********** PRACTICAL NO. 4 **********

Problem Statement:

Write and Execute SQL indexing queries for

data warehouse.

*/

/*Q1. Create individual b-tree indexes on the following columns of the table:

customers_copy_btree

(a) cust_gender

(b) cust_year_of_birth

(c) cust_last_name

(d) cust_street_address

How long does it take to create the indexes?*/

```
create table customer_copy_btree as select * from sh.customers;
create index customer_copy_btree_gender_idx on customer_copy_btree(cust_gender);
```

Elapsed: 00:00:00.20

```
create index customer_copy_btree_yob_idx on customer_copy_btree(cust_year_of_birth);
```

Elapsed: 00:00:00.03

```
create index customer_copy_btree_lname_idx on customer_copy_btree(cust_last_name);
```

Elapsed: 00:00:00.06

```
create index customer_copy_btree_stra_idx on customer_copy_btree(cust_street_address);
```

Elapsed: 00:00:00.04

COLUMN TIME INDEX BTREE

CREATION TIME Elapsed: 00:00:00.29

GENDER Elapsed: 00:00:00.29

YEAR OF BIRTH Elapsed: 00:00:00.23

LAST NAME Elapsed: 00:00:00.04

STREET ADDRESS Elapsed: 00:00:00.04

/*Q2. Create bitmap indexes on the above columns. How long does it take to create bitmap indexes? Compare it with the results of btree index creation.*/

SQL> CREATE TABLE CUSTOMER_BITMAP AS (SELECT * FROM SH.CUSTOMERS);

Table created.

Elapsed: 00:00:00.29

SQL> create bitmap index cus_gender_bitmap_idx on customer_bitmap(cust_gender);

Index created.

Elapsed: 00:00:00.23

SQL> create bitmap index cus_year_of_birth_bitmap_idx on customer_bitmap(cust_year_of_birth);

Index created.

Elapsed: 00:00:00.04

SQL> create bitmap index cus_last_name_bitmap_idx on customer_bitmap(cust_last_name);

Index created.

Elapsed: 00:00:00.03

SQL> create bitmap index cus_street_address_bitmap_idx on customer_bitmap(cust_street_address);

Index created.

Elapsed: 00:00:00.04

COMPARISON CHART

COLUMN	TIME INDEX BTREE	TIME INDEX BITMAP
CREATION TIME	Elapsed: 00:00:03.06	Elapsed: 00:00:02.27
GENDER	Elapsed: 00:00:00.38	Elapsed: 00:00:00.34
YEAR OF BIRTH	Elapsed: 00:00:00.64	Elapsed: 00:00:00.03
LAST NAME	Elapsed: 00:00:00.32	Elapsed: 00:00:00.05
STREET ADDRESS	Elapsed: 00:00:01.03	Elapsed: 00:00:00.14

/*Q3. Do as directed:

(a) Find the size of each segment: customers_copy_bitmap and customers_copy_btree

(b) The b-tree index range for high and low cardinality address index.

(c) The bitmap index range for high and low cardinality address index.

*/

--a

```
select segment_name,
bytes/1024/1024 "Size in MB"
from user_segments
```

where segment_name like '%CUSTOMER_COPY%';

--c

select segment_name,

bytes/1024/1024 "Size in MB"

from user_segments

where segment_name like '%BTREE%';

SEGMENT_NAME	Size in MB
CUS_GENDER_BTREE_IDX	.875
CUS_YEAR_OF_BIRTH_BTREE_IDX	1
CUS_LAST_NAME_BTREE_IDX	2
CUS_GENDER_BTREE_IDXX	.875
CUS_YEAR_OF_BIRTH_BTREE_IDXX	1
CUS_LAST_NAME_BTREE_IDXX	2

--d

select segment_name,

bytes/1024/1024 "Size in MB"

from user_segments

where segment_name like '%BITMAP%';

SEGMENT_NAME	Size in MB

CUSTOMER_BITMAPS 12

CUS_GENDER_BITMAP_IDX .0625

CUS_YEAR_OF_BIRTH_BITMAP_IDX .1875

CUS_LAST_NAME_BITMAP_IDX .125

CUS_STREET_ADDRESS_BITMAP_IDX 3

Elapsed: 00:00:00.06

/*Q4. Use year of birth, which had 75 different values in our test data as filter column. Also show the execution plan for both indexes- btree and bitmap. Compare the cost of the execution plan for b-tree and bitmap indexes.*/

set autotrace on;

select * from customer_copy_btree where cust_year_of_birth=1967;

956 rows selected.

Elapsed: 00:00:12.15

Execution Plan

Plan hash value: 3388583990

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

| 0 | SELECT STATEMENT | | 956 | 278K| 406 (1)| 00:00:05 |

|* 1 | TABLE ACCESS FULL| CUSTOMERTEST | 956 | 278K| 406 (1)| 00:00:05 |

---

select * from customer_copy_bitmap where cust_year_of_birth=1967;

Execution Plan

Plan hash value: 2649141227

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

| 0 | SELECT STATEMENT | | 956 | 278K| 98 (0)| 00:00:02 |

| 1 | TABLE ACCESS BY INDEX ROWID | CUSTOMER_COPY_BITMAP | 956 | 278K| 98 (0)| 00:00:02 |

| 2 | BITMAP CONVERSION TO ROWIDS| | | | | |

|* 3 | BITMAP INDEX SINGLE VALUE | CUST_COPY_BITMAP_YOB_IDX | | | |

For btree indexed table,a full table scan is run.

The cost of the execution plan against the bitmap indexed table

is shown above.

/*Q5. Show that update to the bitmap indexed column takes a bit longer than the b-tree indexed

column.

(a) Create new indexes on cust_id column of btree and bitmap table.

(b) Set the timing on

(c) Write a PL/SQL procedure for each table as directed:

i. Create new columns- upd_cust_id and cust_yob_value with number format.

ii. In loop of 500 allot random values to both columns

iii. set cust_year_of_birth = cust_yob_value and consider cust_id = upd_cust_id

(d) What is the size of the indexes compared to the size as before the updates took place.

*/

--a

```
create index test_cid_btree on customertest(cust_id);
create index test_cid_bitmap on customer_copy_bitmaps(cust_id);
```

```
SQL> create index test_cid_btree on customertest(cust_id);
Index created.

Elapsed: 00:00:00.78

SQL> create index test_cid_bitmap on customer_bitmaps(cust_id);
Index created.

Elapsed: 00:00:00.38
```

--b

```
SET TIMING ON
```

--c.1

```
declare
upd_cust_id number(5);
upd_yob number(4);
begin
for i in 1 .. 500 loop
```

```
upd_cust_id := dbms_random.value(1,55000);
upd_yob := dbms_random.value(1900,2000);
update customer_bitmaps
set cust_year_of_birth=upd_yob
where cust_id = upd_cust_id;
commit;
end loop;
end;
/
PL/SQL procedure successfully completed.

Elapsed: 00:00:00.76

--c.2
declare
upd_cust_id number(5);
upd_yob number(4);
begin
for i in 1 .. 500 loop
upd_cust_id := dbms_random.value(1,55000);
upd_yob := dbms_random.value(1900,2000);
update customertest
set cust_year_of_birth=upd_yob
where cust_id = upd_cust_id;
commit;
end loop;
end;
/
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:01.04

/*Q6. Comparison of time for index creation for normal bitmap index and join bitmap index.
Do as directed:
a) Create table customers_bijx_test_bitmap from customers & sales_bijx_test_bitmap as
from sales
b) create bitmap index sales_bijx_test_bitmap_bix1 on sales_bijx_test_bitmap table and
cust_id column, and bitmap index cust_bijx_test_bitmap_bix1 on
customers_bijx_test_bitmap table and cust_last_name column.
What is the elapsed time for each index creation?
c) Create table customers_bijx_test_bitjoin from customers and Create table
sales_bijx_test_bitjoin from Sales and add constraint of primary key to cust_id column of
cust_bijx_test_bitjoin table.
d) (a) create bitmap index named sales_bijx_test_bitjoin_bjx1 using sales_bijx_test_bitjoin &
customers_bijx_test_bitjoin.cust_id tables.

(b) create bitmap index named sales_bijx_test_bitjoin_bjx2 using tables
sales_bijx_test_bitjoin and customers_bijx_test_bitjoin.cust_last_name
Conclude which index creation takes more time.
*/

--a
create table customers_bijx_test_bitmap as select * from sh.customers;
create bitmap index ccust_bijx_test_bitmap_bix1 on customers_bijx_test_bitmap(cust_last_name);
Elapsed: 00:00:00.01

--b

```
create table sales_bijx_test_bitmap as select * from sh.sales;

create bitmap index sales_bijx_test_bitmap_bix1 on sales_bijx_test_bitmap(cust_id);
```

Elapsed: 00:00:00.39

--c

```
create table customers_bijx_test_bitjoin as select * from(customers);

alter table customers_bijx_test_bitjoin add constraint pk_cust PRIMARY KEY(cust_id);

create table sales_bijx_test_bitjoin as select * from(sh.sales);
```

--d.a

```
create bitmap index  sales_bijx_test_bitjoin_bjx1 on
sales_bijx_test_bitjoin(customers_bijx_test_bitjoin.cust_id)

from sales_bijx_test_bitjoin,customers_bijx_test_bitjoin

where sales_bijx_test_bitjoin.cust_id=customers_bijx_test_bitjoin.cust_id;
```

Elapsed: 00:00:01.09

--d.2

```
create bitmap index  sales_bijx_test_bitjoin_bjx2 on
sales_bijx_test_bitjoin(customers_bijx_test_bitjoin.cust_last_name)

from sales_bijx_test_bitjoin,customers_bijx_test_bitjoin

where sales_bijx_test_bitjoin.cust_id=customers_bijx_test_bitjoin.cust_id;
```

Elapsed: 00:00:01.05

---COMPRESSED INDEX --

/*

1. Create table Student(StudId, StudName)

2. Add 10 Rows

3. Define Index on StudName(First Name and Last Name)

4. Get the Statistics of Index

5. Now add about 10000 rows that will have same last name

6. Get the Statistics of Index

7. Drop Index

8. Create Compressed Index

9. Get the Statistics of Index

10. Compare statics and give your comments*/

--1.

```
create table student(
studid int,
studname varchar2(30));
```

--2.

```
insert into student values(1,'A');
insert into student values(2,'B');
insert into student values(3,'C');
insert into student values(4,'D');
insert into student values(5,'E');
insert into student values(6,'F');
insert into student values(7,'G');
insert into student values(8,'H');
insert into student values(9,'I');
```

--3.

```
CREATE INDEX STUDENT_BTREE_INDEX ON STUDENT(studname);

Index created.

Elapsed: 00:00:00.02


--4

SELECT

  COMPRESSION,

  LEAF_BLOCKS,

  Round(NUM_ROWS/Decode(LEAF_BLOCKS,0,1,LEAF_BLOCKS)) "ROWS PER BLOCK",
DISTINCT_KEYS,

  NUM_ROWS,NUM_ROWS-DISTINCT_KEYS DUP_ROWS

FROM

  USER_INDEXES

WHERE

   INDEX_NAME = 'STUDENT_BTREE_INDEX';



COMPRESS LEAF_BLOCKS ROWS PER BLOCK DISTINCT_KEYS   NUM_ROWS   DUP_ROWS

-------- ----------- -------------- ------------- ---------- ----------

DISABLED          1             9             9         9         0


SELECT t.blocks, t.num_rows, i.clustering_factor

FROM user_tables t, user_indexes i

 WHERE t.table_name = i.table_name AND i.index_name='STUDENT_BTREE_INDEX';


    BLOCKS   NUM_ROWS CLUSTERING_FACTOR

---------- ---------- -----------------

                                  1

--5
```

```
DECLARE v_a NUMBER;
BEGIN
v_a := 11;
WHILE v_a < 10000
LOOP
INSERT INTO STUDENT VALUES(v_a,'Smith');
v_a := v_a + 1;
END LOOP;
COMMIT;
END;
/
```

PL/SQL procedure successfully completed.

--6

```
EXEC DBMS_STATS.gather_table_stats('ARPIT', 'STUDENT');

SELECT
  COMPRESSION,
  LEAF_BLOCKS,
  Round(NUM_ROWS/Decode(LEAF_BLOCKS,0,1,LEAF_BLOCKS)) "ROWS PER BLOCK",
DISTINCT_KEYS,
  NUM_ROWS,NUM_ROWS-DISTINCT_KEYS DUP_ROWS
FROM
  USER_INDEXES
WHERE
   INDEX_NAME = 'STUDENT_BTREE_INDEX';
```

COMPRESS LEAF_BLOCKS ROWS PER BLOCK DISTINCT_KEYS NUM_ROWS DUP_ROWS

-------- ----------- -------------- ------------- ---------- ----------

DISABLED 36 278 10 9998 9988

SQL> SELECT t.blocks, t.num_rows, i.clustering_factor

2 FROM user_tables t, user_indexes i

3 WHERE t.table_name = i.table_name AND i.index_name='STUDENT_BTREE_INDEX';

BLOCKS NUM_ROWS CLUSTERING_FACTOR

---------- ---------- -----------------

28 9998 21

--7

DROP INDEX STUDENT_BTREE_INDEX;

Index dropped.

--8

CREATE INDEX EMP_EMPNAME_IDX ON STUDENT(studname)COMPRESS TABLESPACE USERS;

Index Created.

--9

SELECT

COMPRESSION,

LEAF_BLOCKS,

Round(NUM_ROWS/Decode(LEAF_BLOCKS,0,1,LEAF_BLOCKS)) "ROWS PER BLOCK", DISTINCT_KEYS,

```
  NUM_ROWS,NUM_ROWS-DISTINCT_KEYS DUP_ROWS
FROM
  USER_INDEXES
WHERE
   INDEX_NAME = 'STUDNAME_IDX';

COMPRESS LEAF_BLOCKS ROWS PER BLOCK DISTINCT_KEYS   NUM_ROWS   DUP_ROWS
-------- ----------- -------------- ------------- ---------- ----------
ENABLED          16           625           10       9998       9988

SELECT t.blocks, t.num_rows, i.clustering_factor
FROM user_tables t, user_indexes i
 WHERE t.table_name = i.table_name AND i.index_name='STUDNAME_IDX';

    BLOCKS   NUM_ROWS CLUSTERING_FACTOR
---------- ---------- -----------------
        28       9998                21

--10

When compression is enabled, less leaf blocks are created and
rows per block gets increased.

--FUNCTION BASED INDEX
/*Function Based Indexes:
1. Create function based index on Employee table of HR schema. Function should be on
salary attribute based on commission percentage.
Find out list of employees having commission percentage less than 50000.
2. Create function based index on employee name for Upper and lower function.
```

3. Create user table with attributes (UserId, UserName, Gender)

4. Insert 10000 records in user table

5. Build regular index on Username

6. Build function based index on user name based on Upper function

7. Compare the response time and comment.

*/

--1

```
CREATE TABLE HR AS ( SELECT * FROM HR.EMPLOYEES);
```

```
SQL> CREATE TABLE HR AS ( SELECT * FROM HR.EMPLOYEES);
```

Table created.

Elapsed: 00:00:00.13

```
CREATE INDEX INDEX_FBI_HR ON HR(COMMISSION_PCT*SALARY);
```

```
SQL> CREATE INDEX INDEX_FBI_HR ON HR(COMMISSION_PCT*SALARY);
```

Index created.

Elapsed: 00:00:00.03

```
SELECT * FROM HR
        WHERE SALARY*COMMISSION_PCT < 50000;
```

```
SQL> SELECT * FROM HR WHERE (COMMISSION_PCT*SALARY) <50000;
```

EMPLOYEE_ID FIRST_NAME LAST_NAME EMAIL
PHONE_NUMBER HIRE_DATE JOB_ID SALARY COMMISSION_PCT MANAGER_ID
DEPARTMENT_ID

---

173 Sundita Kumar SKUMAR 011.44.1343.329268 21-APR-08
SA_REP 6100 .1 148

80

167 Amit Banda ABANDA 011.44.1346.729268 21-APR-08
SA_REP 6200 .1 147

80

179 Charles Johnson CJOHNSON 011.44.1644.429262 04-JAN-08
SA_REP 6200 .1 149

80

166 Sundar Ande SANDE 011.44.1346.629268 24-MAR-08
SA_REP 6400 .1 147

80

165 David Lee DLEE 011.44.1346.529268 23-FEB-08
SA_REP 6800 .1 147

80

164 Mattea Marvins MMARVINS 011.44.1346.329268 24-JAN-08
SA_REP 7200 .1 147

80

155 Oliver Tuvault OTUVAULT 011.44.1344.486508 23-NOV-07
SA_REP 7000 .15 145

80

178 Kimberely Grant KGRANT 011.44.1644.429263 24-MAY-07
SA_REP 7000 .15 149

172 Elizabeth Bates EBATES 011.44.1343.529268 24-MAR-07
SA_REP 7300 .15 148

80

171 William Smith WSMITH 011.44.1343.629268 23-FEB-07
SA_REP 7400 .15 148

80

163 Danielle Greene DGREENE 011.44.1346.229268 19-MAR-07 SA_REP 9500 .15 147

80

154 Nanette Cambrault NCAMBRAU 011.44.1344.987668 09-DEC-06 SA_REP 7500 .2 145

80

153 Christopher Olsen COLSEN 011.44.1344.498718 30-MAR-06 SA_REP 8000 .2 145

80

177 Jack Livingston JLIVINGS 011.44.1644.429264 23-APR-06 SA_REP 8400 .2 149

80

176 Jonathon Taylor JTAYLOR 011.44.1644.429265 24-MAR-06 SA_REP 8600 .2 149

80

161 Sarath Sewall SSEWALL 011.44.1345.529268 03-NOV-06 SA_REP 7000 .25 146

80

170 Tayler Fox TFOX 011.44.1343.729268 24-JAN-06 SA_REP 9600 .2 148

80

169 Harrison Bloom HBLOOM 011.44.1343.829268 23-MAR-06 SA_REP 10000 .2 148

80

149 Eleni Zlotkey EZLOTKEY 011.44.1344.429018 29-JAN-08 SA_MAN 10500 .2 100

80

175 Alyssa Hutton AHUTTON 011.44.1644.429266 19-MAR-05 SA_REP 8800 .25 149

80

152 Peter Hall PHALL 011.44.1344.478968 20-AUG-05 SA_REP 9000 .25 145

80

160 Louise Doran LDORAN 011.44.1345.629268 15-DEC-05
SA_REP 7500 .3 146

80

151 David Bernstein DBERNSTE 011.44.1344.345268 24-MAR-05
SA_REP 9500 .25 145

80

159 Lindsey Smith LSMITH 011.44.1345.729268 10-MAR-05
SA_REP 8000 .3 146

80

162 Clara Vishney CVISHNEY 011.44.1346.129268 11-NOV-05
SA_REP 10500 .25 147

80

168 Lisa Ozer LOZER 011.44.1343.929268 11-MAR-05
SA_REP 11500 .25 148

80

150 Peter Tucker PTUCKER 011.44.1344.129268 30-JAN-05
SA_REP 10000 .3 145

80

158 Allan McEwen AMCEWEN 011.44.1345.829268 01-AUG-04
SA_REP 9000 .35 146

80

148 Gerald Cambrault GCAMBRAU 011.44.1344.619268 15-OCT-
07 SA_MAN 11000 .3 100

80

174 Ellen Abel EABEL 011.44.1644.429267 11-MAY-04
SA_REP 11000 .3 149

80

157 Patrick Sully PSULLY 011.44.1345.929268 04-MAR-04
SA_REP 9500 .35 146

80

156 Janette King JKING 011.44.1345.429268 30-JAN-04
SA_REP 10000 .35 146

80

147 Alberto Errazuriz AERRAZUR 011.44.1344.429278 10-MAR-05
SA_MAN 12000 .3 100

80

146 Karen Partners KPARTNER 011.44.1344.467268 05-JAN-05
SA_MAN 13500 .3 100

80

145 John Russell JRUSSEL 011.44.1344.429268 01-OCT-04
SA_MAN 14000 .4 100

80

35 rows selected.

Elapsed: 00:00:00.05

--2

CREATE INDEX EMPNAME_INDEX ON HR(UPPER(FIRST_NAME) || LOWER(LAST_NAME));

SQL> CREATE INDEX EMPNAME_INDEX ON HR(UPPER(FIRST_NAME) ||
LOWER(LAST_NAME));

Index created.

Elapsed: 00:00:00.09

--3

CREATE TABLE user_data (

userid NUMBER(10) NOT NULL,

username VARCHAR2(40) NOT NULL,

gender VARCHAR2(1)

```
);
```

Table created.

Elapsed: 00:00:00.31

```
--4
BEGIN
  FOR userid IN 1 .. 100000 LOOP
    IF MOD(userid, 2) = 0 THEN
      INSERT INTO user_data
      VALUES (userid, 'John', 'M');
    ELSE
      INSERT INTO user_data
      VALUES (userid, 'Jayne', 'F');
    END IF;
    COMMIT;
  END LOOP;
END;
/
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:19.29

```
--5
```

```
SQL> CREATE INDEX INDEX_REGULAR ON USER_DATA(USERNAME);
```

Index created.

****

Elapsed: 00:00:00.67

****

SELECT COUNT(*) FROM USER_DATA;

Elapsed: 00:00:00.67

SQL>

--6

CREATE INDEX INDEX_FBI_USERNAME ON USER_DATA(UPPER(USERNAME));

SQL> CREATE INDEX INDEX_FBI_USERNAME ON USER_DATA(UPPER(USERNAME));

Index created.

SQL> SELECT COUNT(*) FROM USER_DATA;

  COUNT(*)

    100000

Elapsed: 00:00:00.10

SQL>

--7

Function based index gives a faster retrieval than normal BTree index.

--INDEX ORGANIZED TABLE

/*1. Create an IOT look_ups with the attributes (lookup_code, lookup_value,

lookup_description) in tablespace ts_lookup.

Constraint: lookup_code should be primary key

PctThreshold is 20 and and lookup_description should be in overflow area.

Overflow should be in ts_overflow tablespace.

2. Create a Index Organized Table(IOT) emp_iot based on hr.employees

3. Create a Index Organized Table(IOT) emp101_emp based on hr.employees. Place the

column hiredate in overflow area.

4. Compare the timings of executing select all from employees,emp_iot, and emp101_iot.

Comment on your observations.*/

--1

CREATE TABLESPACE LOOKUPS DATAFILE 'C:\Users\admin\Desktop\DWM' SIZE 10M;

SQL> CREATE TABLESPACE LOOKUPTEST DATAFILE
'C:\Users\admin\Desktop\DWM\lookuptest.dbf' SIZE 10M;

Tablespace created.

Elapsed: 00:00:01.51

SQL>

CREATE TABLE IOT_LOOKUPS(

lookup_code NUMBER(10),

lookup_value NUMBER(10),

lookup_description VARCHAR2(40),

```
CONSTRAINT LOOKUP_PK PRIMARY KEY(LOOKUP_CODE))
ORGANIZATION INDEX
PCTTHRESHOLD 20
INCLUDING LOOKUP_DESCRIPTION
OVERFLOW TABLESPACE LOOKUPTEST;

Table created.

BEGIN
  FOR lookup_code IN 1 .. 100000 LOOP
    IF MOD(lookup_code, 2) = 0 THEN
      INSERT INTO IOT_LOOKUPS
      VALUES (lookup_code, lookup_code+1, 'String');
    ELSE
      INSERT INTO IOT_LOOKUPS
      VALUES (lookup_code, lookup_code+2, 'String_2');
    END IF;
    COMMIT;
  END LOOP;
END;
/

PL/SQL procedure successfully completed.

Elapsed: 00:00:06.98

--2

CREATE TABLE EMP_IOT
```

```
(EMP_NO NUMBER,
EMP_NAME VARCHAR2(20),
EMP_DEPT NUMBER,
EMP_ADDRESS VARCHAR2(500),
EMP_HIST VARCHAR2(1000),
CONSTRAINT EMP_PK PRIMARY KEY(EMP_NO))
ORGANIZATION INDEX
INCLUDING EMP_NAME
TABLESPACE LOOKUP_DESCRIPTION
OVERFLOW TABLESPACE LOOKUPTEST;
```

Tablespace Created.

```
BEGIN
  FOR EMP_NO IN 1 .. 100000 LOOP
    IF MOD(EMP_NO, 2) = 0 THEN
      INSERT INTO EMP_IOT
      VALUES (EMP_NO,'SAM', 101, 'MYADDRESS', 'OLDTEXT');
    ELSE
      INSERT INTO EMP_IOT
      VALUES (EMP_NO,'JAM', 501, 'MYADDRESS', 'OLDTEXT');
    END IF;
    COMMIT;
  END LOOP;
END;
/
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:13.96

--3

```
CREATE TABLE EMP_IOT101
(EMP_NO NUMBER,
EMP_NAME VARCHAR2(20),
EMP_DEPT NUMBER,
EMP_ADDRESS VARCHAR2(500),
EMP_HIST VARCHAR2(1000),
HIREDATE DATE,
CONSTRAINT EMP_PUK PRIMARY KEY(EMP_NO))
ORGANIZATION INDEX
INCLUDING HIREDATE
TABLESPACE LOOKUPTEST
OVERFLOW TABLESPACE LOOKUPTEST;
```

Table created.

```
BEGIN
  FOR EMP_NO IN 1 .. 100000 LOOP
    IF MOD(EMP_NO, 2) = 0 THEN
      INSERT INTO EMP_IOT101
      VALUES (EMP_NO,'SAM', 101, 'MYADDRESS', 'OLDTEXT', '1-4-2005');
    ELSE
      INSERT INTO EMP_IOT101
      VALUES (EMP_NO,'JAM', 501, 'MYADDRESS', 'OLDTEXT','1-4-2005');
    END IF;
    COMMIT;
  END LOOP;
```

```
END;
/

PL/SQL procedure successfully completed.

Elapsed: 00:00:20.16
```