Name : Vinayak Soni
Roll No : 67

```
--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?
--set timing on;
create table customers_copy_btree as (select * from sh.customers);
CREATE TABLE CUSTOMERS_copy_BITMAP AS (SELECT * FROM sh.CUSTOMERS);


CREATE INDEX idx_cust_gender ON customers_copy_btree(cust_gender);
CREATE INDEX idx_cust_year_of_birth ON
customers_copy_btree(cust_year_of_birth);
CREATE INDEX idx_cust_last_name ON
customers_copy_btree(cust_last_name);
CREATE INDEX idx_cust_street_address ON
customers_copy_btree(cust_street_address);

--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.

CREATE BITMAP INDEX idx_cust_gender ON
customers_copy_btree(cust_gender);
CREATE BITMAP INDEX idx_cust_year_of_birth ON
customers_copy_btree(cust_year_of_birth);
CREATE BITMAP INDEX idx_cust_last_name ON
customers_copy_btree(cust_last_name);
CREATE BITMAP INDEX idx_cust_street_address ON
customers_copy_btree(cust_street_address);

--Q3. Find the size of each segment: customers_copy_bitmap and
customers_copy_btree
--(Hint : Use users_segment table)
SELECT segment_name, segment_type, bytes/1024/1024 AS mb
FROM user_segments
WHERE segment_name = 'CUSTOMERS_COPY_BITMAP';

SELECT segment_name, segment_type, bytes/1024/1024 AS mb
```

```
FROM user_segments
WHERE segment_name = 'CUSTOMERS_COPY_BTREE';

--Q4. Do as directed :
--a. Create function based index on Employee table of HR schema.
Function should be on salary
--attribute based on commission percentage.
CREATE INDEX idx_salary_comm_pct ON Employee ((SALARY *
COMMISSION_PCT));

--b. Find out list of employees having commission percentage less
than 50000.
SELECT * FROM Employee WHERE COMMISSION_PCT < 0.5;

--c. Create function based index on employee name for Upper and lower
function.

CREATE INDEX idx_emp_upper_lower ON Employee (UPPER(ENAME),
LOWER(ENAME));

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

CREATE TABLE user (
  UserId NUMBER,
  UserName VARCHAR2(50),
  Gender VARCHAR2(10)
);

--e. Insert 10000 records in user table

DECLARE
  i NUMBER;
BEGIN
  FOR i IN 1..10000 LOOP
    INSERT INTO user (UserId, UserName, Gender)
    VALUES (i, 'User' || i, CASE MOD(i, 2) WHEN 0 THEN 'F' ELSE 'M'
END);
  END LOOP;
END;
/

--f. Build regular index on Username
CREATE INDEX idx_username ON user (UserName);

--g. Build function based index on user name based on Upper function
```

```
CREATE INDEX idx_upper_username ON user (UPPER(UserName));

--h. Compare the response time and comment.
-- Query using regular index
SELECT * FROM user WHERE UserName = 'UserName 5000';
-- Query using function-based index
SELECT * FROM user WHERE UPPER(UserName) = 'USERNAME 5000';

--Q5. Do as directed :
--a. Create an IOT look_ups with the attributes (lookup_code,
lookup_value,
--lookup_description).
--b. Constraint: lookup_code should be primary key
--c. lookup_description should be in overflow area.
CREATE TABLE look_ups (
  lookup_code          VARCHAR2(50),
  lookup_value         VARCHAR2(50),
  lookup_description   VARCHAR2(4000),
  CONSTRAINT pk_look_ups PRIMARY KEY (lookup_code)
)
ORGANIZATION INDEX
OVERFLOW TABLESPACE users;

--Q6. Do as directed :
--a. Create a Index Organized Table(IOT) emp_iot based on
hr.employees
--b. Create a Index Organized Table(IOT) emp101_emp based on
hr.employees. Place
--the column hiredate in overflow area.
--c. Compare the timings of executing select all from
employees,emp_iot, and
--emp101_iot. Comment on your observations.
-- Create emp_iot based on hr.employees
CREATE TABLE emp_iot
(
  employee_id    NUMBER PRIMARY KEY,
  first_name     VARCHAR2(20),
  last_name      VARCHAR2(25),
  email          VARCHAR2(25),
  phone_number   VARCHAR2(20),
  hire_date      DATE,
  job_id         VARCHAR2(10),
  salary         NUMBER,
  commission_pct NUMBER,
  manager_id     NUMBER,
```

```
  department_id  NUMBER
)
ORGANIZATION INDEX;

-- Create emp101_iot based on hr.employees
CREATE TABLE emp101_iot
(
  employee_id    NUMBER PRIMARY KEY,
  first_name     VARCHAR2(20),
  last_name      VARCHAR2(25),
  email          VARCHAR2(25),
  phone_number   VARCHAR2(20),
  job_id         VARCHAR2(10),
  salary         NUMBER,
  commission_pct NUMBER,
  manager_id     NUMBER,
  department_id  NUMBER,
  hire_date      DATE
)
ORGANIZATION INDEX OVERFLOW TABLESPACE USERS;

SELECT * FROM hr.employees;
SELECT * FROM emp_iot;
SELECT * FROM emp101_iot;

--Q7. Do as directed :
--a. Create a cluster PERSONNEL containing copy_emp and copy_dept
tables. Cluster key is
--deptno.
CREATE CLUSTER personnel_cluster (deptno NUMBER(4));

CREATE TABLE copy_emp (
  empno NUMBER(4) PRIMARY KEY,
  ename VARCHAR2(10),
  job VARCHAR2(9),
  mgr NUMBER(4),
  hiredate DATE,
  sal NUMBER(7,2),
  comm NUMBER(7,2),
  deptno NUMBER(4)
)
CLUSTER personnel_cluster (deptno);

CREATE TABLE copy_dept (
  deptno NUMBER(4) PRIMARY KEY,
```

```
  dname VARCHAR2(14),
  loc VARCHAR2(13)
)
CLUSTER personnel_cluster (deptno);

--b. create a index on cluster PERSONNEL.
CREATE INDEX personnel_index ON CLUSTER personnel_cluster;

--c. Populate copy_emp and copy_dept with data from emp and dept
tables of scott
--respectively.
INSERT INTO copy_dept SELECT * FROM dept;

INSERT INTO copy_emp
SELECT e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm,
e.deptno
FROM emp e JOIN copy_dept d ON (e.deptno = d.deptno);

--d. Create a dumkmy table &quot;dummy&quot; which references empno
of copy_emp.
CREATE TABLE dummy (
  empno NUMBER(4),
  CONSTRAINT fk_empno FOREIGN KEY (empno) REFERENCES copy_emp(empno)
);

--e. Drop cluster PERSONNEL.
DROP CLUSTER personnel_cluster INCLUDING TABLES;

--f. Create a hash cluster named hash_emp containing table
copy101_emp. Create 10 hashkeys
--and use the hash function (empno mod 100).
CREATE CLUSTER hash_emp_cluster (empno NUMBER(4))
HASHKEYS 10
HASH IS MOD(empno, 100);

CREATE TABLE copy101_emp (
  empno NUMBER(4) PRIMARY KEY,
  ename VARCHAR2(10),
  job VARCHAR2(9),
  mgr NUMBER(4),
  hiredate DATE,
  sal NUMBER(7,2),
  comm NUMBER(7,2),
  deptno NUMBER(4)
)
```

```
CLUSTER hash_emp_cluster (empno);
```