```
REM: Assignment 2 – DML STATEMENTS

REM: LAKSHMI PRIYA B

REM: 185001083

REM: **********************************************************************


SET echo ON;

DROP TABLE classes;
DROP TABLE employees;

CREATE TABLE Classes(
class varchar2(20),
type varchar2(5),
country varchar2(20),
numguns number(5),
bore number(10),
displacement number(20)
);

DESC Classes;


REM: 1.Add first two tuples from the above sample data.
REM: List the columns explicitly in the INSERT clause.
REM: (No ordering of columns)

INSERT INTO
Classes(class, type, country, numguns, bore, displacement)
VALUES ('Bismark', 'bb', 'Germany', 8, 14, 32000);

INSERT INTO
Classes(numguns, bore, displacement, class, type, country)
VALUES (9, 16, 46000, 'Iowa', 'bb', 'USA');


REM: 2.Populate the relation with the remaining set of tuples.
REM: This time, do not list the columns in the INSERT clause.

INSERT INTO
Classes
VALUES ('Kongo', 'bc', 'Japan', 8, 15, 42000);

INSERT INTO
Classes
VALUES ('North Carolina', 'bb', 'USA', 9, 16, 37000);
```

```
INSERT INTO
Classes
VALUES ('Revenge', 'bb', 'Gt. Britain', 8, 15, 29000);

INSERT INTO
Classes
VALUES ('Renown', 'bc', 'Gt. Britain', 6, 15, 32000);


REM: 3.Display the populated relation
SELECT * FROM classes;


REM: 4.Mark an intermediate point here in this transaction.
SAVEPOINT savept1;


REM: 5.Change the displacement of Bismark to 34000.
UPDATE classes
SET displacement=34000
WHERE class='Bismark';

SELECT * FROM classes;


REM: 6.For the battleships having at least 9 number of guns or
REM: the ships with at least 15 inch bore, increase the displacement by 10%.
REM: Verify your changes to the table

UPDATE classes
SET displacement=1.1*displacement
WHERE numguns >= 9 OR bore >= 15;

SELECT * FROM classes;


REM: 7.Delete Kongo class of ship from Classes table.
DELETE FROM classes
WHERE class='Kongo';


REM: 8.Display your changes to the table
SELECT * FROM classes;


REM: 9.Discard the recent updates to the relation
REM: without discarding the earlier INSERT operation(s).
ROLLBACK TO savept1;
```

```
SELECT * FROM classes;


REM: 10. Commit the changes.
COMMIT;

@D:/employees.sql;

SELECT * FROM employees;


REM: 11. Display firsy name, job id and salary of all the employees.
SELECT first_name, job_id, salary
FROM employees;


REM: 12. Display the id, name(first & last), salary and annual salary
REM: of all the employees. Sort the employees by first name.
REM: Label the columns as shown below:
REM: (EMPLOYEE_ID, FULL NAME, MONTHLY SAL, ANNUAL SALARY)
SELECT  employee_id AS EMPLOYEE_ID,
        CONCAT(first_name, CONCAT(' ',last_name)) AS "FULL NAME",
        salary AS "MONTHLY SALARY",
        salary*12 AS "ANNUAL SALARY"
FROM employees
ORDER BY first_name;


REM: 13. List the different jobs in which the employees are working for.
SELECT job_id
FROM employees
GROUP BY job_id;


REM: 14. Display the id, first name, job id, salary and commission
REM: of employees who are earning commissions.
SELECT  employee_id,
        first_name,
        job_id,
        salary,
        commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;


REM: 15. Display the details (id, first name, job id, salary and dept id)
REM: of employees who are MANAGERS.
```

```
SELECT  employee_id,
        first_name,
        job_id,
        salary,
        department_id
FROM employees
WHERE employee_id IN (
                        SELECT DISTINCT manager_id FROM employees
                    );


REM: 16. Display the details of employees other than sales representatives
REM: (id, first name, hire date, job id, salary and dept id)
REM: who are hired after '01-May-1999' or whose salary is at least 10000.
SELECT employee_id, first_name, hire_date, job_id, salary, department_id
FROM employees
WHERE hire_date > '01-MAY-1999' OR salary >= 10000;


REM: 17. Display the employee details (first name, salary, hire date and dept id)
REM: whose salary falls in the range of 5000 to 15000 and his/her name begins
REM: with any of characters (A,J,K,S). Sort the output by first name
SELECT first_name, salary, hire_date, department_id
FROM employees
WHERE salary between 5000 and 15000
          AND
    (first_name LIKE 'A%' OR
        first_name LIKE 'J%' OR
        first_name LIKE 'K%' OR
        first_name LIKE 'S%' )
ORDER BY first_name;


REM: 18. Display the experience of employees in no. of years and months
REM: who were hired after 1998. Label the columns as:
REM: (EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, EXP-YRS, EXP-MONTHS)
SELECT employee_id, first_name, hire_date,
(select extract(year from current_date) from dual) - extract(year from hire_date) AS exp_yrs,
extract(month from hire_date) - (select extract(month from current_date) from dual) AS
exp_months
from EMPLOYEES;


REM: 19. Display the total number of departments.
SELECT COUNT(DISTINCT department_id) AS num_depts
FROM employees;
```

```
REM: 20. Show the number of employees hired by year-wise. Sort the result by year-wise
SELECT extract(year from hire_date) AS hire_year, COUNT(*) as count
FROM employees
GROUP BY extract(year from hire_date)
ORDER BY hire_year;


REM: 21. Display the minimum, maximum and average salary, number of employees
REM: for each department. Exclude the employee(s) who are not in any department.
REM: Include the department(s) with at least 2 employees and the average salary is
REM: more than 10000. Sort the result by minimum salary in descending order.
SELECT department_id, min(salary), max(salary), avg(salary), count(*)
FROM employees
WHERE department_id IS NOT NULL
GROUP BY department_id
HAVING count(*) >= 2 AND avg(salary) > 10000
ORDER BY min(salary) DESC;




REM: **********

REM: LAKSHMI PRIYA B

REM: 185001083

REM: **********
```