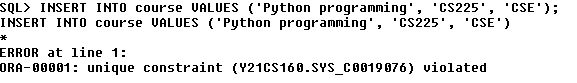
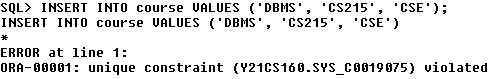
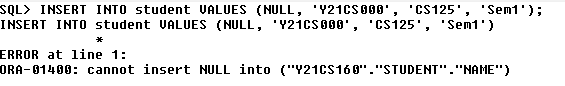
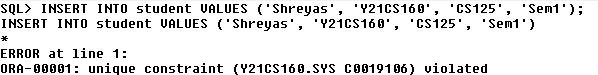
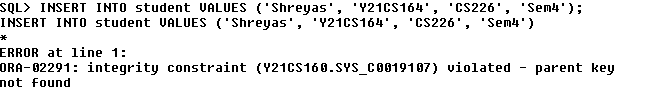
2. Write insert statements that violate the constraints specified in problem 1

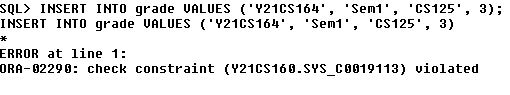




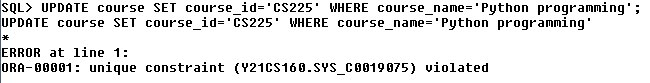


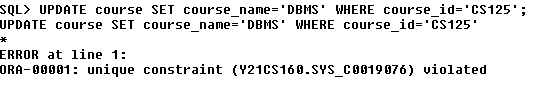


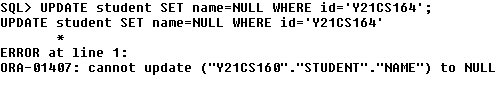


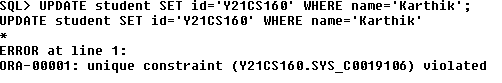


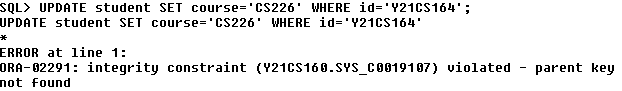
3. Write update statements that violate the constraints specified in problem 1.

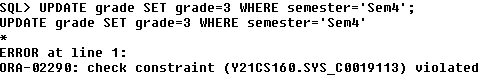
****

****

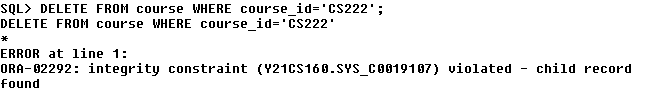
****

****

****

****

**4.** Write delete statements that violate the constraints specified in problem 1.

****

**5.** Drop the constraints specified in problem 1 using alter table statements.

****

****

****

****

****

****

**6.** Add the constraints specified in problem 1 using alter table statements.

****

****

****

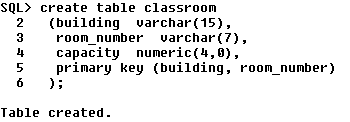
****

****

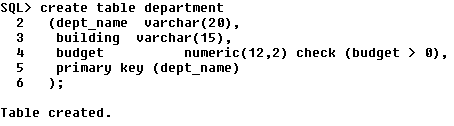
****

**Week 3**

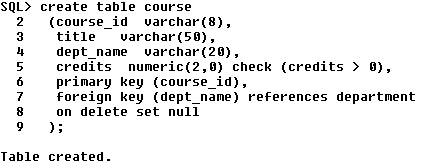
1. Create University database by copying it from other user



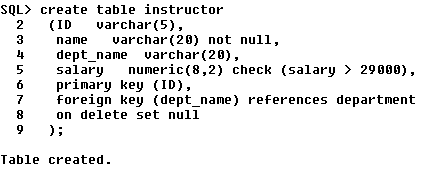




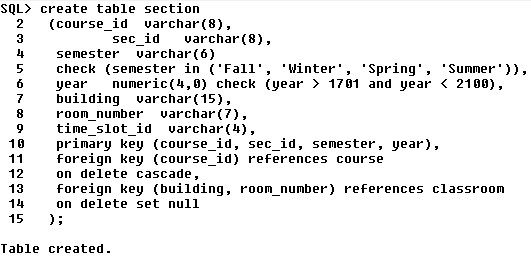




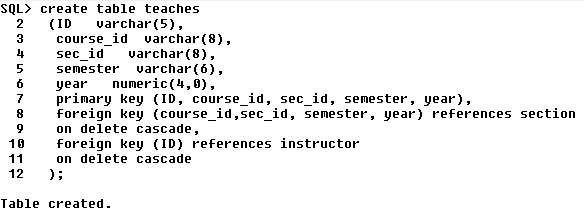




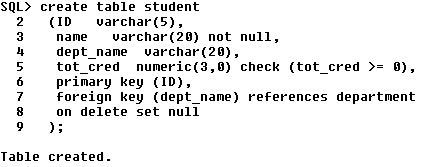




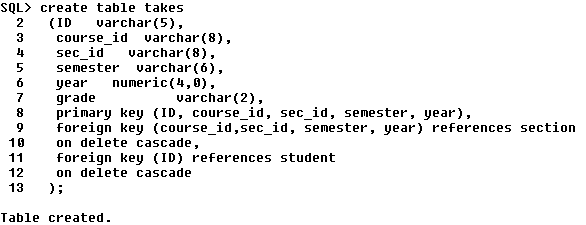




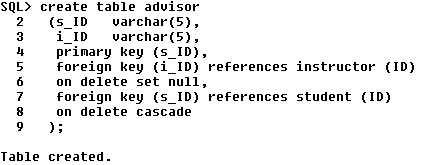




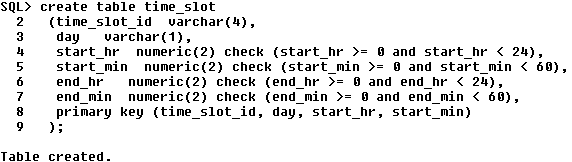




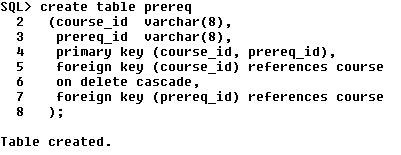






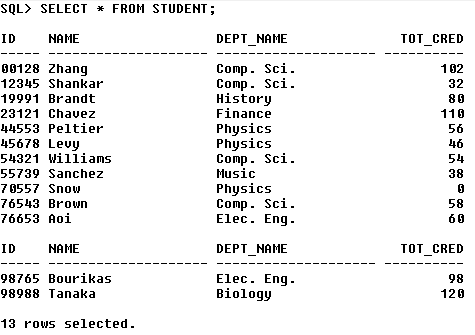




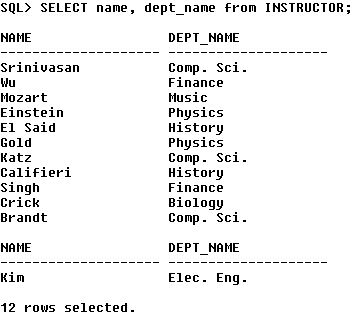




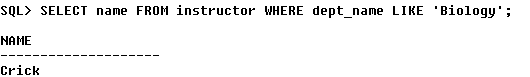
1. Write SQL Select Statements for the following simple queries that retrieve data from a single table:
   1. Find the details of all students



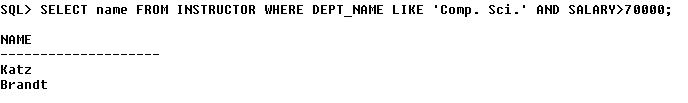
* 1. Find the department names of instructors



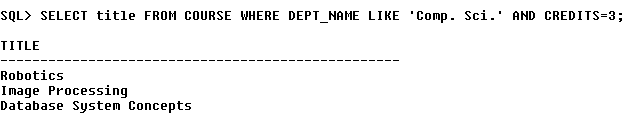
* 1. Find the names of all the instructors from Biology department



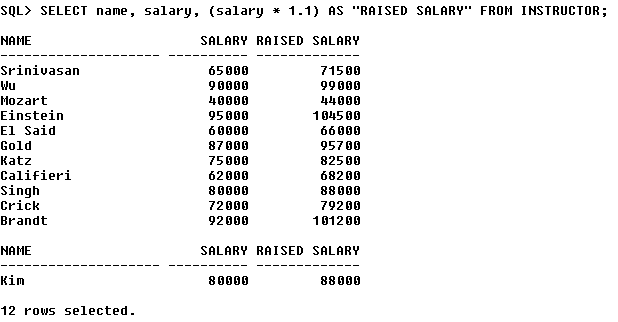
* 1. Find the names of all instructors in the Computer Science department who have salaries greater than $70,000.



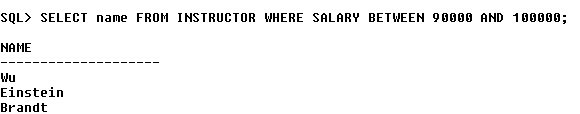
* 1. Find the names of courses in Computer science department which have 3 credits



* 1. Find the names of the instructors, their present salaries and the resulting salaries if they were given a 10% raise



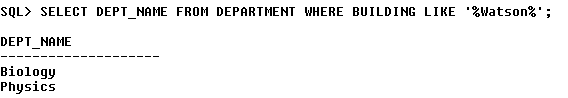
* 1. Find the names of instructors with salary amounts between $90,000 and $100,000,



* 1. Find all instructors whose salary is unknown



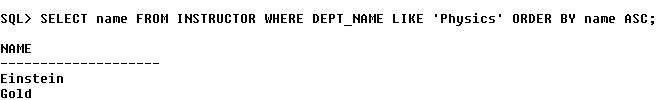
* 1. Find the names of all departments whose building name includes the substring ‘Watson’



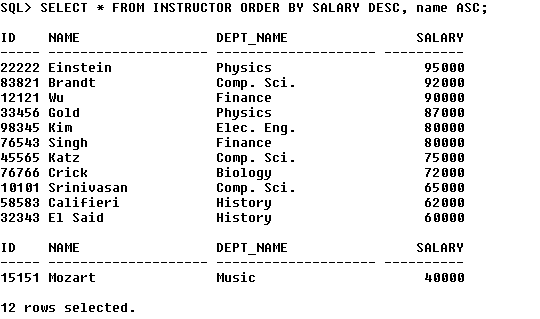
* 1. Find departments whose names contain the string “sci” as a substring, regardless of the case.



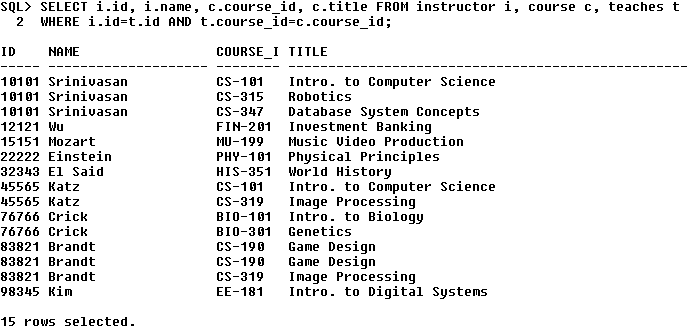
* 1. List the names of all instructors in the Physics department in alphabetic order



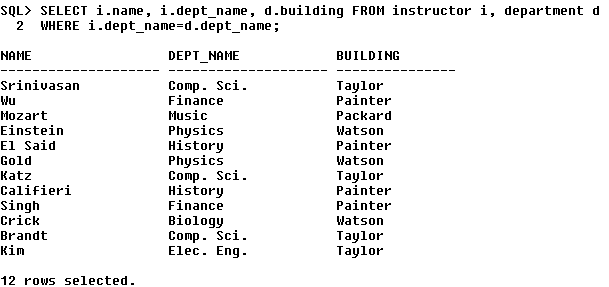
* 1. List the entire instructor relation in descending order of salary. If several instructors have the same salary, order them in ascending order by name



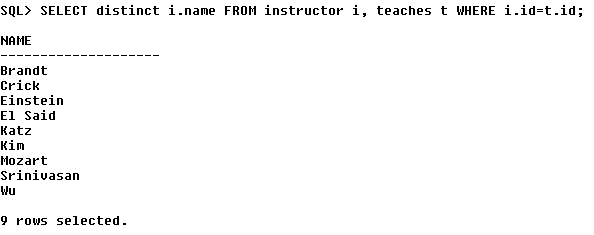
1. Write SQL queries for retrieving data from multiple tables using Joins:
   1. Find all possible combinations of instructors and the courses they teach



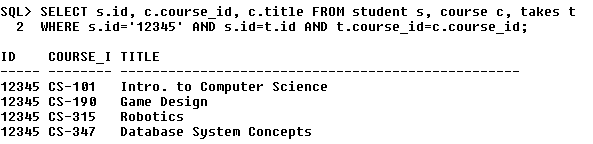
* 1. Retrieve the names of all instructors, along with their department names and department building name.



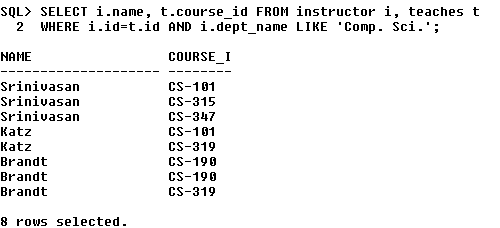
* 1. Find the names of instructors who have taught at least one course



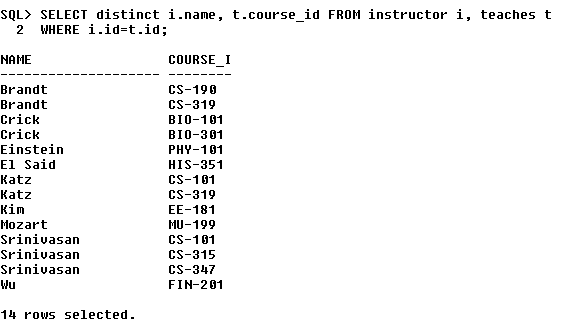
* 1. For the student with ID 12345 (or any other value), show all course\_id and title of all courses registered for by the student



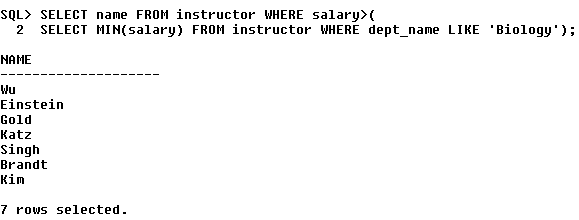
* 1. Find instructor names and course identifiers for instructors in the Computer Science department



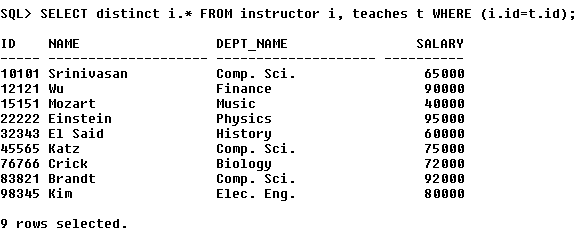
* 1. For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.



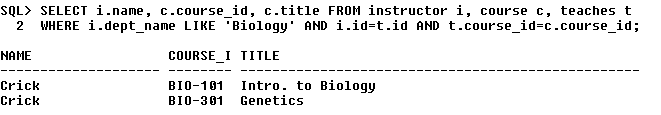
* 1. Find the names of all instructors whose salary is greater than at least one instructor in the Biology department. Or Find the names of all instructors who earn more than the lowest paid instructor in the Biology department.



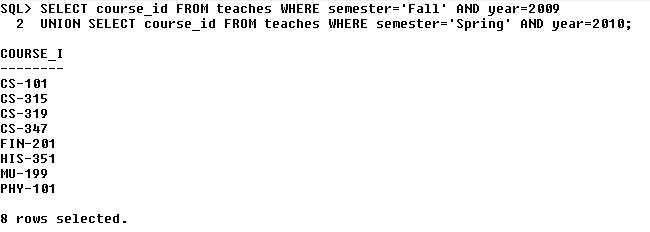
* 1. Find full details of instructors who teach at least one course.



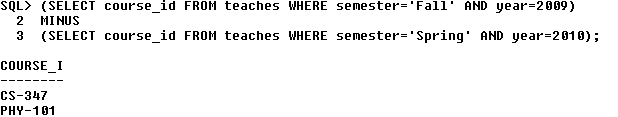
* 1. Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course



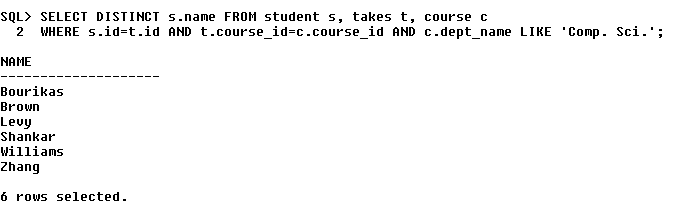
* 1. Find the set of all courses taught either in Fall 2009 or in Spring 2010, or both.



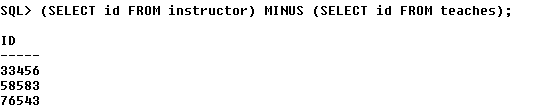
* 1. Find all courses taught in the Fall 2009 semester but not in the Spring 2010 semester



* 1. Find the names of all students who have taken any Comp. Sci. course ever. (there should be no duplicate names)

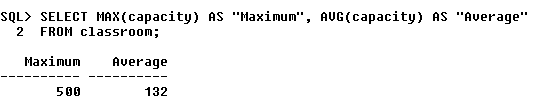


* 1. Display the IDs of all instructors who have never taught a course. (Don’t write nested query)

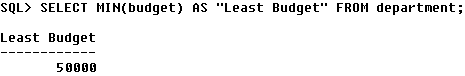


**Week 4**

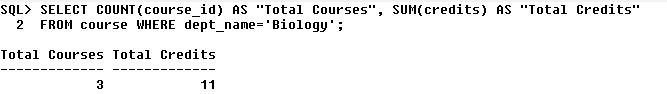
1. Write SQL Select Statements using Aggregate Functions, Group By and Having clauses for the following queries that retrieve data from university database
2. Find the maximum and average capacity of buildings in the university.



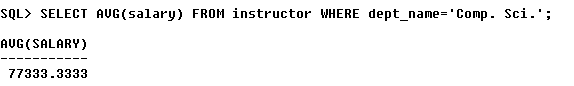
1. Display the least budget of the departments.



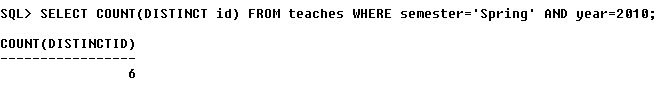
1. Find the total number of courses and credits offered by Biology department



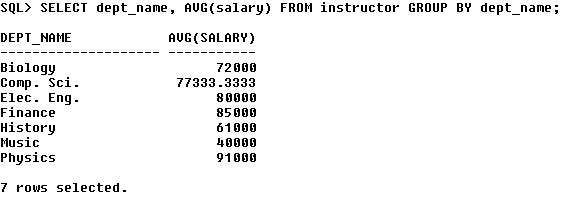
1. Find the average salary of instructors in the Computer Science department



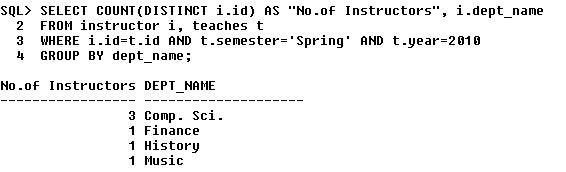
1. Find the total number of instructors who teach a course in the Spring 2010 semester



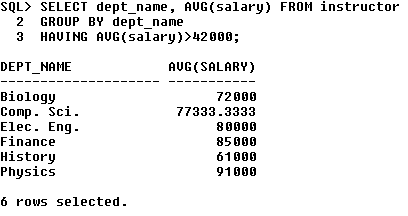
1. Find the average salary in each department.



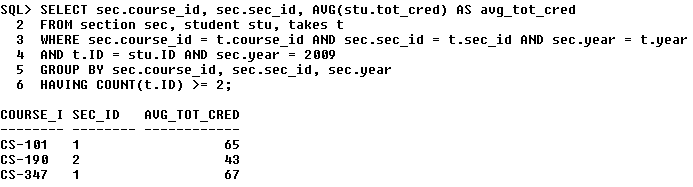
1. Find the number of instructors in each department who teach a course in the Spring 2010 semester.



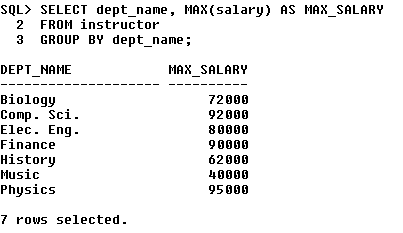
1. Find the department name and average salary of the department for only those departments where the average salary of the instructors is more than $42,000



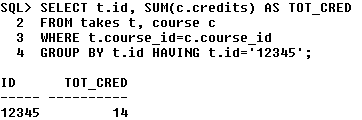
1. For each course section offered in 2009, find the average total credits (tot\_cred) of all students enrolled in the section, if the section had at least 2 students



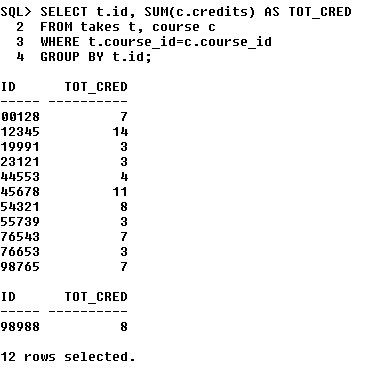
1. For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor



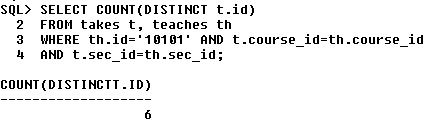
1. For the student with ID 12345 (or any other value), show the total number of credits scored for all courses (taken by that student). Don't display the tot\_creds value from the student table, you should use SQL aggregation on courses taken by the student



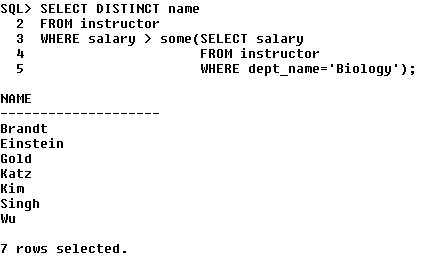
1. Display the total credits for each of the students, along with the ID of the student; don't bother about the name of the student. (Don't display the tot\_creds value from the student table, you should use SQL aggregation on courses taken by the student.)



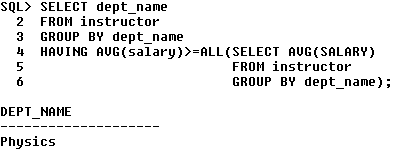
1. Write nested queries for answering the following queries that retrieve data from university database:
2. Find the total number of (distinct) students who have taken course sections taught by the instructor with ID 10101



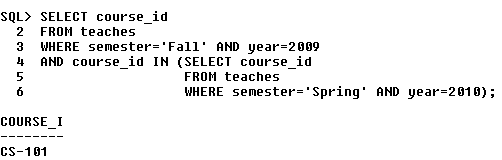
1. Find the names of all instructors whose salary is greater than at least one instructor in the Biology department



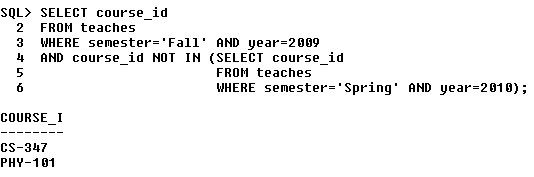
1. Find the departments that have the highest average salary



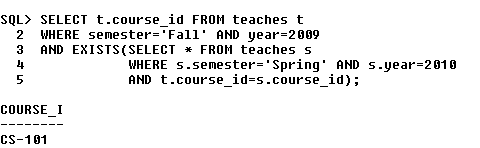
1. Find all the courses taught in the both the Fall 2009 and Spring 2010 semesters



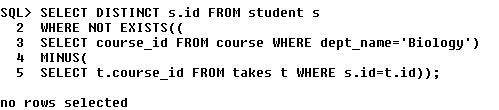
1. Find all the courses taught in the Fall 2009 semester but not in the Spring 2010 semester.



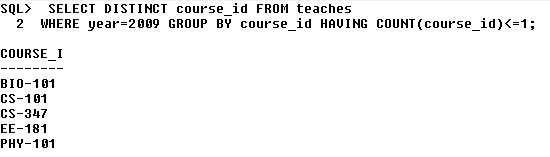
1. Find all courses taught in both the Fall 2009 semester and in the Spring 2010 semester. (Write correlated nested Query)



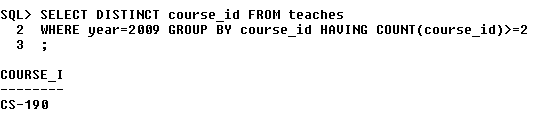
1. Find all students who have taken all courses offered in the Biology department. (Write Correlated nested Query)



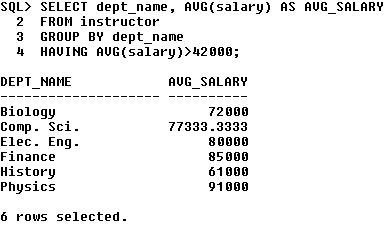
1. Find all courses that were offered at most once in 2009.



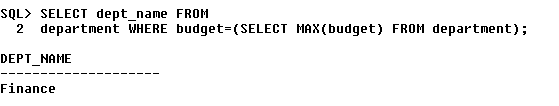
1. Find all courses that were offered at least twice in 2009”



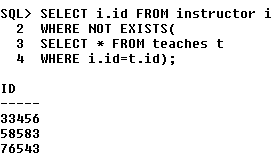
1. Find the average instructors’ salaries of those departments where the average salary is greater than $42,000.



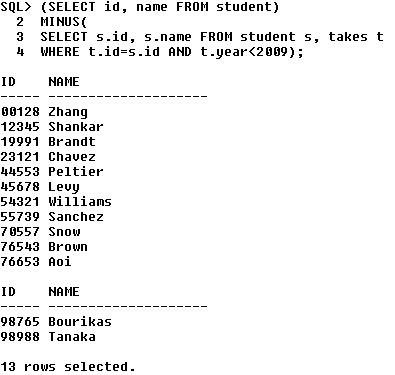
1. Find the departments with the maximum budget.



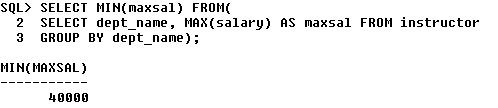
1. Find the names of instructors who have not taught any course.



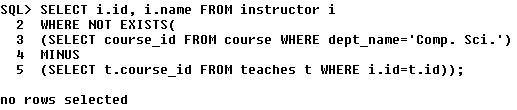
1. Find the IDs and names of all students who have not taken any course offering before Spring 2009



1. Find the lowest, across all departments, of the per-department maximum salary computed.



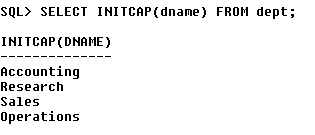
1. Display the IDs and names of the instructors who have taught all Comp. Sci. courses



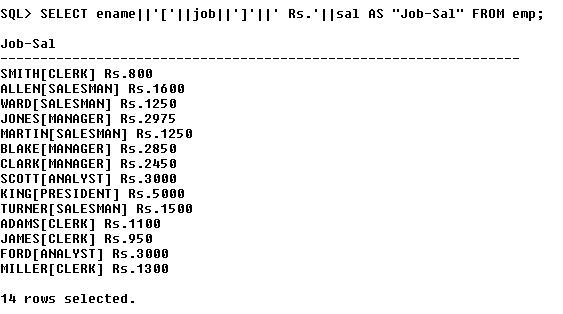
Lab Cycle – 2

Write Select statements for the following queries using SQL single row functions:

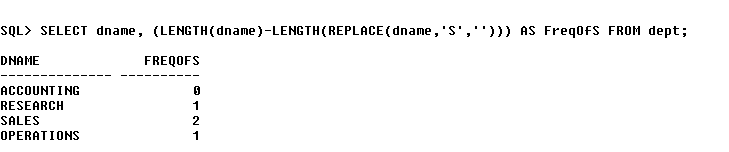
1. Display the department names in the lower case but the initial must be in uppercase.



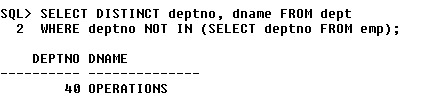
1. Determine the ‘ename’, ‘job’, ‘sal’ rename the title as Job-sal the output must be Job-Sal as SMITH [CLERK] RS.2000



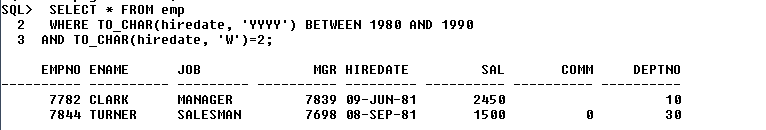
1. For each department, Count the number of times S occurs in department names.



1. Write a query to display the department name which does not contain any employees



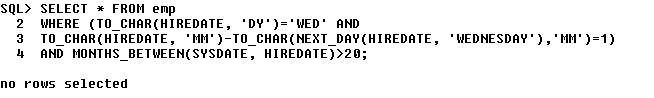
1. Write a query to display all employee details where employee was joined in year date wise 1980 & 1990 and 2nd week of every month.



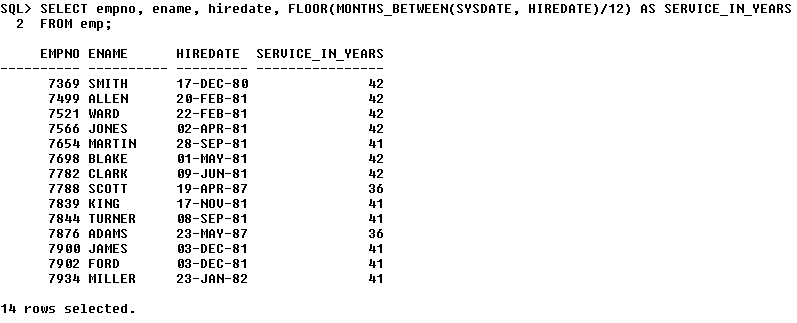
1. Write an SQL statement to convert the current date to new date picture, ex: MONDAY 10th JUNE 2005 10:30:00 PM



1. Write a query to display all employee details who joined last Wednesday of a month and experience should be greater than 20 months



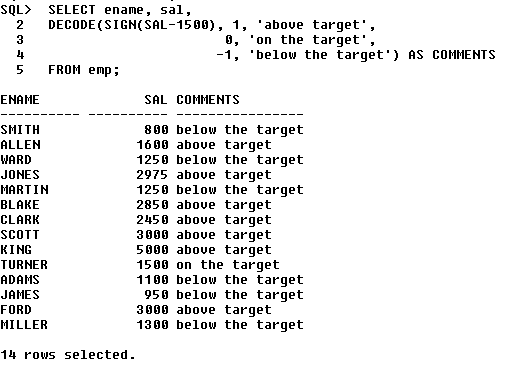
1. Write a query to calculate the service of employees rounded to years.



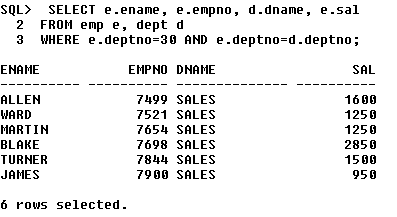
1. Write a query that will display a list of employees and their salary and the comments as follows: a. If the salary is more than 1500 then display “above target”

b. If the salary is equal to 1500 then display “on the target”

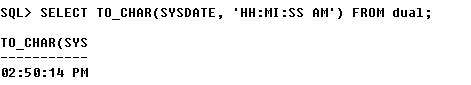
c. If the salary is less than 1500 then display “below the target”



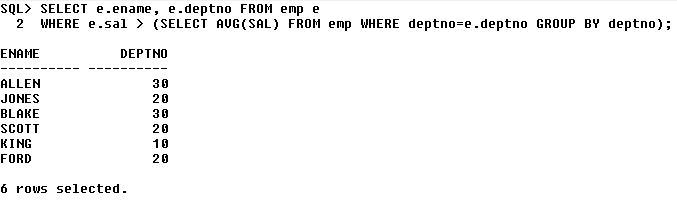
1. Display all employee names, employee number, department names & salary grades for all employees who are working in department 30.



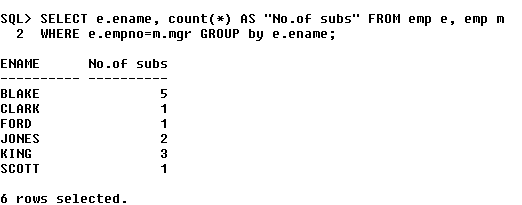
1. Display the time of day



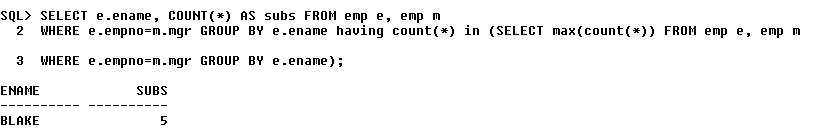
1. Find all employees who earn a salary greater than the average salary of their departments.



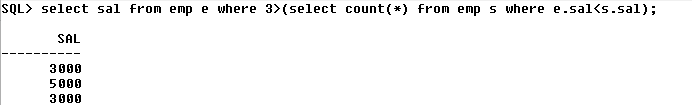
1. Write a query to find the name of the manager and number of sub-ordinates.



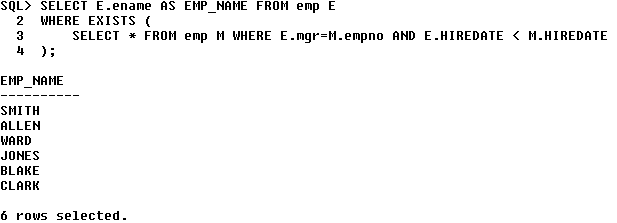
1. Write a query to find out the manager having Maximum number of sub-ordinates.



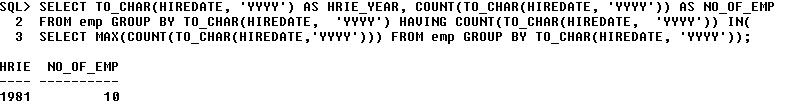
1. Write a query to find the top 3 earners



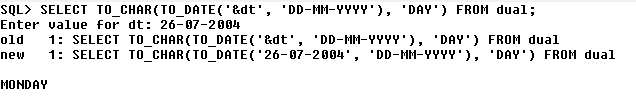
1. Write a query to find out the employees who have joined before their managers.



1. Write a query to find out the year, where most people join in the company displays the year and No. of Employees



1. Write a query which will return the DAY of the week.(ie. MONDAY), for any date entered in the format: DD.MM.YY



Cycle 3

1. Write a PL/SQL block to do the following:
   1. Read a number and print its multiplication table

declare

n number := &n;

i number;

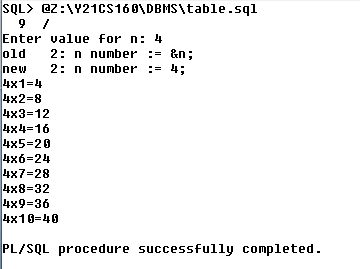
begin

for i in 1..10 loop

dbms\_output.put\_line(n||'x'||i||'='||(n\*i));

end loop;

end;



* 1. Read a number and check whether it is a palindrome or not

declare

n number := &n;

rev number := 0;

temp number := n;

begin

while temp<>0 loop

rev := (rev\*10)+mod(temp,10);

temp := trunc(temp/10);

end loop;

dbms\_output.put\_line('Reverse: '||rev);

if rev=n then

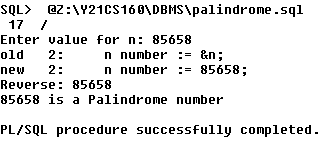
dbms\_output.put\_line(n||' is a Palindrome number');

else

dbms\_output.put\_line(n||' is not a Palindrome number');

end if;

end;



* 1. Read a number n, and print the first n Fibonacci numbers

declare

a number := 0;

b number := 1;

c number;

n number := &n;

i number;

begin

dbms\_output.put\_line('Fibonacci series: ');

dbms\_output.put\_line(a);

dbms\_output.put\_line(b);

for i in 3..n loop

c := a+b;

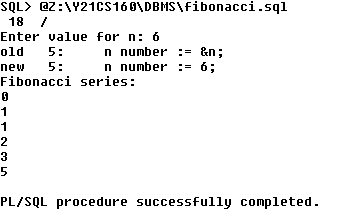
dbms\_output.put\_line(c);

a := b;

b := c;

end loop;

end;



1. Write a PL/SQL block which will accept two numbers and find out their LCM and HCF. The output should be stored in a table called **DEMO\_TAB**.

declare

n1 DEMO\_TAB.Num1%type := &n1;

n2 DEMO\_TAB.Num2%type := &n2;

ma number;

mi number;

gcd DEMO\_TAB.HCF%type := 1;

lcm DEMO\_TAB.LCM%type := 1;

i number;

r DEMO\_TAB%ROWTYPE;

begin

if n1>n2 then

ma := n1;

mi := n2;

else

mi := n1;

ma := n2;

end if;

for i in REVERSE 1..mi loop

if MOD(mi, i)=0 AND MOD(ma, i)=0 then

gcd := i;

EXIT;

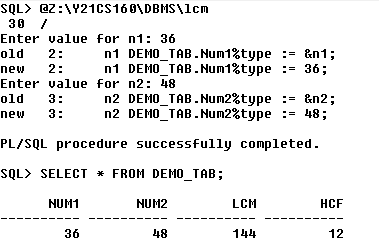
end if;

end loop;

lcm := (n1\*n2)/gcd;

INSERT INTO DEMO\_TAB VALUES(n1, n2, lcm, gcd);

end;



1. Consider the following relation schemas

Inventory

|  |  |  |
| --- | --- | --- |
| Product\_ID | Product\_name | Quantity |

Purchase\_Record

|  |  |  |
| --- | --- | --- |
| Product\_ID | Status | Date |

Write a PL/SQL block to read the quantity of a product from inventory and if it is > 0 reduce the quantity by 1 and record the status of purchase of that product as ‘PURCHASED’. Otherwise record the status of purchase of that product as ‘OUT OF STOCK’. While recording the status of a purchase, record the date of transaction.

declare

prodId Inventory.Product\_Id%type := &prodId;

qnt Inventory.Quantity%type;

begin

SELECT Quantity INTO qnt FROM Inventory WHERE Product\_Id=prodId;

if sql%found then

if qnt > 0 then

INSERT INTO Purchase\_Record VALUES(prodId, 'PURCHASED', SYSDATE);

UPDATE Inventory SET Quantity=Quantity-1 WHERE Product\_Id=prodId;

else

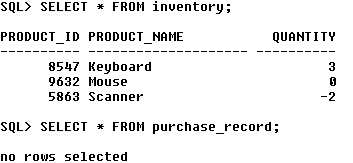
INSERT INTO Purchase\_Record VALUES(prodId, 'OUT OF STOCK', SYSDATE);

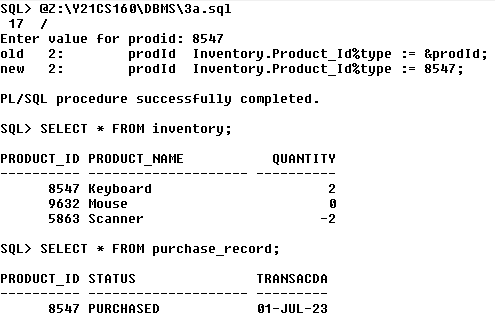
end if;

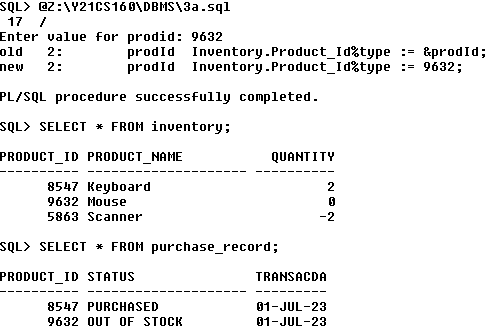
end if;

COMMIT;

end;







1. Write a PL/SQL block to handle the following built-in exceptions

***no\_data\_found***

***too\_many\_rows***

***zero\_divide***

declare

empno emp.empno%type;

empD emp%ROWTYPE;

n number := 10;

a number;

begin

SELECT empno INTO empno FROM emp WHERE deptno=5560;

SELECT \* INTO empD FROM emp;

a := n / 0;

exception

when no\_data\_found then

dbms\_output.put\_line('No data foud with given constraints');

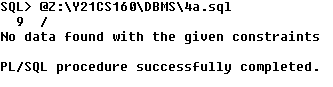
when too\_many\_rows then

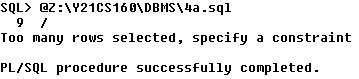
dbms\_output.put\_line('Too many rows selected, specify a constraint');

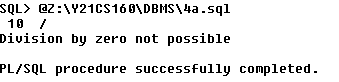
when zero\_divide then

dbms\_output.put\_line('Division by zero not possible');

end;







1. Write a PL/SQL block to check whether the quantity of any product in Inventory table of problem 3 is < 0. If so, using an exception display relevant message and update quantity to 0.

declare

pid inventory.product\_id%type;

invalid\_qty exception;

begin

SELECT product\_id INTO pid FROM inventory WHERE Quantity<0;

if sql%found then

raise invalid\_qty;

end if;

exception

when no\_data\_found then

dbms\_output.put\_line('No data found with given constraint');

when too\_many\_rows then

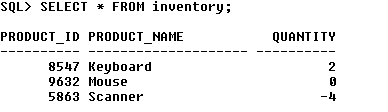
dbms\_output.put\_line('Too many rows select');

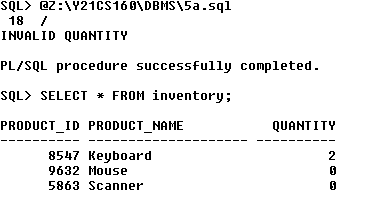
when invalid\_qty then

dbms\_output.put\_line('INVALID QUANTITY');

UPDATE inventory SET Quantity=0 WHERE product\_id=pid;

end;





1. Consider the following relation schemas

Emp

|  |  |  |  |
| --- | --- | --- | --- |
| Empid | name | salary | dno |

Del\_History

|  |  |  |
| --- | --- | --- |
| Dno | Rows\_deleted | date |

Write a PL/SQL block to delete records of all employees who belong to a particular department and then record the dno, no of rows deleted and date on which deletion occurred in the Del\_History table.

declare

dnum emp1.dno%type := &dno;

c number;

begin

DELETE FROM emp1 WHERE dno=dnum;

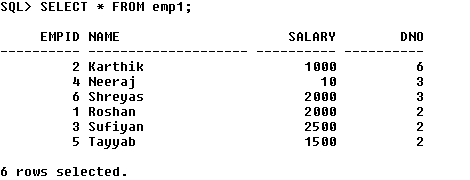
if sql%found then

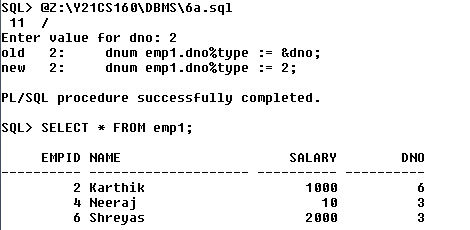
c := sql%rowcount;

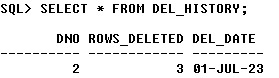
INSERT INTO del\_history VALUES(dnum, c, SYSDATE);

end if;

end;







1. Consider the following relation schemas

Bank\_Main

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acc\_no | Name | Address | Acc-type | Curr\_balance |

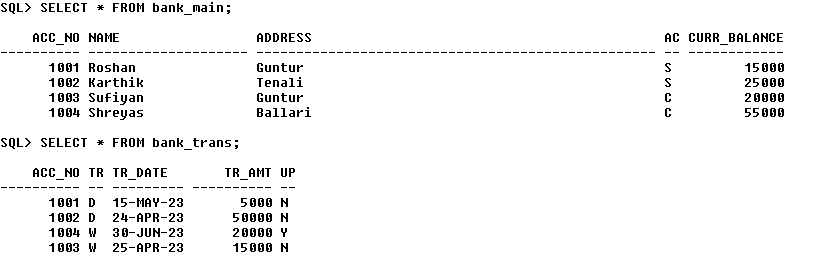
Bank\_Trans

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acc\_no | Tr\_type | Tr\_date | Tr\_amt | Upd\_flag |

Where acc\_type is S – savings C- current

Tr\_type is D – deposit W – withdrawal

Write a PL/SQL block to update master table’s (i.e., Bank\_Main) curr\_balance field with transaction details from transaction file (i.e., Bank\_Trans) and also change the status of the Upd\_flag field of transaction table to ‘Y’ once the updation is complete



declare

anum bank\_trans.acc\_no%type;

tt bank\_trans.tr\_type%type;

amt bank\_trans.tr\_amt%type;

cursor bank\_cursor is SELECT acc\_no, tr\_type, tr\_amt FROM bank\_trans WHERE upd\_flag='N';

begin

open bank\_cursor;

loop

fetch bank\_cursor into anum, tt, amt;

exit when bank\_cursor%notfound;

if tt='D' then

UPDATE BANK\_MAIN SET Curr\_balance=Curr\_balance+amt WHERE acc\_no=anum;

elsif tt='W' then

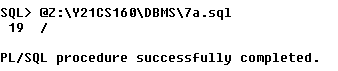
UPDATE BANK\_MAIN SET Curr\_balance=Curr\_balance-amt WHERE acc\_no=anum;

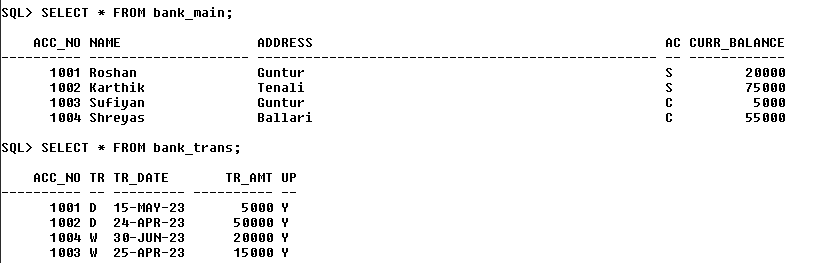
end if;

UPDATE BANK\_TRANS SET UPD\_FLAG='Y' WHERE acc\_no=anum AND tr\_type=tt;

end loop

end;





**CYCLE – 4**

1. Write a PL/SQL stored function to calculate nCr usingfact (m) function that returns factorial of m.

CREATE OR REPLACE FUNCTION fact(n number) RETURN number AS f number:= 1;

i number;

begin

    for i in 1..n loop

        f := f\*i;

    end loop;

    return f;

end;

CREATE OR REPLACE FUNCTION ncr(n number, r number) RETURN number AS

val number;

begin

    val := fact(n)/(fact(n-r)\*fact(r));

    return val;

end;

declare

    n number := &n;

    r number := &r;

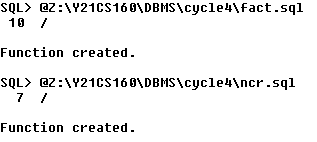
    value number;

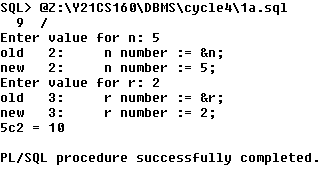
begin

    value = ncr(n, r);

    dbms\_output.put\_line(n||'c'||r||' = '||value);

end;





1. Write a PL/SQL block that updates salary of an employee in employee table by using incr function which takes employee number as argument, calculates increment and returns increment based on the following criteria.

If salary <= 3000 – increment = 30% of salary

If salary > 3000 and <= 6000– increment = 20% of salary

Else increment = 10% of salary.

CREATE OR REPLACE FUNCTION incr(n number) RETURN NUMBER AS

hike number;

begin

    SELECT sal INTO hike FROM emp2 WHERE empno=n;

    if hike < 3000 then

        hike := 0.3 \* hike;

    elsif hike > 3000 AND hike < 6000 then

        hike := 0.2 \* hike;

    else

        hike := 0.1 \* hike;

    end if;

    return hike;

end;

declare

    enum    emp2.empno%type := &enum;

    hike     number;

begin

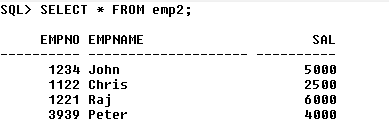
    hike := incr(enum);

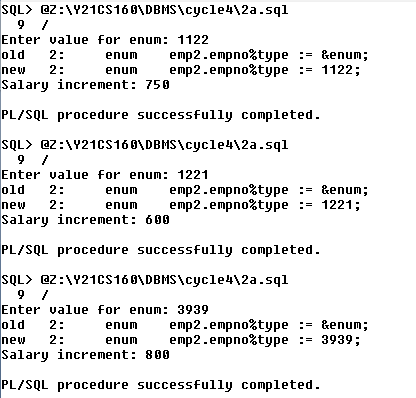
    dbms\_output.put\_line('Salary increment: '|| hike);

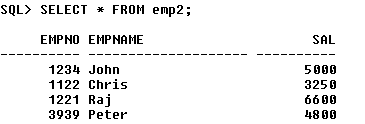
    UPDATE emp2 SET sal = sal+hike WHERE empno=enum;

end;









1. Write a stored procedure, raise\_salary which accepts an employee number. It uses incr function of previous program to get the salary increase amount and uses employee number to select the current salary from employee table. If employee number is not found or if the current salary is null, it should raise an exception. Otherwise, updates the salary.

CREATE OR REPLACE PROCEDURE raiseSal(n number) AS

    s number;

    a number;

    exc exception;

begin

    s := incr(n);

    SELECT sal INTO a FROM emp2 WHERE empno = n;

    if a is null then raise exc;

    else

        UPDATE emp2 SET sal=sal+s WHERE empno = n;

    end if;

    exception

        when no\_data\_found then

            dbms\_output.put\_line('Employee does not exist');

        when exc then

            dbms\_output.put\_line('Salary is null');

end;

declare

    n number := &n;

    r emp2%ROWTYPE;

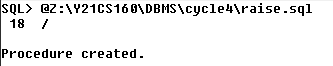
begin

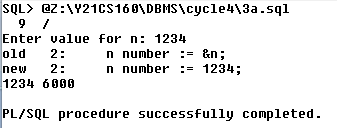
    raiseSal(n);

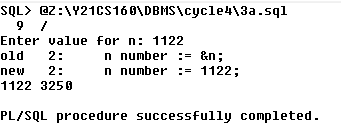
    SELECT \* INTO r FROM emp2 WHERE empno=n;

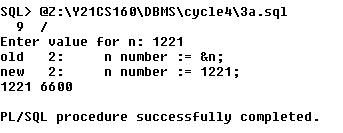
    dbms\_output.put\_line(r.empno||' '||r.sal);

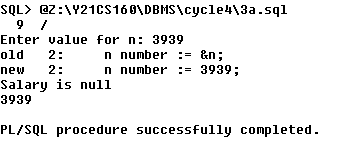
end;











1. Consider the following table

|  |  |  |  |
| --- | --- | --- | --- |
| prod\_id | name | mat\_used | weight |

ITEM

Write a procedure which will check for the existence of prod\_id from the table ITEM. This procedure must have two arguments, one of which will receive a value, which is a matching pattern for prod\_id of the ITEM table, and another which will return a value indicating whether a match has been found or not.

CREATE OR REPLACE PROCEDURE match(n in number, s out varchar) AS

    a item.prod\_id%type;

begin

    SELECT prod\_id INTO a FROM item WHERE prod\_id=n;

    s := 'match found';

    exception

      when no\_data\_found then

        s := 'no match found';

end;

declare

    n number := &pid;

    s varchar(12);

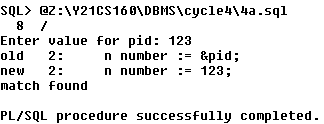
begin

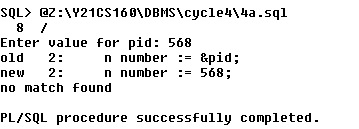
    match(n, s);

    dbms\_output.put\_line(s);

end;







1. Consider the following tables

|  |  |  |
| --- | --- | --- |
| EMPNO | NAME | AGE |

PERSINFO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMPNO | NAME | AGE | OPERATION | ODATE |

AUDITPERSINFO

PERSINFO is the table for which the auditing must be performed and AUDITPERSINFO is the table which keeps track of the records deleted or modified. Create a database trigger audit\_trial. This trigger is forced when an UPDATE or a DELETE is performed on the table PERSINFO. It first checks for the operation being performed on the table. Then depending on the operation, a variable (that corresponds to operation) is assigned the value ‘UPDATE’ or ‘DELETE’ and then inserts the updated/deleted record into AUDITPERSINFO.

CREATE OR REPLACE TRIGGER audit\_trg

AFTER UPDATE OR DELETE ON PERSINFO

FOR EACH ROW

declare

    eno number;

    ename varchar(20);

    eage number;

    op varchar2(12);

begin

    if UPDATING then op := 'UPDATE';

    elsif DELETING  then op := 'DELETE';

    end if;

    eno := :old.empno;

    ename := :old.name;

    eage := :old.age;

    INSERT INTO AUDITPERSINFO VALUES(eno, ename, eage, op, SYSDATE);

end;



