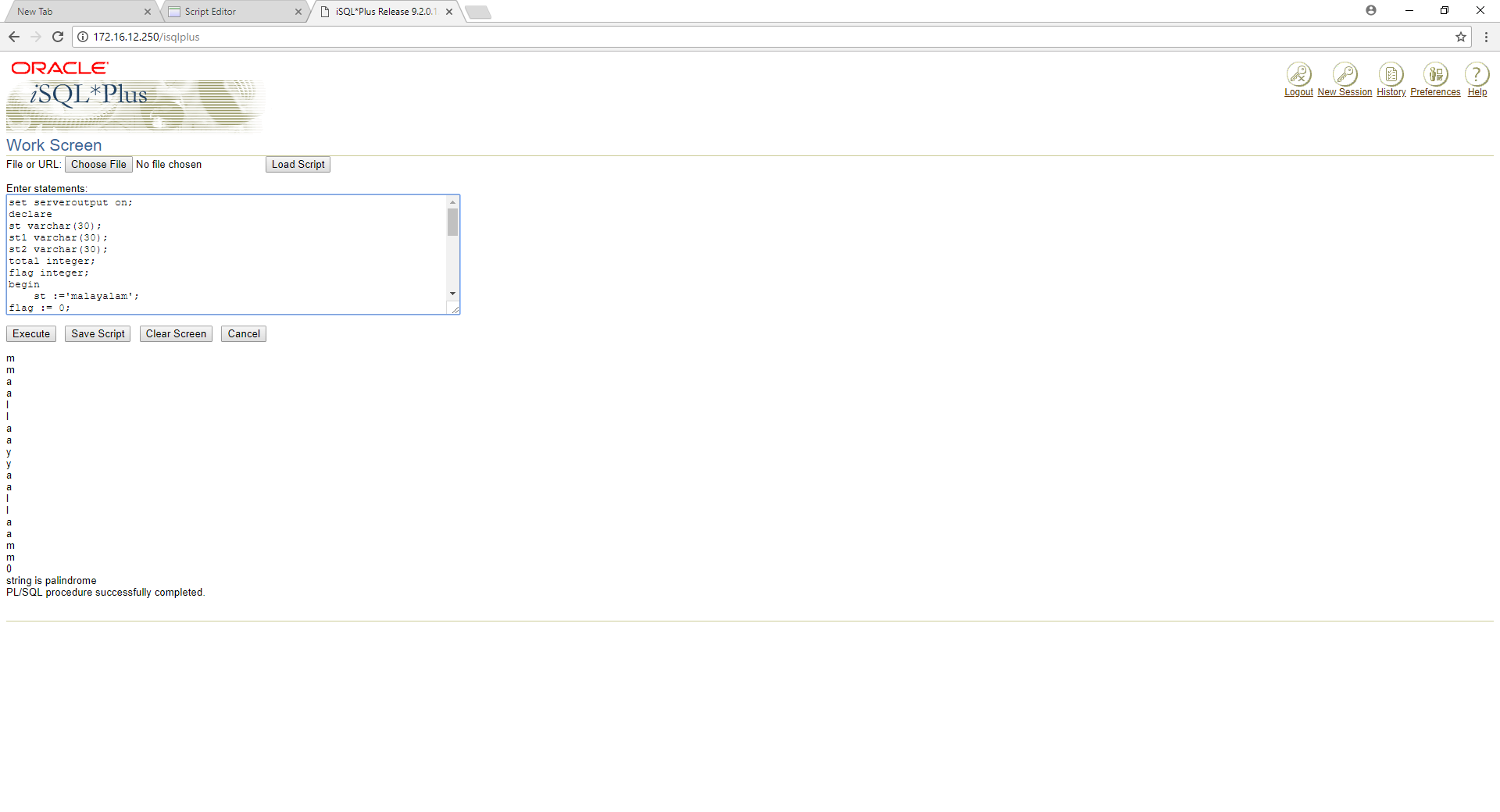
PRACTICAL 10

**To perform the concept of loop**

Find out whether given string is palindrome or not using For, While and Simple

Loop.

declare  
st varchar(30);  
st1 varchar(30);  
st2 varchar(30);  
total integer;  
flag integer;  
begin  
    st :='malayalam';  
flag := 0;  
total :=length(st);  
for var in 1..total loop  
    st1:= SUBSTR (st, var, 1);  
 dbms\_output.put\_line(st1);   
st2:= SUBSTR (st, total-var+1, 1);  
 dbms\_output.put\_line(st2);   
if(st1<>st2) then  
    flag:=1;  
end if;  
end loop;  
 dbms\_output.put\_line(flag);   
  
if flag=1 then  
 dbms\_output.put\_line('string is not palindrome');   
else  
    dbms\_output.put\_line('string is palindrome');   
end if;  
end;



**PRACTICAL 11**

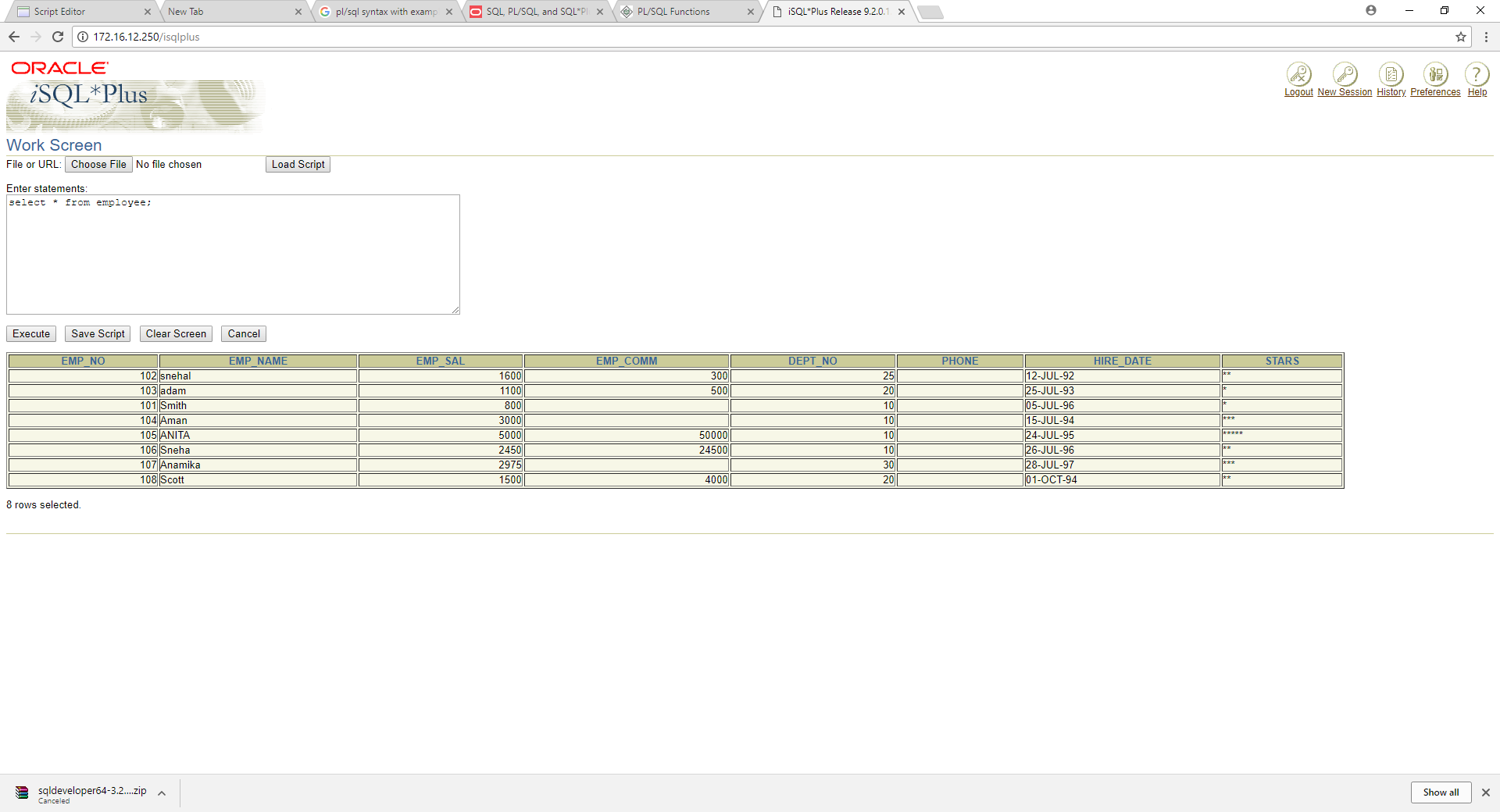
**To understand the concept of “select into” and “% type” attribute**.

Create an EMPLOYEES table that is a replica of the EMP table. Add a new column, STARS, of VARCHAR2 data type and length of 50 to the EMPLOYEES table for storing asterisk (\*).

Create a PL/SQL block that rewards an employee by appending an asterisk in the STARS column for every Rs1000/- of the employee’s salary. For example, if the employee has a salary amount of Rs8000/-, the string of asterisks should contain eight asterisks. If the employee has a salary amount of Rs12500/-, the string of asterisks should contain 13 asterisks.

Update the STARS column for the employee with the string of asterisks.

DECLARE  
EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE;  
EMPS\_NAME "EMPLOYEE".EMP\_NAME%TYPE;  
EMPS\_SAL "EMPLOYEE".EMP\_SAL%TYPE;  
EMPS\_COMM "EMPLOYEE".EMP\_COMM%TYPE;  
DEPT\_NO1 "EMPLOYEE".DEPT\_NO%TYPE;  
EMPS\_STAR VARCHAR2(50);  
i number;  
BEGIN  
  
for i in 101..108  
loop  
select NVL( ROUND (EMP\_SAL/1000),0) INTO EMPS\_SAL FROM EMPLOYEE WHERE EMP\_NO=I;  
EMPS\_STAR:=NULL;  
FOR J IN 1..EMPS\_SAL  
LOOP  
EMPS\_STAR:=EMPS\_STAR || '\*' ;  
END LOOP;  
update employee set stars=emps\_star where emp\_no = i;  
end loop;  
  
END;  
​



**PRACTICAL 12**

**To perform the concept of cursor**

a) Display all the information of EMP table using %ROWTYPE.

(b) Create a PL/SQL block that does the following:

In a PL/SQL block, retrieve the name, salary, and MANAGER ID of the employees working in the particular department. Take Department Id from user.

If the salary of the employee is less than 1000 and if the manager ID is either 7902 or 7839, display the message <<last\_name>> Due for a raise. Otherwise, display the message <<last\_name>> Not due for a raise.

set serveroutput on;

declare

emp\_no employee.emp\_no%TYPE;

emp\_name employee.emp\_name%TYPE;

emp\_sal employee.emp\_sal%TYPE;

EMP\_COMM employee.EMP\_COMM%TYPE;

DEPT\_NO employee.DEPT\_NO%TYPE;

PHONE employee.PHONE%TYPE;

HIRE\_DATE employee.HIRE\_DATE%TYPE;

STARS employee.STARS%TYPE;

cursor c is select \* from employee;

begin

open c;

loop

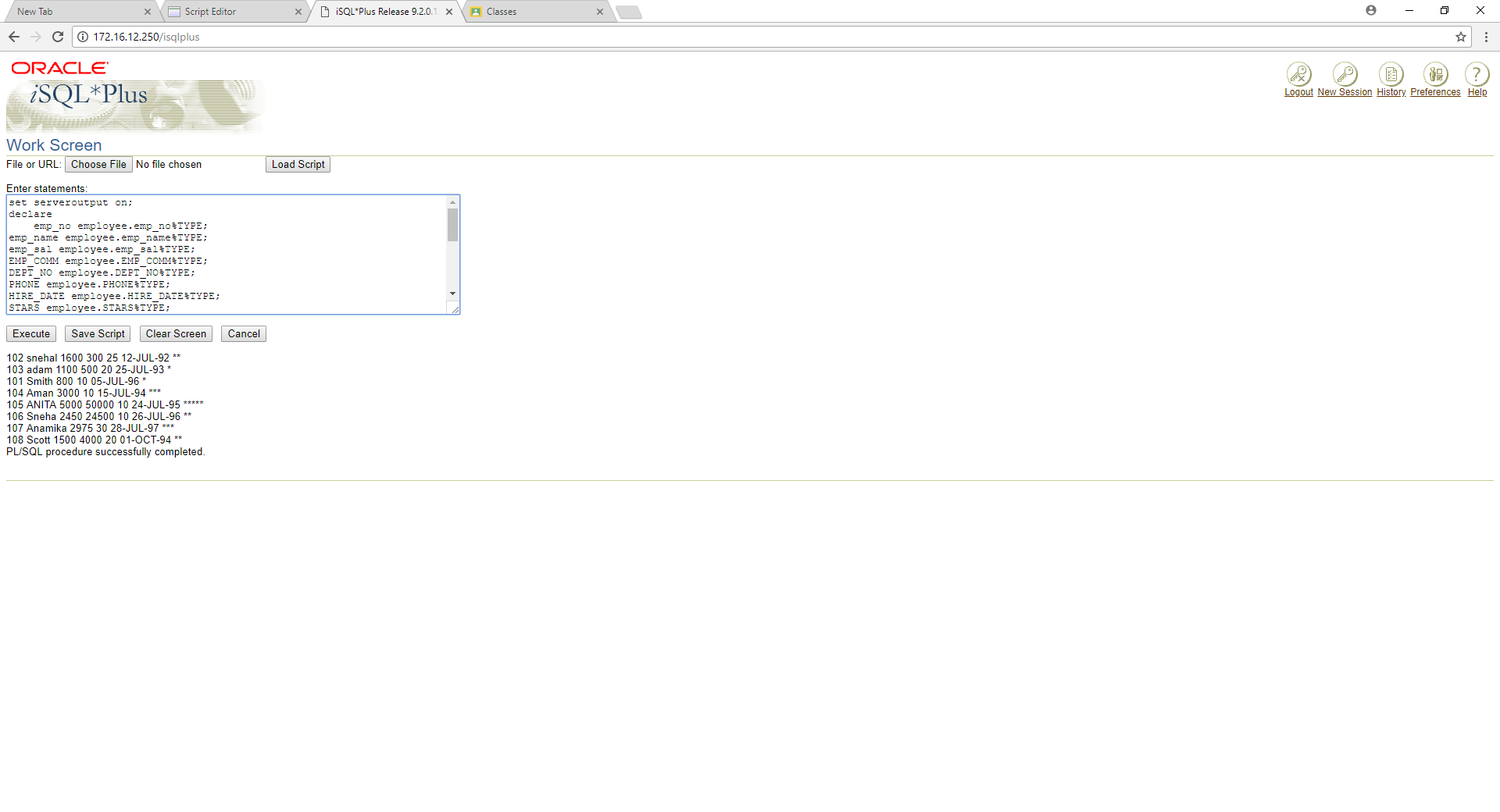
fetch c into emp\_no,emp\_name,emp\_sal,emp\_comm,dept\_no,phone,hire\_date,stars;

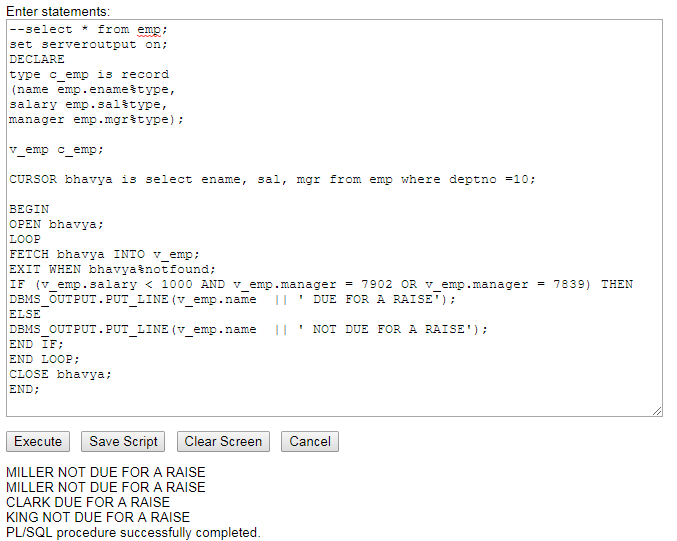
exit when c%NOTFOUND;

dbms\_output.put\_line(emp\_no || ' ' || emp\_name || ' ' || emp\_sal || ' ' || emp\_comm || ' ' || dept\_no || ' ' || phone || ' ' || hire\_date || ' ' || stars);

end loop;

end;





**PRACTICAL 13:**

**To perform the concept of trigger**

Write a PL/SQL block to update the salary where deptno is 10. Generate trigger that will store the original record in other table before updation take place

CREATE OR REPLACE TRIGGER a    
BEFORE DELETE OR INSERT OR UPDATE ON employee    
FOR EACH ROW    
WHEN (NEW.EMP\_ID > 0)    
DECLARE    
sal\_diff number;    
BEGIN    
sal\_diff:= :new.salary - :old.salary;  
dbms\_output.put\_line('salarydifference'||sal\_diff);  
END;  ​

PRACTICAL 14:

**To perform the concept of function and procedure**

Write a PL/SQL block to update the salary of employee specified by empid. If record exist then update the salary otherwise display appropriate message. Write a function as well as procedure for updating salary.

DECLARE

A NUMBER;

SAL NUMBER;

EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE;

EMPS\_NAME "EMPLOYEE".EMP\_NAME%TYPE;

EMPS\_SAL "EMPLOYEE".EMP\_SAL%TYPE;

EMPS\_COMM "EMPLOYEE".EMP\_COMM%TYPE;

DEPT\_NO1 "EMPLOYEE".DEPT\_NO%TYPE;

HIRE\_DATE1 "EMPLOYEE".HIRE\_DATE%TYPE;

STARZ "EMPLOYEE".STARS%TYPE;

CURSOR S\_C IS SELECT EMP\_NO,EMP\_NAME,EMP\_SAL,EMP\_COMM,DEPT\_NO,HIRE\_DATE,STARS FROM EMPLOYEE WHERE EMP\_NO=&EMP\_NO;

PROCEDURE SAL\_C(SAL IN OUT NUMBER)

is

BEGIN

OPEN S\_C;

FETCH S\_C INTO EMPS\_NO,EMPS\_NAME,EMPS\_SAL,EMPS\_COMM,DEPT\_NO1,HIRE\_DATE1,STARZ;

IF S\_C%FOUND THEN

UPDATE EMPLOYEE SET EMP\_SAL = SAL+EMPS\_SAL WHERE EMPS\_SAL=EMP\_SAL;

SAL := SAL+EMPS\_SAL;

DBMS\_OUTPUT.PUT\_LINE(SAL);

ELSE

DBMS\_OUTPUT.PUT\_LINE('WRONG CHOICE');

END IF;

CLOSE S\_C;

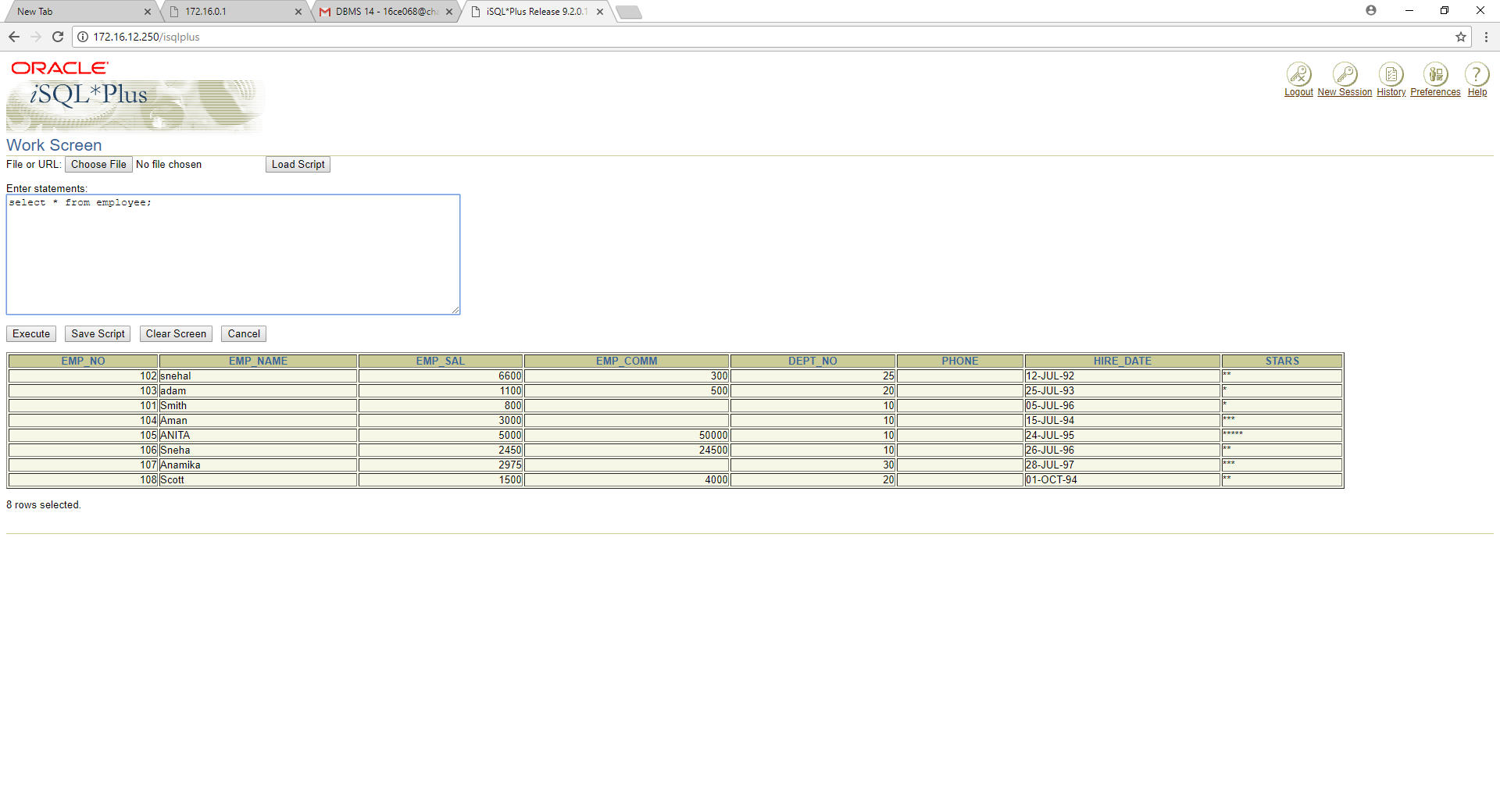
END;

BEGIN

A:=5000;

SAL\_C(A);

END;



CREATE OR REPLACE

FUNCTION "CheckidandUpdate" (id IN NUMBER, sal IN NUMBER)

RETURN NUMBER

AS

cursor c is SELECT \* FROM EMPLOYEE where EMP\_NO = id;

abc c%rowtype;

ret NUMBER(1);

BEGIN

open c;

fetch c into abc;

IF c%notfound THEN

ret:=0;

ELSE

UPDATE EMPLOYEE SET EMP\_SAL=sal WHERE EMP\_NO=id;

ret:=1;

END IF;

close c;

RETURN ret;

END;

DECLARE

retval NUMBER;

id NUMBER;

sal NUMBER;

BEGIN

id:=&Enter\_id;

sal:=&Enter\_sal\_to\_update;

retval := "CheckidandUpdate"(id, sal);

IF retval=1 THEN

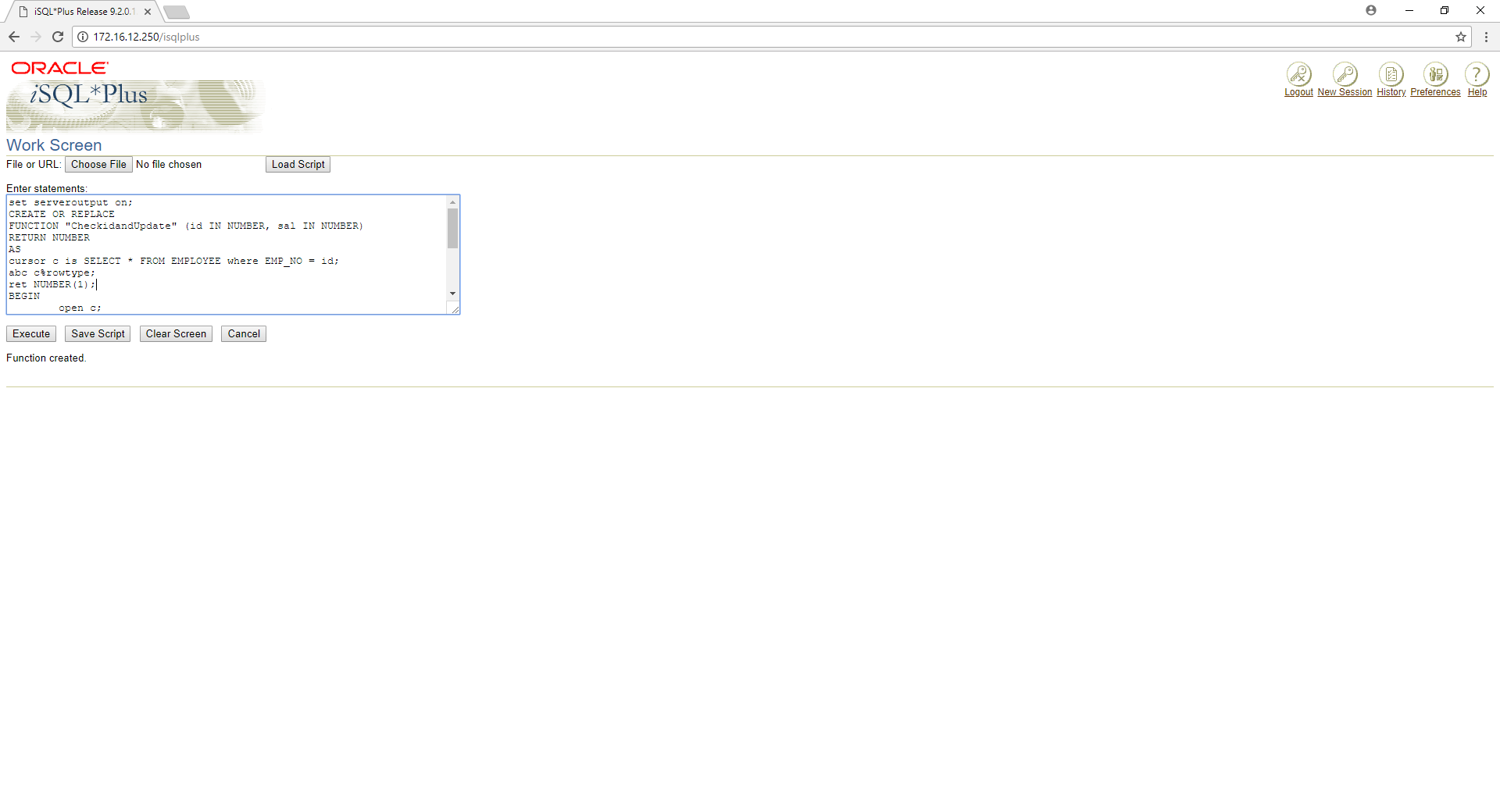
dbms\_output.put\_line('Updated Succesfully');

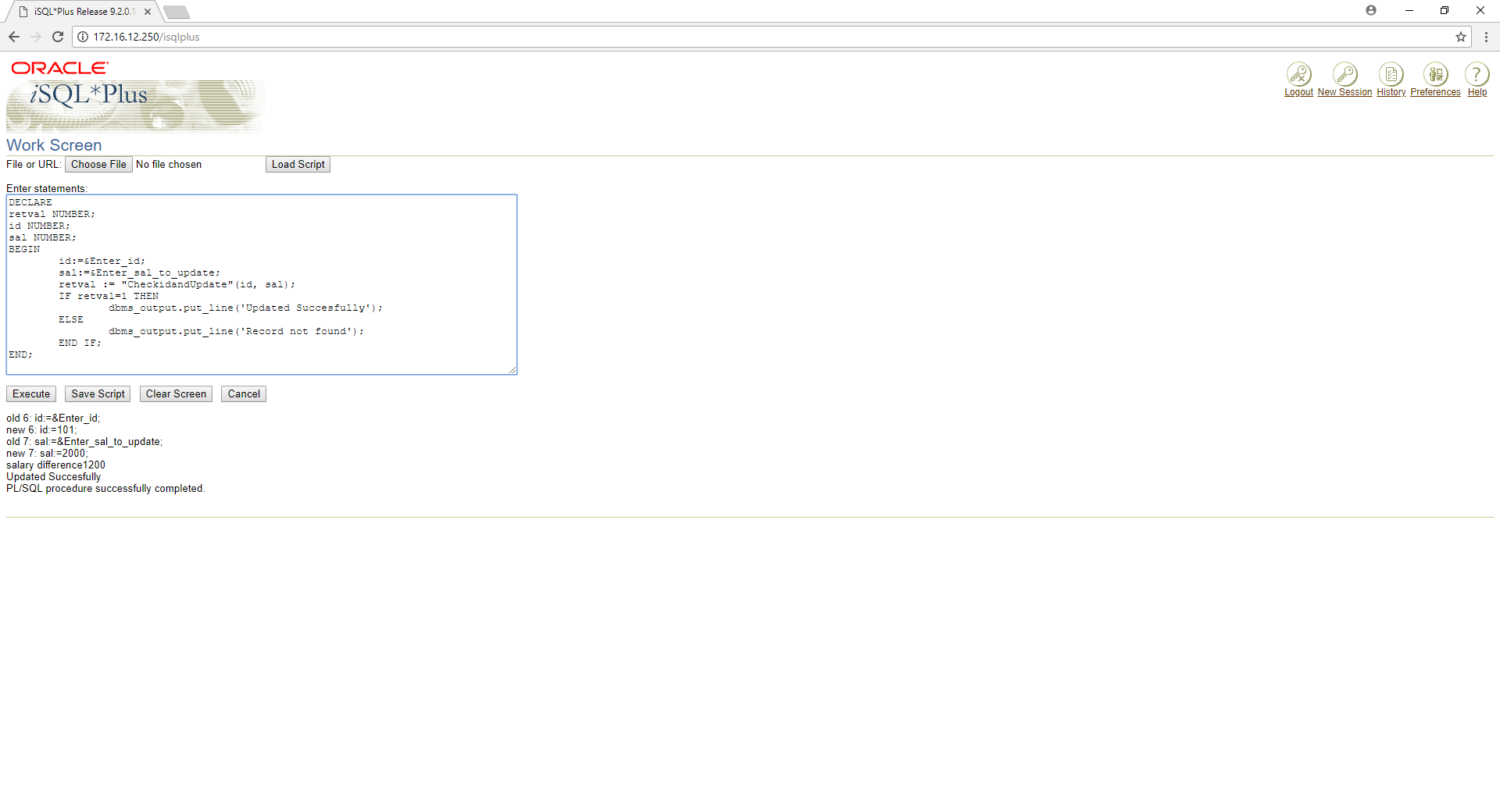
ELSE

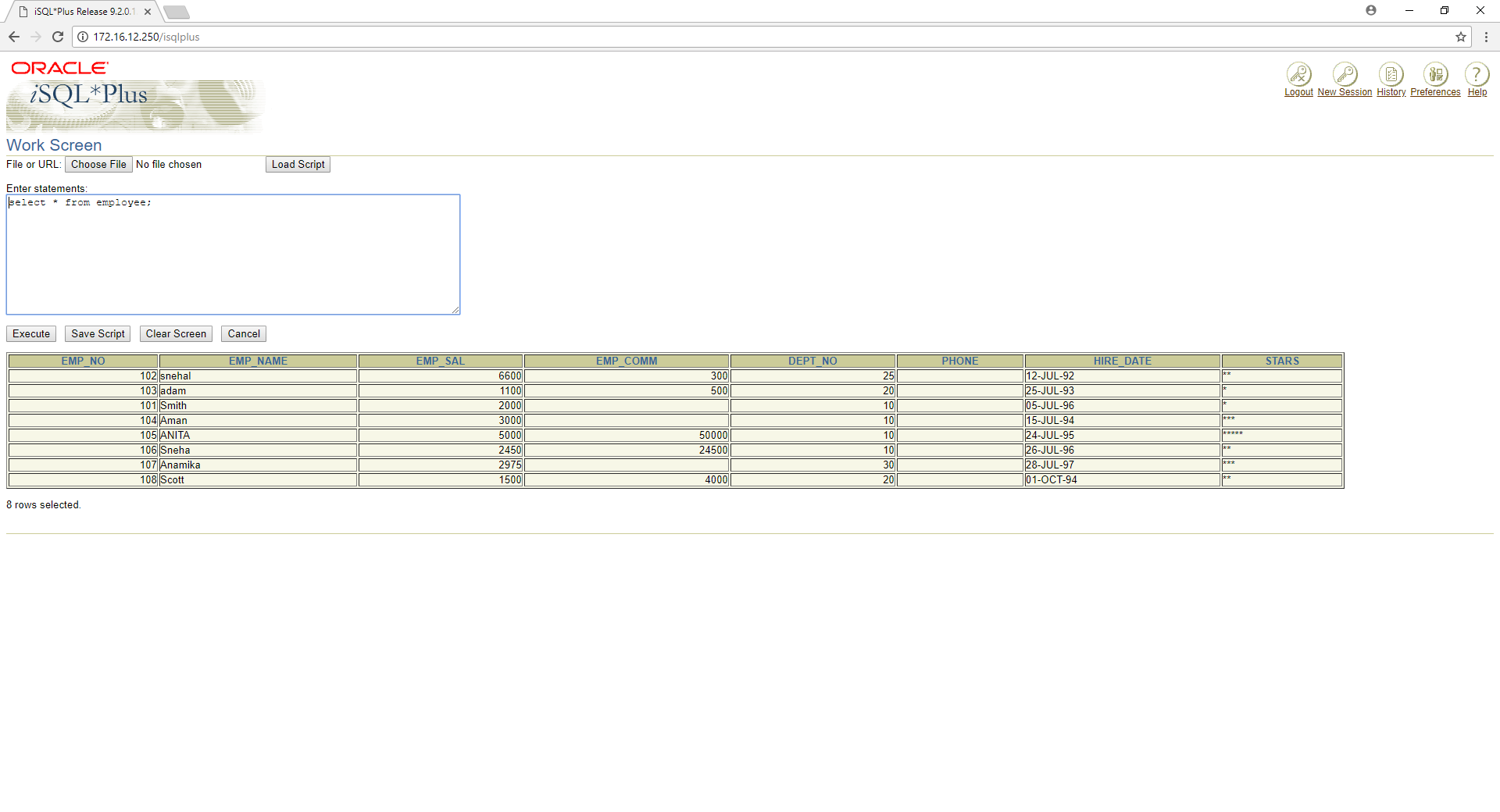
dbms\_output.put\_line('Record not found');

END IF;

END;







PRACTICAL 15:

**To perform the concept of exception handler**

Write a PL/SQL block that will accept the employee code, amount and operation. Based on specified operation amount is added or deducted from salary of said employee. Use user defined exception handler for handling the exception.

DECLARE

EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE;

EMPS\_NAME "EMPLOYEE".EMP\_NAME%TYPE;

EMPS\_SAL "EMPLOYEE".EMP\_SAL%TYPE;

EMPS\_COMM "EMPLOYEE".EMP\_COMM%TYPE;

DEPT\_NO1 "EMPLOYEE".DEPT\_NO%TYPE;

HIREDATE1 "EMPLOYEE".HIREDATE%TYPE;

STARZ "EMPLOYEE".STARS%TYPE;

CURSOR CAMOUNT IS SELECT EMP\_NO,EMP\_NAME,EMP\_SAL,EMP\_COMM,DEPT\_NO,HIRE\_DATE,STARS FROM EMPLOYEE WHERE EMP\_NO=&EMP\_NO;

AMOUNT NUMBER:=&AMOUNT;

CA NUMBER ;

EX\_INVALID\_TRANSACTION EXCEPTION;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('SELECT YOUR CHOICE');

DBMS\_OUTPUT.PUT\_LINE('1... WITHROW');

DBMS\_OUTPUT.PUT\_LINE('2... ADD');

CA:=&1\_FOR\_WITHDRAW\_\_\_\_2\_FOR\_ADD;

OPEN CAMOUNT;

FETCH CAMOUNT INTO EMPS\_NO,EMPS\_NAME,EMPS\_SAL,EMPS\_COMM,DEPT\_NO1,HIRE\_DATE1,STARZ;

CASE CA

WHEN 1 THEN

IF AMOUNT>EMPS\_SAL THEN

RAISE EX\_INVALID\_TRANSACTION;

ELSE

UPDATE EMPLOYEE SET EMP\_SAL=EMPS\_SAL-AMOUNT WHERE EMP\_NO=EMPS\_NO;

DBMS\_OUTPUT.PUT\_LINE('UPDATED!!!' || AMOUNT || ' AMOUNT WITHDRAWED FROM SALARY');

END IF;

WHEN 2 THEN

IF CAMOUNT%NOTFOUND THEN

RAISE NO\_DATA\_FOUND;

ELSE

UPDATE EMPLOYEE SET EMP\_SAL=EMPS\_SAL+AMOUNT WHERE EMP\_NO=EMPS\_NO;

DBMS\_OUTPUT.PUT\_LINE('UPDATED!!!' || AMOUNT || ' AMOUNT ADDED INTO SALARY');

END IF;

ELSE DBMS\_OUTPUT.PUT\_LINE('ONLY TWOCHOICE AVAILABLE');

END CASE;

EXCEPTION

WHEN EX\_INVALID\_TRANSACTION THEN

DBMS\_OUTPUT.PUT\_LINE('TRANSECTION NOT ALLOWED !!! CHECK AGAIN YOUR DETAILS');

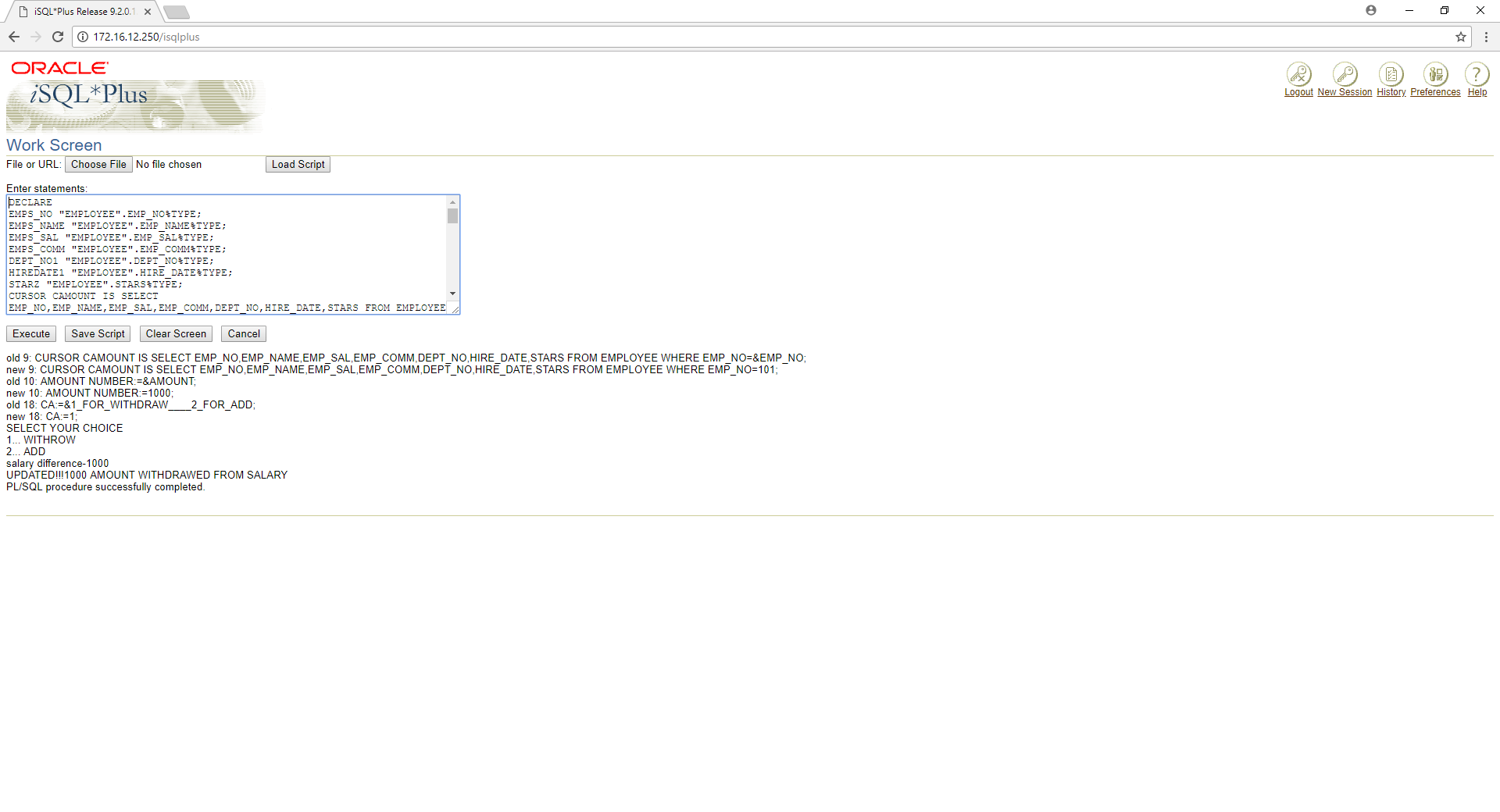
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE NOT SELECTED');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR OCCUR');

END;



PRACTICAL 16:

**To perform the concept of package**

Create and invoke a package that contains private and public constructs.

CREATE OR REPLACE PACKAGE EMP\_package AS

-- Adds a customer

PROCEDURE addCustomer(

EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE,

EMPS\_NAME "EMPLOYEE".EMP\_NAME%TYPE,

EMPS\_SAL "EMPLOYEE".EMP\_SAL%TYPE,

EMPS\_COMM "EMPLOYEE".EMP\_COMM%TYPE,

DEPT\_NO1 "EMPLOYEE".DEPT\_NO%TYPE,

HIREDATE1 "EMPLOYEE".HIRE\_DATE%TYPE,

STARZ "EMPLOYEE".STARS%TYPE);

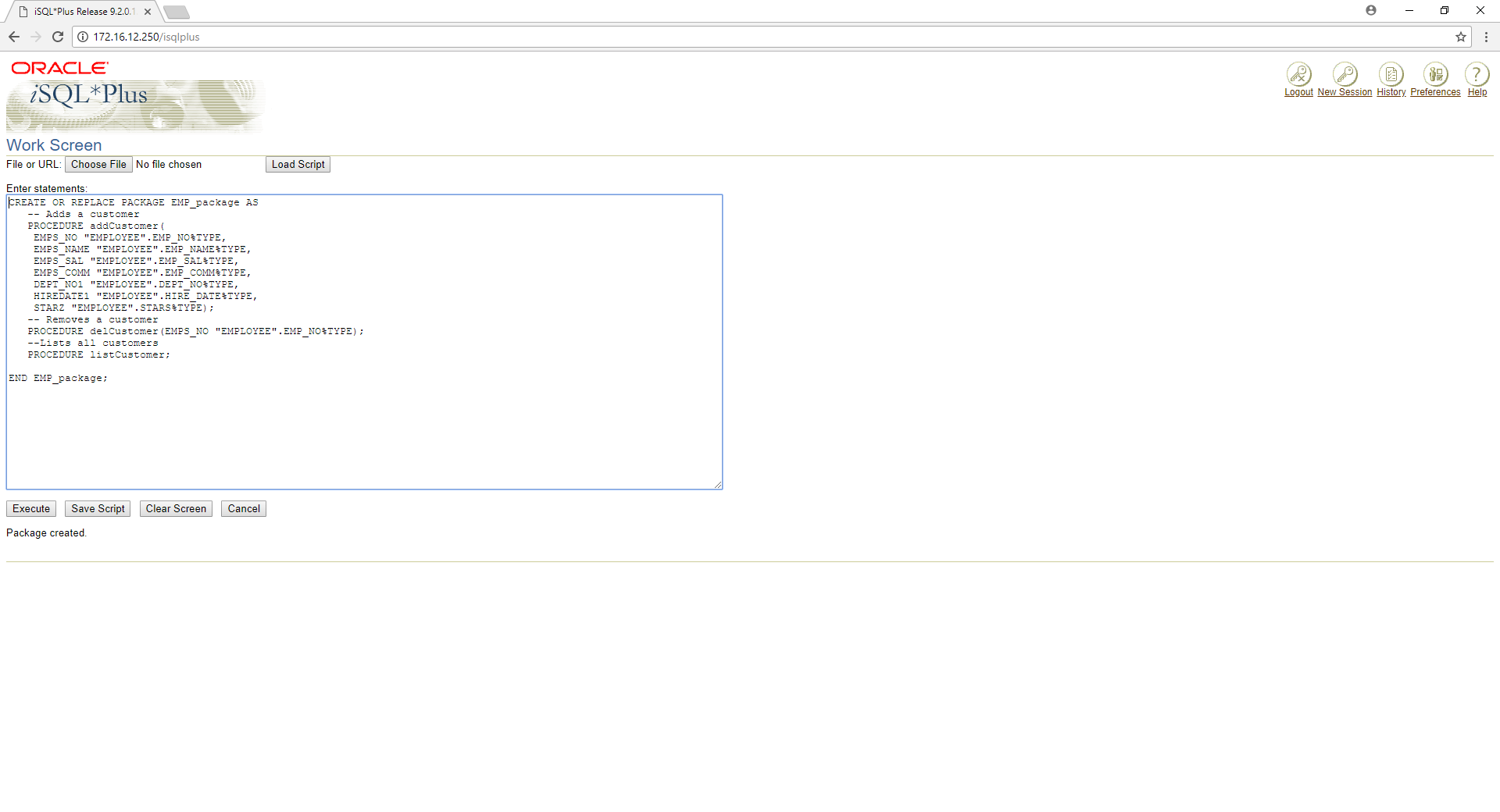
-- Removes a customer

PROCEDURE delCustomer(EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE);

--Lists all customers

PROCEDURE listCustomer;

END EMP\_package;



CREATE OR REPLACE PACKAGE BODY EMP\_package AS

PROCEDURE addCustomer(

EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE,

EMPS\_NAME "EMPLOYEE".EMP\_NAME%TYPE,

EMPS\_SAL "EMPLOYEE".EMP\_SAL%TYPE,

EMPS\_COMM "EMPLOYEE".EMP\_COMM%TYPE,

DEPT\_NO1 "EMPLOYEE".DEPT\_NO%TYPE,

HIREDATE1 "EMPLOYEE".HIREDATE%TYPE,

STARZ "EMPLOYEE".STARS%TYPE)

IS

BEGIN

INSERT INTO EMPLOYEE (EMP\_NO,EMP\_NAME,EMP\_SAL,EMP\_COMM,DEPT\_NO,HIREDATE,STARS)

VALUES(EMPS\_NO,EMPS\_NAME,EMPS\_SAL,EMPS\_COMM,DEPT\_NO1,HIREDATE1,STARZ);

END addCustomer;

PROCEDURE delCustomer(EMPS\_NO "EMPLOYEE".EMP\_NO%TYPE) IS

BEGIN

DELETE FROM EMPLOYEE

WHERE EMP\_NO = EMPS\_NO;

END delCustomer;

PROCEDURE listCustomer IS

CURSOR c\_customers is

SELECT EMP\_NAME FROM EMPLOYEE;

TYPE c\_list is TABLE OF "EMPLOYEE".EMP\_NAME%TYPE;

name\_list c\_list := c\_list();

counter integer :=0;

BEGIN

FOR n IN c\_customers LOOP

counter := counter +1;

name\_list.extend;

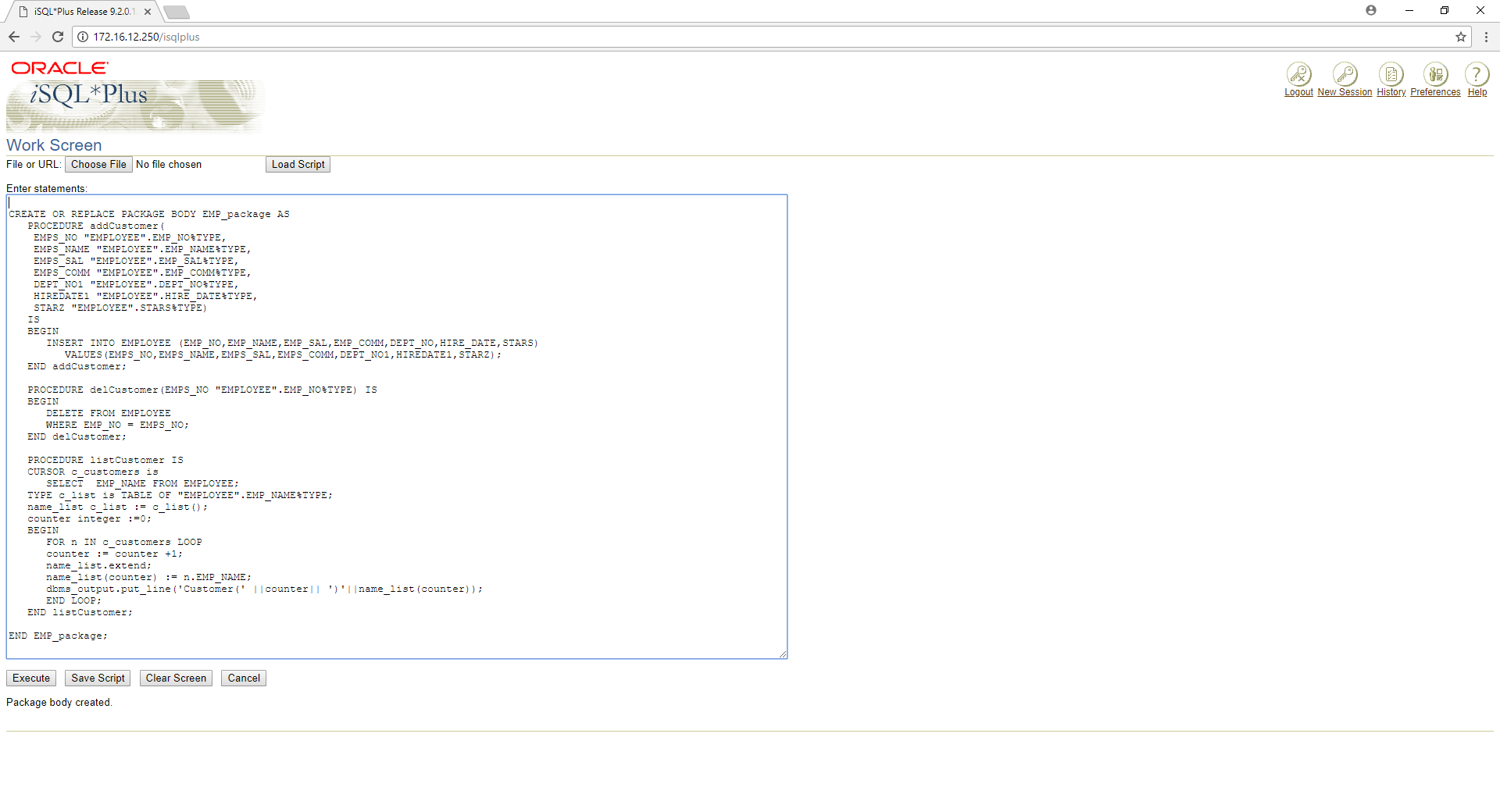
name\_list(counter) := n.EMP\_NAME;

dbms\_output.put\_line('Customer(' ||counter|| ')'||name\_list(counter));

END LOOP;

END listCustomer;

END EMP\_package;



DECLARE

EID "EMPLOYEE".EMP\_NO%TYPE := 111;

BEGIN

EMP\_package.addcustomer(110, 'Rajnish', 2500,5000,15,'09-04-15','\*\*\*');

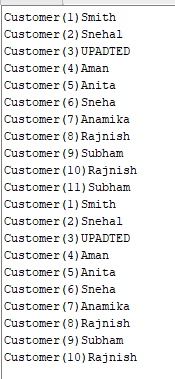
EMP\_package.addcustomer(111, 'Subham', 3200,10000,10,'12-04-12', '\*\*\*');

EMP\_package.listcustomer;

EMP\_package.delcustomer(EID);

EMP\_package.listcustomer;

END;



Practical 17:

**To create, insert values in MongoDB.**

Practical-17

Aim : To create, insert values in MongoDB

1.What is NoSQL Database? How does it work?

2.Types of NoSQL Database

3.Types: Key value Store, Column Store, Document database, Graph based Compare based on these parameters: performance, Scalability, complexity, Flexibility

4.Why NoSQL and not to use SQL?

Snapshots of Create DB, Create Collection, Update, Delete, Select in MongoDB

Software : Mongodb

Knowledge Required

Knowledge of mongodb

How to use mongodb

Theory

1.**What is NoSQL Database? How does it work?**

•NoSQL encompasses a wide variety of different database technologies that were developed in response to the demands presented in building modern applications:

•Developers are working with applications that create massive volumes of new, rapidly changing data types — structured, semi-structured, unstructured and polymorphic data.

•Long gone is the twelve-to-eighteen month waterfall development cycle. Now small teams work in agile sprints, iterating quickly and pushing code every week or two, some even multiple times every day.

•Applications that once served a finite audience are now delivered as services that must be always-on, accessible from many different devices and scaled globally to millions of users.

•Organizations are now turning to scale-out architectures using open source software, commodity servers and cloud computing instead of large monolithic servers and storage infrastructure.

Relational databases were not designed to cope with the scale and agility challenges that face modern applications, nor were they built to take advantage of the commodity storage and processing power available today.

2.**Types of NoSQL Database**

There are four types of mongo db

1.**Document databases**

2.**Graph**

3.**Key-value stores**

4.**Wide-column stores**

3.**Types: Key value Store, Column Store, Document database, Graph based Compare based on these parameters: performance, Scalability, complexity, Flexibility**

•**Document databases**pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-arraypairs, or even nested documents.

•**Graph stores**are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph.

•**Key-value stores**are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value. Examples of key-value stores are Riak and Berkeley DB. Some key-valuestores, such as Redis, allow each value to have a type, such as 'integer', which adds functionality.

•**Wide-column stores**such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

4.**Why NoSQL and not to use SQL?**

***SQL vs NoSQL: MySQL vs MongoDB***

Now that we’ve established the key structural differences between SQL and NoSQL databases, let’s delve into the key functional differences between the two, looking specifically at MySQL and MongoDB as examples.

**MySQL: The SQL Relational Database**

The following are some MySQL benefits and strengths:

•**Maturity**: MySQL is an extremely established database, meaning that there’s a huge community, extensive testing and quite a bit of stability.

•**Compatibility:**MySQL is available for all major platforms, including Linux, Windows, Mac, BSD and Solaris. It also has connectors to languages like Node.js,

Ruby, C#, C++, Java, Perl, Python and PHP, meaning that it’s not limited to SQL query language.

•**Cost-effective:**The database is open source and free.

•**Replicable:**The MySQL database can be replicated across multiple nodes, meaning that the workload can be reduced and the scalability and availability of the application can be increased.

•**Sharding:**While sharding cannot be done on most SQL databases, it can be done on MySQL servers. This is both cost-effective and good for business.

**MongoDB: The NoSQL Non-Relational Database**

The following are some of MongoDB benefits and strengths:

•**Dynamic schema:**As mentioned, this gives you flexibility to change your data schema without modifying any of your existing data.

•**Scalability:**MongoDB is horizontally scalable, which helps reduce the workload and scale your business with ease.

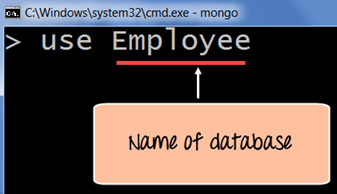
•**Manageability:**The database doesn’t require a database administrator. Since it is fairly user-friendly in this way, it can be used by both developers and administrators.

•**Speed:**It’s high-performing for simple queries.

•**Flexibility:**You can add new columns or fields on MongoDB without affecting existing rows or application performance.

## Creating a database using “use” command

Creating a database in MongoDB is as simple as issuing the "**using**" command. The following example shows how this can be done.

[](https://cdn.guru99.com/images/MongoDB/112115_0607_Introductio2.png)

## Creating a collection using insert()

The easiest way to create a collection is to insert a record (which is nothing but a document consisting of Field names and Values) into a collection. If the collection does not exist a new one will be created.

The following example shows how this can be done.

db.Employee.insert

(

{

"Employeeid" : 1,

"EmployeeName" : "Martin"

}

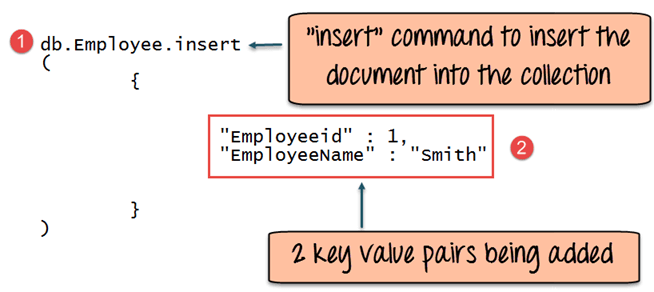
)

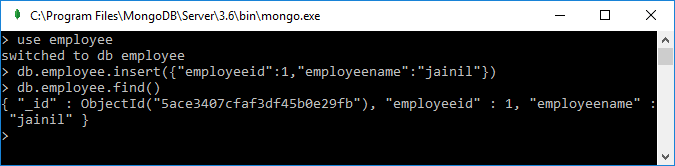
## Adding documents using insert() command

MongoDB provides the **insert () command** to insert documents into a collection. The following example shows how this can be done.

**Step 1)** Write the "insert" command

**Step 2)** Within the "insert" command, add the required Field Name and Field Value for the document which needs to be created.

[](https://cdn.guru99.com/images/MongoDB/112115_0607_Introductio5.png)



Practical 18:

**To create, modify, delete, execute and recompile a stored procedure in SQL Server/ MySQL**

