SQL 2

Handling of dates in Oracle

- Dates are stored differently from the SQL standard
  - standard uses two different types: date and time
  - Oracle uses one type: DATE
    - Stored in internal format contains date and time
    - Output is controlled by formatting
      - select to_char(sysdate,'dd-Mon-yyyy') from dual;
        - » 01-Aug-2012
      - select to_char(sysdate,'dd-Mon-yyyy hh:mi:ss PM') from dual;
        - » 01-Aug-2012 10:56:50 AM
  - Oracle 10G introduced TIMESTAMP datatype – finer granularity on secs
    - select cast(sysdate as TIMESTAMP) from dual;
      - 01-AUG-12 10.27.10.000000000 AM

Handling of dates in Oracle (cont'd)

- DATE data type should be
  - formatted with TO_CHAR when selecting for display
- String literal representing date must be
  - formatted with TO_DATE when comparing or inserting/updating
- Example:

```
select empno, empname, to_char(empbdate,'dd-Mon-yyyy')
from payroll.employee
where empbdate > to_date('01-Feb-1962','dd-Mon-yyyy')
order by empbdate
```

- In the where clause, why not convert empbdate on the left to CHAR?
  - where to_char(empbdate,'dd-Mon-yyyy') > '01-Feb-1962'

Oracle Group Functions

- Used to perform mathematical summaries such as:
  - counting the number of rows
  - finding the minimum and maximum values for some specified attribute
  - summing the values in a column, and
  - averaging the values in a specified column.
- All group functions
  - can be applied only to sets of values
  - return a single aggregated value, derived from a set of values
- NULL values
  - are ignored by the group functions, the only exception is the COUNT(*) function

COUNT Function

- Can use the COUNT function for any datatype
- COUNT(*) returns the number of rows in a table

```
SELECT COUNT(*)
FROM employee;
```

- COUNT(*expr*) returns the number of non-null rows.

```
SELECT COUNT(empno)
FROM employee;
```

```
SELECT COUNT(empcomm)
FROM employee;
```

MAX and MIN Functions

- Use MIN and MAX functions for any datatype.

```
SELECT MIN(empbdate), MAX(empbdate)
FROM employee;

SELECT MIN(empname), MAX(empname)
FROM employee;

SELECT MIN(empcomm), MAX(empcomm)
FROM employee;
```

SUM and AVG Functions

- Use the SUM and AVG functions for numeric data.

```
SELECT SUM(empmsal)
FROM employee;

SELECT SUM(empcomm)
FROM employee;


SELECT AVG(empmsal)
FROM employee;

SELECT AVG(empcomm)
FROM employee;
```

NVL Function

- The NVL function forces group functions to include null values using an assigned value (normally 0).

```
SELECT COUNT(NVL(empcomm, 0))
FROM employee;

SELECT AVG(NVL(empcomm, 0))
FROM employee;
```

NVL(x, y)

- y if x is NULL; otherwise x
- y can be any value

```
SELECT AVG(NVL(empcomm, 10))
FROM employee;
```

GROUP BY clause

- Is used to divide the rows/tuples in a table into groups
- Allows the aggregate functions to return summary information for each group
- All columns in the SELECT list that are not in the group functions must be included in the GROUP BY clause
- The GROUP BY column does not have to be in the SELECT clause

```
SELECT deptno, COUNT(*), MIN(empmsal), MAX(empmsal),
                SUM(empmsal), AVG(empmsal)
FROM employee
GROUP BY deptno
ORDER BY deptno;
```

GROUP BY clause

- All columns in the SELECT list that are not group functions ***must*** be in the GROUP BY clause.

SELECT deptno, AVG(empmsal)
FROM employee;

```
SQL> SELECT deptno, AVG(empmsal)
  2  FROM employee;
SELECT deptno, AVG(empmsal)
       *
ERROR at line 1:
ORA-00937: not a single-group group
function
```

SELECT d.deptno, d.deptname, AVG(e.empmsal)
FROM employee e, department d
WHERE e.deptno = d.deptno
GROUP BY d.deptno;

```
SQL> SELECT d.deptno, d.deptname, AVG(e.empmsal)
  2  FROM employee e, department d
  3  WHERE e.deptno = d.deptno
  4  GROUP BY d.deptno;
SELECT d.deptno, d.deptname, AVG(e.empmsal)
                 *
ERROR at line 1:
ORA-00979: not a GROUP BY expression
```

GROUP BY clause

- The GROUP BY column does not have to be in the SELECT list.

```
SELECT d.deptname, AVG(e.empmsal)
FROM employee e, department d
WHERE e.deptno = d.deptno
GROUP BY d.deptno, d.deptname;
```

- Grouping by more than one column

```
SELECT deptno, empjob, SUM(empmsal)
FROM employee
GROUP BY deptno, empjob
ORDER BY deptno;
```

Illegal queries using GROUP BY

- Who earns more than the average salary?

  SELECT empno, empmsal
  FROM employee
  WHERE empmsal > avg(empmsal);

```
SQL> SELECT empno, empmsal
  2  FROM employee
  3  WHERE empmsal > avg(empmsal);
WHERE empmsal > avg(empmsal)
              *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

- Cannot use the WHERE clause to restrict groups.

HAVING clause

- Is applied to the output of a GROUP BY operation to restrict the selected rows.
- Operates like a WHERE clause, however, the WHERE clause applies to columns and expressions for individual rows, while the HAVING clause is applied to the output of a GROUP BY operation.
- The DBMS evaluates the clauses in a SQL statement in the following order:
  - WHERE clause
  - GROUP BY clause
  - HAVING clause
- Therefore if we wish to restrict the results of a query based on the result of a GROUP BY clause we need to use a HAVING clause rather than the WHERE clause.

HAVING clause

- Use the HAVING clause to restrict groups
  - Rows are grouped.
  - The group function is applied.
  - Groups matching the HAVING clause are displayed.

```
SQL> SELECT deptno, max(empmsal)
       2  FROM   employee
       3  WHERE  max(empmsal)>2900
       4  GROUP BY deptno;

WHERE   max(empmsal)>2900
        *
ERROR at line 3:
  ORA-00934: group function
is not allowed here
```

```
SQL> SELECT deptno, max(empmsal)
       2  FROM   employee
       3  GROUP BY deptno
       4  HAVING max(empmsal)>2900;

    DEPTNO  MAX(EMPMSAL)
---------- -------------
        10          5000
        20          3000
```

HAVING clause

- Find the total monthly salary (i.e., the payroll) for each non-sales job classification with a payroll greater than $5000

```
SELECT      empjob, SUM(empmsal) AS "PAYROLL"
FROM        employee
WHERE       empjob NOT LIKE 'SALES%'
GROUP BY  empjob
HAVING  SUM(empmsal) > 5000
ORDER BY SUM(empmsal);
```

Nesting Functions

- Can nest aggregate functions

  – Display the average monthly salary per department

```
SELECT   deptno, avg(empmsal)
FROM     employee
GROUP BY deptno;
```

  – What is the greatest average monthly salary (of departments)?

```
SELECT   max(avg(empmsal))
FROM     employee
GROUP BY deptno;
```