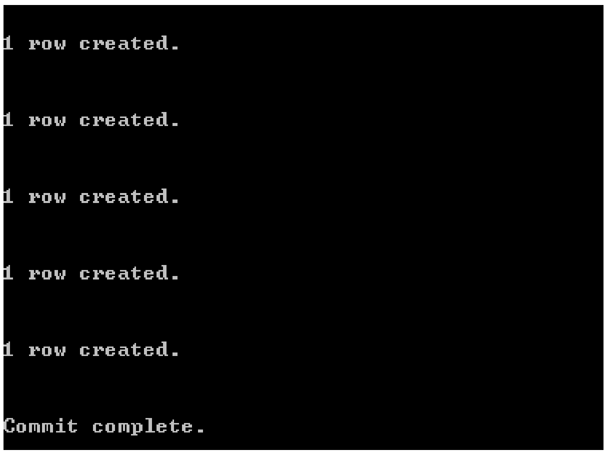
0a)

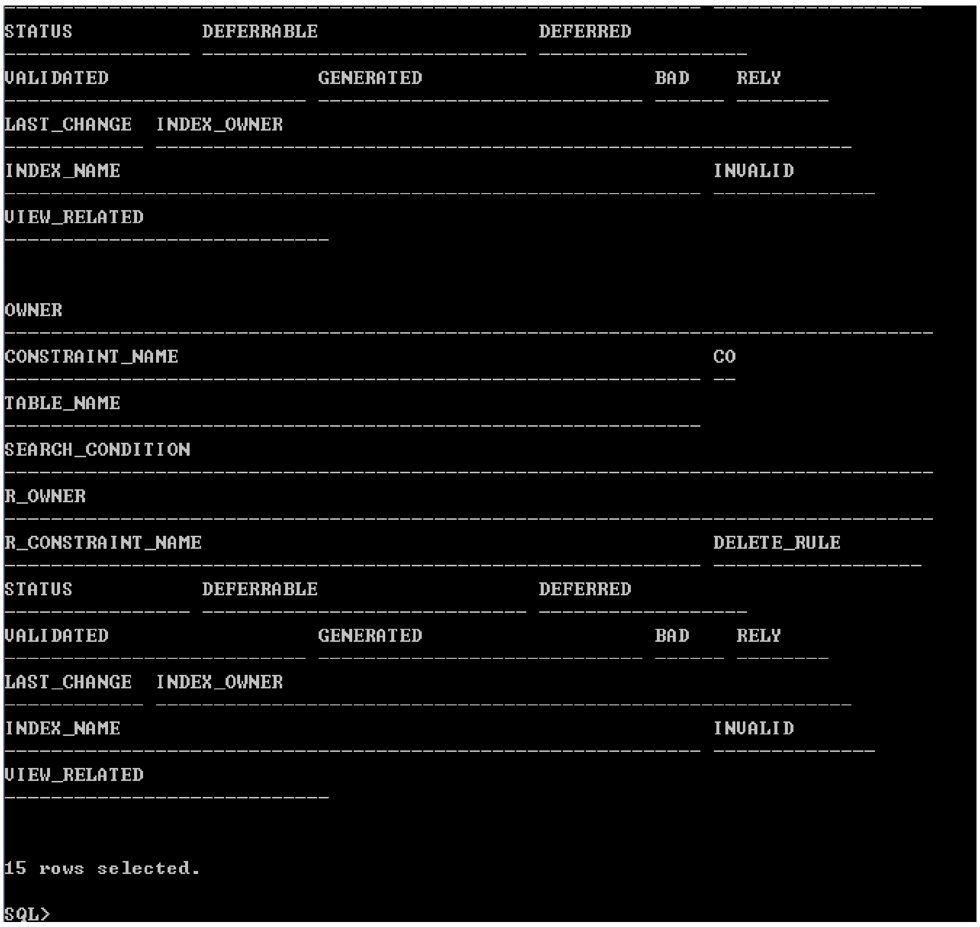
@G:\prjScript.sql.txt;



```
1 row created.


1 row created.


1 row created.


1 row created.


1 row created.


Commit complete.
```

1a)

SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME = 'EMP'
OR TABLE_NAME = 'DEPT' OR TABLE_NAME = 'PURCHASE' OR TABLE_NAME = 'CLIENT';



```
STATUS           DEFERRABLE                    DEFERRED
---------------- ----------------------------- ------------------
VALIDATED                 GENERATED                    BAD    RELY
------------------------- ---------------------------- ------ --------
LAST_CHANGE  INDEX_OWNER
------------ -----------------------------------------------------------
INDEX_NAME                                                   INVALID
------------------------------------------------------------ --------------
VIEW_RELATED
----------------------------


OWNER
--------------------------------------------------------------------------------
CONSTRAINT_NAME                                              CO
------------------------------------------------------------ --
TABLE_NAME
------------------------------------------------------------
SEARCH_CONDITION
--------------------------------------------------------------------------------
R_OWNER
--------------------------------------------------------------------------------
R_CONSTRAINT_NAME                                            DELETE_RULE
------------------------------------------------------------ ------------------
STATUS           DEFERRABLE                    DEFERRED
---------------- ----------------------------- ------------------
VALIDATED                 GENERATED                    BAD    RELY
------------------------- ---------------------------- ------ --------
LAST_CHANGE  INDEX_OWNER
------------ -----------------------------------------------------------
INDEX_NAME                                                   INVALID
------------------------------------------------------------ --------------
VIEW_RELATED
----------------------------


15 rows selected.

SQL>
```

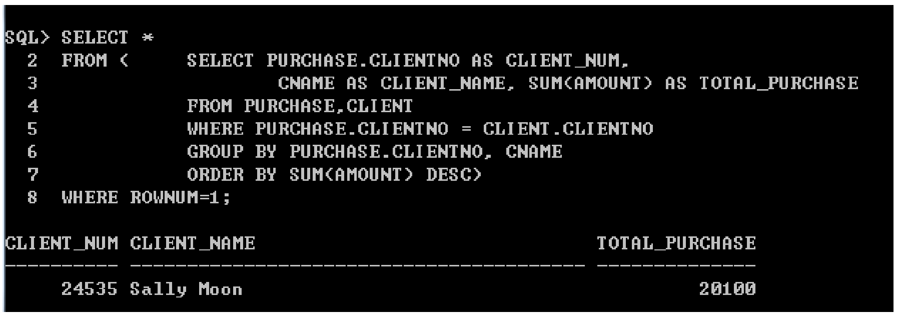
1b)

```
ALTER TABLE DEPT ADD CONSTRAINT UN_DNAME UNIQUE (DNAME);

ALTER TABLE PURCHASE MODIFY AMOUNT NUMBER (4) NOT NULL;

ALTER TABLE EMP MODIFY ENAME VARCHAR2 (20) NOT NULL;

ALTER TABLE DEPT MODIFY DNAME VARCHAR2 (20) NOT NULL;

ALTER TABLE CLIENT MODIFY CNAME VARCHAR2 (20) NOT NULL;

ALTER TABLE PURCHASE MODIFY RECEIPTNO NUMBER (6) NOT NULL;

ALTER TABLE PURCHASE ADD CONSTRAINT CK_SERVICETYPE
        CHECK (SERVICETYPE = 'Training'
                OR SERVICETYPE = 'Data Recovery'
                OR SERVICETYPE = 'Consultation'
                OR SERVICETYPE = 'Software Installation'
                OR SERVICETYPE = 'Software Repair');

ALTER TABLE PURCHASE ADD CONSTRAINT CK_PAYMENTTYPE
        CHECK (PAYMENTTYPE = 'Debit'
                OR PAYMENTTYPE = 'Cash'
                OR PAYMENTTYPE = 'Credit');

ALTER TABLE PURCHASE ADD CONSTRAINT CK_GST
        CHECK (GST = 'Yes' OR GST = 'No');

ALTER TABLE EMP ADD CONSTRAINT FK_DEPTNO
        FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO);

ALTER TABLE PURCHASE ADD CONSTRAINT FK_EMPNO
        FOREIGN KEY (SERVEDBY) REFERENCES EMP (EMPNO);

ALTER TABLE PURCHASE ADD CONSTRAINT FK_CLIENTNO
        FOREIGN KEY (CLIENTNO) REFERENCES CLIENT (CLIENTNO);
```

2a)

```
SELECT *
FROM (  SELECT PURCHASE.CLIENTNO AS CLIENT_NUM,
                CNAME AS CLIENT_NAME, SUM(AMOUNT) AS TOTAL_PURCHASE
        FROM PURCHASE,CLIENT
        WHERE PURCHASE.CLIENTNO = CLIENT.CLIENTNO
        GROUP BY PURCHASE.CLIENTNO, CNAME
```

```
                ORDER BY SUM(AMOUNT) DESC)
WHERE ROWNUM=1;
```



```
SQL> SELECT *
  2  FROM (     SELECT PURCHASE.CLIENTNO AS CLIENT_NUM,
  3                     CNAME AS CLIENT_NAME, SUM(AMOUNT) AS TOTAL_PURCHASE
  4             FROM PURCHASE,CLIENT
  5             WHERE PURCHASE.CLIENTNO = CLIENT.CLIENTNO
  6             GROUP BY PURCHASE.CLIENTNO, CNAME
  7             ORDER BY SUM(AMOUNT) DESC)
  8  WHERE ROWNUM=1;

CLIENT_NUM CLIENT_NAME                                    TOTAL_PURCHASE
---------- ---------------------------------------------- --------------
     24535 Sally Moon                                              20100
```

2b)

```
CREATE OR REPLACE TRIGGER TOP_CLIENT_DISCOUNT
BEFORE INSERT ON "PURCHASE"
FOR EACH ROW
DECLARE
TOP_CLIENT NUMBER(5);
BEGIN
        SELECT CLIENT_NUM INTO TOP_CLIENT FROM (SELECT *
                                                FROM (  SELECT PURCHASE.CLIENTNO AS CLIENT_NUM,
                                                                CNAME AS CLIENT_NAME,
                                                                SUM(AMOUNT) AS TOTAL_PURCHASE
                                                        FROM PURCHASE,CLIENT
                                                        WHERE PURCHASE.CLIENTNO = CLIENT.CLIENTNO
                                                        GROUP BY PURCHASE.CLIENTNO, CNAME
                                                        ORDER BY SUM(AMOUNT) DESC)
                                                WHERE ROWNUM=1);
        IF :NEW.CLIENTNO = TOP_CLIENT THEN :NEW.AMOUNT := (:NEW.AMOUNT)*0.85;
        END IF;
END;
/
```

2c)

```
CREATE OR REPLACE TRIGGER SUNSHINE_DISCOUNT
BEFORE INSERT ON "PURCHASE"
FOR EACH ROW
DECLARE
COUNT_NUM NUMBER(1);
BEGIN
        SELECT NVL(COUNT(*),0) INTO COUNT_NUM FROM (SELECT SERVEDBY
                                                    FROM PURCHASE, EMP, DEPT
                                                    WHERE PURCHASE.SERVEDBY = EMP.EMPNO
```

```
                                                        AND EMP.DEPTNO = DEPT.DEPTNO
                                                        AND DEPT.DNAME = 'SALES - Sunshine'
                                                    GROUP BY SERVEDBY)
    WHERE :NEW.SERVEDBY = SERVEDBY;
    IF COUNT_NUM > 0 THEN
            :NEW.PAYMENTTYPE := 'Cash';
            IF :NEW.SERVICETYPE = 'Data Recovery' THEN :NEW.AMOUNT := (:NEW.AMOUNT)*0.7;
            END IF;
    END IF;
END;
/
```

3a)

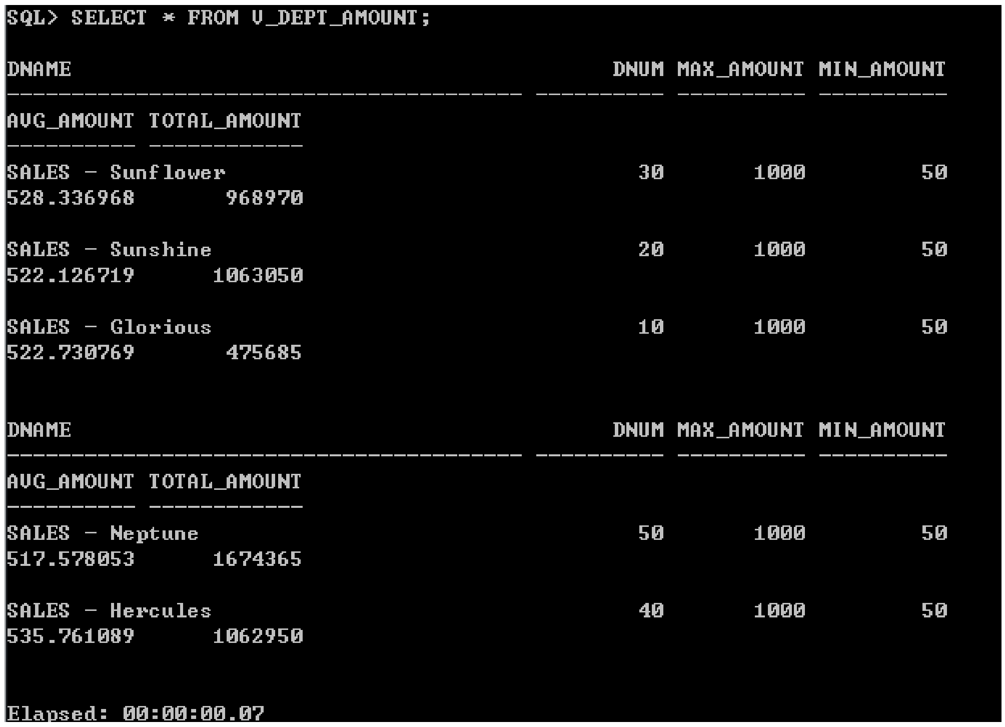
```
CREATE VIEW V_DEPT_AMOUNT AS
SELECT DEPT.DNAME AS DNAME, DEPT.DEPTNO AS DNUM, MAX(AMOUNT) AS MAX_AMOUNT,
        MIN(AMOUNT) MIN_AMOUNT, AVG(AMOUNT) AS AVG_AMOUNT, SUM(AMOUNT) AS TOTAL_AMOUNT
FROM PURCHASE, EMP, DEPT
WHERE PURCHASE.SERVEDBY = EMP.EMPNO AND EMP.DEPTNO = DEPT.DEPTNO
GROUP BY DEPT.DNAME, DEPT.DEPTNO;
```

3b)

```
CREATE MATERIALIZED VIEW MV_DEPT_AMOUNT
BUILD IMMEDIATE AS
SELECT DEPT.DNAME AS DNAME, DEPT.DEPTNO AS DNUM, MAX(AMOUNT) AS MAX_AMOUNT,
        MIN(AMOUNT) MIN_AMOUNT, AVG(AMOUNT) AS AVG_AMOUNT, SUM(AMOUNT) AS TOTAL_AMOUNT
FROM PURCHASE, EMP, DEPT
WHERE PURCHASE.SERVEDBY = EMP.EMPNO AND EMP.DEPTNO = DEPT.DEPTNO
GROUP BY DEPT.DNAME, DEPT.DEPTNO;
```

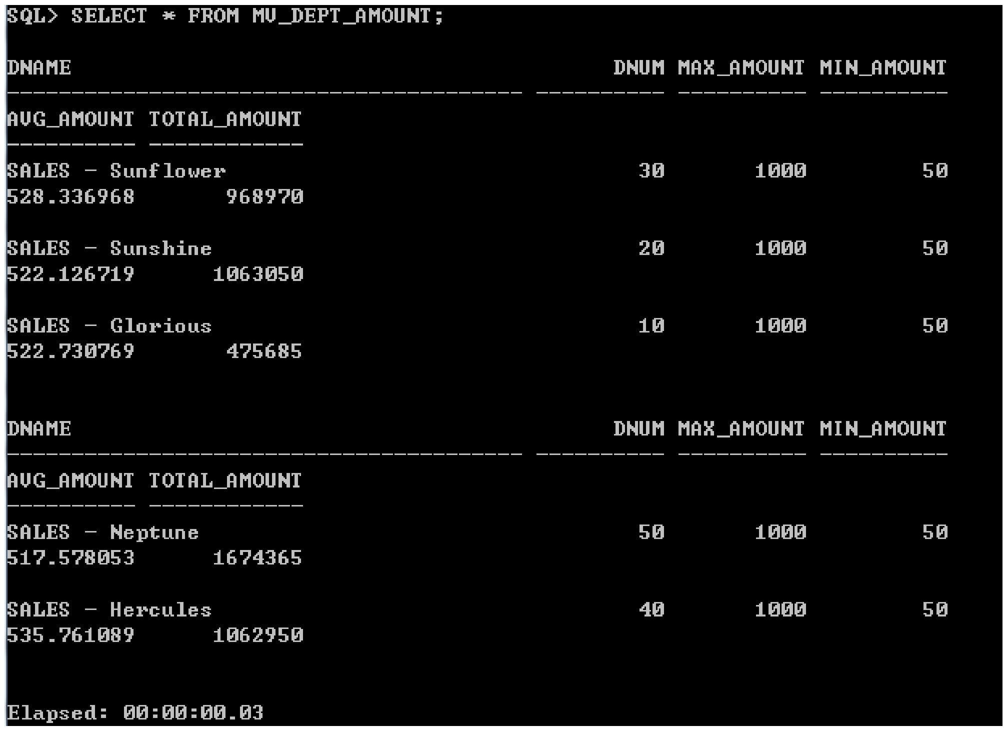
3c)

```
SET TIMING ON;
Q1: SELECT * FROM V_DEPT_AMOUNT;
```



```
SQL> SELECT * FROM V_DEPT_AMOUNT;

DNAME                                          DNUM MAX_AMOUNT MIN_AMOUNT
---------------------------------------- ---------- ---------- ----------
AVG_AMOUNT TOTAL_AMOUNT
---------- ------------
SALES - Sunflower                                30       1000         50
528.336968       968970

SALES - Sunshine                                 20       1000         50
522.126719      1063050

SALES - Glorious                                 10       1000         50
522.730769       475685


DNAME                                          DNUM MAX_AMOUNT MIN_AMOUNT
---------------------------------------- ---------- ---------- ----------
AVG_AMOUNT TOTAL_AMOUNT
---------- ------------
SALES - Neptune                                  50       1000         50
517.578053      1674365

SALES - Hercules                                 40       1000         50
535.761089      1062950


Elapsed: 00:00:00.07
```

Q2: SELECT * FROM MV_DEPT_AMOUNT;



```
SQL> SELECT * FROM MV_DEPT_AMOUNT;

DNAME                                          DNUM MAX_AMOUNT MIN_AMOUNT
---------------------------------------- ---------- ---------- ----------
AVG_AMOUNT TOTAL_AMOUNT
---------- ------------
SALES - Sunflower                                30       1000         50
528.336968       968970

SALES - Sunshine                                 20       1000         50
522.126719      1063050

SALES - Glorious                                 10       1000         50
522.730769       475685


DNAME                                          DNUM MAX_AMOUNT MIN_AMOUNT
---------------------------------------- ---------- ---------- ----------
AVG_AMOUNT TOTAL_AMOUNT
---------- ------------
SALES - Neptune                                  50       1000         50
517.578053      1674365

SALES - Hercules                                 40       1000         50
535.761089      1062950


Elapsed: 00:00:00.03
```

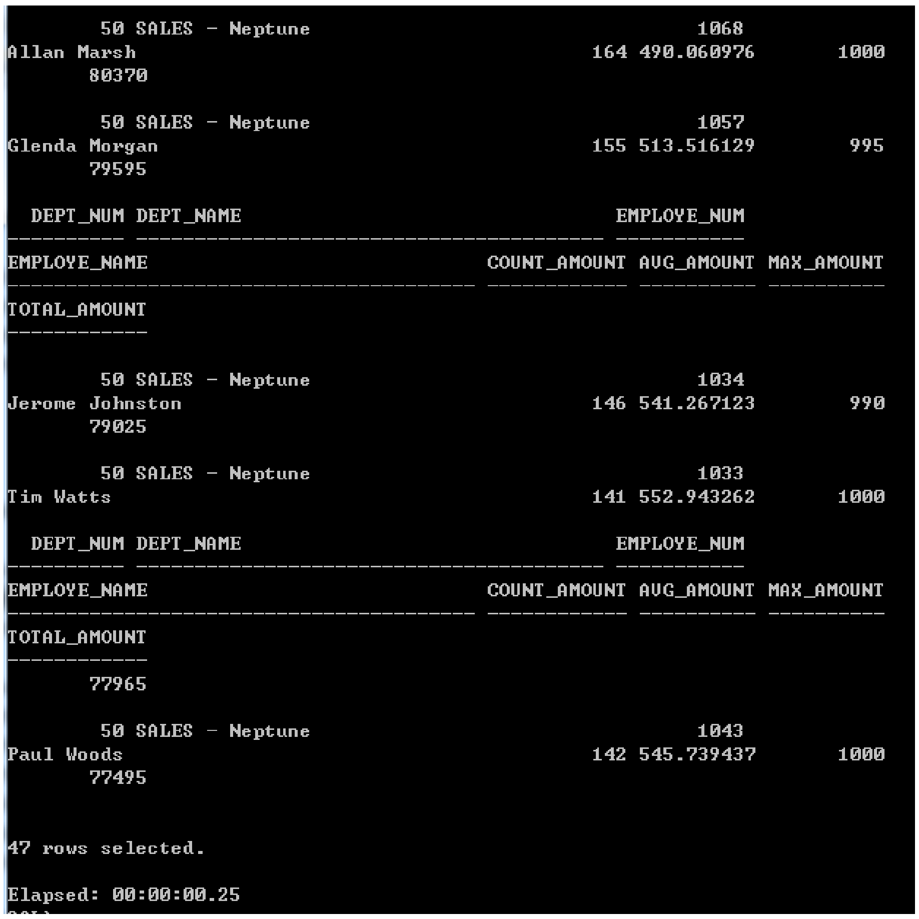
The execution time of Q1 is greater than execution time of Q2. Because Q2 is a materialized view and Q1 is a regular view, and materialized view store the result of the view's query, so materialized view is more efficient than regular view because it does not need to recompute the query every time they are used.

3d)

```
CREATE VIEW V_DEPT_TOP_EMPS AS
SELECT DEPT_NUM, DEPT_NAME, EMPLOYE_NUM, EMPLOYE_NAME, COUNT_AMOUNT, AVG_AMOUNT,
MAX_AMOUNT, TOTAL_AMOUNT
FROM ( SELECT DNAME AS DEPT_NAME, DEPT.DEPTNO AS DEPT_NUM, ENAME AS EMPLOYE_NAME,
EMP.EMPNO AS EMPLOYE_NUM, COUNT(AMOUNT) AS COUNT_AMOUNT, AVG(AMOUNT) AS AVG_AMOUNT,
MAX(AMOUNT) AS MAX_AMOUNT, SUM(AMOUNT) AS TOTAL_AMOUNT, ROW_NUMBER() OVER(PARTITION BY
DNAME ORDER BY SUM(AMOUNT) DESC) RM
        FROM EMP, PURCHASE, DEPT
        WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND DEPT.DEPTNO = EMP.DEPTNO
        GROUP BY DNAME, DEPT.DEPTNO, ENAME, EMP.EMPNO
        ORDER BY DEPT.DEPTNO,SUM(AMOUNT) DESC)
WHERE RM <= 10;

CREATE MATERIALIZED VIEW MV_DEPT_TOP_EMPS
BUILD IMMEDIATE AS
SELECT DEPT_NUM, DEPT_NAME, EMPLOYE_NUM, EMPLOYE_NAME, COUNT_AMOUNT, AVG_AMOUNT,
MAX_AMOUNT, TOTAL_AMOUNT
FROM ( SELECT DNAME AS DEPT_NAME, DEPT.DEPTNO AS DEPT_NUM, ENAME AS EMPLOYE_NAME,
EMP.EMPNO AS EMPLOYE_NUM, COUNT(AMOUNT) AS COUNT_AMOUNT, AVG(AMOUNT) AS AVG_AMOUNT,
MAX(AMOUNT) AS MAX_AMOUNT, SUM(AMOUNT) AS TOTAL_AMOUNT, ROW_NUMBER() OVER(PARTITION BY
DNAME ORDER BY SUM(AMOUNT) DESC) RM
        FROM EMP, PURCHASE, DEPT
        WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND DEPT.DEPTNO = EMP.DEPTNO
        GROUP BY DNAME, DEPT.DEPTNO, ENAME, EMP.EMPNO
        ORDER BY DEPT.DEPTNO,SUM(AMOUNT) DESC)
WHERE RM <= 10;
```

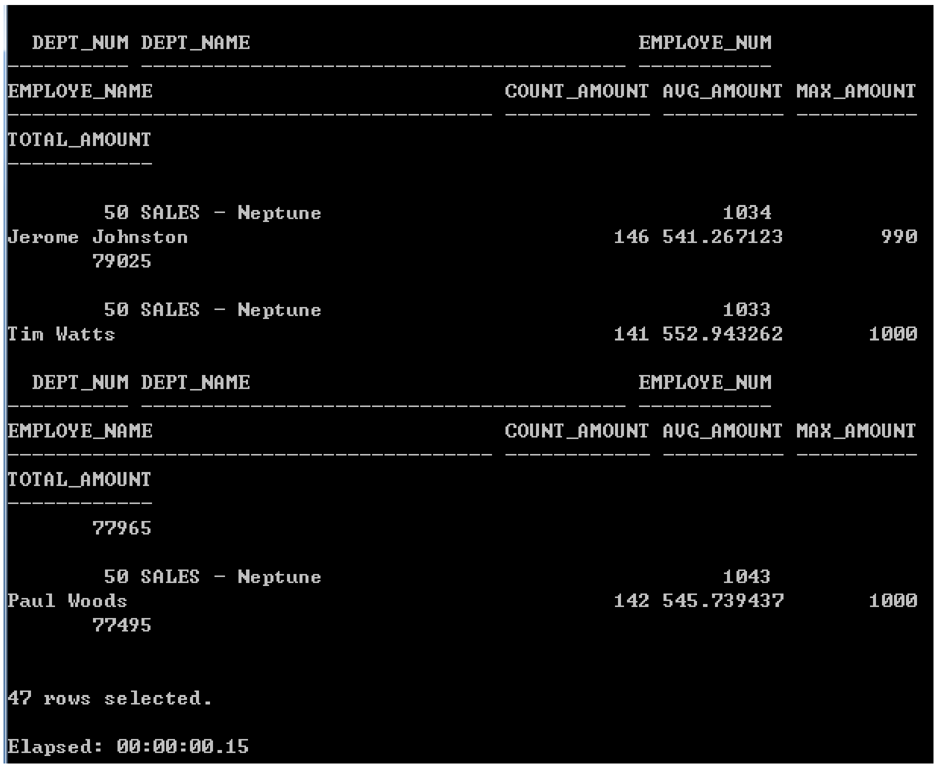
3e)

Q1: SELECT * FROM V_DEPT_TOP_EMPS;



```
        50 SALES - Neptune                                      1068
Allan Marsh                                         164 490.060976       1000
       80370

        50 SALES - Neptune                                      1057
Glenda Morgan                                       155 513.516129        995
       79595

  DEPT_NUM DEPT_NAME                                 EMPLOYE_NUM
---------- ---------------------------------------- -----------
EMPLOYE_NAME                             COUNT_AMOUNT AVG_AMOUNT MAX_AMOUNT
---------------------------------------- ------------ ---------- ----------
TOTAL_AMOUNT
------------

        50 SALES - Neptune                                      1034
Jerome Johnston                                     146 541.267123        990
       79025

        50 SALES - Neptune                                      1033
Tim Watts                                           141 552.943262       1000

  DEPT_NUM DEPT_NAME                                 EMPLOYE_NUM
---------- ---------------------------------------- -----------
EMPLOYE_NAME                             COUNT_AMOUNT AVG_AMOUNT MAX_AMOUNT
---------------------------------------- ------------ ---------- ----------
TOTAL_AMOUNT
------------
       77965

        50 SALES - Neptune                                      1043
Paul Woods                                          142 545.739437       1000
       77495


47 rows selected.

Elapsed: 00:00:00.25
```

Q2: SELECT * FROM MV_DEPT_TOP_EMPS;

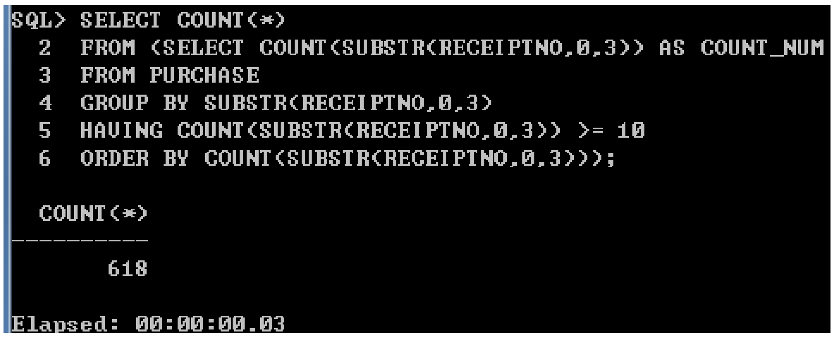


```
  DEPT_NUM DEPT_NAME                                 EMPLOYE_NUM
---------- ---------------------------------------- -----------
EMPLOYE_NAME                             COUNT_AMOUNT AVG_AMOUNT MAX_AMOUNT
---------------------------------------- ------------ ---------- ----------
TOTAL_AMOUNT
------------

        50 SALES - Neptune                                      1034
Jerome Johnston                                     146 541.267123        990
       79025

        50 SALES - Neptune                                      1033
Tim Watts                                           141 552.943262       1000

  DEPT_NUM DEPT_NAME                                 EMPLOYE_NUM
---------- ---------------------------------------- -----------
EMPLOYE_NAME                             COUNT_AMOUNT AVG_AMOUNT MAX_AMOUNT
---------------------------------------- ------------ ---------- ----------
TOTAL_AMOUNT
------------
       77965

        50 SALES - Neptune                                      1043
Paul Woods                                          142 545.739437       1000
       77495


47 rows selected.

Elapsed: 00:00:00.15
```

The execution time of Q1 is greater than execution time of Q2. Because Q2 is a materialized view and Q1 is a regular view, and materialized view store the result of the view's query, so materialized view is more efficient than regular view because it does not need to recompute the query every time they are used.

4a)

SELECT COUNT(*)
FROM (SELECT COUNT(SUBSTR(RECEIPTNO,0,3)) AS COUNT_NUM
FROM PURCHASE
GROUP BY SUBSTR(RECEIPTNO,0,3)
HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3)));



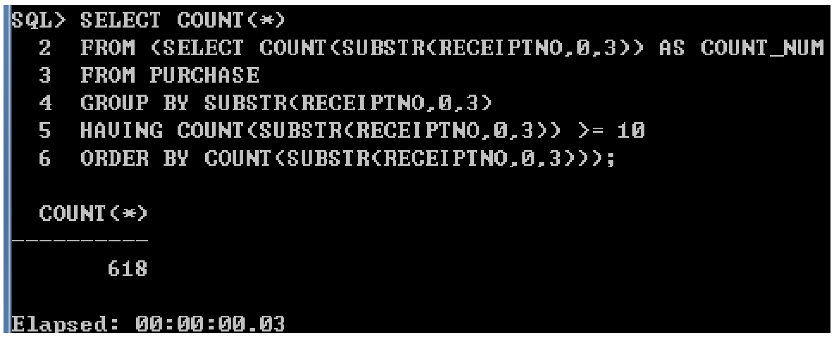
```
SQL> SELECT COUNT(*)
  2  FROM (SELECT COUNT(SUBSTR(RECEIPTNO,0,3)) AS COUNT_NUM
  3  FROM PURCHASE
  4  GROUP BY SUBSTR(RECEIPTNO,0,3)
  5  HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
  6  ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3)));

  COUNT(*)
----------
       618

Elapsed: 00:00:00.03
```

4b)

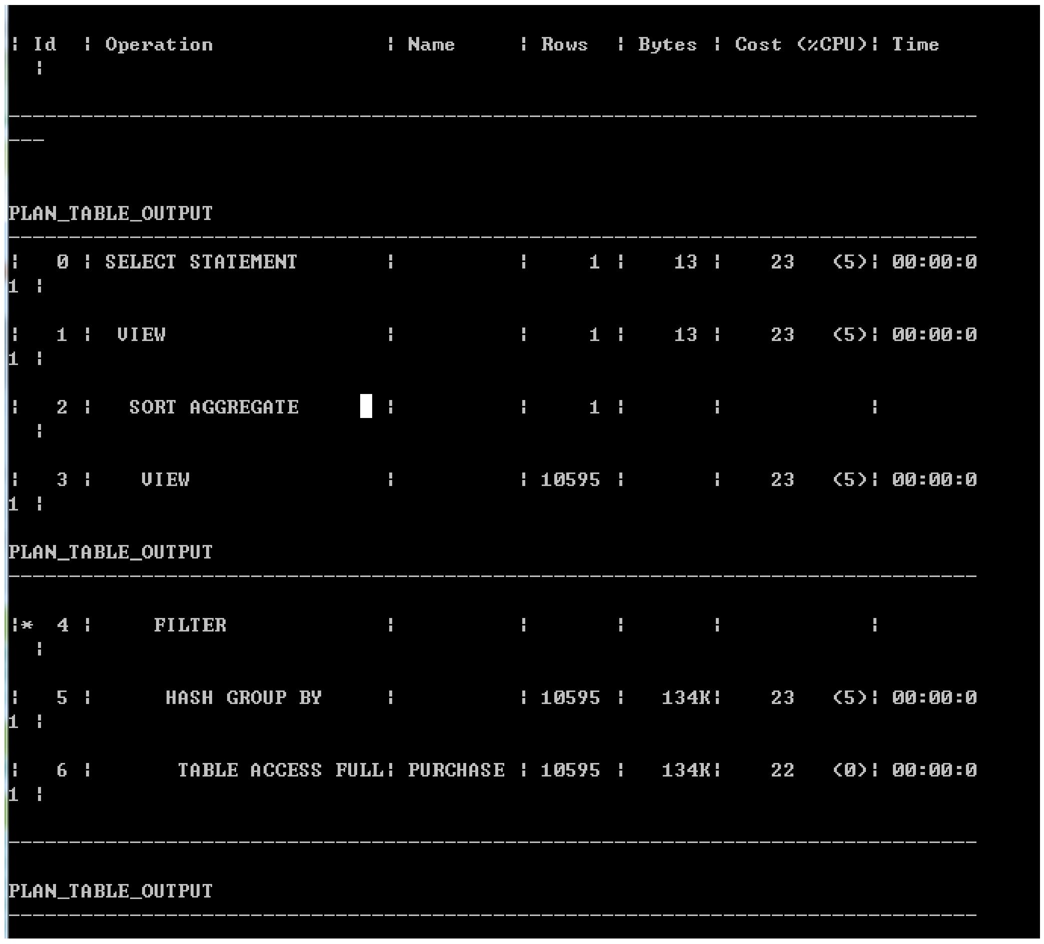
CREATE INDEX RECEIPT_BOOK ON PURCHASE(SUBSTR(RECEIPTNO,0,3));

EXPLAIN PLAN FOR SELECT * FROM (SELECT COUNT(*)
FROM (SELECT COUNT(SUBSTR(RECEIPTNO,0,3)) AS COUNT_NUM
FROM PURCHASE
GROUP BY SUBSTR(RECEIPTNO,0,3)
HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3))));
SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

Before:

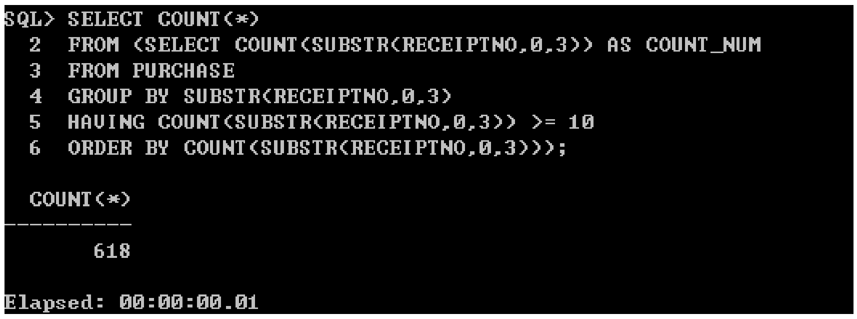


```
SQL> SELECT COUNT(*)
  2  FROM (SELECT COUNT(SUBSTR(RECEIPTNO,0,3)) AS COUNT_NUM
  3  FROM PURCHASE
  4  GROUP BY SUBSTR(RECEIPTNO,0,3)
  5  HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
  6  ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3)));

  COUNT(*)
----------
       618

Elapsed: 00:00:00.03
```

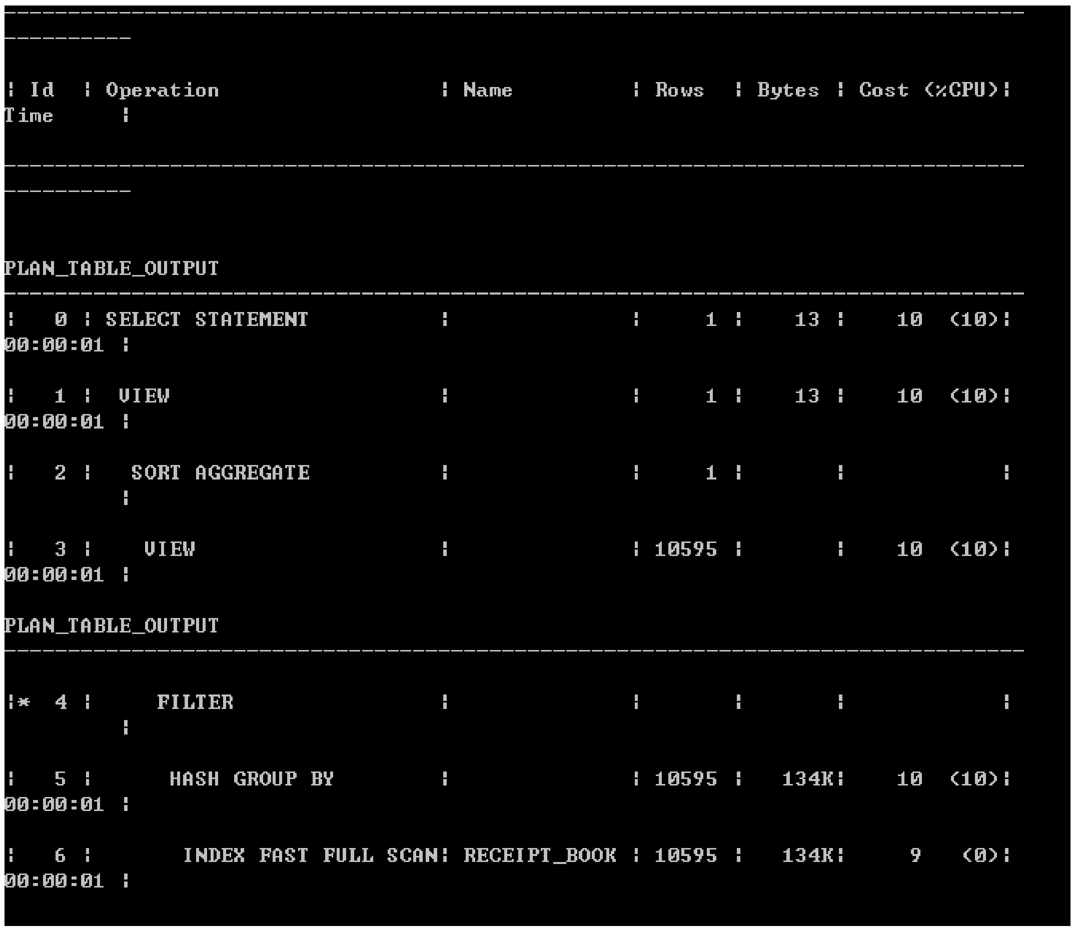


```
| Id  | Operation              | Name     | Rows  | Bytes | Cost (%CPU)| Time
  |

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |          |     1 |    13 |    23   (5)| 00:00:0
1  |

|   1 |  VIEW                  |          |     1 |    13 |    23   (5)| 00:00:0
1  |

|   2 |   SORT AGGREGATE       |          |     1 |       |            |
  |

|   3 |    VIEW                |          | 10595 |       |    23   (5)| 00:00:0
1  |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|*  4 |     FILTER             |          |       |       |            |
  |

|   5 |      HASH GROUP BY     |          | 10595 |   134K|    23   (5)| 00:00:0
1  |

|   6 |       TABLE ACCESS FULL| PURCHASE | 10595 |   134K|    22   (0)| 00:00:0
1  |

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
```

After:



```
SQL> SELECT COUNT(*)
  2  FROM (SELECT COUNT(SUBSTR(RECEIPTNO,0,3)) AS COUNT_NUM
  3  FROM PURCHASE
  4  GROUP BY SUBSTR(RECEIPTNO,0,3)
  5  HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
  6  ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3)));

  COUNT(*)
----------
       618

Elapsed: 00:00:00.01
```

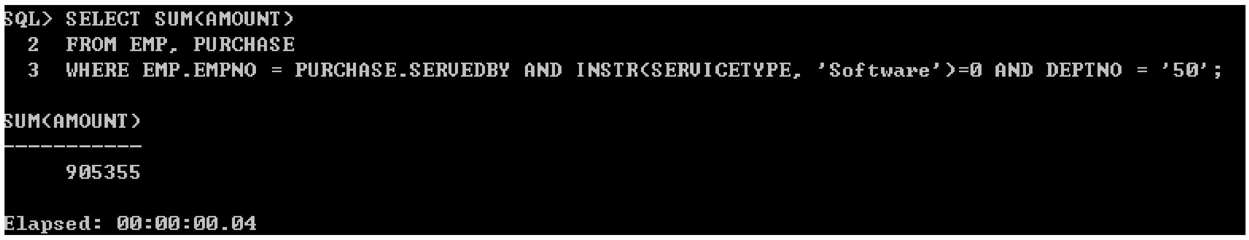


```
--------------------------------------------------------------------------------
----------

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |              |     1 |    13 |    10  (10)|
00:00:01 |

|   1 |  VIEW                     |              |     1 |    13 |    10  (10)|
00:00:01 |

|   2 |   SORT AGGREGATE          |              |     1 |       |            |
         |

|   3 |    VIEW                   |              | 10595 |       |    10  (10)|
00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|*  4 |     FILTER                |              |       |       |            |
         |

|   5 |      HASH GROUP BY        |              | 10595 |   134K|    10  (10)|
00:00:01 |

|   6 |       INDEX FAST FULL SCAN| RECEIPT_BOOK | 10595 |   134K|     9   (0)|
00:00:01 |
```

Yes, the index speeds up the query. The cost is 23<5> and row is 10595 when we do not use index, but when we use index, the cost is 10<10> and row is 10595 in the execution plan. Because if we create an index, it affects the way the data is physically ordered on the disk. It's better to add the index after the fact and let the database engine reorder the rows when it knows how the data is distributed.

4c)

SELECT SUM(AMOUNT)
FROM EMP, PURCHASE
WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND INSTR(SERVICETYPE, 'Software')=0 AND DEPTNO = '50';



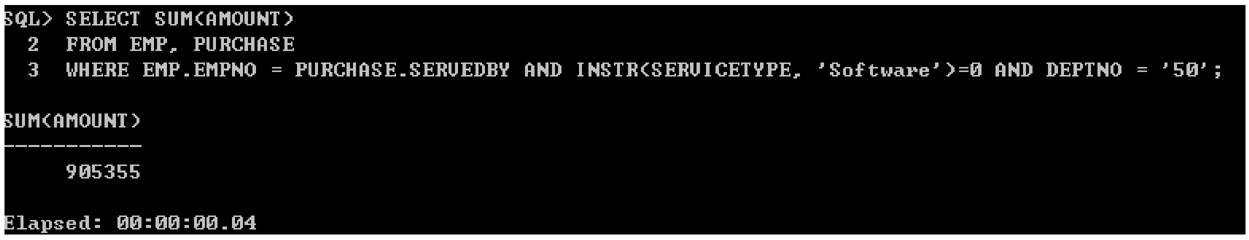
```
SQL> SELECT SUM(AMOUNT)
  2  FROM EMP, PURCHASE
  3  WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND INSTR(SERVICETYPE, 'Software')=0 AND DEPTNO = '50';

SUM(AMOUNT)
-----------
     905355

Elapsed: 00:00:00.04
```

4d)

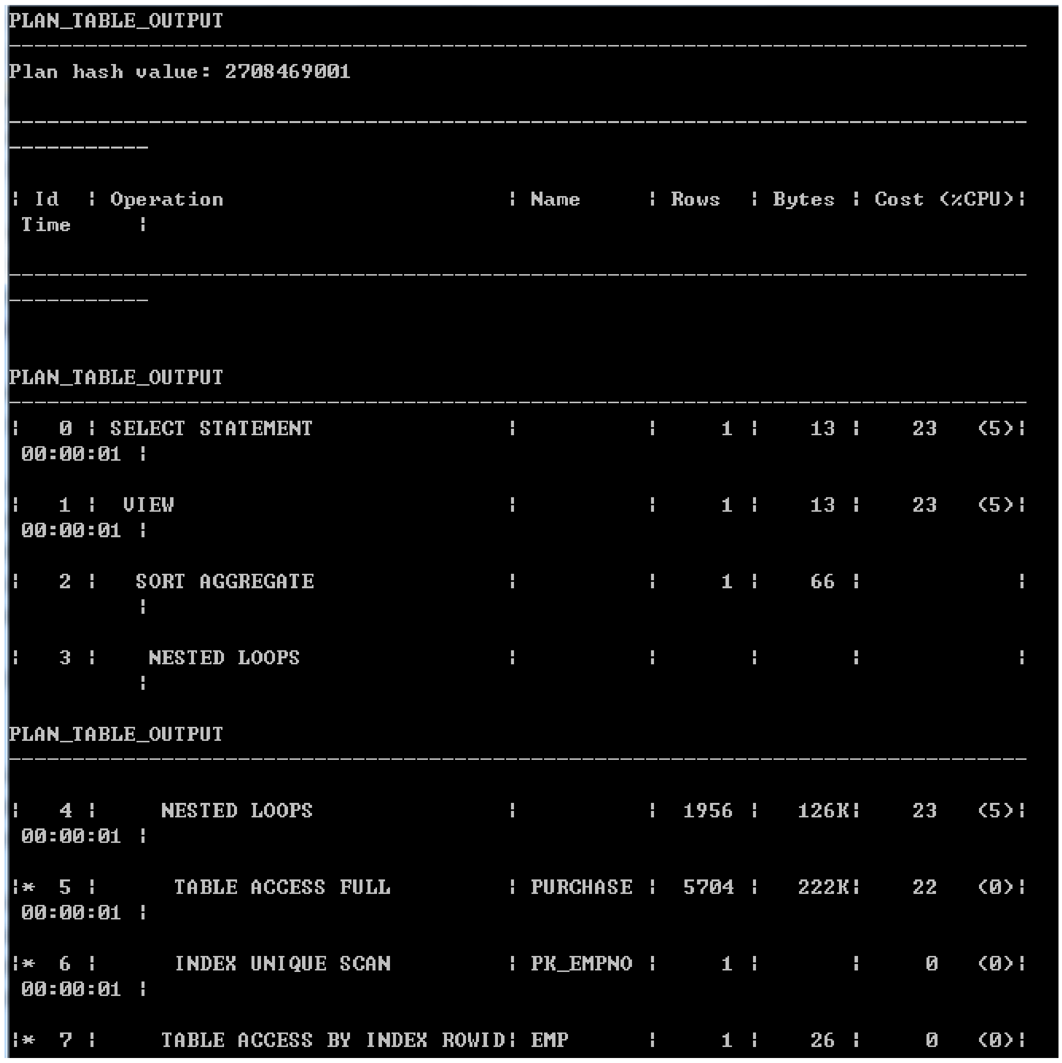
CREATE INDEX SERVICE_AMOUNT ON PURCHASE(INSTR(SERVICETYPE, 'Software'));

EXPLAIN PLAN FOR SELECT * FROM (SELECT SUM(AMOUNT)
FROM EMP, PURCHASE
WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND INSTR(SERVICETYPE, 'Software')=0 AND DEPTNO = '50');
SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

Before:

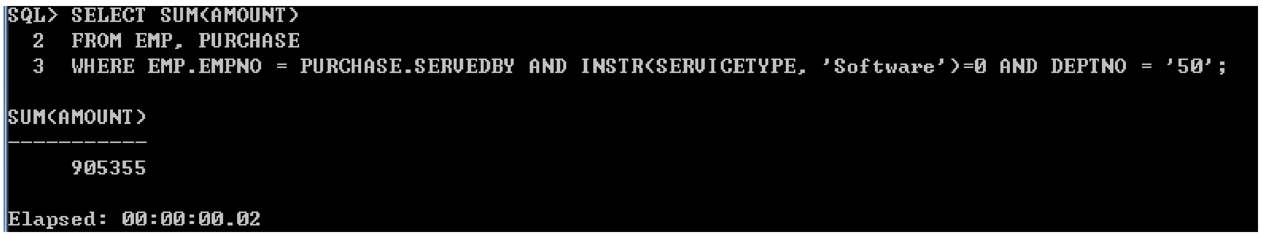


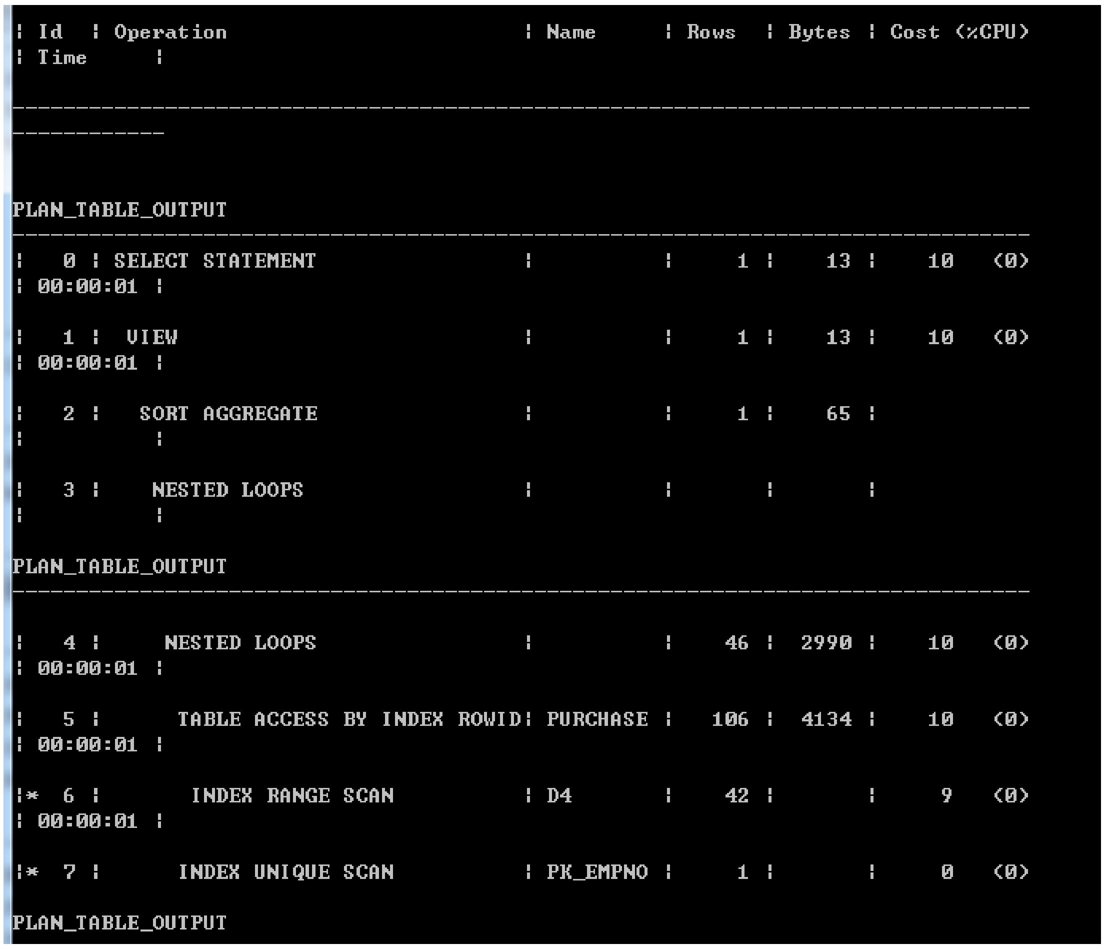
```
SQL> SELECT SUM(AMOUNT)
  2  FROM EMP, PURCHASE
  3  WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND INSTR(SERVICETYPE, 'Software')=0 AND DEPTNO = '50';

SUM(AMOUNT)
-----------
     905355

Elapsed: 00:00:00.04
```



```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2708469001

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |          |     1 |    13 |    23   (5)|
 00:00:01 |

|   1 |  VIEW                        |          |     1 |    13 |    23   (5)|
 00:00:01 |

|   2 |   SORT AGGREGATE             |          |     1 |    66 |            |
          |

|   3 |    NESTED LOOPS              |          |       |       |            |
          |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |     NESTED LOOPS             |          |  1956 |  126K|    23   (5)|
 00:00:01 |

|*  5 |      TABLE ACCESS FULL       | PURCHASE |  5704 |  222K|    22   (0)|
 00:00:01 |

|*  6 |      INDEX UNIQUE SCAN       | PK_EMPNO |     1 |       |     0   (0)|
 00:00:01 |

|*  7 |     TABLE ACCESS BY INDEX ROWID| EMP    |     1 |    26 |     0   (0)|
```

After:

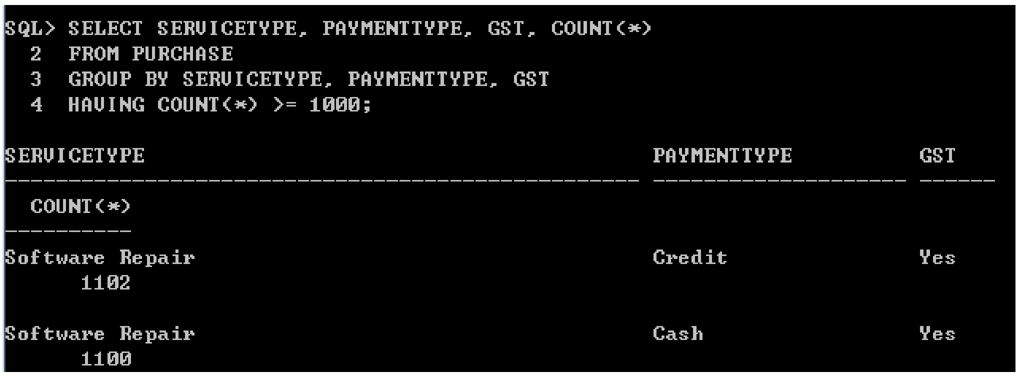

```
SQL> SELECT SUM(AMOUNT)
  2  FROM EMP, PURCHASE
  3  WHERE EMP.EMPNO = PURCHASE.SERVEDBY AND INSTR(SERVICETYPE, 'Software')=0 AND DEPTNO = '50';

SUM(AMOUNT)
-----------
     905355

Elapsed: 00:00:00.02
```


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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |           |     1 |    13 |    10   (0)
| 00:00:01 |

|   1 |  VIEW                            |           |     1 |    13 |    10   (0)
| 00:00:01 |

|   2 |   SORT AGGREGATE                 |           |     1 |    65 |
|          |

|   3 |    NESTED LOOPS                  |           |       |       |
|          |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |     NESTED LOOPS                 |           |    46 |  2990 |    10   (0)
| 00:00:01 |

|   5 |      TABLE ACCESS BY INDEX ROWID| PURCHASE  |   106 |  4134 |    10   (0)
| 00:00:01 |

|*  6 |       INDEX RANGE SCAN           | D4        |    42 |       |     9   (0)
| 00:00:01 |

|*  7 |      INDEX UNIQUE SCAN           | PK_EMPNO  |     1 |       |     0   (0)

PLAN_TABLE_OUTPUT
```

Yes, the index speeds up the query. The cost is 23<5> and row is 5704 when we do not use index, but when we use index, the cost is 10<0> and row is 106 in the execution plan. Because if we create an index, it affects the way the data is physically ordered on the disk. It's better to add the index after the fact and let the database engine reorder the rows when it knows how the data is distributed.

5a)

SELECT SERVICETYPE, PAYMENTTYPE, GST, COUNT(*)
FROM PURCHASE
GROUP BY SERVICETYPE, PAYMENTTYPE, GST
HAVING COUNT(*) >= 1000;



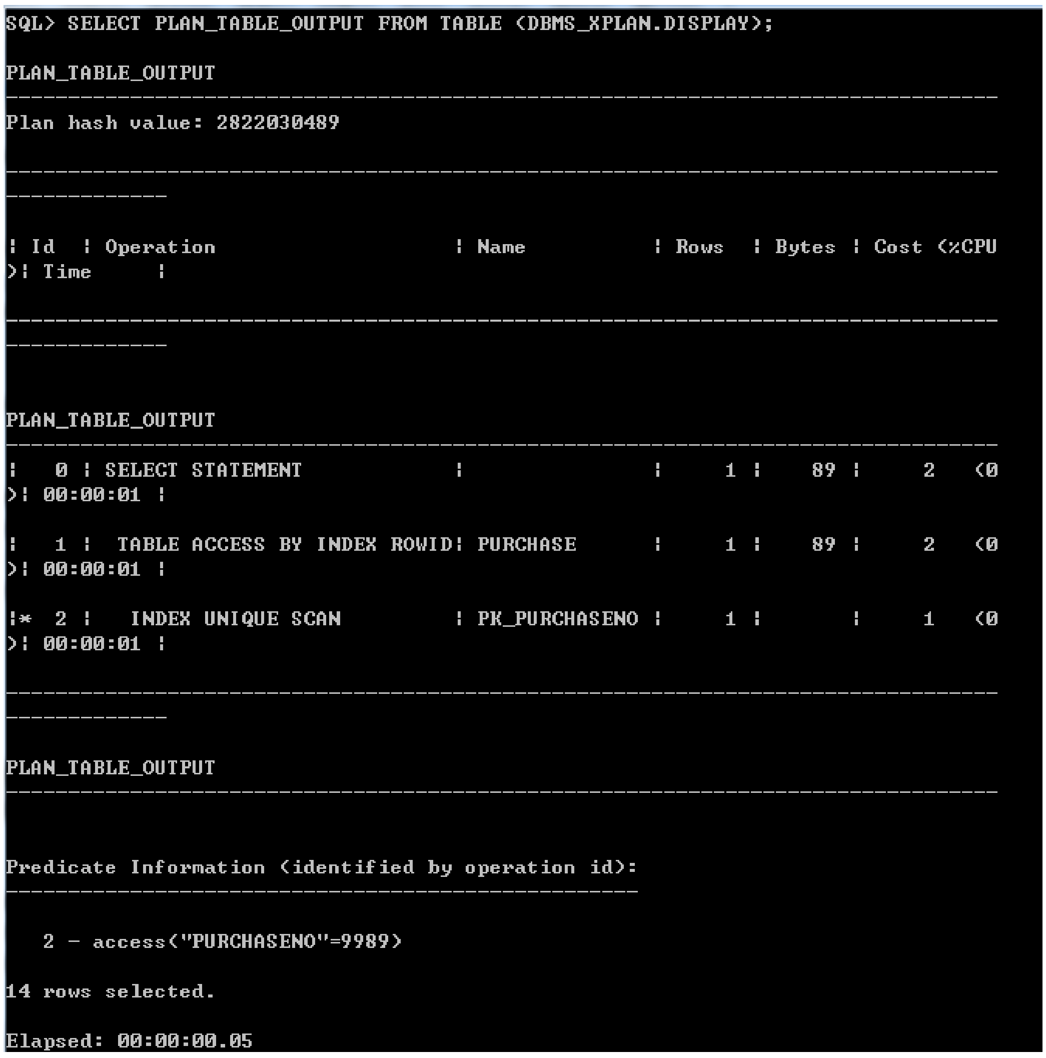
```
SQL> SELECT SERVICETYPE, PAYMENTTYPE, GST, COUNT(*)
  2  FROM PURCHASE
  3  GROUP BY SERVICETYPE, PAYMENTTYPE, GST
  4  HAVING COUNT(*) >= 1000;

SERVICETYPE                                        PAYMENTTYPE          GST
-------------------------------------------------- -------------------- ------
  COUNT(*)
----------
Software Repair                                    Credit               Yes
      1102

Software Repair                                    Cash                 Yes
      1100
```

5b)

The bitmap index, because it facilitates querying on multiple keys. When we use bitmap index in multiple column, one of the advantage is that multiple bitmap indexes can be merged and the column does not have to selective.

6a)

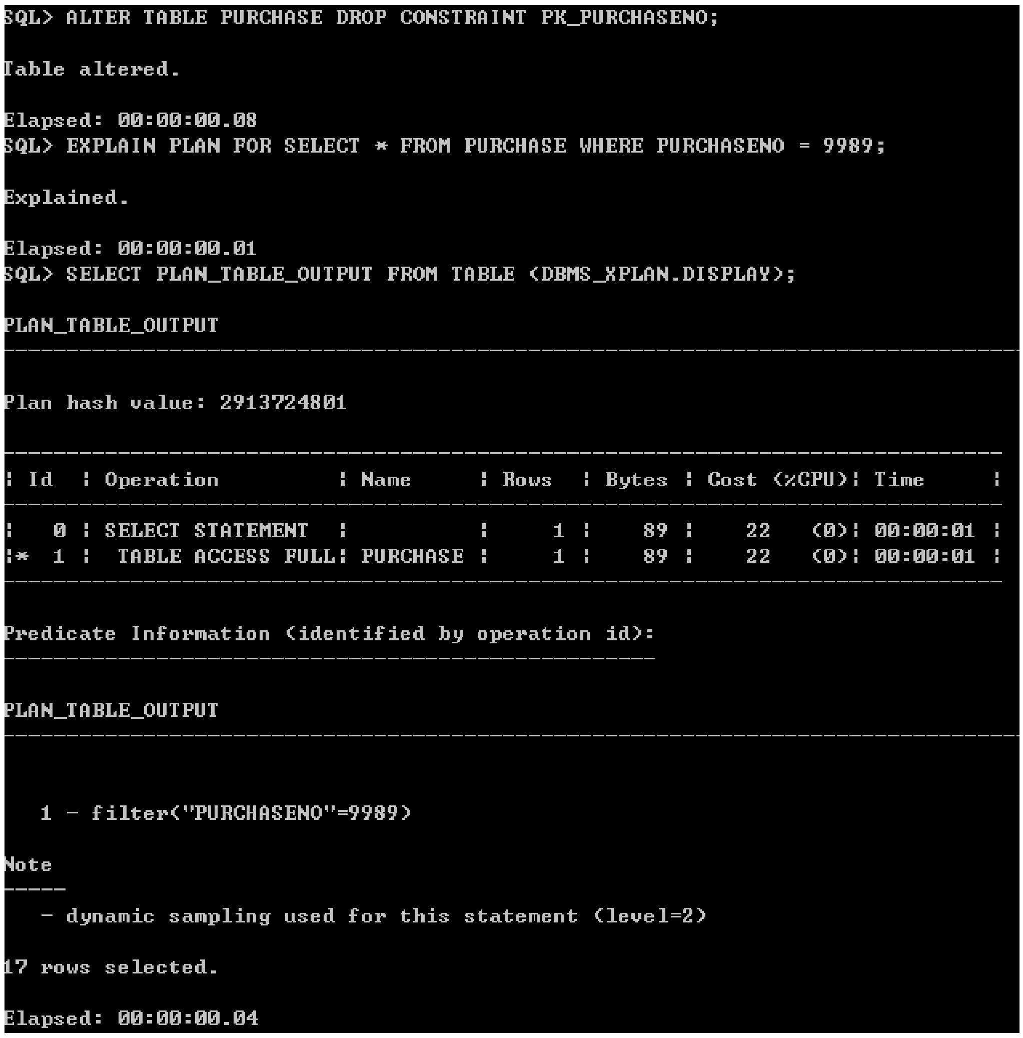
EXPLAIN PLAN FOR SELECT * FROM PURCHASE WHERE PURCHASENO = 9989;
SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);



```
SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2822030489

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

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

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


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |               |     1 |    89 |     2   (0
)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| PURCHASE      |     1 |    89 |     2   (0
)| 00:00:01 |

|*  2 |   INDEX UNIQUE SCAN         | PK_PURCHASENO |     1 |       |     1   (0
)| 00:00:01 |

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("PURCHASENO"=9989)

14 rows selected.

Elapsed: 00:00:00.05
```

6b)

ALTER TABLE PURCHASE DROP CONSTRAINT PK_PURCHASENO;
EXPLAIN PLAN FOR SELECT * FROM PURCHASE WHERE PURCHASENO = 9989;
SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);



```
SQL> ALTER TABLE PURCHASE DROP CONSTRAINT PK_PURCHASENO;

Table altered.

Elapsed: 00:00:00.08
SQL> EXPLAIN PLAN FOR SELECT * FROM PURCHASE WHERE PURCHASENO = 9989;

Explained.

Elapsed: 00:00:00.01
SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 2913724801

------------------------------------------------------------------------------
| Id  | Operation         | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |     1 |    89 |    22   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| PURCHASE |     1 |    89 |    22   (0)| 00:00:01 |
------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------


   1 - filter("PURCHASENO"=9989)

Note
-----
   - dynamic sampling used for this statement (level=2)

17 rows selected.

Elapsed: 00:00:00.04
```

In this 6b plan table, the operation is that table access full in purchase table, this means it has been executed first, and its output is then fed to the select operation (the purchase table is accessed using a full table scan). But in 6a plan table, the operation is that table access by index rowed and it also has index unique scan in purchase table.