TABLE OF CONTENTS

Chanter 7 DDMC	142-177
Chapter - 7 DBMS 7.1 Write SQL procedure to illustrate DDL Commands.	143
7.1 Write SQL procedure to illustrate DML Commands. 7.2 Write SQL procedure to illustrate DML Commands.	145
7.2 Write SQL procedure to illustrate different operators available in SQL. 7.3 Write SQL procedure to illustrate different operators available in SQL.	147
7.3 Write SQL procedure to illustrate different open	149
7.4 Write SQL procedure to illustrate aggregate functions.	151
7.5 Write SQL procedure to illustrate grouping data.	153
7.6 Write SQL procedure to illustrate Virtual Tables (Views).	
7.7 Write SQL procedure to illustrate Relational Set Operators.	156
7.8 Write SQL Procedure to illustrate Joins.	159
7.9 Write SQL procedure to illustrate Sub Queries.	163
7.10 Write about SQL predefined functions.	166
7.11 Write a PL/SQL block of code to calculate the area of a circle for a value	169
of radius varying from 3 to 7. Store the radius and the corresponding	
values of calculated area in a table AREAS.	
7.12 Write PL/SQL code to illustrate Triggers using Student table.	170
7.13 Write a PL/SQL block of code to create a trigger for the Employee table to	172
display the salary difference between the old values and new values before	
performing INSERT or UPDATE operations.	
7.14 Write a PL/SQL Stored Procedure to insert rows into Employee table.	173
7.15 Write a PL/SQL Stored Procedure to update Product table.	174
7.16 Write a PL/SQL Stored Procedure to illustrate Cursor.	176

1) Write SQL procedure to illustrate DDL Commands.

AIM: Illustrating create table, alter table and drop table commands using Student Table,

PROCEDUR

Step 1: Creating student table.

SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4), S_NAME VARCHAR2(20), S_ADDR VARCHAR2(30));

Step 2: Displaying table structure. SQL> DESCRIBE STUDENT_101;

	S NAME VARO		Name Null?	SOCIATION
VARCHAR2(30)	VARCHAR2(20)	NUMBER(4)	Type	

Step 3: Changing (Modifying) column's data characteristics.

SQL> DESCRIBE STUDENT_101; SQL> ALTER TABLE STUDENT_101 MODIFY(S_NO NUMBER(6));

 S ADDR	S_NAME	S_NO	Name
			Null?
VARCHAR2(30)	VARCHAR2(20)	NOMBER(0)	Type

Step 4: Adding a column.

SQL > DESCRIBE STUDENT_101; SQL> ALTER TABLE STUDENT_101 ADD(S_PHONE NUMBER(10));

INOMBER(10)		O_I IIOINE
NIIIMBED(10)		C PHONE
VARCHAR2(30)		S_ADDR
VARCHAR2(20)		S_NAME
NUMBER(6)		S_NO
Type	Null?	Name

Step 5: Adding PRIMARY KEY to a column.

SQL > DESCRIBE STUDENT_101; SQL > ALTER TABLE STUDENT_101 ADD PRIMARY KEY(S_NO);

Name	Null?	Туре
S_NO	Not Null	NUMBER(6)
S_NAME		VARCHAR2(20
SADDR		
3_ADDK		VARCHAR2(30),
S_PHONE		NIMBERGO

Page 143

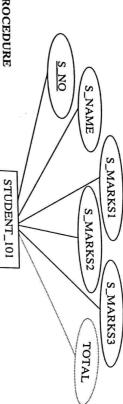
step 6: Removing (Dropping) a table.

				500	101.7	×405	6: KC.
RADE			BONUS	TNAME	SELECT FROM	SOLY DIST	TABLE STUDENT
TABLE	TABLE	TABLE	TABLE	TABTYPE CLUSTERII			_101;

2) Write SQL procedure to illustrate DML Commands.

AIM: Illustrating insert, update, select and delete commands using Student Table,

ER-DIAGRAM



PROCEDURE

Step 1: Creating student table.

SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4) PRIMARY KEY, S_NAME VARCHAR2(20), NUMBER(3), S_MARKS3 NUMBER(3), TOTAL NUMBER(3)); S_MARKS1 NUMBER(3), S_MARKS2

Step 2: Inserting 5 rows into STUDENT_101 table.

I row created. SQL> INSERT INTO STUDENT_101 VALUES(101, 'RAVI', 66, 88, 77, 0);

I row created. SQL> INSERT INTO STUDENT_101 VALUES(102, 'HAMEED', 56, 78, 67, 0);

SQL> INSERT INTO STUDENT_101 VALUES(103, 'THARUN', 76,58,77,0);

SQL> INSERT INTO STUDENT_101 VALUES(104, 'PHANI', 66, 68, 87, 0); I row created.

SQL> INSERT INTO STUDENT_101 VALUES(105, 'ANITHA', 86,67,57,0) I row created.

I row created.

SQL> COMMIT;

Commit complete.

Step 3: Selecting (Displaying) all rows from the STUDENT_101 table.

SQL> SELECT * FROM STUDENT_101;

Page 145

CHAPTER - 7: DBMS

step 4: Updating column TOTAL. # UPUMATE STUDENT_101 SET TOTAL=S_MARKS1+S_MARKS2+S_MARKS3;

5 rows updated.

Step 5; Selecting (Displaying) all rows with updated TOTAL column from the STUDENT_101 table.

SQL> SELECT * FROM STUDENT_101;

_	_			_		_
105	ANITHA	103 PHANI	102 THARUN	101 HAMEED	SNO RAVI	S NAME
	86	66	76	56	66	S_MARKS1
210		68 87 211	58 77 201	78 67 231	88 77 NIARKS3 TOTAL	S MARKS2 C MARKS2

Step 6: Removing (Deleting) a row from the table.

SQL> DELETE FROM STUDENT_101 WHERE S_NO=105;

1 row deleted.

Step 6: Selecting (Displaying) rows after deleting a row from the STUDENT_101 table. SQL> SELECT * FROM STUDENT_101;

104	103	102	0 1	S NO
PHANI	THARUN	HAMEED	RAVI	S_NAME
66	76	56	66	S_MARKS1
68 8	58 77	78 6	-	S_MARKS2 S
 87 221		57 201	-	MARKES TOTAL

3) Write SQL procedure to illustrate different operators available in SQL.

AIM: Illustrating different operators available in SQL on Student table.

PROCEDURE

Step 1: Creating student table.

SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4) PRIMARY KEY S_NAME VARCHAR2(20), S_MARKS1 NUMBER(3), S_MARKS3 NUMBER(3), TOTAL NUMBER(3)); NUMBER(3), S_MARKS2

Step 2: Inserting 5 rows into STUDENT_101 table.

SQL> INSERT INTO STUDENT_101 VALUES(101, 'RAVI', 66,88,77,0); 1 row created.
SQL> INSERT INTO STUDENT_101 VALUES(102, 'HAMEED', 56, 78, 67, 0);

1 row created. lrow creacu.
SQL> INSERT INTO STUDENT_101 VALUES(103, 'THARUN', 76,58,77,0);

SQL> INSERT INTO STUDENT_101 VALUES(105,'ANITHA',86,67,57,0); 110W INTO STUDENT_101 VALUES(104, 'PHANI', 26, 68, 87, 0);

SQL> COMMIT; row created.

Step 3: Updating column TOTAL Commit complete.

SQL> UPDATE STUDENT_101 SET TOTAL=S_MARKS1+S_MARKS2+S_MARKS3;

5 rows updated.

Step 4: Adding new columns AVG and RESULT into STUDENT_101 table.

RESULT VARCHAR2 (6)); SQL> ALTER TABLE STUDENT_101 ADD (AVG NUMBER (5,2),

Table altered.

Step 5: Updating column AVG.

UPDATE STUDENT_101 SET AVG=TOTAL/3;

5 rows updated.

Step 6: Updating column RESULT.

SQL> UPDATE STUDENT_101 SET RESULT='FAIL' WHERE S_MARKS1<35 OR

1 row updated

SQL> UPDATE STUDENT_101 SET RESULT='PASS' WHERE S_MARKS1>=35 AND S_MARKS2>=35 AND S_MARKS3>=35;

4 rows updated

S_MARKS2<35 OR S_MARKS3<35;

Slep 7:Selecting (Displaying) all rows from the STUDENT_101 table. SOL'S SELECT * FROM STUDENT_101;

S NAME ANITHA THARUN RAVI PHANI HAMEED 86 76 S_MARKS1 S_MARKS2 S_MARKS3 TOTAL 58 68 78 67 87 210 22 201 231 60.33 FAII 70.33 67 AVG RESUL PASS PASS PASS

SELECT S_NO, S_NAME, AVG FROM STUDENT_101 WHERE AVG>=75;

101 S_NAME AVG

Step 9; Display S_NO,S_NAME and AVG of all passed students who got above 60% and step 9; Display S_NO,S_NAME and AVG of all passed students who got above 60% and below 75% marks.

SQL> SELECT S_NO,S_NAME,AVG FROM STUDENT_101 WHERE AVG>=60 AND AVG<=75;

SNO 103 102 S_NAME | AVG ANITHA THARUN 70.33 HAMEED

Step 10: Display the name of the students having 'H' as second letter in their name. SQL> SELECT S_NAME FROM STUDENT_101 WHERE S_NAME LIKE '_H%';

THARUN SNAME PHANI

Step 11: Display the Students Number and Names of the students whose S_NO between 102 and 104.

SQL> SELECT S_NO,S_NAME FROM STUDENT_101 WHERE S_NO BETWEEN 102 AND 104;

102	103	102	S_NO	
PHANI	THARUN	HAMEED	S_NAME	

Step 12: Display the Students Number and Names of the students whose S_NO not equal to 102 and 104

SQL> SELECT S_NO,S_NAME FROM STUDENT_101 WHERE S_NO NOT IN(102,104);

ANITHA	105
THARUN	103
RAVI	101
S_NAME	S_NO

Page 148

4) Write SQL procedure to illustrate aggregate functions.

AIM: Illustrating aggregate functions on Employee table.

PROCEDURE

Step 1: Creating Employee table. SQL> CREATE TABLE EMPLOYEE_101 (EMP_NO NUMBER (4), EMP_NAME

EMP_SALARY VARCHAR2(20), EMP_DOJ NUMBER (9,2));

Step 2: inserting rows into EMPLOYEE_101 table.

SQL> INSERT INTO EMPLOYEE 101 VALUES (111, 'RAZAQ', '01-JAN-1990', 'ANANTAPURAMU', 20000);

SQL>INSERT INTO EMPLOYEE_101 VALUES(112, 'RAMESH', I row created. '02-FEB-1989', 'HINDUPUR',25000);

I row created.

'04-MAR-1985', 'DHARMAVARAM',18000); SQL> INSERT INTO EMPLOYEE_101 VALUES(113, 'RADHIKA', I row created.

I row created. '07-FEB-1979', 'ANANTAPURAMU',15000); SQL> INSERT INTO EMPLOYEE_101 VALUES(114, 'RAMANI',

'06-APR-1980', 'HUNDUPUR',17000);

SQL> INSERT INTO EMPLOYEE_101 VALUES(116, 'RAJU', '05-MAY-2000', 'ANANTAPURAMU', 20000);

I row created.

SQL> INSERT INTO EMPLOYEE_101 VALUES(117, 'RAJESH', '03-JUN-2005', 'DHARMAVARAM',16000);

I row created

SQL> INSERT INTO EMPLOYEE_101 VALUES(118, 'REKHA', '02-DEC-2010',

'HINDUPUR', 19000);

I row created.

SQL> COMMIT;

Commit complete.

SQL> INSERT INTO EMPLOYEE_101 VALUES(115, 'RAJESH', DATE, EMP_CITY VARCHAR2 (15), SQL> SELECT MAX (EMP_SALARY) FROM EMPLOYEE_101; SQL> SELECT MIN(EMP_SALARY) FROM EMPLOYEE_101; SQL> SELECT AVG(EMP_SALARY) FROM EMPLOYEE_101; SQL> SELECT SUM (EMP_SALARY) FROM EMPLOYEE_101; SQL> SELECT COUNT (*) FROM EMPLOYEE_101; SQL> SELECT COUNT (EMP_SALARY) FROM EMPLOYEE_101; MAX(EMP_SALARY) MIN(EMP_SALARY) AVG(EMP_SALARY) SUM(EMP_SALARY) COUNT(*) 15000 18750 20000

5(6P 3: Selecting (Displaying) all rows from the EMPLOYEE_101 table.

g rows selected.	118 REKHA	RAJESH	116 RAJU	115 RAJESH	RAMANI	RADHIKA	RAMESH	RAZAQ	NO EMP_NAME
LimaduPUR 16000	HIMAVARAM	3-Jun-05 DHAR APURAMU 17000	5-May-00 ANANT 15000	HIDIDITAPURAMU	7-Feb-79 ANANAVARAM 25000	DUADUPUR	-	AND CITY	EMP_DOJ

Step 4: Performing aggregate functions on the column EMP_SALARY.

COUNT(EMP_SALARY)

25000

Write SQL procedure to illustrate grouping data.

AIM: Illustrating grouping data on Employee table.

PROCEDURE

Step 1: Creating Employee table

SQL> CREATE TABLE EMPLOYEE_101 (EMP_NO EMP_SALARY VARCHAR2 (20), EMP_DOJ NUMBER (9,2)); DATE, EMP_CITY VARCHAR2 (15), NUMBER (4), EMP_NAME

Step 2: inserting rows into EMPLOYEE_101 table.

SQL> INSERT INTO EMPLOYEE_101 VALUES(111, 'RAZAQ', '01-JAN-1990', 'ANANTAPURAMU', 20000);

SQL>INSERT INTO EMPLOYEE_101 VALUES(112, 'RAMESH', '02-FEB-1989, I row created. 'HINDUPUR', 25000);

I row created.

SQL> INSERT INTO EMPLOYEE_101 VALUES(113, 'RADHIKA', 'DHARMAVARAM', 18000);

I row created.

104-MAR-1985',

SQL> INSERT INTO EMPLOYEE_101 VALUES(114, 'RAMANI',

'07-FEB-1979', 'ANANTAPURAMU',15000);

I row created.

SQL> INSERT INTO EMPLOYEE_101 VALUES(115,'RAJESH',

'06-APR-1980', 'HUNDUPUR',17000);

I row created.

SQL> INSERT INTO EMPLOYEE_101 VALUES(116,'RAJU','05-MAY-2000',

'ANANTAPURAMU',20000);

I row created.

SQL> INSERT INTO EMPLOYEE_101 VALUES(117,'RAJESH',

103-JUN-2005', 'DHARMAVARAM', 16000);

I row created.

SQL> INSERT INTO EMPLOYEE_101 VALUES(118,'REKHA','02-DEC-2010',

'HINDUPUR', 19000);

I row created.

SQL> COMMIT;

Commit complete.

Page 151

Selecting (Displaying) all rows from the EMPLOYEE_101 table.

SELECT + FROM EMPLOYEE_101; 3: 3" SELECT + FROM EMPLOYEE 101;

EMP NO RAJESH RAJESH RAJU EMP_NAME REKHA RAMANI RADHIKA RAMESH RAZAQ EMP_DOJ EMP_CITY 2-Dec-10 | HINDUPUR 5-May-00 6-Apr-80 4-Mar-85 DHARMAVARAM 3-Jun-05 7-Feb-79 ANANTAPURAMU 2-Feb-89 I-Jan-90 DHARMAVARAM HUNDUPUR HINDUPUR ANANTAPURAMU ANANTAPURAMU EMP_SALARY 20000 20000 18000 25000 19000 16000 17000 15000

4: Displaying number of employees working in each city. 8 rows selected.

SOL'S SELECT EMP_CITY, COUNT (EMP_NO) FROM EMPLOYEE_101 GROUP BY EMP_CITY;

DHARMAVARAM	HINDUPUR	ANANTAPURAMU	EMP_CITY
w	2	3	COUNT(EMP NO.

Step 5: Displaying rows in ascending order.

SQL> SELECT * FROM EMPLOYEE_101 ORDER BY EMP_NAME;

8 rows selected.	811		111	112	114	111	116		117		115	113	:	EMP_NO	\
cted.	KEKHA	PEIZIT	RAZAQ	KAMESH	N. HATTELL	PAMANI	KAJU	2 111	RAJESH	S. IEGII	RAJESH		RADHIKA	EMP_NAME	
	2-Dec-10		1-Jan-90	2-Feb-89	/-I.cu-/9	7 504 70	5-May-00		3-Jun-05		6-Apr-80		4-Mar-85	EMP_DOJ	
	2-Dec-10 HINDUPUR	OIMUNO HITTER	1-Jan-90 ANANTARITAMIT	2-Feb-89 HINDUPUR	/-I'EU-/9 ANANTAPURAMU	ANT	5-May-00 ANANTAPURAMII	MINDE	DHARMAVARAM	. O. O.	HUNDUPLIR	DITALVIAVARAM	4-Mar-85 DHADMAUADA	EMP_DOJ EMP_CITY	
	19000	20000	2000	25000	15000	20000	20000	10000	17000	17000		18000	-SALAKY	FMP CALARY	1

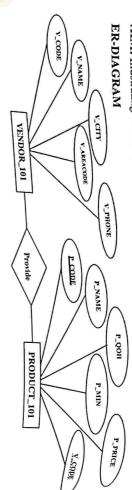
Step 6: Displaying which cities employees taking salary more than 50000.

SQL> SELECT EMP_CITY, SUM (EMP_SALARY) FROM EMPLOYEE_101 GROUP BY EMP_CITY HAVING SUM (EMP_SALARY) >50000;

HINDUPUR	ANANTAPURAMU	EMP_CITY
61000	55000	SUM(EMP_SALARY)

6) Write SQL procedure to illustrate Virtual Tables (Views).

AIM: Illustrating Virtual Tables using VENDOR and PRODUCT tables.



PROCEDURE

Step 1: Creating Vendor table.

SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5),V_NAME V_PHONE NUMBER(10), PRIMARY KEY(V_CODE)); VARCHAR2(20), V_CITY VARCHAR2(15), V_AREACODE VARCHAR2(4),

Table created.

Step 2: inserting rows into table VENDOR_101.

SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM', 'ANANTAPURAMU', 'ATP', 9999888877);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES (102,

'KAMAL','ANANTAPURAMU', 'ATP', 8899889977);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI',

'TDP', 8877997799);

1 row created.

SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI',

'TDP',7373839399);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR',

'HDP',7374565656);

I row created.

Page 153

SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR', 'HDP',7473656565);

> Step 5: Creating Virtual Table(View) to show V_CODE, V_NAME, P_NAME, P_PRICE sep 3: Creating Product table. sep 4: Inserting rows into the table PRODUCT_101. WHERE VENDOR_101.V_CODE=PRODUCT_101.V_CODE P_NAME, P_PRICE, SQL-CREATE VIEW PROD_VIEW AS SELECT VENDOR_101.V_CODE, V_NAME P_QOH, V_PHONE Commit complete. SQL> COMMIT; 1 row created. I row created. SQL> INSERT INTO PRODUCT_101 I row created. I row created. SQL> INSERT INTO PRODUCT_101 MONITORS',40,5800,10,106); DISK',30,4200,10,103); I row created. SQL> INSERT INTO PRODUCT_101 VALUES('LED_18','LED INSERT INTO PRODUCT_101 VALUES('HD_1TB', 'HARD I row created. BOARD', 35, 2800, 10, 103); Commit complete. INSERT INTO PRODUCT_101 VALUES('MB','MOTHER SQL'> 2800.10.103): SOL INSERT INTO PRODUCT_101 SULY ('PCR', 'PROCESSOR', 50, 3200, 10, 101); SOLY COMMIT; 3; Creatine TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY
> 50^{L7} NAME VARCHAR2(20), P_QOH NUMBER(4), P_DDT-I row created. KEY, P. NUMBER (2), V_CODE NUMBER (5) REFERENCES VENDOR_101); SQL' CRUSQL' NAME VARCHAR2 (20), P_QOH NUMBER (4), P_PRICE NUMBER (8,2),
> KEY, P_NAME (2), V_CODE NUMBER (5) REFERENCES VENTON VALUES ('PRTR', 'PRINTER', 50, 7000, 10, 101); VALUES ('KB_MS', 'KEYBOARD_MOUSE_COMBO', 100,670,20,104); P_QOH, V_PHONE FROM VENDOR_101, PRODUCT_101

η Write SQL procedure to illustrate Relational Set Operators.

η Water Properties (UNION, UNION ALL, INTERSECT, MINUS) AIM: UENDOR and PRODUCT tables. on VENDOR and PRODUCT tables.

 $_{PROCEDURE}$

Step 1: Creating Vendor table. SQL CREATE TABLE VENDOR_101(V_CODE NUMBER(5), V_NAME V_PHONE NUMBER(10), PRIMARY KEY(V_CODE)); VARCHAR2(20), V_CITY VARCHAR2(15), V_AREACODE VARCHAR2(4),

Table created.

6 rows selected.

106

KUMAR KIRAN

LED MONITORS

103

2

KAJAL

KEYBOARD_MOUSE_COMBO

7000 670 5800 4200 2800 3200

50

9999888877 7373839399 7473656565 8877997799 8877997799 9999888877

100

40 30 35

KAREEM PRINTER

101

103

KIRAN

MOTHER BOARD HARD DISK

101

V_CODE V_NAME P_NAME

KAREEM PROCESSOR

Step 6: Displaying rows from virtual table PROD_VIEW.

SQL> SELECT * FROM PROD_VIEW;

P_PRICE P_QOH V_PHONE

50

Step 2: inserting rows into table VENDOR_101. SQL INSERT INTO VENDOR_101 VALUES (101, 'KAREEM', 'ANANTAPURAMU', 'ATP', 9999888877);

1 row created.

SQL> INSERT INTO VENDOR_101 VALUES (102,

'KAMAL','ANANTAPURAMU', 'ATP', 8899889977); 1 row created.

I row created. SQL> INSERT INTO VENDOR_101 VALUES (103, 'KIRAN', 'TADIPATRI', 'TDP', 8877997799);

SQL> INSERT INTO VENDOR_101 VALUES(105, 'KESAV', 'HINDUPUR', I row created. SQL> INSERT INTO VENDOR_101 VALUES(104, 'KAJAL', 'TADIPARTI', 'TDP',7373839399);

I row created. 'HDP',7473656565);

I row created.

'HDP', 7374565656);

SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR',

SQL> COMMIT;

Step 3: Creating Product table.

Compile complete.

SQL> CREATE TABLE PRODUCT_101 (P_CODE VARCHAR2 (10) PRIMARY KEY, P_NAME VARCHAR2 (20), P_QOH NUMBER (4), P_PRICE

NUMBER (8,2), P_MIN NUMBER (2), V_CODE NUMBER (5) REFERENCES VENDOR_101);

Table created.

Step 4: Inserting rows into the table PRODUCT_101.

SQL> INSERT INTO PRODUCT_101 VALUES ('PCR', 'PROCESSOR', 50, 3200, 10, 101);

I row created.

35,2800,10,103); SQL> INSERT INTO PRODUCT_101 VALUES('MB','MOTHER BOARD',

SQL> INSERT INTO PRODUCT_101 VALUES('HD_1TB', 'HARD DISK', I row created.

30,4200,10,103);

SQL> INSERT INTO PRODUCT_101 VALUES('LED_18','LED MONITORS', I row created.

40,5800,10,106);

I row created.

SQL> INSERT INTO PRODUCT_101

VALUES('KB_MS','KEYBOARD_MOUSE_COMBO', 100,670,20,104);

I row created.

SQL> INSERT INTO PRODUCT_101

VALUES('PRTR','PRINTER',50,7000,10,101);

1 row created.

SQL> COMMIT;

Commit complete.

Step 5: Implementing Relational Set operators.

1. UNION

SQL> SELECT V_CODE FROM VENDOR_101 UNION SELECT V_CODE FROM PRODUCT_101;

106	105	104	103	102	101	V_CODE

6 rows selected.

Page 157

UNION ALL UNION SELECT V_CODE FROM VENDOR_101 UNION ALL SELECT V_CODE FROM PRODUCT_101;

106 103 104 103 101 106 105 104 102 103 101 V_CODE

12 rows selected.

101

3. INTERSECT

SQL> SELECT V_CODE FROM VENDOR_101 INTERSECT SELECT V_CODE FROM PRODUCT_101;

_			
104	103	101	V_CODI

106

4 rows selected.

MINUS

SQL> SELECT V_CODE FROM VENDOR_101 MINUS SELECT V_CODE FROM PRODUCT_101;

10	10	<
5	2	CO
		DE

4: Inserting rows into the table PRODUCT_101.

INSERT INTO PRODUCT_101

VALUES ('PCR', 'PROCESSOR', 50, 3200, 10, 101);

[10" INSERT INTO PRODUCT_101 VALUES('MB','MOTHER 60^{L^2} ____npn'.35,2800,10,103).

BOARD',35,2800,10,103);

LAB PROGRAM -8

8) Write SQL Procedure to illustrate Joins

AIM: Illustrating SQL Join operations on VENDOR and PRODUCT tables.

Step 1: Creating Vendor table.

SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5), V_NAME V_PHONE NUMBER(10), PRIMARY KEY(V_CODE)); VARCHAR2 (20), V_CITY VARCHAR2 (15), V_AREACODE VARCHAR2 (4),

1 row created.

INSERT INTO PRODUCT_101 VALUES('HD_1TB','HARD

pisk',30,4200,10,103);

1 row created.

INSERT INTO PRODUCT_101 VALUES('LED_18','LED

MONITORS',40,5800,10,106);

SQL> INSERT INTO PRODUCT_101

I row created.

Table created.

Step 2: inserting rows into table VENDOR_101.

SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM', 'ANANTAPURAMU', 'ATP', 9999888877);

1 row created.

SQL> INSERT INTO VENDOR_101 VALUES (102, 'KAMAL','ANANTAPURAMU', 'ATP', 8899889977);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI', 'TDP', 8877997799);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI', 'TDP',7373839399);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR', 'HDP',7374565656);

I row created

SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR', 'HDP',7473656565);

I row created.

SQL> COMMIT;

Commit complete.

Step 3 : Creating Product table.

SQL> CREATE TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY NUMBER(8,2), P_MIN NUMBER(2), V_CODE NUMBER(5) REFERENCES KEY, P_NAME VARCHAR2(20), P_QOH NUMBER(4), P_PRICE

103

101

V

106 103

KUMAR KIRAN KIRAN KAREEM

> HARD DISK MOTHER BOARD PROCESSOR

LED MONITORS

Step 5: Performing join operations. SQL> SELECT V.V_CODE, V_NAME, P_NAME FROM VENDOR_101 SQL> COMMIT; 1 row created. PRODUCT_101; PRODUCT_101 P WHERE V.V_CODE=P.V_CODE; SQL> SELECT V.V_CODE, V_NAME, P_NAME FROM VENDOR_101 V, 1. Natural Join (Inner Join) Commit complete. I row created. SQL> INSERT INTO PRODUCT_101 Code for Oracle 9i, 10g Code for Oracle 8 V_CODE VALUES ('PRTR', 'PRINTER', 50, 7000, 10, 101); VALUES ('KB_MS', 'KEYBOARD_MOUSE_COMBO', 100,670,20,104); V_NAME P_NAME

104 KAJAL KEYBOARD_MOUSE_COMBO 101 KAREEM PRINTER

6 rows selected.

2. Cross Join

Code for Oracle8

SQL> SELECT * FROM VENDOR_101, PRODUCT_101;

Code for Oracle 9i, 10g

SQL> SELECT * FROM VENDOR_101 CROSS JOIN PRODUCT_101; It displays 36 rows (product of VENDER_101 and PRODUCT_101).

Left Outer Join

Code for Oracle8

SQL> SELECT V.V_CODE,V_NAME,P_NAME FROM VENDOR_101 V,
PRODUCT_101 P WHERE V.V_CODE=P.V_CODE(+);

Code for Oracle 9i,10g

SQL> SELECT V.V_CODE,V_NAME,P_NAME FROM VENDOR_101 V

LEFT OUTER JOIN PRODUCT_101 P ON V.V_CODE=P.V_CODE;

		אוגגונד
V_CODE	V_CODE V_NAME	P_NAME
101	KAREEM	PROCESSOR
101	KAREEM	PRINTER
102	KAMAL	
103	KIRAN	MOTHER BOARD
103	KIRAN	HARD DISK
104	KAJAL	KEYBOARD_MOUSE_COMBO
105	KESAV	
106 / 1	KUMAR	LED MONITORS

8 rows selected.

. Right Outer Join

Code for Oracle8

SQL> SELECT V.V_CODE,V_NAME,P_NAME FROM VENDOR_101 V,
PRODUCT_101 P WHERE V.V_CODE(+)=P.V_CODE;

Code for Oracle 9i,10g

SQL> SELECT V.V_CODE, V_NAME, P_NAME FROM VENDOR_101 V
RIGHT OUTER JOIN PRODUCT_101 P ON V.V_CODE=P.V_CODE;

Dated	<u>1</u> 0]	<u>i</u> /2	8/3	<u></u>	3/3	<u> </u>	CODI
	KAREEM	KAJAL	KUMAR	KIRAN	KIRAN	KAREEM	CODE V_NAME
	PRINTER COMBO	KEYBOARD MOT	LED MONITORS	HARD DISK	MOTHER BOAR	PROCESSOR	P_NAME

6 rows selected.

5. Full Outer Join Code for Oracle8

Code:

Co

Code for Oracle 9i,10g

COUTER JOIN PRODUCT_101 P ON V.V_CODE_B TO FULL

101 KA							105 K	104 K	103 K	103 K	102 K	101	101 I	V_CODE
KAREEM	KAJAL	KUMAR	KIRAN		KAREEM	KUMAR	KESAV	KAJAL	KIRAN	KIRAN	KAMAL	KAREEM	KAREEM	V_NAME
PRINTER	KEYBOARD_MOUSE_COMBO	LED MONITORS	HARD DISK	MOTHER BOARD	PROCESSOR	LED MONITORS		KEYBOARD_MOUSE COMBO	HARD DISK	MOTHER BOARD		PRINTER	PROCESSOR	P_NAME

inserting rows into the table PRODUCT_101.

SOL INSERT INTO PRODUCT_101

50L7 ('PCR', 'PROCESSOR', 50, 3200, 10, 101);

LAB PROGRAM -9

9) Write SQL procedure to illustrate Sub Queries

AIM: Illustrating Sub Queries using Vendor and Product tables.

PROCEDURE

Step 1: Creating Vendor table.

SQL> CREATE TABLE VENDOR_101(V_CODE NUMBER(5),V_NAME V_PHONE NUMBER(10), PRIMARY KEY(V_CODE)); VARCHAR2 (20), V_CITY VARCHAR2 (15), V_AREACODE VARCHAR2 (4),

Table created.

Step 2: inserting rows into table VENDOR_101.

SQL> INSERT INTO VENDOR_101 VALUES (101, 'KAREEM', 'ANANTAPURAMU', 'ATP', 9999888877);

SQL> INSERT INTO VENDOR_101 VALUES (102, `KAMAL','ANANTAPURAMU', 'ATP', 8899889977);

I row created.

I row created.

SQL> INSERT INTO VENDOR_101 VALUES (103,'KIRAN','TADIPATRI', 'TDP', 8877997799);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES(104,'KAJAL', 'TADIPARTI', 'TDP',7373839399);

I row created.

SQL> INSERT INTO VENDOR_101 VALUES(105,'KESAV','HINDUPUR', 'HDP',7374565656);

I row created.

SQL>INSERT INTO VENDOR_101 VALUES(106,'KUMAR','HINDUPUR', 'HDP',7473656565);

I row created.

SQL> COMMIT;

Commit complete.

Step 3: Creating Product table.

SQL> CREATE TABLE PRODUCT_101(P_CODE VARCHAR2(10) PRIMARY NUMBER(8,2), P_MIN NUMBER(2), V_CODE NUMBER(5) REFERENCES KEY, P_NAME VARCHAR2(20), P_QOH NUMBER(4), P_PRICE VENDOR_101);

> Sup 5: Implementing Sub Queries. IN(SELECT V_CODE FROM PRODUCT_101); 1. Display vendors who are providing products I row created. SQL> SELECT V_CODE, V_NAME, V_PHONE FROM VENDOR_101 WHERE V_CODE Commit complete. SQL> COMMIT; 1 row created. 591 INSERT INTO PRODUCT_101 I row created. VALUES ('PRTR', 'PRINTER', 50, 7000, 10, 101); I row created. I row created. 40,5800,10,106); INSERT INTO PRODUCT_101 VALUES('KB_MS', 30,4200,10,103); 35,2800,10,103); 'KEYBOARD_MOUSE_COMBO', 100,670,20,104); I row created. INSERT INTO PRODUCT_101 VALUES ('LED_18', 'LED MONITORS', 10W INSERT INTO PRODUCT_101 VALUES('HD_ITB','HARD DISK', I TOW INSERT INTO PRODUCT_101 VALUES ('MB', 'MOTHER BOARD', \$0^1, 10, 103); 103 104 101 V_CODE KIRAN V_NAME KAJAL KAREEM V_PHONE 7373839399 9999888877 8877997799

4 rows selected.

106

KUMAR

7473656565

Page 163

2. Display products with a price greater than or equal to the average product

SQL SELECT P NAME, P PRICE FROM PRODUCT 101 WHERE P PRICE >= (SELECT AVG(P_PRICE) FROM PRODUCT_101);

P NAME	P_PRICE
HARD DISK	4200
LED MONITORS	5800
PRINTER	7000

3 rows selected.

3. Delete the products that are provided by vendors with area code 'HDp'.

FROM VENDOR_101 WHERE V_AREACODE='HDP'); SQL> DELETE FROM PRODUCT_101 WHERE V_CODE IN (SELECT V_CODE

I row deleted.

SQL's SELECT * FROM PRODUCT_101;

r	_			V		7
	PRTR	KB_MS	нр_ітв	МВ	PCR	P_CODE
	PRINTER	KEYBOARD_MOUSE_COMBO	HARD DISK	MOTHER BOARD	PROCESSOR	P_CODE P_NAME
	50	100	30	35	50	P_QOH
	7000	670	4200	2800	3200	P_QOH P_PRICE
	10 101	20 104	10 103	10 103	10 101	P_MIN V_CODE

5 rows selected.

in 1: Number functions. MOCEDURE Mustrating SQL predefined functions. While about SQL predefined functions. 1. ABS SELECT ABS (-500) FROM DUAL; ABS(-500)

500

3. FLOOR 2 CEIL SELECT CEIL (123.55) FROM DUAL; CEIL(123.55)

4. MOD SOL> SELECT FLOOR (123.55) FROM DUAL; 123 FLOOR(123.55)

SQL> SELECT MOD(10,3) FROM DUAL, SQL> SELECT MOD(10,5) FROM DUAL; MOD(10,5) MOD(10,5)

5. POWER

SQL> SELECT POWER (2,4) FROM DUAL;

POWER(2,4)

6. ROUND

SQL> SELECT ROUND (123.55,0), ROUND (123.55,1) FROM DUAL;

124 12	ROUND(123.55,0) RC
123.6	ROUND(123.55,1)

7. SQRT

SQL> SELECT SQRT(64) FROM DUAL;

SQRT(64)

8. TRUNC

SQL> SELECT TRUNC(123.55,0), TRUNC(123.55,1) FROM DUAL;

123 TRUNC(123.55,0) TRUNC(123.55,1) 123.5

Step 2: Character functions.

1. INITCAP

SQL> SELECT INITCAP('anantapuramu') FROM DUAL;

Anantapuramu INITCAP('ANANTAPURAMU')

LENGTH('ANANTAPURAMU')

SQL> SELECT LENGTH ('ANANTAPURAMU') FROM DUAL; 2. LENGTH

SUBSTR

SQL> SELECT SUBSTR('ANANTAPURAMU', 3, 3) FROM DUAL; 3. SUBSTR

ANT

4. LOWER

SQL> SELECT LOWER ('ANANTAPURAMU') FROM DUAL;

anantapuramu LOWER ('ANANT APURAMU')

5. UPPER

SQL> SELECT UPPER ('anantapuramu') FROM DUAL;

UPPER('ANANT APURAMU')

ANANTAPURAMU

SQL> SELECT ADD_MONTHS('15-JAN-2014',5) FROM DUAL;

ADD_MONTH

15-JUN-14

Step 3: Date functions.

1. ADD_MONTHS

Page 167

FROM DUAL; 1. LAST_DAY 3. MONTHS_BETWEEN 3. MON'SELECT MONTHS_BETWEEN('15-MAR-2014','15-JAN-2014') 1. LAST_DAY('15-JAN-2014') FROM DUAL;

31-JAN-14

4. NEXT_DAY 4. NW. SELECT NEXT_DAY('20-JAN-2014','MON') FROM DUAL; MONTHS_BETWEEN('15-MAR-2014','15-JAN-2014')

11) Write a PL/SQL block of calculated area in a varying from 3 to 7. Store the radius and the corresponding values of calculated area in a large in a lar 11) Write a PL/SQL block of code to calculate the area of a circle for a Value of Value of Calculated area.

AIM: Creating PL/SQL block of code to insert rows into AREAS table.

PROCEDURE

Step 1: Creating AREAS table

Creating ALLIAN SQL> CREATE TABLE AREAS (RADIUS NUMBER (5), AREA NUMBER (10,2));

Step 2: Writing PL/SQL code.

AREA NUMBER (14,2); RADIUS NUMBER (5); PI CONSTANT NUMBER (4,2):=3.14;

BEGIN

WHILE RADIUS <=7 RADIUS :=3;

RADIUS:=RADIUS+1; INSERT INTO AREAS VALUES (RADIUS, AREA); AREA := PI*POWER(RADIUS,2);

END LOOP;

END;

Step 3: Displaying rows from the table AREAS. PL/SQL procedure successfully completed.

SQL> SELECT * FROM AREAS;

7	6	U	4		RADIUS
153.86	113.04	78.5	50.24	28.26	AREA

LAB PROGRAM -12

step 1: Creating student table. PROCEDURE AM: Illustrating triggers using student table. Write PL/SQL code to illustrate Triggers using Student table. S MARKS NUMBER(3), TOTAL NUMBER(3), AVERAGE NUMBER(5,2), RESULT S NAME VARCHAR2(20), S MARKS1 NUMBER(3), S MARKS2 NUMBER(3), SQL> CREATE TABLE STUDENT_101 (S_NO NUMBER(4) PRIMARY KEY,

VARCHAR2 (6));

Step 2: Creating trigger to update columns TOTAL, AVERAGE, RESULT after inserting a row. SQL> CREATE OR REPLACE TRIGGER STUDENT_TRG AFTER

UPDATE STUDENT_101 SET TOTAL=S_MARKS1+S_MARKS2+S_MARKS3; UPDATE STUDENT_101 SET AVERAGE=TOTAL/3; UPDATE STUDENT_101 SET RESULT='PASS' WHERE S_MARKS1>=35 AND STUDENT_101

UPDATE STUDENT_101 SET RESULT='FAIL' WHERE S_MARKS1<35 OR S_MARKS2<35 OR S_MARKS3<35; S_MARKS2>=35 AND S_MARKS3>=35;

Trigger created.

SQL> INSERT INTO STUDENT_101 VALUES(101,'RAVI',66,88,77,0,0,' '); Step 3: Inserting 5 rows into STUDENT_101 table.

SQL> INSERT INTO STUDENT_101 VALUES(102,'HAMEED',56,78,67,0,0,' '); SQL> INSERT INTO STUDENT_101 VALUES(103, 'THARUN', 76,58,77,0,0,' ');

SQL> INSERT INTO STUDENT_101 VALUES(104, 'PHANI', 26, 68, 87, 0, 0, '');

1 row created.

SQL> INSERT INTO STUDENT_101 VALUES(105,'ANITHA', 86,67,57,0,0,'''); 1 row created.

SQL> COMMIT;

Step 4: Selecting (Displaying) all rows from the STUDENT_101 table.

SELECT * FROM STUDENT_101;

105	104	103	102	101	ON O	2 10
ANITHA	PHANI	THARUN 76	НАМЕЕД 36	KAVI	S_NAME	SQL> St
86	26	76	30	00	O_MANAGE	LECI A FRO
0,	67	20	50	78	88	S MARKS2
	57	87	77	67	71	S_MARKS3
	210	181	211	201	231	TOTAL
	70	60.33	70.33	67	77	SQL> SELECT FINE S MARKS2 S_MARKS3 TOTAL AVERAGE
FASS	A	ASS	86.			

LAB PROGRAM -13

Write a 1... Write while a PL/SQL block of code to create a trigger for the Employee table to display the

UPDATE operations. (PDAILE or the Employee table to display the salary differences, lM: Create a trigger for the Employee table to display the salary differences.

PROCEDURE Step 1: Creating Employee table. SQL' CREATE TABLE EMPLOYEE_101 (EMP_NO EMP_SALARY VARCHAR2 (20), EMP_DOJ DATE, EMP_CITY VARCHAR2 (15), NUMBER (9,2)); NUMBER(4),EMP_NAME

Sep 2: Creating Trigger.
Sep Set Server out on;
SQL> CREATE OR REPLACE FOR EACH ROW WHEN (NEW.EMP_NO>0) UPDATE OR INSERT ON EMPLOYEE REPLACE TRIGGER EMP_SALARY_CHANGE BEFORE

DECLARE

END; SAL_DIFF NUMBER(8,2); DBMS_OUTPUT.PUT_LINE('NEW SALARY: ' | :NEW.EMP_SALARY); DBMS_OUTPUT_LINE('OLD SALARY : ' | | :OLD.EMP_SALARY); DBMS_OUTPUT.PUT_LINE('EMPLOYEE NO. : '|| :NEW.EMP_NO); SAL_DIFF:= :NEW.EMP_SALARY-:OLD.EMP_SALARY; DBMS_OUTPUT.PUT_LINE('SALARY DIFFERENCE : ' | | SAL_DIFF);

Trigger created.

Step 3: inserting a row into EMPLOYEE_101 table.

SQL> INSERT INTO EMPLOYEE_101 VALUES(119, 'RAJINI', '02-MAR-1998', 'TADIPATRI',25000)

EMPLOYEE NO.: 119

OLD SALARY:

NEW SALARY: 25000

SALARY DIFFERENCE:

Step 4: Updating column EMP_SALARY. SQL> UPDATE EMPLOYEE_101 SET EMP_NO=119; EMPLOYEE NO.: 119

EMP_SALARY=EMP_SALARY+1000 WHERE

OLD SALARY: 25000 NEW SALARY: 26000

SALARY DIFFERENCE: 1000

I row updated.

Page 172

AIM: PL/SQL Stored Procedure to insert rows into Employee table. 14) Write a PL/SQL Stored Procedure to insert rows into Employee table.

PROCEDURE

SQL> CREATE TABLE EMPLOYES_102 (EMP_NO VARCHAR2(20), EMP_DOJ DATE, EMP_CITY Step 1: Creating Employee table. NUMBER (9,2)); VARCHAR2(15), EMP_SALARY NUMBER (4), EMP_NAME

Step 2: Creating Stored Procedure. VARCHAR, E_DOJ DATE, E_CITY VARCHAR, E_ SALARY SQL'S CREATE OR REPLACE PROCEDURE EMP_ADD (E_NO NUMBER, E_NAME NUMBER)

E_CITY, E_SALARY);

BEGIN INSERT INTO EMPLOYEE_102 VALUES(E_NO,E_NAME,E_DOJ,

DBMS_OUTPUT_LINE('Employee '||E_NAME||' is Added.');

Step 3: Executing Stored Procedure.

Procedure created

SQL> EXEC EMP_ADD(110,'RAMANA','02-MAR-1999', 'TADIPATRI',35000);

Employee RAMANA is Added

PL/SQL procedure successfully completed

Step 4: Selecting (Displaying) all rows from the EMPLOYEE_102 table.

SQL> SELECT * FROM EMPLOYEE_102;

00000	THE PARTY IN	02-141-70	CAMPAGA	TIV
35000	TADIPATRI	02-MAB-99	BAMANA	110
TWITT TOTAL TANK		F1411 15 Co	Train Training	_ TATE
EMP SAI ADV	EMP CITY	EMP DOI	EMP NAME	EMB NO

sup 1: Creating Product table. PROCEDURE While a PL/SQL Stored Procedure to update Product table. While A Fre stored Procedure to update column P_DISCOUNT of the PRODUCT table.

IN : pUSQL Stored Procedure to update column P_DISCOUNT of the PRODUCT table. FOREATE TABLE PRODUCT_102(P_CODE VARCHAR2(10) PRIMARY SQLP NAME VARCHAR2(20), P_QOH NUMBER(A) NUMBER(8,2), P_MIN NUMBER(2), P_DISCOUNT NUMBER(4,2)); KEY, P. NAME VARCHAR2 (20), P_QOH NUMBER (4), P_PRICE

Table created.

Step 2: Inserting rows into the table PRODUCT_102. SOL INSERT INTO PRODUCT_102 VALUES('PCR', 'PROCESSOR', 50, 3200, 10, 0.5);

1 row created. SQL> INSERT INTO PRODUCT_102 VALUES('MB','MOTHER BOARD',35,2800,10,0.75);

1 row created. SQL> INSERT INTO PRODUCT_102 VALUES('HDD', 'HARD DISK', 30, 4200, 10, 1.2);

SQL> INSERT INTO PRODUCT_102 1 row created. VALUES ('MNTR', 'LED MONITORS', 40,5800,10,0.8);

SQL> INSERT INTO PRODUCT_102 1 row created.

VALUES ('PRTR', 'PRINTER', 50, 7000, 10, 1);

1 row created.

SQL> COMMIT;

Commit complete.

Step 3: Displaying rows.

SQL> SELECT * FROM PRODUCT_102;

PRTR	MNTR LED	HDD HAR	MB MOT	PCR PRO	P_CODE P_NAME	
PRINTED	LED MONITORS	HARD DISK	MOTHER BOARD	PROCESSOR	ME	
50	40	30	35	50	P_QOH	
7000	5800	4200	2800	3200	P_QOH P_PRICE	
10	10	10	10	10	P_MIN	
_	.8	1.2	.75	is	P_DISCOUNT	

Step 4: Creating Stored Procedure to add additional 5% discount for all products when the bound of the minimum quantity. SQL> CREATE OR REPLACE PROCEDURE PROD_DISCOUNT

BEGIN

UPDATE PRODUCT_102 SET P_DISCOUNT=P_DISCOUNT+0.05 WHERE P_QOH>=P_MIN*3;

DBMS_OUTPUT.PUT_LINE(' Table Updated');

SQL> EXEC PROD_DISCOUNT; Table Updated

Step 5: Executing Stored Procedure.

Procedure created.

PL/SQL procedure successfully completed.

Step 6: Displaying rows.

SQL> SELECT * FROM PRODUCT_102;

				_	
PRTR	MNTR	HDD	MB	PCR	P_CODE
PRINTER	LED MONITORS	HARD DISK	MOTHER BOARD	PROCESSOR	P_NAME
50	40	30	35	50	P_QOH
7000	5800	4200	2800	3200	P_QOH P_PRICE
	10	10	10	10	P_MIN
1.05	.85	1.25	.∞	.55	P DISCOUR
	50 7000 10	R LED MONITORS 40 5800 10 PRINTER 50 7000 10	HARD DISK 30 4200 10 LED MONITORS 40 5800 10 PRINTER 50 7000 10	MOTHER BOARD 35 2800 10 HARD DISK 30 4200 10 R LED MONITORS 40 5800 10 PRINTER 50 7000 10	PROCESSOR 50 3200 10 MOTHER BOARD 35 2800 10 HARD DISK 30 4200 10 R LED MONITORS 40 5800 10 PRINTER 50 7000 10

Write a PL/SQL Stored Procedure to illustrate Cursor.

16) Write A PL/SQL Stored Procedure to Cursor.

16) Write PL/SQL Stored Procedure to Cursor using PRODUCT table, M. PL/SQL Stored Procedure to Cursor using PRODUCT table, M. P. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. P. Stored Procedure to Cursor using PRODUCT table, M. P. Stored Procedure to Cursor using PRODUCT table, M. P. Stored Procedure to Cursor using Procedu

PROCEDURE

Sup 1: Creating Product table. CREATE TABLE PRODUCT_102(P_CODE VARCHAR2(10) PRIMARY KEY, P_NAME VARCHAR2(20), P_QOH NUMBER(4), P_PRICE

Table created.

NUMBER (8,2), P_MIN NUMBER (2), P_DISCOUNT NUMBER (4,2));

sup 2: Inserting rows into the table PRODUCT_102.

SOL INSERT INTO PRODUCT_102

VALUES('PCR','PROCESSOR',50,3200,10,0.5);

I row created.

SQL> INSERT INTO PRODUCT_102 VALUES ('MB', 'MOTHER BOARD',35,2800,10,0.75);

I row created.

SQL> INSERT INTO PRODUCT_102 VALUES ('HDD', 'HARD DISK',30,4200,10,1.2);

I row created.

SQL> INSERT INTO PRODUCT_102 VALUES('MNTR','LED MONITORS', 40, 5800, 10, 0.8);

I row created.

SQL> INSERT INTO PRODUCT_102

VALUES('PRTR','PRINTER',50,7000,10,1);

I row created.

SQL> COMMIT;

Commit complete

Step 3: Displaying rows.

SQL> SELECT * FROM PRODUCT_102:

-	10	7000	50	PRINTER	PRTR
ò	10	5800	40	LED MONITORS	MNTR
1.3	10	4200	30	HARD DISK	HDD
.75	10	2800	35	MOTHER BOARD	MB
į,	10	3200	50	PROCESSOR	PCR
P_DISCOUNT	P_MIN	P_PRICE P_MIN	P_QOH	P_CODE P_NAME	P_CODE

Page 176

```
Step 4: Creating Stored Procedure to list all products that have P_QOH is greater than the
      SQL>CREATE OR REPLACE PROCEDURE PRO_CURSOR IS
          CODE PRODUCT_102.P_CODE%TYPE;
          NAME PRODUCT 102.P_NAME%TYPE;
           CURSOR P CURSOR IS
                SELECT P_CODE, P_NAME FROM PRODUCT_102 WHERE
                     P_QOH>=(SELECT AVG(P_QOH) FROM PRODUCT_102);
          BEGIN
          DBMS_OUTPUT_LINE('PRODUCT WITH P_QOH > AVG(P_QOH)');
          DBMS_OUTPUT.PUT_LINE('----');
          DBMS_OUTPUT.PUT_LINE('PRODUCT_CODE PRODUCT_NAME');
          DBMS_OUTPUT.PUT_LINE('----');
          OPEN P CURSOR;
          LOOP
               FETCH P_CURSOR INTO CODE, NAME;
               EXIT WHEN P_CURSOR%NOTFOUND;
               DBMS_OUTPUT_LINE(' '||CODE||' '|NAME);
          END LOOP;
          DBMS_OUTPUT.PUT_LINE('----');
          DBMS_OUTPUT_LINE('TOTAL NUMBER OF PRODUCTS PROCESSED:
                '||P_CURSOR%ROWCOUNT);
          CLOSE P_CURSOR;
     END;
     1
     Procedure created.
     SQL> SET SERVEROUT ON;
     SQL> EXEC PRO_CURSOR;
     PRODUCT WITH P_QOH > AVG(P_QOH)
     PRODUCT_CODE PRODUCT_NAME
     PCR
                   PROCESSOR
     PRTR
```

TOTAL NUMBER OF PRODUCTS PROCESSED: 2 PL/SQL procedure successfully completed.

PRINTER