LAB WORK

DML COMMANDS

1. CREATE TABLE EMPLOYEE (ID INT , NAME VARCHAR(30) , DEPT VARCHAR(20), SALARY INT , DOB DATE , GENDER CHAR(1)) ;
2. DESC EMPLOYEE ;
3. INSERT INTO EMPLOYEE VALUES(1,'PRAVEEN' , 'CSE' , 4000 , '1980-12-12' , 'M') , (2 , 'RAJU ' , 'CSE ' , 5000 , '1982-02-01' , 'M') , (3 , 'RANI' , 'CSE' , 6000 , '1985-12-25' , 'F') , (4 , 'RAJITHA' , 'ECE' , 6500 , '1989-01-24' , 'F') ,(5 , 'KARUN' , 'EEE' , 4500 , '1990-08-14' , 'M') ;
4. SELECT * FROM EMPLOYEE ;
5. SELECT NAME FROM EMPLOYEE ;
6. SELECT NAME,SALARY FROM EMPLOYEE ;
7. SELECT * FROM EMPLOYEE WHERE ID =1 ;
8. SELECT * FROM EMPLOYEE WHERE DEPT = 'CSE' ;
9. SELECT * FROM EMPLOYEE WHERE YEAR(DOB) = '1980' ;

ID	NAME	DEPT	SALARY	DOB	GENDER
1	PRAVEEN	CSE	4000	1980-12-12	M

10. SELECT * FROM EMPLOYEE WHERE SALARY < 5000 ;
11. SELECT * FROM EMPLOYEE WHERE DEPT<> 'CSE' ;
12. SELECT * FROM EMPLOYEE WHERE SALARY BETWEEN 4000 AND 5000 ;

ID	NAME	DEPT	SALARY	DOB	GENDER
1	PRAVEEN	CSE	4000	1980-12-12	M
2	RAJU	CSE	5000	1982-02-01	M
5	KARUN	EEE	4500	1990-08-14	M

13. SELECT * FROM EMPLOYEE WHERE ID BETWEEN 3 AND 4 ;
14. SELECT * FROM EMPLOYEE WHERE DEPT = 'CSE' OR ID =4 ;

ID	NAME	DEPT	SALARY	DOB	GENDER
1	PRAVEEN	CSE	4000	1980-12-12	M
2	RAJU	CSE	5000	1982-02-01	M
3	RANI	CSE	6000	1985-12-25	F

ID	NAME	DEPT	SALARY	DOB	GENDER
4	RAJITHA	ECE	6500	1989-01-24	F

15. SELECT * FROM EMPLOYEE WHERE ID NOT BETWEEN 1 AND 3 ;
16. SELECT * FROM EMPLOYEE WHERE LENGTH(NAME) = 4 ;

ID	NAME	DEPT	SALARY	DOB	GENDER
3	RANI	CSE	6000	1985-12-25	F

17. SELECT * FROM EMPLOYEE WHERE NAME LIKE '_R%' ;
18.

ID	NAME	DEPT	SALARY	DOB	GENDER
1	PRAVEEN	CSE	4000	1980-12-12	M

NOTE : THE MOST COMMONLY USED OPERATION ON STRING IS PATTERN MATCHING USING THE OPERATOR **LIKE .** WE DESCRIBE PATTERN BY USING TWO SPECIAL CHARACTERS .

a) PERCENT (%) : THE % CHARACTER MATCHES ANY SUBSTRING.
b) UNDERSCORE(_) : THE _ CHARACTER MATCHES ANY CHARACTER .

PATTERNS ARE CASE SENSITIVE

19. ELECT * FROM EMPLOYEE WHERE NAME LIKE 'R%' ;
20. SELECT * FROM EMPLOYEE WHERE NAME LIKE '%N' ;
21. UPDATE EMPLOYEE SET NAME ='MOHAN' WHERE ID =1 ;
22. SELECT * FROM EMPLOYEE ;
23. DELETE FROM EMPLOYEE WHERE ID =2 ;
24. SELECT * FROM EMPLOYEE ;
25. TRUNCATE TABLE EMP ;

VIEWS

1. CREATE TABLE STUD (ID INT , NAME VARCHAR(20) , AGE INT(11) , GENDER VARCHAR(2)) ;
2. INSERT INTO STUD VALUES (1,'A',19,'F'),(2,'B',20,'M'),(3,'C',22,'M'),(4,'D',21,'F')
3. SELECT * FROM STUD
4. CREATE VIEW STUD_VIEW AS SELECT ID ,NAME FROM STUD
5. DESC STUD_VIEW
6. SELECT * FROM STUD_VIEW
7. INSERT INTO STUD_VIEW VALUES(5,'E')
8. SELECT * FROM `STUD

OUTPUT

1 A 19 F

2	B	20	M
3	C	22	M
4	D	21	F
5	E	*NULL*	*NULL*

9. DELETE FROM STUD WHERE ID = 5
10. SELECT * FROM STUD

ORDER BY

SQL USES ORDER BY CLAUSE TO DISPLAY THE TUPLES IN THE RESULT OF THE QUERY TO APPEAR IN SORTED ORDER

11. CREATE TABLE STUD1 (ID INT , NAME VARCHAR(20) , AGE INT(11) , GENDER VARCHAR(2)) ;
12. INSERT INTO STUD1 VALUES (1,'A',19,'F'),(2,'B',20,'M'),(3,'C',22,'M'),(4,'D',21,'F')
13. SELECT * FROM STUD1
14. SELECT NAME FROM STUD1 ORDER BY NAME
15. SELECT AGE FROM STUD1 ORDER BY AGE
16. SELECT AGE FROM STUD1 ORDER BY AGE DESC
17. SELECT * FROM STUD1 WHERE ID > 2 ORDER BY AGE DESC
18. SELECT * FROM STUD1 ORDER BY AGE ASC , ID DESC

OUTPUT

ID 1	NAME	AGE	GENDER
1	A	19	F
2	B	20	M
3	C	22	M
4	D	21	F

AGGREGATION

AGGREGATION FUNCTIONS ARE FUNCTIONS THAT TAKE A COLLECTION OF VALUES AS INPUT AND RETURN A SINGLE VALUE .SQL OFFERS FIVE BUILT IN AGGREGATE FUNCTIONS

a) AVERAGE : avg
b) MINIMUM : min
c) MAXIMUM :max
d) TOTAL: sum

e) COUNT : count

19. CREATE TABLE EMP2(ID INT , NAME VARCHAR(20) , DOB DATE , GENDER VARCHAR(3) , SALARY INT , DEPT VARCHAR(5) , COLLEGE VARCHAR(20) , AGE INT) ;
20. DESC EMP2 ;
21. INSERT INTO EMP2 VALUES (1, 'ANUSHA' , '2000-10-10' , 'F' , 10000 , 'CSE' , 'KITS',20) ,(2,'AKANKSHA','1990-12-10' , 'F',8000,'ECE' , 'KITS' , 30) , (3,'ANJALI' ,'1995-08-14' , 'F' ,15000 ,'CSE' , 'KITSW' ,40) , (4,'KARUN' ,'1990-12-06' , 'M' , 9000 , 'ECE' , 'KITSW' , 38) , (5,'KARTHIK' , '1995-12-09' , 'M' ,8500 , 'CSE' , 'KITS' , 30) ,(6 , 'PRATHIK' , '1992-09-13' , 'M' , 9500 , 'ECE' , 'KITSW' , 30) , (7 , 'NULL' , '1994-05-06' , 'F' , 120000 , 'CIVIL' , 'KITSW' , 32) ;
22. SELECT * FROM EMP2
23. SELECT COUNT(NAME) FROM EMP2 ;

OUTPUT

COUNT(NAME)
7

24. SELECT MIN(SALARY) FROM EMP2 ;

OUTPUT

MIN(SALARY)
8000

25. SELECT MIN(AGE) FROM EMP2 ;
26. SELECT MAX(SALARY) FROM EMP2 ;
27. SELECT MAX(SALARY) AS EMP_SAL FROM EMP2 ;

OUTPUT

EMP_SAL
120000

28. LECT AVG(SALARY) FROM EMP2 ;
29. SELECT SUM(SALARY) FROM EMP2 ;

GROUP BY

GROUP BY CLAUSE IS USED TO GROUP THE ROW BASED ON CERTAIN CRITERIA . FOR EXAMPLE IN AN EMPLOYEE TABLE(EMP2) YOU CAN GROUP THE EMPLOYEE BY DEPARTMENT (DEPT) AND

SO ON . **GROUP BY** IS USUALLY USED IN CONJUCTION WITH AGGREGATE FUNCTION LIKE **SUM , AVG , MIN , MAX** ETC

30. SELECT DEPT ,COUNT(*) FROM EMP2 GROUP BY DEPT ;

OUTPUT

DEPT	COUNT(*)
CIVIL	1
CSE	3
ECE	3

31. SELECT DEPT , MIN(SALARY) FROM EMP2 GROUP BY DEPT ;

OUTPUT :

DEPT	MIN(SALARY)
CIVIL	120000
CSE	8500
ECE	8000

32. SELECT DEPT , SUM(SALARY) FROM EMP2 GROUP BY DEPT ;

OUTPUT:

DEPT	SUM(SALARY)
CIVIL	120000
CSE	33500
ECE	26500

33. SELECT GENDER , COUNT(*) FROM EMP2 GROUP BY GENDER ;

GENDER	COUNT(*)
F	4

GENDER	COUNT(*)
M	3

34. SELECT COLLEGE , GENDER ,COUNT(*) FROM EMP2 GROUP BY COLLEGE , GENDER ;

COLLEGE	GENDER	COUNT(*)
KITS	F	2
KITS	M	1
KITSW	F	2
KITSW	M	2

35. SELECT COLLEGE , DEPT ,GENDER ,COUNT(*) FROM EMP2 GROUP BY COLLEGE ,DEPT , GENDER ;

COLLEGE	DEPT	GENDER	COUNT(*)
KITS	CSE	F	1
KITS	CSE	M	1
KITS	ECE	F	1
KITSW	CIVIL	F	1
KITSW	CSE	F	1
KITSW	ECE	M	2

HAVING CLAUSE :

THE HAVING CLAUSE TELLS SQL TO INCLUDE ONLY CERTAIN GROUPS PRODUCED BY THE GROUP BY CLAUSE IN THE QUERY RESULT SET. HAVING IS EQUIVALENT TO THE WHERE CLAUSE AND IS USED TO SPECIFY THE SEARCH CRITERIA OR SEARCH CONDITION WHEN GROUP BY CLAUSE IS SPECIFIED .

36. SELECT DEPT , COUNT(*) FROM EMP2 GROUP BY DEPT HAVING COUNT(*) > 1 ;

DEPT	COUNT(*)
CSE	3

DEPT	COUNT(*)
ECE	3

JOIN

1. CREATE TABLE MOBILE (MODEL VARCHAR(20) , PRICE INT) ;
2. DESC MOBILE ;
3. CREATE TABLE LAPTOP (MODEL VARCHAR(20) , PRICE INT) ;
4. DESC LAPTOP ;
5. INSERT INTO MOBILE VALUES ('NOKIA' , 10000) ,('SAMSUNG' , 20000) , ('IPHONE' , 500000) ;
6. SELECT * FROM MOBILE ;
7. INSERT INTO LAPTOP VALUES ('DELL' , 30000) ,('ACER' , 20000) , ('AUES' , 100000) ;
8. SELECT * FROM LAPTOP ;
9. SELECT MOBILE.MODEL , LAPTOP.MODEL FROM MOBILE JOIN LAPTOP ON MOBILE.PRICE < LAPTOP.PRICE ;

OUTPUT

MODEL	MODEL
NOKIA	DELL
SAMSUNG	DELL
NOKIA	ACER
NOKIA	AUES
SAMSUNG	AUES

10. SELECT MOBILE.MODEL , LAPTOP.MODEL FROM MOBILE JOIN LAPTOP ON MOBILE.PRICE = LAPTOP.PRICE ;

OUTPUT

MODEL	MODEL
SAMSUNG	ACER
NOKIA	AUES

NATURAL JOIN

11. CREATE TABLE DEPT2 (DNAME VARCHAR(20) , MANAGER VARCHAR(20) , PRIMARY KEY (DNAME)) ;
12. DESC DEPT2 ;
13. CREATE TABLE EMP3(EMPID INT , ENAME VARCHAR(20) , DNAME VARCHAR(20) , PRIMARY KEY(EMPID) , FOREIGN KEY(DNAME) REFERENCES DEPT2(DNAME)) ;
14. DESC EMP3;
15. INSERT INTO DEPT2 VALUES ('FIN' , 'M1') ,('SALES' , 'M2') ;
16. SELECT * FROM DEPT2 ;
17. INSERT INTO EMP3 VALUES (1,'A' ,'FIN') ,(2 , 'B' ,'SALES') ,(3 , 'C' , 'SALES') ;
18. SELECT * FROM EMP3 ;
19. SELECT * FROM DEPT2 NATURAL JOIN EMP3 ;

OUTPUT:

DNAME	MANAGER	EMPID	ENAME
FIN	M1	1	A
SALES	M2	2	B
SALES	M2	3	C

FULL JOIN

20. CREATE TABLE COUNTRY (CID INT , CNAME VARCHAR(20) NOT NULL , PRIMARY KEY(CID)) ;
21. DESC COUNTRY ;
22. CREATE TABLE STATE (SID INT , CID INT ,SNAME VARCHAR(20) , PRIMARY KEY (SID) , FOREIGN KEY(CID) REFERENCES COUNTRY(CID)) ;
23. DESC STATE ;
24. INSERT INTO COUNTRY VALUES (1 , 'NEPAL') , (2 , 'INDIA') , (3 , 'SRILANKA') ;
25. SELECT * FROM COUNTRY ;
26. INSERT INTO STATE VALUES (1 , 1 , 'TS') , (2,1,'AP') , (3,2,'A') ,(4,1,'KATMANDU') ;
27. SELECT * FROM STATE ;

a) **FULL JOIN**

28. SELECT * FROM COUNTRY LEFT JOIN STATE ON COUNTRY.CID = STATE.CID ;

OUTPUT :

CID	CNAME	SID	CID	SNAME
1	NEPAL	1	1	TS
1	NEPAL	2	1	AP
2	INDIA	3	2	A
1	NEPAL	4	1	KATMANDU
3	SRILANKA	*NULL*	*NULL*	*NULL*

29. SELECT * FROM COUNTRY RIGHT JOIN STATE ON COUNTRY.CID = STATE.CID ;

OUTPUT :

CID	CNAME	SID	CID	SNAME
1	NEPAL	1	1	TS
1	NEPAL	2	1	AP
1	NEPAL	4	1	KATMANDU
2	INDIA	3	2	A

NESTED QUERIES

30. CREATE TABLE DEPT4(DNAME VARCHAR(20) , DNO INT , PRIMARY KEY(DNO)) ;

31. DESC DEPT4 ;
32. INSERT INTO DEPT4 VALUES('HR' ,01) ,('HR' , 02) ;
33. SELECT * FROM DEPT4 ;
34. CREATE TABLE EMP4 (DNO INT , ENO INT) ;
35. DESC EMP4 ;
36. INSERT INTO EMP4 VALUES (01 , 123) , (02 , 145) ;
37. SELECT * FROM EMP4 ;
38. SELECT DNAME FROM DEPT4 WHERE DNO =(SELECT DNO FROM EMP4 WHERE ENO = 123)
39. OUTPUT :

DNAME
HR

SET OPERATIONS

1. CREATE TABLE DEPOSITOR(CUSTOMER_NAME VARCHAR(20) , ACCOUNT_NO INT) ;
2. INSERT INTO DEPOSITOR VALUES ('JOHN' ,1001) , ('SITA' ,1002) ,('VISHAL' , 1003) , ('RAM' , 1004) ;
3. CREATE TABLE BORROWER(CUSTOMER_NAME VARCHAR(20) , LOAN_NO INT) ;
4. INSERT INTO BORROWER VALUES ('JOHN' ,2001) , ('TONNY' ,2003) ,('VISHAL' , 2002) , ('ROHIT' , 2004) ;

UNION

UNION MAY CAUSE MERGE THE OUTPUT OF TWO OR MORE QUERIES INTO A SINGLE SET OF ROWS AND COLUMN .

5. SELECT CUSTOMER_NAME FROM BORROWER UNION SELECT CUSTOMER_NAME FROM DEPOSITOR ;
   OUTPUT:

CUSTOMER_NAME
JOHN
TONNY
VISHAL
ROHIT
SITA
RAM

6. SELECT CUSTOMER_NAME FROM BORROWER UNION ALL SELECT CUSTOMER_NAME FROM DEPOSITOR ;

OUTPUT:

CUSTOMER_NAME
JOHN
TONNY
VISHAL
ROHIT
JOHN
SITA
VISHAL
RAM

NOTE:

a) IF WE WANT TO RETAIN ALL DUPLICATE , WE MUST WRITE UNION ALL IN PLACE OF UNION .
b) UNION CANNOT BE USED IN SUBQUERIES.
c) YOU CANNOT USE AGGREGATE FUNCTIONS IN UNION .

THE INTERSECT OPERATION

THE INTERSECT CLAUSE OUTPUTS ONLY ROWS PRODUCED BY BOTH THE QUERIES INTERSECTED i.e THE INTERSECT OPERATION RETURNS COMMON RECORDS FROM THE OUTPUT OF BOTH QUERIES .

7. SELECT CUSTOMER_NAME FROM BORROWER INTERSECT SELECT CUSTOMER_NAME FROM DEPOSITOR ; (DOES NOT WORK IN APP SERVER)
8. SELECT CUSTOMER_NAME FROM BORROWER INTERSECT ALL SELECT CUSTOMER_NAME FROM DEPOSITOR ; (DOES NOT WORK IN APP SERVER)

THE EXCEPT OPERATION

THE EXCEPT ALSO CALLED AS MINUS OUTPUTS ROWS THAT ARE IN FIRST TABLE BUT NOT IN SECOND TABLE

9. SELECT CUSTOMER_NAME FROM DEPOSITOR MINUS SELECT CUSTOMER_NAME FROM BORROWER ; (DOES NOT WORK IN APP SERVER)

NULL VALUES

SQL ALLOWS THE USE OF NULL VALUES TO INDICATE ABSENCE OF INFORMATION ABOUT THE VALUE OF AN ATTRIBUTE.

10. CREATE TABLE CUSTOMER (CUST_NO INT , CUST_NAME VARCHAR(20) , CUST_PH_NO INT) ;
11. INSERT INTO CUSTOMER VALUES (101 , 'TELCO' , 5346772) , (102 , 'BAJAJ' , 5429810) , (103 , 'MAHINDRA' , NULL) ;
12. DESC CUSTOMER ;
13. SELECT * FROM CUSTOMER ;
14. SELECT CUST_NAME FROM CUSTOMER WHERE CUST_PH_NO IS NULL ;

OUTPUT :

CUST_NAME
MAHINDRA

15. SELECT CUST_NAME FROM CUSTOMER WHERE CUST_PH_NO IS NOT NULL ;

NESTED SUBQUERIES

SQL PROVIDES A MECHANISM FOR NESTING SUBQUERIES . A SUBQUERY IS A SELECT – FROM WHERE EXPRESSION THAT IS NESTED WITHIN ANOTHER QUERY . MOSTLY SUBQUERIES ARE USED TO PERFORM TESTS FOR SET MEMBERSHIP TO MAKE SET COMPARISIONS.

A) SET MEMBERSHIP

SQL USES **IN** AND **NOT IN** CONSTRUCTS FOR SET MEMBERSHIP TESTS

- **IN** : THE IN CONNECTIVE TESTS FOR SET MEMBERSHIP , WHERE THE SET IS A COLLECTION OF VALUES PRODUCED BY **SELECT** CLAUSE
- **NOT IN** : THE NOT IN CONNECTIVE TESTS FOR THE ABSENCE OF SET MEMBERSHIP .

16. SELECT CUST_NO , CUST_NAME FROM CUSTOMER WHERE CUST_NO IN(101 , 102 ,103 ,104)

OUTPUT :

CUST_NO	CUST_NAME
101	TELCO
102	BAJAJ
103	MAHINDRA

17. SELECT CUST_NO , CUST_NAME FROM CUSTOMER WHERE CUST_NAME NOT IN ('RAMCO' , 'BAJAJ' , 'TCS') ;

OUTPUT :

CUST_NO	CUST_NAME
101	TELCO
103	MAHINDRA

18. SELECT CUSTOMER_NAME FROM DEPOSITOR WHERE CUSTOMER_NAME NOT IN (SELECT CUSTOMER_NAME FROM BORROWER) ;

OUTPUT :

CUSTOMER_NAME
SITA
RAM

B) SET COMPARISON :

NESTED SUBQUERIES ARE USED TO COMPARE SETS . SQL USES VARIOUS COMPARISION OPERATORS SUCH AS **< , <= , > , > = , = , < > , ANY ,ALL AND SOME** ETC TO COMPARE SETS .

19. CREATE TABLE BOOK (TITLE VARCHAR(15) , AUTHOR_NAME VARCHAR(20) , PUBLISHER_NAME VARCHAR(20) , PUB_YEAR DATE) ;
20. DESC BOOK ;
21. INSERT INTO BOOK VALUES('ORACLE' , 'ARORA' , 'PHI' ,'2004-01-01') , ('DBMS' ,'BASU', 'TECHNICAL' , '2004-02-01') , ('ADBMS' , 'BASU' ,'TECHNICAL' ,'2004-02-02') , ('UNIX' , 'KAPOOR' , 'SCITECH' ,'2000-01-01') ;
22. SELECT * FROM BOOK ;
23. ALTER TABLE BOOK ADD UNIT_PRICE INT ;
24. UPDATE BOOK SET UNIT_PRICE=399 WHERE TITLE = 'ORACLE' ;
25. UPDATE BOOK SET UNIT_PRICE=400 WHERE TITLE = 'DBMS' ;
26. UPDATE BOOK SET UNIT_PRICE=450 WHERE TITLE = 'ADBMS' ;
27. UPDATE BOOK SET UNIT_PRICE=300 WHERE TITLE = 'UNIX' ;
28. INSERT INTO BOOK VALUES('DOS' , 'KUMAR' , 'SCAND' ,'2000-01-01' ,250) ;
29. SELECT * FROM BOOK ;

OUTPUT :

TITLE	AUTHOR_NAME	PUBLISHER_NAME	PUB_YEAR	UNIT_PRICE
ORACLE	ARORA	PHI	2004-01-01	399
DBMS	BASU	TECHNICAL	2004-02-01	400

TITLE	AUTHOR_NAME	PUBLISHER_NAME	PUB_YEAR	UNIT_PRICE
ADBMS	BASU	TECHNICAL	2004-02-02	450
UNIX	KAPOOR	SCITECH	2000-01-01	300
DOS	KUMAR	SCAND	2000-01-01	250

30. SELECT DISTINCT B1.TITLE FROM BOOK B1 , BOOK B2 WHERE B1.UNIT_PRICE > B2.UNIT_PRICE AND YEAR(B2.PUB_YEAR) ='2004' ;

OUTPUT :

TITLE
DBMS
ADBMS

SOME

31. SELECT DISTINCT TITLE FROM BOOK WHERE UNIT_PRICE > SOME (SELECT UNIT_PRICE FROM BOOK WHERE YEAR(PUB_YEAR) = '2004') ;

OUTPUT :

TITLE
DBMS
ADBMS

NOTE:

1. THE SUBQUERY GENERATES THE SET OF ALL UNIT PRICE VALUE OF BOOKS PUBLISHED IN THE YEAR 2004 . THE **>SOME** COMPARISION IN THE WHERE CLAUSE OF THE OUTER SELECT IS TRUE IF THE UNIT PRICE VALUE OF THE TUPLE IS GREATER THAN AT LEAST ONE MEMBER OF THE SET OF ALL UNIT PRICE VALUES OF THE BOOK PUBLISHED IN THE YEAR 2004 .
2. SQL ALLOWS **<SOME , >SOME , <=SOME , >=SOME , =SOME AND <>SOME** COMPARISION . =SOME IS IDENTICAL TO **IN** AND <> SOME IS IDENTICAL TO **NOT IN** . THE KEYWORD **ANY** IS SIMILAR TO **SOME** .

ALL

32. SELECT DISTINCT TITLE FROM BOOK WHERE UNIT_PRICE > ALL (SELECT UNIT_PRICE FROM BOOK WHERE YEAR(PUB_YEAR) = '2004') ;

OUTPUT :

TITLE

NO ROW SELECTED

NOTE:

1. THE “>ALL” CORRESPONDS TO THE PHRASE ‘GREATER THAN ALL’ .
2. SQL ALLOWS <ALL , <=ALL , <=ALL , >=ALL , < >ALL , =ALL COMPARISIONS. < >ALL IS IDENTICAL TO NOT IN CONSTRUCT.

33. SELECT AUTHOR_NAME FROM BOOK GROUP BY AUTHOR_NAME HAVING SUM(UNIT_PRICE * 2) >=ALL (SELECT SUM(UNIT_PRICE*2) FROM BOOK GROUP BY AUTHOR_NAME) ;

OUTPUT :

AUTHOR_NAME
BASU

TEST FOR EMPTY RELATION

EXISTS IS A TEST FOR NON EMPTY SET . IT IS REPRESENTED BY AN EXPRESSION OF THE FORM 'EXISTS (SELECT …… FROM ……) . SUCH AN EXPRESSION EVALUATES TO TRUE ONLY IF THE RESULT OF EVALUATING THE SUBQUERY REPRESENTED BY THE (SELECT …… FROM ……) IS NON EMPTY.

34. ALTER TABLE BOOK ADD BOOK_ID INT ;
35. UPDATE BOOK SET BOOK_ID =1001 WHERE TITLE = 'ORACLE' ;
36. UPDATE BOOK SET BOOK_ID =1002 WHERE TITLE = 'DBMS' ;
37. UPDATE BOOK SET BOOK_ID =2002 WHERE TITLE = 'ADBMS' ;
38. UPDATE BOOK SET BOOK_ID =2001 WHERE TITLE = 'DOS' ;
39. UPDATE BOOK SET BOOK_ID =2003 WHERE TITLE = 'UNIX' ;
40. SELECT * FROM BOOK ;

OUTPUT :

TITLE	AUTHOR_NAME	PUBLISHER_NAME	PUB_YEAR	UNIT_PRICE	BOOK_ID
ORACLE	ARORA	PHI	2004-01-01	399	1001
DBMS	BASU	TECHNICAL	2004-02-01	400	1002
ADBMS	BASU	TECHNICAL	2004-02-02	450	2002
UNIX	KAPOOR	SCITECH	2000-01-01	300	2003
DOS	KUMAR	SCAND	2000-01-01	250	2001

41. CREATE TABLE ORDER_INFO (ORDER_NO INT(3) , BOOK_ID INT , ORDER_DATE DATE , QTY INT(5) , PRICE INT(5)) ;
42. INSERT INTO ORDER_INFO VALUES(1,1001, '2004-10-10' , 100 , 399) , (2,1002 , '2004-01-11' ,50 , 400) ;
43. GET THE NAMES OF ALL BOOKS FOR WHICH ORDER WAS PLACED SELECT TITLE FROM BOOK WHERE EXISTS (SELECT * FROM ORDER_INFO WHERE BOOK.BOOK_ID = ORDER_INFO.BOOK_ID) ;

OUTPUT :

TITLE
ORACLE
DBMS

SIMILAR TO THE EXISTS WE CAN USE **NOT EXISTS** ALSO

44. DISPLAY THE TITLE OF BOOKS FOR WHICH ORDER IS NOT PLACED
    SELECT TITLE FROM BOOK WHERE NOT EXISTS (SELECT * FROM ORDER_INFO WHERE BOOK.BOOK_ID = ORDER_INFO.BOOK_ID) ;

OUTPUT :

TITLE

ADBMS

UNIX

DOS

DEFAULT VALUE CONCEPT

AT THE TIME OF COLUMN CREATION A 'DEFAULT VALUE' CAN BE ASSIGNED TO IT . WHEN THE USER IS LOADING A RECORD WITH A VALUE AND LEAVES THIS COLUMN EMPTY , THE DBA WILL AUTOMATICALLY LOAD THIS COLUMN WITH THE DEFAULT VALUE SPECIFIED . THE DATATYPE OF THE DEFAULT VALUE SHOULD MATCH THE DEFAULT TYPE OF THE COLUMN. YOU CAN USE THE DEFAULT CALUSE TO SPECIFY ANY DEFAULT VALUE YOU WANT.

45. CREATE TABLE DETAILS (EMPCODE INT ,EMPNAME VARCHAR(20) , MARRIED CHAR(1) DEFAULT 'M') ;
46. INSERT INTO DETAILS VALUES(1001 , 'RAM', 'N') , (1002,'SHYAM' ,' ')

NOTE : Q.NO 45 AND 46 NOT WORKING IN APP-SERVER

CHECK INTEGRITY CONSTRAINT

THE CHECK CONSTRAINT DEFINES A CONDITION THAT EVERY ROW MUST SATISFY . THERE CAN BE MORE THAN ONE CHECK CONSTRAINT ON ACOLUMN , AND THE CHECK CONSTRAINT CAN BE DEFINED AT COLUMN AS WELL AS THE TABLE LEVEL . AT COLUMN LEVEL , THE CONSTRAINT IS DEFINED BY

DEPTID INT(2) CONSTRAINT CK-DEPTID CHECK ((

DEPTID >=10) AND (DEPTID <=99)) ,

AND AT TABLE LEVEL BY

CONSTRAINT CK-DEPTID CHECK ((

DEPTID >=10) AND (DEPTID <=99))

EX. a) ADD CHECK CONSTRAINT ON EMP_CODE COLUMN OF EMPLOYEE1 SO THAT EVERY EMP_CODE SHOULD START WITH 'E' .

b) ADD CHECK CONSTRAINT ON CITY COLUMN OF EMPLOYEE1 SO THAT ONLY THE CITIES 'WGL' , 'KMGR' , 'JAMMI' ,'HYD' ARE ALLOWED

47. CREATE TABLE EMPLOYEE1 (EMP_CODE INT(5) CONSTRAINT CK_EMPCODE CHECK(EMP_CODE LIKE 'E%') , EMP_NAME VARCHAR(20) NOT NULL , CITY VARCHAR(20) CONSTRAINT CK_CITY CHECK (CITY IN ('WGL' , 'KMGR' , 'JAMMI' ,'HYD')) ,SAL INT(5) , PRIMARY KEY (EMP_CODE)) ----- **NOT WORKING IN APP SERVER**

EXAMPLES QUERIES

1. FIND THE NAMES AND AGES OF ALL SAILORS ?
   SELECT DISTINCT S.SNAME , S.AGE FROM SAILORS S WHERE RATING >7 ;
2. FIND THE NAMES OF SAILOR WHO HAVE RESERVED BOAT NUMBER 103 ? SELECT DISTINCT S.SNAME FROM SAILORS S RESERVE R WHERE S.SID = R.SID AND R.BID =103 ;
3. FIND THE SID OF SAILORS WHO HAVE RESERVED A RED BOAT ?
   SELECT R.SID FROM BOATS B , RESERVE R WHERE B.BID = R.BID AND B.COLOR='RED' ;
4. FIND THE COLOR OF BOAT RESERVED BY LUBBER ?
   SELECT B.COLOR FROM SAILORS S , RESERVE R , BOATS B WHERE S.SID =R.SID AND R.BID = B.BID AND S.SNAME='LUBBER'
5. COMPUTE INCREMENT FOR THE RATING OF PERSONS WHO HAVE SAILED TWO DIFFERENT BOATS ON THE SAME DATE ?
   SELECT S.SNAME ,S.RATING + 1 AS RATING FROM SAILORS S , RESERVE R1 , RESERVE R2 WHERE S.SID = R1.SID AND S.SID=R2.SID AND R1.DAY=R2.DAY AND R1.BID<> R2.BID
6. GENERAL EXAMPLE ?
   SELECT S1.SNAME AS NAME1,S2.SNAME AS NAME2 FROM SAILORS S1, SAILORS S2 WHERE 2*S1.RATING =S2.RATING-1
7. FIND THE NAME OF SAILOR WHO HAVE RESERVED A RED OR A GREEN BOAT ? USING UNION ALSO CAN BE WRITTEN
   SELECT S.SNAME FROM SAILORS S ,RESERVE R ,BOATS B WHERE S.SID = R.SID AND R.BID = B.BID AND (B.COLOR ='RED' OR B.COLOR='GREEN')
8. FIND THE NAME OF SAILOR WHO HAVE RESERVED A RED AND A GREEN BOAT? USING INTERSECT CAN ALSO BE WRITTEN
   SELECT S.SNAME FROM SAILORS S ,RESERVE R1 ,RESERVE R2 ,BOATS B1,BOATS B2 WHERE S.SID = R1.SID AND R1.BID = B1.BID AND S.SID =R2.SID AND R2.BID = B2.BID AND B1.COLOR ='RED' AND B2.COLOR='GREEN'
9. FIND THE SIDS OF ALL SAILORS WHO HAVE RESERVED RED BOAT BUT NOT A GREEN BOAT ?
   **SELECT S.SNAME FROM SAILORS S , RESERVE R , BOATS B WHERE S.SID = R.SID AND R.BID=B.BID AND B.COLOR='RED '

   EXCEPT

   (SELECT S1.SNAME FROM SAILORS S1 ,RESERVE R1 , BOATS B1 WHERE S1.SID=R1.SID AND R1.BID=B1.BID AND B1.COLOR= 'GREEN ');

10. FIND THE NAMES OF SAILOR WHO HAVE RESERVED BOAT NUMBER 103 USING IN OPERATOR ?
    SELECT S.SNAME FROM SAILORS S WHERE S.SID IN(SELECT R.SID FROM RESERVE R WHERE R.BID=103)
11. FIND THE NAME OF SAILORS WHO HAVE RESERVED A RED BOAT USING IN OPERATOR ?
    SELECT S.SNAME FROM SAILORS S WHERE S.SID IN(SELECT R.SID FROM RESERVE R WHERE R.BID IN (SELECT B.BID FROM BOATS B WHERE B.COLOR ='RED'))
12. FIND THE NAMES OF SAILOR WHO HAVE NOT RESERVED A RED BOAT USING IN OPERATOR ?
    SELECT S.SNAME FROM SAILORS S WHERE S.SID NOT IN(SELECT R.SID FROM RESERVE R WHERE R.BID IN (SELECT B.BID FROM BOATS B WHERE B.COLOR ='RED'))

COMPLEX QUERIES

COMPLEX QUERIES ARE OFTEN HARD OR IMPOSSIBLE TO WRITE AS A SINGLE SQL BLOCK. THERE ARE TWO WAYS FOR COMPOSING MULTIPLE SQL BLOCKS TO EXPRESS A COMPLEX QUERIES.

a) DERIVED RELATION
b) WITH CLAUSE

DERIEVED RELATION

SQL ALLOWS A SUBQUERIEY EXPRESSION TO BE USED IN THE FROM CLAUSE . IF WE USE SUCH A EXPRESSION , THEN WE MUST GIVE THE RESULT RELATION ANAME AND WE CAN RENAME THE ATTRIBUTES. FOR RENAMING **AS** CLAUSE IS USED.

48. SELECT * FROM EMP2 ;

OUTPUT:

ID	NAME	DOB	GENDER	SALARY	DEPT	COLLEGE	AGE
1	ANUSHA	2000-10-10	F	10000	CSE	KITS	20
2	AKANKSHA	1990-12-10	F	8000	ECE	KITS	30
3	ANJALI	1995-08-14	F	15000	CSE	KITSW	40
4	KARUN	1990-12-06	M	9000	ECE	KITSW	38
5	KARTHIK	1995-12-09	M	8500	CSE	KITS	30
6	PRATHIK	1992-09-13	M	9500	ECE	KITSW	30
7	NULL	1994-05-06	F	120000	CIVIL	KITSW	32