```
set echo on:

DROP TABLE classes;

REM:**********************Part – I : DML Update operations & TCL
statements**********************

REM:Consider the Classes relation that can be described as below:
REM:The relation Classes records the name of the class – ship class, the type
of ships (bb for battleship
REM:or bc for battle cruiser), the country that built the ship, the number of
main guns, the bore
REM:(diameter of the gun barrel, in inches) of the main guns, and the
displacement (weight, in tons).
REM:Note: Define the relation Classes appropriately to accommodate the
following tuples:

CREATE TABLE classes(
class VARCHAR(20) PRIMARY KEY,
type VARCHAR(4) CHECK(type IN('bb', 'bc')),
country VARCHAR(20),
numguns NUMBER(3),
bore NUMBER(3),
displacement NUMBER(10));

DESC classes;

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

INSERT INTO classes(class, type, country, numguns, bore, displacement)
VALUES('Bismark', 'bb', 'Germany', 8, 14, 32000);
INSERT INTO classes(type, class, country, numguns, bore, displacement) VALUES(
'bb', 'Iowa', 'USA', 9, 16, 46000);

REM: 2. Populate the relation with the remaining set of tuples. 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 table_created_display;

REM: 5. Change the displacement of Bismark to 34000.

SELECT * FROM classes;
UPDATE classes SET displacement=34000 WHERE class='Bismark';
SELECT * FROM classes;

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

UPDATE classes SET displacement=displacement+0.1*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 without discarding the
earlier INSERT operation(s).

ROLLBACK TO table_created_display;
SELECT * FROM classes;

REM: 10. Commit the changes.

COMMIT;
SELECT * FROM classes;

REM:**********************Part – II : DML Retrieval
operations**********************
REM:Use the employees.sql to create the database and write the SQL statements
for the following:

@z:/employees.sql;

REM: 11. Display first name, job id and salary of all the employees.

SELECT first_name, job_id, salary FROM employees;

REM: 12. Display the id, name(first and last), salary and annual salary of all
the employees.
REM: Sort the employees by first name.
REM: Label the columns as shown below:(EMPLOYEE_ID, FULL NAME, MONTHLY SAL,
ANNUAL SALARY)

SELECT employee_id, first_name || ' ' || last_name AS full_name, salary AS
monthly_sal, salary*12 AS annual_sal
FROM employees
ORDER BY first_name;

REM: 13. List the different jobs in which the employees are working for.

SELECT DISTINCT(job_id) FROM employees;

REM: 14. Display the id, first name, job id, salary and commission 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) of
employees who are MANAGERS.

SELECT DISTINCT(e2.employee_id), e2.first_name, e2.job_id, e2.salary,
e2.department_id
FROM employees e1, employees e2
WHERE e1.manager_id=e2.employee_id;

REM: 16. Display the details of employees other than sales representatives
(id, first name, hire date, job id, salary and dept id)
REM:     who are hired after '01May1999' or whose salary is at least 10000.
```

```
SELECT employee_id, first_name, hire_date, job_id, salary, department_id
FROM employees
WHERE (hire_date > TO_DATE('1999-05-01','YYYY-MM-DD') OR salary>=10000) AND
job_id <> 'SA_REP';

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 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 who
were hired after 1998.
REM:     Label the columns as: (EMPLOYEE_ID, FIRST_NAME, HIRE_DATE, EXPYRS,
EXPMONTHS).

SELECT employee_id, first_name, hire_date, EXTRACT(YEAR FROM SYSDATE) -
EXTRACT(YEAR FROM hire_date) AS expyrs,
(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM hire_date))*12 AS expmonths
FROM employees
WHERE hire_date > TO_DATE('31-12-1998','DD-MM-YYYY');

REM: 19. Display the total number of departments.

SELECT COUNT(DISTINCT(DEPARTMENT_ID)) FROM employees;

REM: 20. Show the number of employees hired by yearwise. Sort the result by
yearwise.

SELECT COUNT(*) AS num_employees, EXTRACT(year from hire_date) AS hire_yr
FROM employees
GROUP BY EXTRACT(year from hire_date)
ORDER BY EXTRACT(year from hire_date);

REM: 21. Display the minimum, maximum and average salary, number of employees
for each department.
REM:     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 more than 10000.
REM:     Sort the result by minimum salary in descending order.

SELECT MIN(salary) AS min_sal, MAX(salary) AS max_sal, AVG(salary) AS avg_sal,
COUNT(*) AS num_employees, department_id
FROM employees WHERE department_id IS NOT NULL
GROUP BY department_id
HAVING COUNT(*) > 1 AND AVG(salary) > 10000 ORDER BY min_sal DESC;
```