**Cycle 3**

1. Create a view called EMPLOYEES\_VU based on the employee numbers, employee names, and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.
2. Display the contents of the EMPLOYEES\_VU view.
3. Select the view name and text from the USER\_VIEWS data dictionary view.
4. Using your EMPLOYEES\_VU view, enter a query to display all employee names and department numbers.
5. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. Label the view columns EMPNO, EMPLOYEE, and DEPTNO. Do not allow an employee to be reassigned to another department through the view.
6. Display the structure and contents of the DEPT50 view.
7. Attempt to reassign Matos to department 80.
8. Create a view called SALARY\_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the EMPLOYEES, DEPARTMENTS, and JOB\_GRADES tables. Label the columns Employee, Department, Salary, and Grade, respectively.
9. Create a sequence to be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1000. Have your sequence increment by ten numbers. Name the sequence DEPT\_ID\_SEQ.
10. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script lab12\_2.sql. Run the statement in your script.
11. Write a script to insert two rows into the DEPT table. Name your script lab12\_3.sql. Be sure to use the sequence that you created for the ID column. Add two departments named Education and Administration. Confirm your additions. Run the commands in your script.
12. Create a nonunique index on the foreign key column (DEPT\_ID) in the EMP table.
13. Display the indexes and uniqueness that exist in the data dictionary for the EMP table. Save the statement into a script named lab12\_5.sql.
14. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.
15. Query all the rows in your DEPARTMENTS table.
16. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team’s table.
17. Create a synonym for the other team’s DEPARTMENTS table.
18. Query all the rows in the other team’s DEPARTMENTS table by using your synonym.

**Cycle 4**

**Exercises on the HR schema: PL/SQL programs**

1. Write a program to interchange the salaries of employee 120 and 122.
2. Increase the salary of employee 115 based on the following conditions:If experience is more than 10 years, increase salary by 20%If experience is greater than 5 years, increase salary by 10%Otherwise 5% Case by Expression.
3. Change commission percentage as follows for employee with ID = 150.If salary is more than 10000 then commission is 0.4%, if Salary is less than 10000 but experience is more than 10 years then 0.35%, if salary is less than 3000 then commission is 0.25%.In the remaining cases commission is 0.15%.
4. Find out the name of the employee and name of the department for the employee who is managing for employee 103.
5. Display missing employee IDs.
6. Display the year in which maximum number of employees joined along with how many joined in each month in that year.
7. Change salary of employee 130 to the salary of the employee with first name ‘Joe’. If Joe is not found then take average salary of all employees. If more than one employee with first name ‘Joe’ is found then take the least salary of the employees with first name Joe.
8. Display Job Title and Name of the Employee who joined the job first day.
9. Display 5th and 10th employees in Employees table.
10. Update salary of an employee based on department and commission percentage. If department is 40 increase salary by 10%. If department is 70 then 15%,if commission is more than .3% then 5% otherwise 10%.
11. Create a function that takes department ID and returns the name of the manager of the department.
12. Create a function that takes employee ID and return the number of jobs done by the employee in the past.
13. Create a procedure that takes department ID and changes the manager ID for the department to the employee in the department with highest salary. (Use Exceptions).
14. Create a function that takes a manager ID and return the names of employees who report to this manager. The names must be returned as a string with comma separating names.
15. Ensure no changes can be made to EMPLOYEES table before 6am and after 10pm in a day.
16. Create a Trigger to ensure the salary of the employee is not decreased.
17. Create a trigger to ensure the employee and manager belongs to the same department.
18. Whenever the job is changed for an employee write the following details into job history.Employee ID, old job ID, old department ID, hire date of the employee for start date, system date for end date.But if a row is already present for employee job history then the start date should be the end date of that row +1.