# CDBM 280 – Database Management Systems

## Final Assessment

Name:\_\_\_\_\_\_\_\_\_\_\_Jaydeep\_\_Patel\_\_\_\_\_\_\_\_\_\_\_\_\_\_

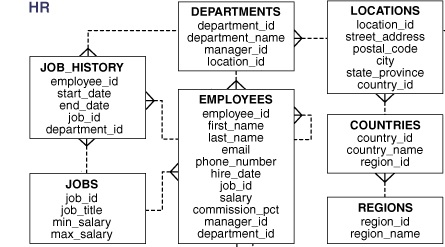
Total Marks: \_\_\_\_\_ / 55

**Read the following carefully:**

* You have **3 hours** to complete the exam.
* Each question has the marks indicated to the left of the question.
* Create a word document with your name as the filename and add a location within it for each question in the exam.
* Add the appropriate answer to each question to your word document
* Save often!
* Submit the finished document to the CDBM280 drop box titled “Final Examination”
* You may use Brightspace content and notes you have gathered.
* DO NOT access any other online resources.

**VIEWS**

[4] Given the following tables and view created from the tables, answer the following:



CREATE OR REPLACE VIEW DetailsView

AS

SELECT j.job\_id, j.job\_title, e.employee\_ID, e.last\_name, e.first\_name,

e.salary, d.department\_id, d.department\_name, d.manager\_id

FROM Employees e JOIN Jobs j

ON e.job\_id = j.job\_id

JOIN Departments d

ON e.department\_id = d.department\_id;

* 1. What columns can be updated/inserted/deleted and why?

Here, employees table is key preserved table. Because The key preserved table will generally be on the “Many” side of a one-to-many relationship. Here all the columns coming from employee can be updated/inserted/deleted.

* e.employee\_ID, e.last\_name, e.first\_name, e.salary.

(You can **insert** into a view based on multiple tables if Oracle can determine the proper rows to insert. In a multitable view, Oracle determines which of the tables are *key-preserved.* and here employee table is key preserved)

* 1. What columns cannot be updated/inserted/deleted and why?

In the given view, j.job\_id, j.job\_title, d.department\_id, d.department\_name, d.manager\_id. These columns cannot be updated/inserted/deleted.

Rows from these columns will be repeated in the result view. It also means that every key from these table will not be a key of the resulting join. Also these tables are on the “one” side of a one-to-many relationship.

**QUERIES**

* **Note: use no subqueries in the SELECT or FROM clauses (they can be used in WHERE and HAVING), and use no views for this section**. Do not use functions that weren’t covered in class.
* Use aliases for fields as appropriate.

1. Using the tables from the **HR** schema, provide the SQL statements to solve the following. (Marks as indicated in [ ] for each part.)

**The HR schema exists outside your user login – therefore when referring to tables you must use HR.tablename, for example:**

**SELECT \* FROM HR.Employees;**

* 1. [4] Display the job title in uppercase, start date (listing the full month name, two digit date, a comma, and four digit year), and employee id where the end date is from January 1 1998 to the end of December 1999. Sort by start date ascending.

|  |
| --- |
| SELECT UPPER(j.job\_title) AS TITLE, to\_char(jh.start\_date, 'Month DD,YYYY') AS STARTDAY, e.employee\_id  FROM HR.jobs J JOIN HR.employees E ON j.job\_id = E.job\_id  JOIN HR.job\_history JH ON e.job\_id = JH.job\_id  WHERE jh.end\_date between '1-JAN-98' and '31-DEC-99'  ORDER BY jh.start\_date ; |
|  |

* 1. [7] Find the highest commission percent for employees not managed by Steven King that is smaller than the lowest commission percent of those employees that are managed by Steven King.

|  |
| --- |
| SELECT MAX(emp.commission\_pct)  FROM HR.Employees emp where emp.manager\_id <> 100  AND emp.commission\_pct < (SELECT MAX(mgr.commission\_pct) FROM HR.Employees mgr where manager\_id = 100 ); |
|  |

* 1. [4] List all country names and the number of locations for each in the form “<Country> has <##> locations”.

|  |
| --- |
| SELECT \* FROM HR.Employees e;  SELECT C.country\_name || ' has ' || COUNT(L.location\_id) || ' locations ' AS Locations\_OF\_Countries  FROM HR.Countries C  LEFT JOIN HR.Locations L ON C.country\_ID = L.country\_ID  GROUP BY C.country\_name; |
|  |

* 1. [ 5] Determine the employee last names, salary, and job\_id for employees who earn a salary more than 30% higher than the average salary for employees with the same manager.

|  |
| --- |
| SELECT E.last\_name, E.salary, E.job\_id  FROM HR.Employees E  WHERE E.salary > (SELECT AVG(salary)  FROM HR.Employees E  WHERE E.manager\_id = E.manager\_id)\* 1.3; |
|  |

1. [ 6 ] Generate an entry in the plan table for the following query: find all of the employees who earn both the less than the average salary and also have the lowest commission rate. Display their first name, last name, salary, and commission percent. Sort by last name. The generated plan should be named BROKE\_SALESPEOPLE.

|  |
| --- |
| SELECT \* FROM PLAN\_TABLE;  EXPLAIN PLAN SET STATEMENT\_ID = 'BROKE\_SALESPEOPLE' FOR  SELECT e.first\_name, e.last\_name, e.salary, e.commission\_pct  FROM HR.EMPLOYEES E  WHERE e.salary < (SELECT AVG(E.SALARY)  FROM HR.EMPLOYEES E)  AND e.commission\_pct = (SELECT MIN(e.commission\_pct)  FROM HR.EMPLOYEES E)  ORDER BY E.LAST\_NAME ASC; |
|  |

**PL/SQL**

1. [7 ] Create a PL/SQL function called Manager\_Emps that receives as a parameter the manager ID and returns the names of every employee who reports to that manager. The names must be returned as a string with commas separating names.

|  |
| --- |
| create or replace function Manager\_Emps(managerID char)  return varchar2  is  v\_employees varchar2(1000);  cursor employee\_cur is  select emp.first\_name from HR.Employees emp  JOIN HR.Employees mgr on emp.manager\_id = mgr.employee\_id  where emp.manager\_id = managerID;  begin  for employeerecord in employee\_cur  loop  v\_employees := v\_employees || ', ' || employeerecord.first\_name;  end loop;  -- remove extra , at the beginning  return ltrim(v\_employees,', ');  end;  /  select DISTINCT Manager\_Emps(HR.Employees.employee\_id) FROM HR.Employees ; |

|  |
| --- |
|  |

1. [3] Write a select query that will display the manager\_id and also use the function from #1 to display the employees for that manager id number. Manager ids should only appear once. Sort by manager id.

|  |
| --- |
| select DISTINCT emp.manager\_id ,Manager\_Emps(manager\_id) FROM HR.Employees emp ; |

|  |
| --- |
|  |

1. [ 15 ] Write an anonymous PL/SQL to display a report as follows.
   1. For each country, ordered by name, display :

"Country: <country\_id>: <country\_name>: list of employee first names"

“Total Employees for <country\_name> is X”

* 1. You MUST use at least one For Loop in your solution.
  2. Sample output:

|  |
| --- |
| DECLARE  country\_tempid VARCHAR2(1024);  vemployee VARCHAR2(1024);  ncount NUMBER := 0;  CURSOR country\_cur IS SELECT \* FROM HR.countries;  CURSOR employee\_cur IS  SELECT \*  FROM HR.Employees e  JOIN HR.Departments d on d.department\_id = e.department\_id  JOIN HR.locations loc on d.location\_id = loc.location\_id  Right JOIN HR.countries coun on loc.country\_id = coun.country\_id  WHERE loc.country\_id = country\_tempid  ORDER BY coun.country\_name;  BEGIN  FOR country in country\_cur LOOP  DBMS\_OUTPUT.PUT('Country: ' || country.country\_id || ': ' || country.country\_name || ': ');  country\_tempid := country.country\_id;  FOR employee in employee\_cur LOOP  vemployee := vemployee || employee.first\_name || ', ';  ncount := ncount + 1;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vemployee, 1, LENGTH(vemployee)-2));  DBMS\_OUTPUT.PUT\_LINE('Total employees for ' || country.country\_name || ' is ' || ncount);  vemployee := '';  ncount := 0;  END LOOP;  END; |
|  |

Sample output (note: some countries are omitted in this sample, there are many in the table):

Country: AR: Argentina:

Total Employees for Argentina is 0

Country: AU: Australia:

Total Employees for Australia is 0

Country: BE: Belgium:

Total Employees for Belgium is 0

.

.

.

Country: UK: United Kingdom: Susan, John, Karen, Alberto, Gerald, … (continues)

Total Employees for United Kingdom is 35

.

.

Country: ZW: Zimbabwe:

Total Employees for Zimbabwe is 0

