Oracle SQL

DBS 2017/2018 Spring

PUBLIC TRANSPORTATION SYSTEM (PTS)

We need to store personal data of employees, completed work hours, information about vehicle fleet and lines. Important data about employees are: name, birth date, address, hire date and monthly salary and phone numbers. For each employee we need to know her/his immediate boss, (the person to whom she/he reports). We need additional information on drivers. We need to know their types of driving license, how many kilometers have they driven, and how many accidents they have had. For the purpose of quality assurance, the number of accidents per kilometer is continuously monitored for drivers. Since our company has family incentives (e.g., free monthly tickets for family members, child care center, wives' club, etc.), we need to know the relatives of our employees. However, the only thing we really need is the family member status, that is spouse, child1, child2, mother, father, sibling, etc., and their names.

ORACLE: TABLE DUAL

DUAL-A DUMMY RELATION for output purposes

Examples:

SQL> SELECT SYSDATE FROM **DUAL**;

Result:

SYSDATE

06-MARC. -13

SQL>SELECT 2*2 FROM DUAL;

Result:

2*2

4

SET OPERATIONS IN SQL

```
SELECT JOB
FROM EMP
WHERE DEPTNO=10
UNION / INTERSECT/MINUS
SELECT JOB FROM EMP
WHERE DEPTNO=20;
```

RESULT:

```
JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
```

```
JOB
---------
CLERK
MANAGER
```

```
JOB
---------
PRESIDENT
```

SOME OTHER ORACLE METATABLES

```
SQL> select   owner, object_name
     from     all_objects
     where    owner='B_NOVAK';
```

owner	object_name
B_NOVAK	JOBHIST_PK
B_NOVAK	SALESEMP
B_NOVAK	DATUMOK
B_NOVAK	SAL
B_NOVAK	EMP_SALS
B_NOVAK	DEPT10

MORE ON SELECT CLAUSE-ORACLE FUNCTIONS

```
SQL>    SELECT DNAME AS DEPARTMENT
        FROM DEPT;
```

Result:

```
DEPARTMENT
--------------
OPERATIONS
SALES
RESEARCH
ACCOUNTING
```

```
SQL>    SELECT LOWER(DNAME) AS DEPARTMENT
        FROM DEPT;
```

Result:

```
DEPARTMENT
--------------
operations
sales
research
accounting
```

More on SELECT CLAUSE-ORACLE STRING FUNCTIONS

```
SQL>    SELECT DNAME, LOC
        FROM DEPT;

DNAME          LOC
-------------- ----------
OPERATIONS      BOSTON
SALES          CHICAGO
RESEARCH        DALLAS
```

```
SQL>    SELECT INITCAP(LOWER(DNAME)),INITCAP(LOWER(LOC))
        FROM DEPT;

INITCAP(LOWER(DNAME)        INITCAP(LOWER(LOC))
--------------              ----------
Operations                  Boston
Sales                       Chicago
Research                    Dallas
Accounting                  New York
```

More on SELECT CLAUSE-ORACLE STRING FUNCTIONS

```
SQL> SELECT LENGTH(DNAME), DNAME
     FROM   DEPT;

LENGTH(DNAME)       DNAME
-------------       --------------
           15       OPERATIONS
           15       SALES
           15       RESEARCH
           15       ACCOUNTING
```

DNAME was defined as CHAR(15)

SELECT CLAUSE AGGREGATE FUNCTIONS

```
SELECT DEPTNO, MIN(SAL)
FROM EMP
GROUP BY DEPTNO;
```

DEPTNO	MIN(SAL)
10	1300
20	800
30	950

SELECT CLAUSE AGGREGATE FUNCTIONS

```
SQL>SELECT DEPTNO, COUNT(EMPNO)
      FROM EMP
      GROUP BY DEPTNO;
```

DEPTNO	COUNT(EMPNO)
10	3
20	5
30	6

SELECT CLAUSE AGGREGATE FUNCTIONS

FIND THE AVERAGE ANNUAL SALARY OF THE NON MANAGERIAL STAFF:

```
SQL>
SELECT DEPTNO, 12*AVG(SAL)
FROM EMP
WHERE JOB NOT IN ('MANAGER', 'PRESIDENT')
GROUP BY DEPTNO;
```

Result:

DEPTNO	12*AVG(SAL)
10	15600
20	23700
30	15720

SELECT CLAUSE AGGREGATE FUNCTIONS

```
SQL>SELECT DEPTNO, AVG(SAL)
        FROM EMP
        GROUP BY DEPTNO;
```

DEPTNO	AVG(SAL)
10	2916.66667
20	2175
30	1566.66667

CONTROLLING OUTPUT-**ORDER BY**

```
SQL>
SELECT JOB, COUNT(*), 12*AVG(SAL) a
FROM EMP
GROUP BY JOB
HAVING COUNT(*)>2
ORDER BY a desc;
```

SELECT CLAUSE AGGREGATE FUNCTIONS:HAVING

```
SQL> SELECT DEPTNO, AVG(SAL)
        FROM EMP
        GROUP BY DEPTNO
        HAVING AVG(SAL)>2000;
```

INSTEAD OF WHERE – HAVING:
condition as in WHERE

DEPTNO	AVG(SAL)
10	2916.66667
20	2175

SET MEMBERSHIP-EMBEDDING SQL QUERIES INTO ANOTHER SQL QUERY

Example:

Find employees with the smallest salary in each department:

```
SQL> select ename, sal, deptno
        from emp
        where (sal, deptno)
                in
                        (select min(sal), deptno
                         from emp
                         group by deptno);
```

ENAME	SAL	DEPTNO
SMITH	800	20
JAMES	950	30
MILLER	1300	10

SET COMPARISON

Example:

Find the department (by its number) in which the average salary is greater than the one in dept 30:

```
SQL> select deptno, avg(sal)
        from emp
        group by deptno
        having avg(sal) >(          select avg(sal)
                                    from emp
                                    where deptno=30);

DEPTNO     AVG(SAL)
 ---------          ----------
10              2916.66667
11               20 2175
```

USING SOME

Find the names of employees with greater salary than the smallest in department 20.

```
SQL> select ename, sal
            from emp
            where sal> SOME (select sal
                             from emp
                             where deptno=20);
```

```
 ENAME SAL
---------- ----------
ALLEN 1600
WARD 1250
JONES 2975
MARTIN 1250
BLAKE 2850
CLARK 2450
SCOTT 3000
KING 5000
TURNER 1500
```

USING ALL

Find the names of employees with greater salary than everyone's salary in department 20.

```
SQL> select ename, sal
        from emp
        where sal> ALL (select sal
                        from emp
                         where deptno=20);
```

ENAME	SAL
KING	5000

USING EXISTS

- EXISTS: IS there any row in the result of the inner selecion?)
- VALUES: -TRUE -FALSE
- Find the name of employees having at least one other employee to riport them.

```
SQL>select empno, ename, job
        from emp e
        where EXISTS ( select*
                        from emp
                        where mgr=e.empno);
```

QUERY LANGUAGE - MORE ON WHERE

```
SQL> SELECT   EMPNO, ENAME, JOB, SAL
     FROM     EMP
     WHERE    SAL BETWEEN 1000 AND 2000
     AND      JOB='CLERK';
```

RESULT:

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

QL: MORE ON WHERE

```
SQL>  SELECT    EMPNO, ENAME, JOB, SAL
      FROM      EMP
      WHERE     SAL BETWEEN 1000 AND 2000
      OR        JOB='CLERK';
```

RESULT:

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300

QL: MORE ON WHERE

FIND THE NAME, JOB, SALARY SALARY GRADE AND DEPTNO
FOR THOSE EMPLOYEES WHOSE JOB IS NOT CLERK
AND ORDER BY DESCENDING SALARY:

```
SELECT ENAME, JOB, SAL, GRADE, DNAME
FROM EMP, SAL, DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO
AND SAL BETWEEN LOSAL AND HISAL
AND JOB!='CLERK'
ORDER BY SAL DESC;
```

ENAME	JOB	SAL	GRADE	DNAME
KING	PRESIDENT	5000	5	ACCOUNTING
SCOTT	ANALYST	3000	4	RESEARCH
FORD	ANALYST	3000	4	RESEARCH
JONES	MANAGER	2975	4	RESEARCH
BLAKE	MANAGER	2850	4	SALES
CLARK	MANAGER	2450	4	ACCOUNTING
ALLEN	SALESMAN	1600	3	SALES
TURNER	SALESMAN	1500	3	SALES
WARD	SALESMAN	1250	2	SALES
MARTIN	SALESMAN	1250	2	SALES

QL: extended structure

SELECT attribute list with possible **aggregates:**

In this case attributes must have aggregates on it except the one listed in the GROUP BY CLAUSE

FROM table list

WHERE condition without aggregates

GROUP BY attribute list

HAVING condition with aggregate

//where must not used

ORDER BY attribute list desc;/increasing is the default

RELATIONAL ALGEBRA AND SQL: CROSS PRODUCT- **DO NOT USE!!!**

SELECT * FROM SAL, DEPT;

Result:

GRADE	LOSAL	HISAL	DEPTNO	DNAME	LOC
1	700	1200	40	OPERATIONS	BOSTON
2	1201	1400	40	OPERATIONS	BOSTON
3	1401	2000	40	OPERATIONS	BOSTON

RELATIONAL ALGEBRA AND SQL: NATURAL JOIN

```
SQL>
SELECT EMPNO, ENAME, EMP.DEPTNO, DEPT.DNAME
FROM EMP, DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO;
```

Result:

EMPNO	ENAME	DEPTNO	DNAME
7369	SMITH	20	RESEARCH
7499	ALLEN	30	SALES
7521	WARD	30	SALES
7566	JONES	20	RESEARCH
7654	MARTIN	30	SALES
7698	BLAKE	30	SALES
7782	CLARK	10	ACCOUNTING
7788	SCOTT	20	RESEARCH
7839	KING	10	ACCOUNTING
7844	TURNER	30	SALES
7876	ADAMS	20	RESEAR

USING ALIAS NAMES

```
SQL>
SELECT EMPNO, ENAME, E.DEPTNO,
                D.DNAME
FROM EMP E, DEPT D
WHERE E.DEPTNO=D.DEPTNO;
```

JOINING A TABLE TO ITSELF-ALIAS MUST USE

FIND THE EMPLOYEES AND THEIR MANAGERS:

```
SQL>
SELECT E.EMPNO, E.ENAME,M.ENAME  BOSS
FROM EMP E, EMP M
WHERE E.MGR=M.EMPNO;
```

EMPNO	ENAME	BOSS
7902	FORD	JONES
7788	SCOTT	JONES
7934	MILLER	CLARK
7876	ADAMS	SCOTT
7369	SMITH	FORD

OTHER EXAMPLE FOR ALIAS NAMES

- FIND THOSE EMPLOYEES WHO EARN MORE, THAN THEIR MANAGERS (**must use** aliases):
- **********

```
SQL>
SELECT E.ENAME, E.SAL
FROM EMP E, EMP M
WHERE E.MGR=M.EMPNO
AND E.SAL>M.SAL;
ENAME             SAL
---------- ----------
SCOTT             3000
FORD              3000
```

DDL-MORE ABOUT CREATE-CONSTRAINTS

```
CREATE TABLE jobhist
( empno       NUMBER(4) NOT NULL,
  startdate   DATE NOT NULL,
  enddate     DATE,
  job         VARCHAR2(9),
  sal         NUMBER(7,2),
  comm        NUMBER(7,2),
  deptno      NUMBER(2),
  chgdesc     VARCHAR2(80),

  CONSTRAINT  jobhist_pk          PRIMARY KEY (empno, startdate),
  CONSTRAINT  jobhist_ref_emp_fk FOREIGN KEY (empno)
  REFERENCES  emp(empno)          ON DELETE CASCADE,
  CONSTRAINT  jobhist_ref_dept_fk FOREIGN KEY (deptno)
  REFERENCES  dept (deptno)       ON DELETE SET NULL,

  CONSTRAINT jobhist_date_chk CHECK (startdate <= enddate)  );
```

DDL – a QUERY in TABLE DEFINITION

```
CREATE TABLE SALARY_GRADE
AS SELECT * FROM SAL ;
```

DDL – a QUERY in TABLE DEFINITION

```
CREATE TABLE DEPT30
AS
SELECT EMPNO, ENAME, JOB, SAL
FROM EMP
WHERE DEPTNO=30;

SQL> DESC DEPT30;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                     NOT NULL CHAR(10)
 JOB                                                CHAR(9)
 SAL                                                NUMBER(10,2)
```

QUERIES – TABLE CONTENT

SQL> SELECT* FROM DEPT30;

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7844	TURNER	SALESMAN	1500
7900	JAMES	CLERK	950

DDL – a **QUERY** in TABLE DEFINITION

```
SQL>
CREATE TABLE EMP_SALS(NAME,
   SALARY, GRADE)
AS
SELECT EMP.ENAME, EMP.SAL,
   SALARY_GRADES.GRADE
FROM EMP, SALARY_GRADES
WHERE EMP.SAL BETWEEN
   SALARY_GRADES.LOSAL AND
   SALARY_GRADES.HISAL
```

DDL – **a QUERY** in TABLE DEFINITION

SQL> **DESC** EMP_SALS;

Name	Null?	Type
NAME	NOT NULL	CHAR(10)
SALARY		NUMBER(10,2)
GRADE		NUMBER(1)

DDL:example for view creation (using aggregates and where)

```
SQL>
   CREATE VIEW
   dept_summary (dname, minsal, maxsal,
   avsal)
   AS
   SELECT dname, min(sal), max(sal), avg(sal)
   FROM emp e, dept d
   WHERE e.deptno=d.deptno group by dname;

View created.
```

DDL:example for view creation

Using select in table creation an instance is „inserted” also

SQL>SELECT * FROM dept_summary;

Result:

DNAME	MINSAL	MAXSAL	AVSAL
ACCOUNTING	1300	5000	2916.66667
RESEARCH	800	3000	2175
SALES	950	2850	1566.66667

DML - a QUERY in insertion

```
SQL> INSERT INTO
       EMP_SALS(NAME, SALARY, GRADE)
     (SELECT EMP.ENAME, EMP.SAL,
     SALARY_GRADES.GRADE
     FROM EMP, SALARY_GRADES
     WHERE EMP.SAL BETWEEN
     SALARY_GRADES.LOSAL AND
     SALARY_GRADES.HISAL);
```

RESULT OF PREVIOUS INSERTION

```
SELECT * FROM EMP_SALS;
NAME       SALARY     GRADE
---------- ---------- ----------
SMITH         800          1
ADAMS        1100          1
JAMES         950          1
WARD         1250          2
MARTIN       1250          2
MILLER       1300          2
ALLEN         1600          3
TURNER        1500          3
JONES         2975          4
BLAKE         2850          4
CLARK         2450          4
NAME       SALARY     GRADE
---------- ---------- ----------
SCOTT         3000          4
FORD          3000          4
KING          5000          5
14 rows selected.
```

NULL VALUES=VALUE IS NOT KNOWN

„...any relational database, the concept of NULL can be both confusing and hazardess to your employment. Not understanding how to work with NULLs can lead to incorrect responses to queries and poor business decisions being made. New author Michael Coles brings us four rules that can help you work with NULL values in your tables.”

- http://www.sqlservercentral.com/columnists/mcoles/fourrulesfornulls.asp

NULL VALUES=VALUE IS NOT KNOWN

VALUE IS NOT KNOWN – the cell is not filled for several reason:

- **It is really not known**
- **Not applicable**
- **Not public**
- **Etc.**

NULL values may be queried:

```
SQL> SELECT ENAME FROM EMP where COM
        IS NOT NULL;
```

TRUTH VALUE FOR NULL= UNKNOWN

NULL VALUES=UNKNOWN VALUES

Three-Valued NOT

P	NOT P
True	False
False	True
Unknown	Unknown

NULL VALUES=UNKNOWN VALUES

Three-Valued logic table

IF: T:=1, F:=0, UNKNOWN:=1/2

AND:= MIN(Truth-value(P), Truth-value(Q)),

OR:= MAX(Truth-value(P), Truth-value(Q)),

P	Q	P AND Q	P OR Q
T	F	F	T
T	T	T	T
F	F	F	F
F	T	F	T
UNKNOWN	T	UNKNOWN	T
UNKNOWN	F	F	UNKNOWN
T	UNKNOWN	UNKNOWN	T
F	UNKNOWN	F	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN

NULL VALUES=UNKNOWN VALUES

Comparison in where:

Arithmetical calculation can not be carry out with NULLS:

NULL*VALUE=NULL,

even if eg. 0*anything in math is 0, here NOT!