1. SELECT FIRST\_NAME||' '||LAST\_NAME AS EMPNAME,SALARY AS MONTHLY\_PAY,SALARY\*12 AS ANNUAL\_PAY,DEPARTMENT\_NAME AS DNAME FROM DEPARTMENTS,EMPLOYEES WHERE DEPARTMENTS.DEPARTMENT\_ID=EMPLOYEES.DEPARTMENT\_ID ORDER BY SALARY DESC;

2. SELECT FIRST\_NAME||' '||LAST\_NAME AS EMPNAME , SALARY\*NVL(COMMISSION\_PCT,0)/100 + SALARY AS MONTHLY\_PAY, (SALARY\*NVL(COMMISSION\_PCT,0)/100+SALARY)\*12 AS ANNUAL\_PAY , DEPARTMENT\_NAME AS DNAME FROM DEPARTMENTS,EMPLOYEES WHERE DEPARTMENTS.DEPARTMENT\_ID = EMPLOYEES.DEPARTMENT\_ID ORDER BY SALARY DESC;

3. SELECT FIRST\_NAME||' '||LAST\_NAME AS EMPNAME,TO\_CHAR(SYSDATE,'YYYY')-TO\_CHAR( HIRE\_DATE,'YYYY') AS EXPERIENCE\_YRS FROM EMPLOYEESORDER BY EXPERIENCE\_YRS DESC;

4. SELECT EMPLOYEE\_ID ,FIRST\_NAME||' '||LAST\_NAME AS EMPNAME , SALARY AS OLD\_SALARY,(SALARY\*1.1) AS NEW\_SALARY,DEPARTMENT\_ID FROM EMPLOYEES WHERE DEPARTMENT\_ID = 90;

5. SELECT DEPARTMENT\_ID,DEPARTMENT\_NAME,TOTAL\_SALARY FROM (SELECT E.DEPARTMENT\_ID AS DEPT\_ID,SUM(SALARY) AS TOTAL\_SALARY FROM EMPLOYEES E GROUP BY E.DEPARTMENT\_ID HAVING SUM(SALARY)>100000),DEPARTMENTS WHERE DEPARTMENTS.DEPARTMENT\_ID=DEPT\_ID ;

6. CREATE OR REPLACE VIEW MAXMIN AS SELECT E.FIRST\_NAME,E.JOB\_ID FROM EMPLOYEES E WHERE (E.SALARY=(SELECT MIN(SALARY) FROM EMPLOYEES E1 WHERE E1.JOB\_ID=E.JOB\_ID))OR (E.SALARY=(SELECT MAX(SALARY) FROM EMPLOYEES E1 WHERE E1.JOB\_ID=E.JOB\_ID));

7. CREATE VIEW LOCATDEP AS SELECT EMPLOYEE\_ID,D.DEPARTMENT\_NAME,CITY,COUNTRY\_NAME FROM EMPLOYEES E,DEPARTMENTS D,LOCATIONS L,COUNTRIES C WHERE E.DEPARTMENT\_ID=D.DEPARTMENT\_ID AND D.LOCATION\_ID=L.LOCATION\_ID AND L.COUNTRY\_ID=C.COUNTRY\_ID;

8. SELECT DEPARTMENT\_ID,TO\_CHAR(HIRE\_DATE,'YYYY') AS YEAR,COUNT(\*) FROM EMPLOYEES GROUP BY DEPARTMENT\_ID,TO\_CHAR(HIRE\_DATE,'YYYY') ORDER BY DEPARTMENT\_ID,YEAR;

9. SELECT E.EMPLOYEE\_ID AS EMPID,E.FIRST\_NAME AS ENAME,E1.HIRE\_DATE AS MAN\_JOIN , E.HIRE\_DATE AS EMP\_JOIN FROM EMPLOYEES E,EMPLOYEES E1 WHERE E.MANAGER\_ID =E 1.EMPLOYEE\_ID AND E1.HIRE\_DATE>E.HIRE\_DATE;

10. SELECT CNAME,LNAME,COUNT(\*) AS NO FROM ( SELECT C.COUNTRY\_NAME AS CNAME,L.CITY S LNAME FROM COUNTRIES C, LOCATIONS L,DEPARTMENTS D WHERE D.LOCATION\_ID = L.LOCATION\_ID AND L.COUNTRY\_ID = C.COUNTRY\_ID AND D.DEPARTMENT\_ID IN (SELECT E1.DEPARTMENT\_ID FROM EMPLOYEES E1 GROUP BY E1.DEPARTMENT\_ID HAVING COUNT(\*)>5 )) GROUP BY CNAME,LNAME;

11. SELECT E.FIRST\_NAME,E.SALARY FROM EMPLOYEES E WHERE (SELECT COUNT(\*) FROM EMPLOYEES E1 WHERE E1.MANAGER\_ID=E.EMPLOYEE\_ID)>5;

12. CREATE OR REPLACE PROCEDURE UDPATESALARY(EMPNO INT) IS

BEGIN

UPDATE EMPLOYEES SET SALARY=SALARY\*1.2 WHERE TO\_CHAR(SYSDATE,'YYYY')-TO\_CHAR(HIRE\_DATE,'YYYY') > 10 AND EMPLOYEE\_ID=EMPNO;

UPDATE EMPLOYEES SET SALARY=SALARY\*1.1 WHERE TO\_CHAR(SYSDATE,'YYYY')-TO\_CHAR(HIRE\_DATE,'YYYY')>5 AND TO\_CHAR(SYSDATE,'YYYY')-TO\_CHAR(HIRE\_DATE,'YYYY')<=10 AND EMPLOYEE\_ID=EMPNO;

UPDATE EMPLOYEES SET SALARY=SALARY\*1.05 WHERE TO\_CHAR(SYSDATE,'YYYY') - TO\_CHAR(HIRE\_DATE,'YYYY')<5 AND EMPLOYEE\_ID=EMPNO;

END;

13. SELECT YEAR FROM(SELECT TO\_CHAR(HIRE\_DATE,'YYYY') AS YEAR,COUNT(\*)FROM EMPLOYEES GROUP BY TO\_CHAR(HIRE\_DATE,'YYYY') ORDER BY COUNT(\*) DESC) WHERE ROWNUM=1;

14. CREATE OR REPLACE PROCEDURE CHANGESAL IS SAL NUMBER;

BEGIN

SELECT SALARY INTO SAL FROM (SELECT SALARY FROM EMPLOYEES WHERE FIRST\_NAME='JOE' ORDER BY SALARY ) WHERE ROWNUM=1;

IF(SAL IS NULL) THEN

SELECT AVG(SALARY) INTO SAL FROM EMPLOYEES ;

UPDATE EMPLOYEES SET SALARY=SAL WHERE EMPLOYEE\_ID=130;

ELSE

UPDATE EMPLOYEES SET SALARY=SAL WHERE EMPLOYEE\_ID=130;

END IF;

END;

15. CREATE OR REPLACE PROCEDURE CURS IS

CURSOR EMP IS SELECT EMPLOYEE\_ID,FIRST\_NAME,SALARY FROM EMPLOYEES;

EMPID INT;

NAME VARCHAR2(40);

SAL INT;

COUT INT;

BEGIN

COUT :=0;

OPEN EMP;

LOOP

FETCH EMP INTO EMPID,NAME,SAL;

EXIT WHEN EMP%NOTFOUND;

COUT:=COUT+1;

IF (COUT=5 OR COUT=10) THEN

DBMS\_OUTPUT.PUT\_LINE(EMPID||' '||NAME||' '||SAL);

END IF;

END LOOP;

CLOSE EMP;

END;

16. CREATE OR REPLACE INCSALARY IS

BEGIN

UPDATE EMPLOYEES SET SALARY=SALARY\*1.1 WHERE DEPARTMENT\_ID=40;

UPDATE EMPLOYEES SET SALARY=SALARY\*1.15 WHERE DEPARTMENT\_ID=70;

UPDATE EMPLOYEES SET SALARY=SALARY\*1.05 WHERE DEPARTMENT\_ID<>70 AND

DEPARTMENT\_ID<>40 AND COMMISSION\_PCT>0.3;

UPDATE EMPLOYEES SET SALARY=SALARY\*1.1 WHERE DEPARTMENT\_ID<>70 ANDDEPARTMENT\_ID <>40 AND COMMISSION\_PCT<0.3;

END

17. CREATE OR REPLACE FUNCTION MAN\_NAME (DEPTNO IN INT, NAME OUT VARCHAR2)

RETURN VARCHAR IS

BEGIN

SELECT E.FIRST\_NAME INTO NAME FROM EMPLOYEES E,DEPARTMENTS D WHERE D.MANAGER\_ID =E.EMPLOYEE\_ID ;

END;

18.

CREATE OR REPLACE PROCEDURE MANCHNG(DNO INT) IS

TEMP INT;

BEGIN

SELECT E.EMPLOYEE\_ID INTO TEMP FROM EMPLOYEES E WHERE (SELECT COUNT (\*) FROM EMPLOYEES E1 WHERE E.SALARY>E1.SALARY) = 0;

UPDATE EMPLOYEES SET MANAGER\_ID=TEMP WHERE DEPARTMENT\_ID=DNO;

END;

19. CREATE OR REPLACE TRIGGER CHECK\_SAL BEFORE UPDATE ON EMPLOYEES FOR EACH ROW

DECLARE

SAL INT;

BEGIN

IF ( :NEW.SALARY < :OLD.SALARY) THEN

RAISE\_APPLICATION\_ERROR(-20002,'CAN NOT DECREASE SALARY');

ELSE

DBMS\_OUTPUT.PUT\_LINE('SUCCESFULL UPDATE');

END IF;

END;

20. CREATE OR REPLACE TRIGGER CHECK\_SAL \_MIN BEFORE INSERT ON EMPLOYEES FOR EACH ROW

DECLARE

MINI INT;

MAXI INT;

BEGIN

SELECT MIN\_SALARY INTO MINI FROM JOBS J WHERE J.JOB\_ID=:NEW.JOB\_ID;

SELECT MAX\_SALARY INTO MAXI FROM JOBS J WHERE J.JOB\_ID=:NEW.JOB\_ID;

IF ( :NEW.SALARY>MAXI ) THEN

RAISE\_APPLICATION\_ERROR(-20002,' SALARY NOT BETWEEN MIN MAX');

ELSIF ( :NEW.SALARY < MINI ) THEN

RAISE\_APPLICATION\_ERROR(-20002,' SALARY NOT BETWEEN MIN MAX');

ELSE

DBMS\_OUTPUT.PUT\_LINE('SUCCESFULL INSERT');

END IF;

END;

21. SELECT FIRST\_NAME,SALARY FROM EMPLOYEES WHERE DEPARTMENT\_ID=20 AND EMPLOYEE\_ID IN (SELECT EMP\_ID FROM PROJECT WHERE START\_DATE<'31-DEC-1990');

22. SELECT EMPLOYEE\_ID,FIRST\_NAME,SALARY FROM EMPLOYEES WHERE DEPARTMENT\_ID=10 AND SALARY> ANY (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT\_ID=30);

23. SELECT E.EMPLOYEE\_ID,E.FIRST\_NAME,E.SALARY FROM EMPLOYEES E WHERE (SELECT COUNT(\*)) FROM EMPLOYEES E1 WHERE E1.SALARY<=E.SALARY )< CEIL((SELECT COUNT(\*) FROM EMPLOYEES K )/2) ORDER BY SALARY;

24. CREATE OR REPLACE VIEW EMPLOYEE\_DEPT AS SELECT E.FIRST\_NAME , J.JOB\_TITLE , E.COMMISSION\_PCT , E.SALARY\*12 AS ANNUAL\_SALARY FROM EMPLOYEES E,JOBS JWHERE E.JOB\_ID=J.JOB\_ID AND E.DEPARTMENT\_ID=20;

UPDATE EMPLOYEES SET SALARY=1.1\*SALARY;

UPDATE EMPLOYEES SET

25. CREATE OR REPLACE PROCEDURE MGR\_DATE(DT DATE, DEPT INT) IS

CURSOR EMP IS

SELECT E.EMPLOYEE\_ID,E.FIRST\_NAME,J.JOB\_TITLE,E.MANAGER\_ID FROM EMPLOYEES E,JOBS J WHERE E.JOB\_ID=J.JOB\_ID AND HIRE\_DATE>DT AND MANAGER\_ID IN (SELECT M.EMPLOYEE\_ID FROM EMPLOYEES M WHERE M.DEPARTMENT\_ID=DEPT);

EMP\_ID INT;

F\_NAME VARCHAR2(300);

J\_TITLE VARCHAR2(200);

MGR\_ID INT ;

BEGIN

OPEN EMP;

LOOP

FETCH EMP INTO EMP\_ID,F\_NAME,J\_TITLE,MGR\_ID;

EXIT WHEN EMP%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(EMP\_ID||' '||F\_NAME||' '||J\_TITLE||' '||MGR\_ID);

END LOOP;

END;

26. CREATE OR REPLACE PROCEDURE INC\_KING(NAME VARCHAR2) IS

BEGIN

UPDATE EMPLOYEES E SET E.SALARY=E.SALARY\*1.05 WHERE E.MANAGER\_ID IN

(SELECT M.EMPLOYEE\_ID FROM EMPLOYEES M WHERE M.FIRST\_NAME=NAME

OR M.LAST\_NAME=NAME);

END;

CREATE OR REPLACE TRIGGER CHECK\_JOB BEFORE UPDATE OF SALARY ON

EMPLOYEES FOR EACH ROW

DECLARE

MAXI INT;

MINI INT;

BEGIN

SELECT MAX\_SALARY INTO MAXI FROM JOBS WHERE JOB\_ID=:NEW.JOB\_ID;

SELECT MIN\_SALARY INTO MINI FROM JOBS WHERE JOB\_ID=:NEW.JOB\_ID;

IF(:NEW.SALARY>MAXI) THEN

RAISE\_APPLICATION\_ERROR(-20002,'TOO HIGH SALARY');

ELSIF (:NEW.SALARY<MAXI) THEN

RAISE\_APPLICATION\_ERROR(-20002,'TOO LOW SALARY');

ELSE

DBMS\_OUTPUT.PUT\_LINE('SUCCESSFUL UPDATE');

END IF;

END;

27. CREATE TABLE EMPLOYMENT\_CHANGE(EMPNO INT, OLD\_JOBTYPE VARCHAR2(300) , NEW\_JOBTYPE VARCHAR2(300) , OLD\_DEPTNO INT , NEW\_DEPTNO INT,CHANGE\_DATE DATE,USER\_NAME VARCHAR2(300),TYPE\_CHANGE VARCHAR2(200));

CREATE OR REPLACE TRIGGER CHNG AFTER UPDATE OF DEPARTMENT\_ID,JOB\_ID ON

EMPLOYEES FOR EACH ROW

DECLARE

USER\_N VARCHAR2(300);

BEGIN

SELECT USER INTO USER\_N FROM DUAL;

IF (:OLD.JOB\_ID <> :NEW.JOB\_ID ) THEN

INSERT INTO EMPLOYMENT\_CHANGE VALUES(:OLD.EMPLOYEE\_ID,:OLD.JOB\_ID,:NEW.JOB\_ID,

:OLD.DEPARTMENT\_ID,:NEW.DEPARTMENT\_ID,SYSDATE,USER\_N,'JOBTYPE');

ELSIF (:OLD.DEPARTMENT\_ID <> :NEW.DEPARTMENT\_ID) THEN

INSERT INTO EMPLOYMENT\_CHANGE VALUES(:OLD.EMPLOYEE\_ID,:OLD.JOB\_ID,

:NEW.JOB\_ID,:OLD.DEPARTMENT\_ID,:NEW.DEPARTMENT\_ID,SYSDATE,USER\_N,'DEPT\_NO');

END IF;

END;

CREATE OR REPLACE TRIGGER EXEMP AFTER DELETE ON EMPLOYEES FOR EACH ROW

DECLARE

BEGIN

INSERT INTO EXEMPLOYEE

VALUES(:OLD.EMPLOYEE\_ID,:OLD.FIRST\_NAME,:OLD.DEPARTMENT\_ID,SYSDATE);

END;

28. CREATE OR REPLACE PROCEDURE INC\_SAL\_DEPT(DEPT\_ID INT) IS

CURSOR EMP IS

SELECT EMPLOYEE\_ID,SALARY FROM EMPLOYEES WHERE DEPARTMENT\_ID=DEPT\_ID;

EMP\_ID INT;

SAL INT;

BEGIN

OPEN EMP;

LOOP

FETCH EMP INTO EMP\_ID,SAL;

UPDATE EMPLOYEES SET SALARY=SAL\*1.1 WHERE EMPLOYEE\_ID=EMP\_ID;

END LOOP;

END;

29.

import java.util.Scanner;

import java.sql.\*;

public class Assignment29 {

public static void main(String args[]) {

System.out.println("-------- Oracle JDBC Connection Testing ------");

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

} catch (ClassNotFoundException e) {

System.out.println("Where is your Oracle JDBC Driver?");

e.printStackTrace();

return;

}

System.out.println("Oracle JDBC Driver Registered!");

Connection connection = null;

try {

connection =DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",

"HR", "H");

} catch (SQLException e) {

System.out.println("Connection Failed! Check output console");

e.printStackTrace();

return;

}

if (connection != null) {

System.out.println("You made it, take control your database now!");

try{

Scanner sc=new Scanner(System.in);

String call= "call UpdMgr(?,?)";

CallableStatement update=connection.prepareCall(call);

System.out.println("Please Enter Department, New Manager");

update.setInt(1,sc.nextInt());

update.setInt(2,sc.nextInt());

update.executeUpdate();

System.out.println("Done");

}

catch(Exception E){

E.printStackTrace(); } } } }

30

public class Assignment30 {

public static void main(String args[]) {

System.out.println("-------- Oracle JDBC Connection Testing ------");

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

} catch (ClassNotFoundException e) {

System.out.println("Where is your Oracle JDBC Driver?");

e.printStackTrace();

return; }

System.out.println("Oracle JDBC Driver Registered!");

Connection connection = null;

try {

connection =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",

"HR", "H");

} catch (SQLException e) {

System.out.println("Connection Failed! Check output console");

e.printStackTrace();

return; }

if (connection != null) {

System.out.println("You made it, take control your database now!");

try{

String update1= "SELECT DEPARTMENT\_ID,DEPARTMENT\_NAME,TOTAL\_SALARY

FROM (SELECT E.DEPARTMENT\_ID AS DEPT\_ID,SUM(SALARY) AS TOTAL\_SALARY "

+ "FROM EMPLOYEES E GROUP BY E.DEPARTMENT\_ID HAVING

SUM(SALARY)>100000),DEPARTMENTS "

+ "WHERE DEPARTMENTS.DEPARTMENT\_ID=DEPT\_ID ";

Statement Dept=connection.createStatement();

ResultSet DeptData= Dept.executeQuery(update1);

System.out.println("Department\_Id"+"\t"+"Department\_Name"+"\t"+"Total\_Salary");

while(DeptData.next()){

int dept\_id=DeptData.getInt("DEPARTMENT\_ID");

String dept\_name=DeptData.getString("DEPARTMENT\_NAME");

int sal=DeptData.getInt("TOTAL\_SALARY");

System.out.println(dept\_id+"\t"+dept\_name+"\t"+sal);

}

}

catch(Exception E){

E.printStackTrace(); } } } }