INFS2200 Assignment report

Student Name：DINGXIN YU

Student number：42834821

Task 0

@C:\prjScript.sql.txt

```
--1A

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


--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);


--2B
CREATE OR REPLACE TRIGGER DISCOUNT_TOP_CLIENT
BEFORE INSERT ON PURCHASE
FOR EACH ROW
DECLARE
NEWAMT NUMBER:=:NEW.AMOUNT*0.85;
TC NUMBER:=24535;
begin
	IF :NEW.CLIENTNO=TC THEN
		IF INSERTING THEN
			:NEW.AMOUNT :=NEWAMT;
		END IF;
	END IF;
END DISCOUNT_TOP_CLIENT;
/


--2C
CREATE OR REPLACE TRIGGER DISCOUNT_SUN
BEFORE INSERT ON PURCHASE
FOR EACH ROW
DECLARE
DN NUMBER(2);
BEGIN
	SELECT DEPTNO INTO DN FROM EMP WHERE EMPNO=:NEW.SERVEDBY;
```

```
        IF (DN=20) THEN
                IF INSERTING THEN
                        :NEW.PAYMENTTYPE:='Cash';
                        IF (:NEW.SERVICETYPE='Data Recovery') THEN
                                :NEW.AMOUNT:=:NEW.AMOUNT*0.7;
                        END IF;
                END IF;
        END IF;
END DISCOUNT_SUN;
/


--3A
CREATE VIEW V_DEPT_AMOUNT AS
SELECT DEPT.DEPTNO AS NUM,DEPT.DNAME AS NAME, MAX(PURCHASE.AMOUNT) AS MAXIUM,
MIN(PURCHASE.AMOUNT) AS MINIUM, AVG(PURCHASE.AMOUNT)AS
AVERAGE,SUM(PURCHASE.AMOUNT) AS TOTAL
FROM DEPT, EMP, PURCHASE
WHERE PURCHASE.SERVEDBY=EMP.EMPNO AND DEPT.DEPTNO=EMP.DEPTNO
GROUP BY DEPT.DEPTNO,DEPT.DNAME;


--3B
CREATE MATERIALIZED VIEW MV_DEPT_AMOUNT
BUILD IMMEDIATE AS
SELECT DEPT.DEPTNO AS NUM,DEPT.DNAME AS NAME, MAX(PURCHASE.AMOUNT) AS MAXIUM,
MIN(PURCHASE.AMOUNT) AS MINIUM, AVG(PURCHASE.AMOUNT)AS
AVERAGE,SUM(PURCHASE.AMOUNT) AS TOTAL
FROM DEPT, EMP, PURCHASE
WHERE PURCHASE.SERVEDBY=EMP.EMPNO AND DEPT.DEPTNO=EMP.DEPTNO
GROUP BY DEPT.DEPTNO,DEPT.DNAME;
--3C
SET TIMING ON;
Q1：SELECT * FROM V_DEPT_AMOUNT;
```

Q2：SELECT * FROM MV_DEPT_AMOUNT;

```
Run SQL Command Line

SQL> SET TIMING ON;
SQL> SELECT * FROM V_DEPT_AMOUNT;

       NUM NAME                     MAXIUM     MINIUM    AVERAGE      TOTAL
---------- -------------------- ---------- ---------- ---------- ----------
        30 SALES - Sunflower          1000         50 528.336968     968970
        40 SALES - Hercules           1000         50 535.761089    1062950
        20 SALES - Sunshine           1000         50 522.126719    1063050
        10 SALES - Glorious           1000         50 522.730769     475685
        50 SALES - Neptune            1000         50 517.578053    1674365

Elapsed: 00:00:00.13
SQL> SELECT * FROM MV_DEPT_AMOUNT;

       NUM NAME                     MAXIUM     MINIUM    AVERAGE      TOTAL
---------- -------------------- ---------- ---------- ---------- ----------
        30 SALES - Sunflower          1000         50 528.336968     968970
        40 SALES - Hercules           1000         50 535.761089    1062950
        20 SALES - Sunshine           1000         50 522.126719    1063050
        10 SALES - Glorious           1000         50 522.730769     475685
        50 SALES - Neptune            1000         50 517.578053    1674365

Elapsed: 00:00:00.01
SQL> ^A_
```

Because of the materialized view is more effictive than regular view. It takes less time.

--3D

CREATE VIEW V_DEPT_TOP_EMPS AS

SELECT * FROM(

SELECT DEPT.DNAME,DEPT.DEPTNO AS DNO,EMP.EMPNO AS ENUM, EMP.ENAME AS ENAME, MAX(PURCHASE.AMOUNT) AS MAXIUM, AVG(PURCHASE.AMOUNT)AS AVERAGE,SUM(PURCHASE.AMOUNT) AS TOTAL, COUNT(*) AS CNT

FROM EMP, PURCHASE,DEPT

WHERE PURCHASE.SERVEDBY=EMP.EMPNO AND DEPT.DEPTNO=EMP.DEPTNO

GROUP BY EMP.EMPNO,EMP.ENAME,DEPT.DEPTNO,DEPT.DNAME) T

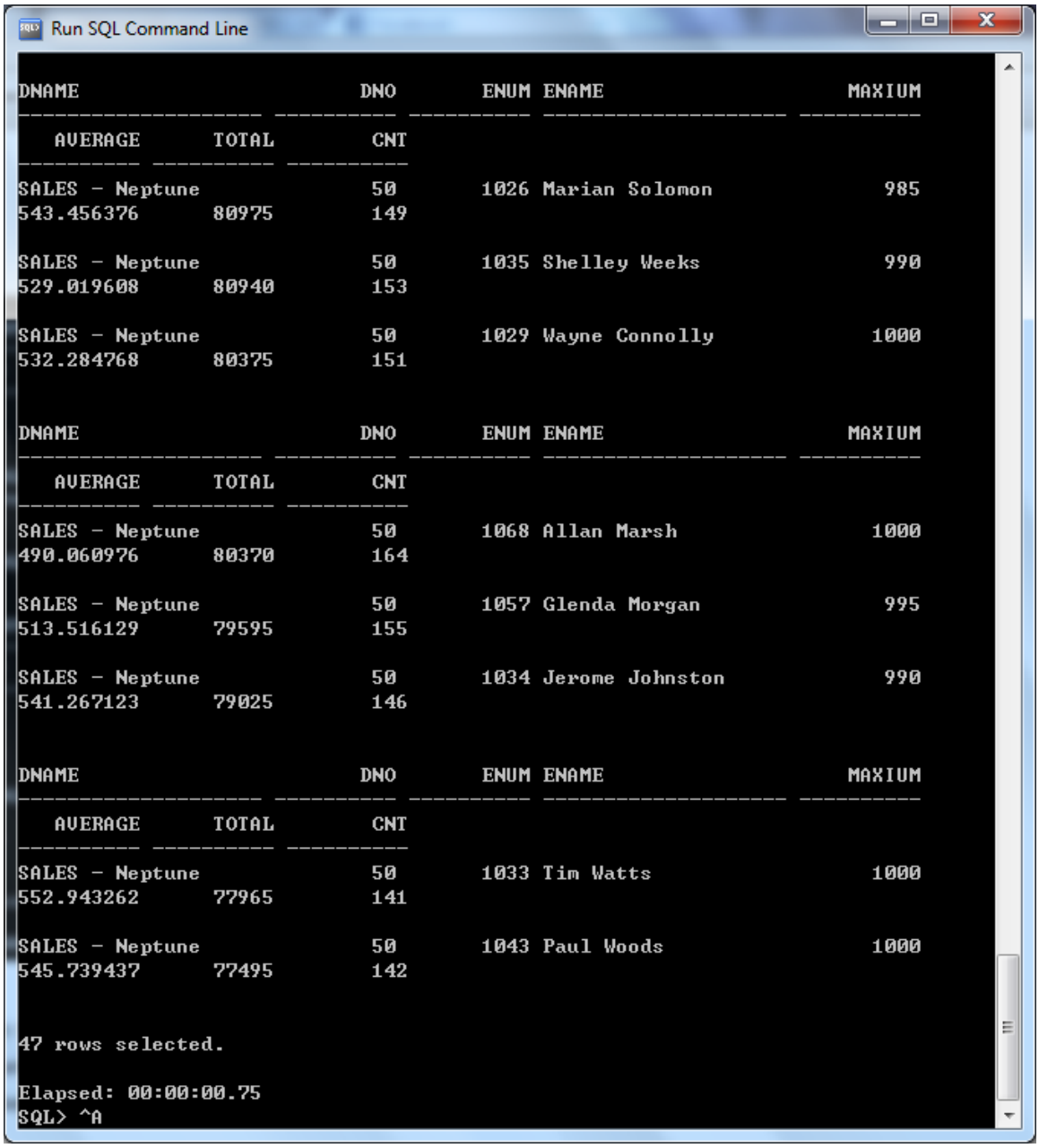
WHERE(

SELECT COUNT (*)

FROM(

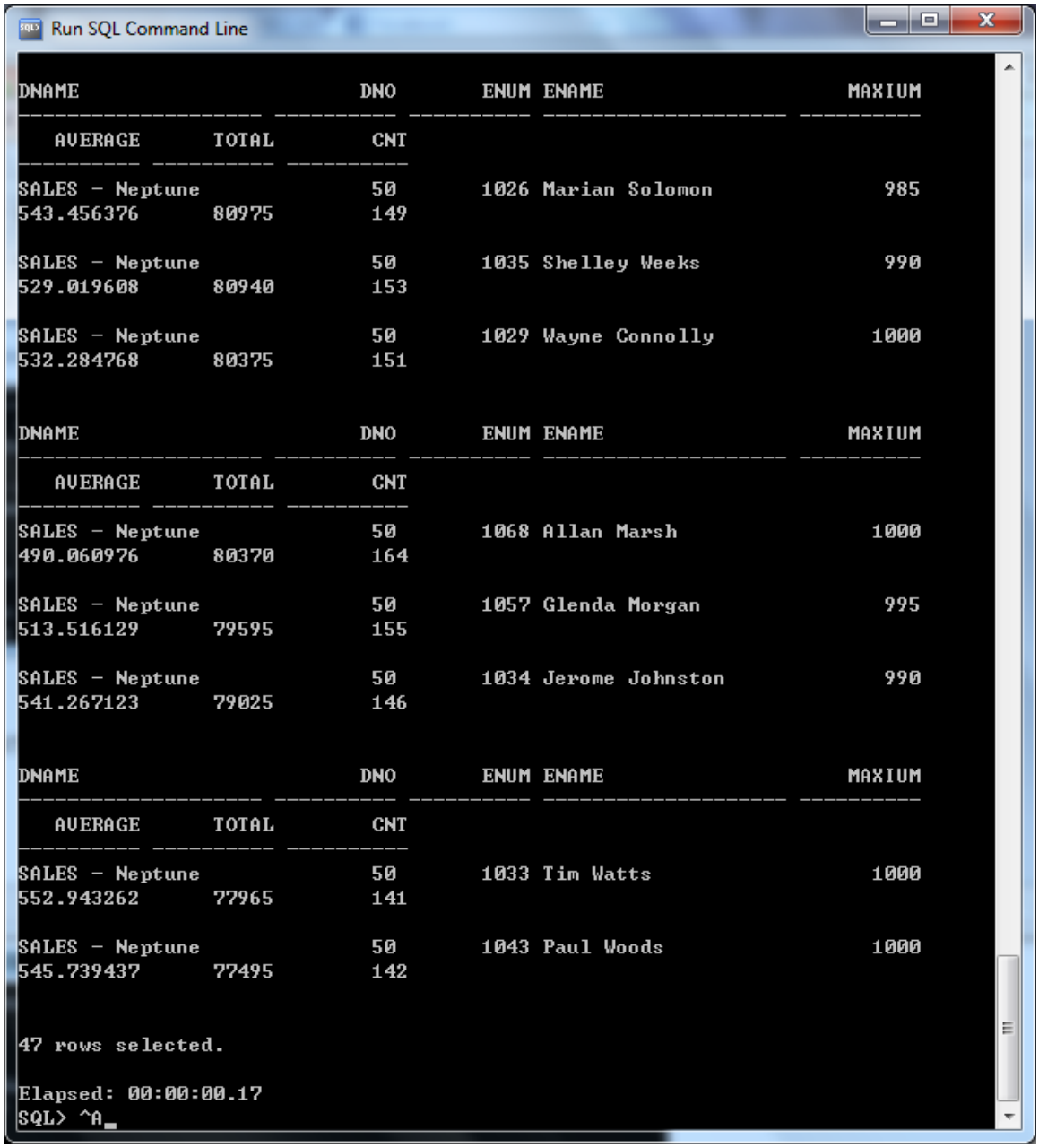
SELECT DEPT.DNAME,DEPT.DEPTNO AS DNO,EMP.EMPNO AS ENUM, EMP.ENAME AS ENAME, MAX(PURCHASE.AMOUNT) AS MAXIUM, AVG(PURCHASE.AMOUNT)AS AVERAGE,SUM(PURCHASE.AMOUNT) AS TOTAL

```
                FROM EMP, PURCHASE,DEPT

                WHERE PURCHASE.SERVEDBY=EMP.EMPNO AND DEPT.DEPTNO=EMP.DEPTNO

                GROUP BY EMP.EMPNO,EMP.ENAME,DEPT.DEPTNO,DEPT.DNAME)

        WHERE DNO=T.DNO AND TOTAL>=T.TOTAL

        )<=10

ORDER BY DNO ASC,TOTAL DESC;


CREATE MATERIALIZED VIEW MV_DEPT_TOP_EMPS

BUILD IMMEDIATE AS

SELECT * FROM(

        SELECT DEPT.DNAME,DEPT.DEPTNO AS DNO,EMP.EMPNO AS ENUM, EMP.ENAME AS
ENAME, MAX(PURCHASE.AMOUNT) AS MAXIUM, AVG(PURCHASE.AMOUNT)AS
AVERAGE,SUM(PURCHASE.AMOUNT) AS TOTAL, COUNT(*) AS CNT

        FROM EMP, PURCHASE,DEPT

        WHERE PURCHASE.SERVEDBY=EMP.EMPNO AND DEPT.DEPTNO=EMP.DEPTNO

        GROUP BY EMP.EMPNO,EMP.ENAME,DEPT.DEPTNO,DEPT.DNAME) T

WHERE(

        SELECT COUNT (*)

        FROM(

                SELECT DEPT.DNAME,DEPT.DEPTNO AS DNO,EMP.EMPNO AS ENUM, EMP.ENAME
AS ENAME, MAX(PURCHASE.AMOUNT) AS MAXIUM, AVG(PURCHASE.AMOUNT)AS
AVERAGE,SUM(PURCHASE.AMOUNT) AS TOTAL

                FROM EMP, PURCHASE,DEPT

                WHERE PURCHASE.SERVEDBY=EMP.EMPNO AND DEPT.DEPTNO=EMP.DEPTNO

                GROUP BY EMP.EMPNO,EMP.ENAME,DEPT.DEPTNO,DEPT.DNAME)

        WHERE DNO=T.DNO AND TOTAL>=T.TOTAL

        )<=10

ORDER BY DNO ASC,TOTAL DESC;

--3E

Q1:SELECT * FROM V_DEPT_TOP_EMPS;
```



```
Run SQL Command Line

DNAME                      DNO       ENUM ENAME                   MAXIUM
-------------------- ---------- ---------- -------------------- ----------
   AVERAGE      TOTAL        CNT
---------- ---------- ----------
SALES - Neptune              50       1026 Marian Solomon             985
543.456376      80975        149

SALES - Neptune              50       1035 Shelley Weeks              990
529.019608      80940        153

SALES - Neptune              50       1029 Wayne Connolly            1000
532.284768      80375        151


DNAME                      DNO       ENUM ENAME                   MAXIUM
-------------------- ---------- ---------- -------------------- ----------
   AVERAGE      TOTAL        CNT
---------- ---------- ----------
SALES - Neptune              50       1068 Allan Marsh               1000
490.060976      80370        164

SALES - Neptune              50       1057 Glenda Morgan              995
513.516129      79595        155

SALES - Neptune              50       1034 Jerome Johnston            990
541.267123      79025        146


DNAME                      DNO       ENUM ENAME                   MAXIUM
-------------------- ---------- ---------- -------------------- ----------
   AVERAGE      TOTAL        CNT
---------- ---------- ----------
SALES - Neptune              50       1033 Tim Watts                 1000
552.943262      77965        141

SALES - Neptune              50       1043 Paul Woods                1000
545.739437      77495        142


47 rows selected.

Elapsed: 00:00:00.75
SQL> ^A
```

Q2:SELECT * FROM MV_DEPT_TOP_EMPS;



```
Run SQL Command Line

DNAME                       DNO       ENUM ENAME                   MAXIUM
-------------------- ---------- ---------- -------------------- ----------
   AVERAGE      TOTAL        CNT
---------- ---------- ----------
SALES - Neptune              50       1026 Marian Solomon             985
543.456376      80975        149

SALES - Neptune              50       1035 Shelley Weeks              990
529.019608      80940        153

SALES - Neptune              50       1029 Wayne Connolly            1000
532.284768      80375        151


DNAME                       DNO       ENUM ENAME                   MAXIUM
-------------------- ---------- ---------- -------------------- ----------
   AVERAGE      TOTAL        CNT
---------- ---------- ----------
SALES - Neptune              50       1068 Allan Marsh               1000
490.060976      80370        164

SALES - Neptune              50       1057 Glenda Morgan              995
513.516129      79595        155

SALES - Neptune              50       1034 Jerome Johnston            990
541.267123      79025        146


DNAME                       DNO       ENUM ENAME                   MAXIUM
-------------------- ---------- ---------- -------------------- ----------
   AVERAGE      TOTAL        CNT
---------- ---------- ----------
SALES - Neptune              50       1033 Tim Watts                 1000
552.943262      77965        141

SALES - Neptune              50       1043 Paul Woods                1000
545.739437      77495        142


47 rows selected.

Elapsed: 00:00:00.17
SQL> ^A_
```

Because the metrialize view is more effictive than regular one. Hence it take less time.

--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 GROUP BY SUBSTR(RECEIPTNO,0,3)
  4  HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
  5  ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3)));

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

Elapsed: 00:00:00.02
SQL>
```

--4B

CREATE INDEX BOOK_RECEIPT ON PURCHASE(SUBSTR(RECEIPTNO,1,3));

EXPLAIN PLAN FOR 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 GROUP BY SUBSTR(RECEIPTNO,0,3)
  4  HAVING COUNT(SUBSTR(RECEIPTNO,0,3)) >= 10
  5  ORDER BY COUNT(SUBSTR(RECEIPTNO,0,3)));

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

Elapsed: 00:00:00.03
```

After:

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

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

Elapsed: 00:00:00.02
```

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

| 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, index does speed up the query.Also, it helps to reduce the elapsed time when query are run. It affect the data storage on disk. And also it reduce the cost of CPU. Besides, sometime the elapsed may not correct show the efficiency because of disk.

--4C

SELECT SUM(AMOUNT) FROM PURCHASE,EMP

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

--4D

CREATE INDEX D4 ON PURCHASE(INSTR(SERVICETYPE, 'Software'));

EXPLAIN PLAN FOR SELECT SUM(AMOUNT) FROM PURCHASE,EMP

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

SELECT PLAN_TABLE_OUTPUT FROM TABLE (DBMS_XPLAN.DISPLAY);

Before:

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

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

Elapsed: 00:00:00.03
```

```
Run SQL Command Line
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

Plan hash value: 1715138504

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

----------

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

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

----------


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

|   0 | SELECT STATEMENT               |          |     1 |    66 |    22   (0)|
00:00:01 |

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

|   2 |   NESTED LOOPS                 |          |       |       |            |
         |

|   3 |    NESTED LOOPS                |          |  1627 |  104K|    22   (0)|
00:00:01 |

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


|*  4 |     TABLE ACCESS FULL          | PURCHASE |  5086 |  198K|    22   (0)|
00:00:01 |

|*  5 |     INDEX UNIQUE SCAN          | PK_EMPNO |     1 |       |     0   (0)|
00:00:01 |

|*  6 |    TABLE ACCESS BY INDEX ROWID| EMP      |     1 |    26 |     0   (0)|
00:00:01 |
```

After:

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

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

Elapsed: 00:00:00.02
```

```
Run SQL Command Line
-----------


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

|   0 | SELECT STATEMENT                |          |     1 |    65 |    10   (0)|

 00:00:01 |

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

          |

|   2 |   NESTED LOOPS                  |          |       |       |            |

          |

|   3 |    NESTED LOOPS                 |          |    42 |  2730 |    10   (0)|

 00:00:01 |

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


|   4 |     TABLE ACCESS BY INDEX ROWID| PURCHASE |    94 |  3666 |    10   (0)|

 00:00:01 |

|*  5 |      INDEX RANGE SCAN           | D4       |    38 |       |     9   (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)|


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

Yes, index does speed up the query.Also, it helps to reduce the elapsed time when query are run. It affect the data storage on disk. And also it reduce the cost of CPU. Besides, sometime the elapsed may not correct show the efficiency because of disk.

--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                PAYMENTTYP GST   COUNT(*)
-------------------------- ---------- --- ----------
Software Repair            Credit     Yes       1102
Software Repair            Cash       Yes       1100
```

--5B

Bitmap is choosen in this situation.Bitmap index facilitates querying on multiples keys. Under this index type, Row id should be easily mapped to a physical address.

(block address)

--6A

EXPLAIN PLAN FOR

SELECT * FROM PURCHASE WHERE PURCHASENO=9989;

SELECT PLAN_TABLE_OUTPUT FROM
TABLE(DBMS_XPLAN.DISPLAY);

```
Run SQL Command Line
--------------------------------------------------------------------------------

Plan hash value: 2822030489

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

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

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

>| Time     |

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

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


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

|   0 | SELECT STATEMENT             |               |     1 |    89 |     0   (0

>| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| PURCHASE      |     1 |    89 |     0   (0

>| 00:00:01 |

|*  2 |   INDEX UNIQUE SCAN          | PK_PURCHASENO |     1 |       |     0   (0

>| 00:00:01 |

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

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

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



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

--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);

```
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
```

One of the unique constraint was dropped. It leads to the system will not create index automatically. Afterwards the cost increase.