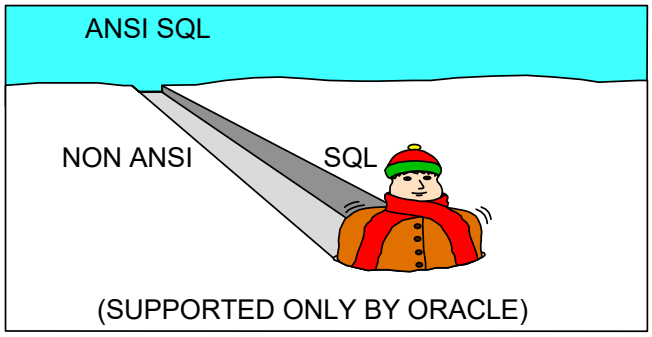
CHAPTER 3

STRUCTURED QUERY LANGUAGE EXTENSIONS



TOPIC OBJECTIVES

This section introduces the **NON-ANSI** standard SQL statements as used in Oracle.

When you finish this section, you will be able to:

- Understand how to expand a column heading's width for display purposes.
- Create queries which prompt users for input data to a query.
- Utilize non-standard built-in functions such as **INITCAP** and **INSTR**.
- Produce edited numeric columns.
- Understand what ***SMART DATES*** are, and how to use them in calculations.

ORACLE SQL PLUS COMMANDS

FORMATS

Oracle provides capabilities to change the width and format of datatypes:

- Column width
- Numeric
- Character
- Date

CHANGING THE COLUMN WIDTH

Example:

DALLAS Mayor DALLAS Mayor

'COLUMN colname FORMAT An' with 'n' being the desired column width.

```
COLUMN NAME FORMAT A13
SELECT DEPT, NAME FROM STAFF
```

NOTE: The only thing that may vary in a character column is the column width.

NUMBER FORMATS

SQL:

```
SET NUMFORMAT $99,999.99
select id,name,salary,comm
from staff where id = 10
```

RESULT:

ID	NAME	SALARY	COMM
10	SANDERS	$18,357.50	-

To reset **NUMFORMAT** back to the original settings type:

```
SET NUMFORMAT ""
```

NUMBER FORMATS

The following formats can be used for numeric data:

INITIAL FORMAT	DATA	RESULT	DESCRIPT
999.99	56.478	56.48	Rounds to 2 decimal places
9,999	8410	8,410	Comma separates thousands
09999	607	00607	Leading zeros fill format
9999	-5609	-5609	Minus sign normally precedes
9999MI	-5609	5609-	Minus sign follows number
9999PR	-5609	<5609>	Negative number in parenthesis
B999	0		Blank when zero
99.99	124.98	##.##	Value too large for format
$99.99	45.23	$45.23	Dollar sign displayed
$99.99PR	-45.23	<$45.23>	Combined formats
9.99EEEE	1200	1.20E+03	Exponential notation

You can set the numeric format for individual columns by using the following command:

```
     COLUMN SALARY FORMAT $99,999.99
     SELECT NAME,TO_CHAR(SALARY,'$999,999.99'),COMM
FROM STAFF
     WHERE ID = 10;
```

RESULT:

NAME	SALARY	COMM
PERNAL	$18,171.25	612.45

NUMERIC FUNCTIONS

- Trigonometric – sine, cosine, tangent, etc.
- Ceiling floor – largest integer not greater than, smallest integer not less than
- Absolute value – make a number positive
- Modulus – take the remainder of a division operation
- Power, root – raise a number to a power, find a root of a number
- Log, natural log = calculate a logarithm base 10, base e, or a specific base
- Rounding and truncation – round off to the nearest whole number, shortening a number by taking off one or more numbers after a decimal point

Function	Statement	Result
Abs(n)	select abs(-35.4) from dual	35.4
Ceil(n)	select ceil(1.1) from dual	2
Cos(n)	select cos(45) from dual	.525321989
Exp(n)	select exp(3) from dual	20.0855369
Log(g,n)	select log(3,2) from dual	.630929754
Mod(m,n)	select mod(8,3) from dual	.2
Power(m,n)	select power(2,4) from dual	16
Round(m,n)	select round(3.14159,3) from dual	3.142
Trunk(m,n)	select trunk(3.14159,3) from dual	3.141

CHARACTER

Functions are used to modify character columns.

The character functions are listed below:
INITCAP (char) - capitalizes the first character of the specified string.

```
SELECT ENAME,INITCAP(ENAME) FROM EMP;
```

INSTR(b) (char1,char2,n,m) - returns the position of the **Mth** occurrence of **char2** in **char1** beginning at position **n**.

Example : Find the employees that have the string **'AM'** beginning in position 2 of the **ENAME** column.

```
SELECT ENAME FROM EMP WHERE INSTR(ENAME,'AM')=2;
```

LENGTH (char) - Returns the length of a string.

Example : Find the length of all of the **ENAME** occurrences.

```
SELECT ENAME, LENGTH (ENAME) FROM EMP;
```

RTRIM or **LTRIM OR TRIM** (char) - **RTRIM** trims trailing spaces from a string.

Example : Using the Length function of **RTRIM**, find the actual number of characters each name takes in the **ENAME** column by eliminating trailing spaces from all of the **ENAME** occurrences.

```
SELECT ENAME, LENGTH(RTRIM(ENAME)) FROM EMP;
```

SUBSTR (char,n,m) - A substring of char beginning at position **n** positions **m** long.

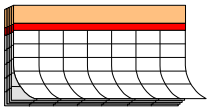
Example : Find the third character in the ENAME column for all employees in the EMP table.

```
SELECT SUBSTR(ENAME,3, 1) FROM EMP;
```

LOWER or **UPPER** (char) - Changes uppercase characters to lowercase.

```
SELECT * FROM EMP WHERE ENAME like UPPER('&NAME%');
```

DATE FORMATS



SQL:

```
SELECT TO_CHAR(MDATE,'MM-DD-YYYY') NEWDATE FROM ORDERS
```

RESULT:

NEW DATE
02-08-1998

YYYY or YYY or YY or Y	Produces 1998 or 998 or 98 or 8.
BC or AD	BC/AD indicator.
Q	Quarter of year. 1st or 2nd or 3rd or 4th.
MM	Month.
MONTH	Name of Month, padded with blanks to length of 9.

SQL:

```
SELECT EMPNO ID, ENAME,TO_CHAR(HIREDATE,'MONTH DD,YYYY')
HIREDATE FROM EMP WHERE empno = 7900
```

RESULT:

ID	E*NAME*	*HIREDATE*
7900	JAMES	FEBRUARY 08, 1998

FMMONTH	Name of Month variable length.
MON	Name of Month as 4-letter abbreviation.

SQL:

```
SELECT EMPNO ID,ENAME,TO_CHAR(HIREDATE,'MON DD,YYYY')
HIREDATE FROM EMP  WHERE empno = 7900
```

RESULT:

ID	*NAME*	*HIREDATE*
7900	JAMES	FEB. 08, 1998

WW or W	Week of Year or Month.
DDD or DD or D	Day of Year, Month or Week.
DAY	Name of Day, padded with blanks to a length of 9.
FMDAY	Name of day variable length.
DY	Name of Day, 4-letter abbreviation (Mon,Tue,etc.)

SQL:

```
SELECT EMPNO ID,ENAME,TO_CHAR(HIREDATE,'MON DD,DY,YYYY')
HIREDATE FROM EMP  WHERE EMPNO = 7900
```

RESULT:

ID	*NAME*	*HIREDATE*
7900	JAMES	FEB. 08, WED, 1998

J	Julian day; number of days since 12/31/4713 B.C.
AM or PM	Meridian indicator.
HH or HH12	Hour of day in 12 hour format.
HH24	Military clock time
/.,	Punctuation is printed as desired in format.

SQL:

```
SELECT EMPNO ID,ENAME,TO_CHAR(HIREDATE,'MM/DD/YY')
HIREDATE FROM EMP WHERE EMPNO = 7900
```

RESULT:

ID	*NAME*	*HIREDATE*
7900	JAMES	02/12/98

anything	Quoted string is printed in format.
codeTH	Suffix to make ordinal number (e.g. DDTH for 4TH).

NOTE: capitalization in spelled-out word or abbreviation follows format elements.

SQL:

```
SELECT EMPNO,ENAME,
TO_CHAR(HIREDATE,'DDth  "of" Month,YYYY') FROM EMP
```

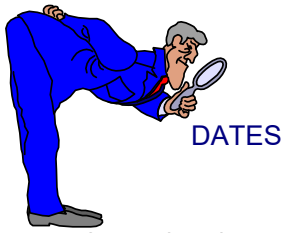
RESULT:

would yield 8th of February, 1998 for example.

Microsecond retrieval

```
SELECT EMPNO,ENAME,
       TO_CHAR(HIREDATE,'dd/Month/YYYY hh:mi:ss:ssss') FROM EMP
```

THE DATE FUNCTIONS



Allows for greater flexibility in using dates. Some of the functions refer to **date** and others to **date and time**.

ADD_MONTHS(d,n) - Date 'd' plus 'n' months.

SQL:

```
SELECT EMPNO, ADD_MONTHS(Hiredate, 6)
FROM EMP
```

RESULT:

EMPNO	ADD_MONTHS
10	08-Feb-98

LAST_DAY(d) - Returns the last day of the month specified.

SQL:

```
SELECT EMPNO, LAST_DAY (Hiredate)
FROM EMP
```

RESULT:

EMPNO	LAST_DAY
10	28-Feb-98

TRUNC(*d*) - Truncates the time in a date field.

SQL:

```
SELECT  EMPNO, HIREDATE
FROM EMP
WHERE TRUNC(HIREDATE) =
TRUNC(SYSDATE);
```

RESULT:

EMPNO	HIREDATE
10	02-JAN-98

SYNTAX OPERATORS & PARAMETER SUBSTITUTION

The **SYNTAX** operators are special characters used to define queries to Oracle.

'&n' - Ampersand operator allows parameter substitution.

***Example*:** `SELECT * FROM EMP WHERE EMPNO= &1 OR EMPNO= &2.`

'&var-name' - Substitution of value of variable. When variable is undefined, then SQL*PLUS prompts for value each execution.

***Example*:** `SELECT * FROM EMP WHERE EMPNO = &empno;`

'&&var-name' - Similar to '&var-name' except prompts only the first time in an execution for 'var-name'.

Parenthesis ( ) - Surround a sub-query or function.

***Example*:**
```
select substr(name,1,instr(name,' ')) FIRSTNAME
from PERS_PERSON_TBL;
```

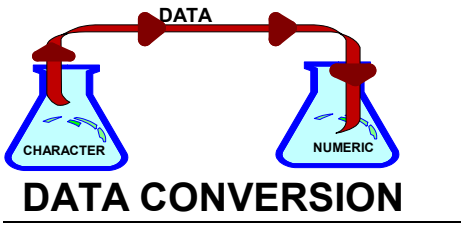
Single quote (') - Surrounds a character or date constant. Use two single quotes to represent one single quote.

Double quotes (") - Surrounds a column name or alias that contains special characters (such as spaces) or literal data in a date format.

At (@) sign -Used to define tables linked from a remote site through **SQL*NET**.

Example: `SELECT * FROM EMP@BOSTON.`

THE CONVERSION FUNCTIONS



The data conversion functions are used to change values from one format to another. A conversion may need to be done where inconsistent data types are not allowed, such as in arithmetic functions. These functions may also be used to convert between generic SQL formats and SQL*PLUS formats.

Descriptions of the conversion functions below:

TO_CHAR(n[,fmt]) - converts number **n** to a **CHAR** value in the format specified.

Example: `SELECT EMPNO,ENAME,TO_CHAR(YEARS) FROM EMP;`

NOTE: When **fmt** is omitted, **n** is converted to a **CHAR** value exactly long enough to hold the significant digits.

TO_CHAR(d[,fmt]) - converts date **d** to a **CHAR** value in the format specified.

Example:
```
SELECT EMPNO,ENAME,TO_CHAR(HIREDATE,'MM-DD-YY')
FROM EMP;
```

TO_DATE(char[,fmt]) - converts a date from a **CHAR** value of format **fmt** to a date value.

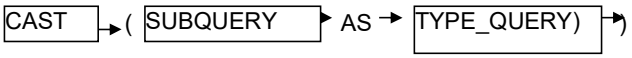
Example:
```
SELECT EMPNO,ENAME,
TO_DATE('FEB, 08,1995','MON,DD,YYYY')
FROM EMP;
```

TO_NUMBER(char) - converts a **CHAR** value containing a number to a **NUMBER** value.

Example: `SELECT ID,NAME,TO_NUMBER(SSN#) FROM EMP;`

THE CONVERSION FUNCTIONS(CONTINUED)

CAST EXAMPLES



The CAST operator converts one built-in datatype or collection-type into a different built-in datatype or collection-typed value.

Convert a character string into a date:

```
SQL>  Select CAST ('05-may-56' as DATE) from dual;

CAST ( '05-
----------
05-MAY-56
```

OTHER CONVERSION FUNCTIONS

ANALYTICAL FUNCTIONS

A NUMBER OF NEW ANALYTICAL FUNCTIONS WERE INTRODUCED IN ORACLE8I AND ENHANCED IN ORACLE9I TO SUPPORT DECISION SUPPORT SYSTEMS. THESE ARE COVERED IN ANOTHER COURSE BUT DISCUSSED HERE TO IDENTIFY THEIR CAPABILITIES.

ANALYTICAL FUNCTIONS
DESCRIPTION

RANKING RANK
CALCULATE PERCENTILES
DENSE_RANK N
TILES AND RANKS FOR
ROW_NUMBER A RESULT
SET
NTILE
RANK IN ORACLE SQL

RANK IS AN ANALYTICAL FUNCTION THAT CAN BE USED TO GET THE RANK OF A ROW IN RESPECT TO A GROUP OF ROWS. THIS LITTLE EXAMPLE WILL DEMONSTRATE THIS. FIRST YOU HAVE TO CREATE AND LOAD A TABLE THAT CONTAINS EACH MONTH'S AVERAGE TEMPERATURE IN PHILADELPHIA IN THE YEARS 1764-1820. THE SCRIPT TO DO THAT CAN BE FOUND AT PENNLOAD.SQL IN YOUR HOME DIRECTORY..
AFTER FILLING THIS TABLE, RANK CAN BE USED TO QUERY THE HOTTEST MONTH IN EACH YEAR:

```
SET FEEDBACK OFF
SET PAGES 50000

SELECT MONTH,YEAR,AVG_TEMP FROM
  (SELECT RANK() OVER (PARTITION BY YEAR ORDER BY
AVG_TEMP DESC) R, AVG_TEMP, MONTH, YEAR FROM
SCOTTISH_WEATHER)
WHERE R=1;
```

Formatted: PAGE-TITLE

Formatted: PAGE-TITLE

Formatted: PAGE-TITLE, Right: 0"

Formatted: PAGE-TITLE

```
        THIS RETURNS:
          MONTH      YEAR   AVG_TEMP
        --------- --------- ---------
                7      1764      59,9
                7      1765      58,5
                8      1766      59,5
                8      1767      59,8
                8      1768      58,7
                7      1769      60,1
                8      1770      58,2
                7      1771      57,4
                7      1772        58
                8      1773      58,3
                7      1774      56,8
                7      1775      59,7
                7      1776      59,6
                8      1777      59,2
                7      1778      61,2
                7      1779      65,2
                8      1780      63,2
                7      1781      60,4
                7      1782      60,1

        NOTE: TWO ROWS ARE RETURNED FOR THE YEAR 1787
BECAUSE THE HOTTEST AVERAGE TEMPERATURE ARE THE SAME
FOR JULY AND AUGUST.
        USING RANK TO SELECT A MONTH'S LAST RECORD
        CREATE TABLE TEST_MONTH (
          VAL      NUMBER,
          DT       DATE
        );

        ALTER SESSION SET NLS_DATE_FORMAT = 'DD.MM.YYYY';

        INSERT INTO TEST_MONTH (VAL,DT) VALUES (18,'28.08.2000');
        INSERT INTO TEST_MONTH (VAL,DT) VALUES (19,'02.08.2000');
        INSERT INTO TEST_MONTH (VAL,DT) VALUES (22,'27.09.2000');
        INSERT INTO TEST_MONTH (VAL,DT) VALUES (23,'04.09.2000');
        INSERT INTO TEST_MONTH (VAL,DT) VALUES (20,'12.08.2000');
        INSERT INTO TEST_MONTH (VAL,DT) VALUES (24,'15.09.2000');
```

```
INSERT INTO TEST_MONTH (VAL,DT) VALUES (19,'27.07.2000');
INSERT INTO TEST_MONTH (VAL,DT) VALUES (18,'01.07.2000');
INSERT INTO TEST_MONTH (VAL,DT) VALUES (21,'26.07.2000');
INSERT INTO TEST_MONTH (VAL,DT) VALUES (24,'03.06.2000');
INSERT INTO TEST_MONTH (VAL,DT) VALUES (22,'11.07.2000');
INSERT INTO TEST_MONTH (VAL,DT) VALUES (21,'14.06.2000');
SELECT VAL,DT FROM (SELECT
   VAL,DT,RANK() OVER(PARTITION BY TO_CHAR(DT,'YYYY.MM')
ORDER BY DT DESC) RN
FROM TEST_MONTH
)
WHERE RN = 1;
      VAL DT
--------- ---------
       21 14.06.2000
       19 27.07.2000
       18 28.08.2000
       22 27.09.2000
```

Formatted: PAGE-TITLE, Right: 0"

-THE SAMPLE CLAUSE

The SAMPLE clause allows a user to generate a set of results based upon a random sampling of a whole table. This feature can be used to data mine a large table.

The SAMPLE clause can only be used in certain types of queries:

- Simple SELECT queries (no joins)
- Simple CREATE TABLE AS SELECT queries

This feature can be useful for data mining tools or in a data warehousing environment where it can be used to avoid full table scans.

Perform a random row sampling:

```
SQL>  SELECT name, salary
        from staff  SAMPLE (15);
```

The keyword BLOCK is used to have ORACLE perform a random block sampling instead of row sampling.

```
SQL> SELECT name, salary
        from staff   SAMPLE BLOCK (15);
```

NOTE: Must be using the CBO (Cost-Based Optimizer.)

OTHER FUNCTIONS

DECODE(char,val,code,val,code,...default) - translates coded column values into the meanings of the code.

SQL:

```
SELECT EMPNO,ENAME,
DECODE(DEPTNO,10,'RESEARCH',20,'ACCT',30,'WAREHOUSE',
DEPTNO)
FROM EMP;
```

Using **DECODE** to order rows in a non-standard order:

SQL:

```
SELECT CODE,EMPNO,ENAME FROM EMP
ORDER BY DECODE(CODE,2,-3,3,-2,CODE);
```

NVL(x,expr) - displays 'expression' if **x** is null else **x** is displayed.

SQL:

```
SELECT EMPNO,ENAME,NVL(TO_CHAR(COMM),'NO VALUE')
FROM EMP;
```

Selecting today's date:

SQL:

```
select empno,ename,sysdate from emp;
```

TOP-N ANALYSIS

THIS REPORT WAS FASTER

TOP-N Analysis returns the largest and smallest result sets in a defined order. TOP-N queries can display a specified number of rows from a sorted order.

- Top 10 sales people by sales
- Top 5 stores grossing largest net revenue
- Top 3 customers

TOP-N query example:

```
sql>  SELECT ROWNUM AS TOPSALARY, NAME, SALARY, COMM
        FROM (SELECT NAME, SALARY, COMM FROM STAFF
                ORDER BY SALARY DESC)
        WHERE ROWNUM <= 10
```

Note: This is an in-line view.

LAB 3 - USING SQL*PLUS COMMANDS FOR REPORTING

(See LAB5.LST for answers)

1. Select all division managers from the **org** table. Change the department heading to **DEPARTMENT** in sql*plus.

2. Display all employees information who have a commission greater than a user-defined value. Use a temporary variable to prompt the user for the commission value.

3 Retrieve all information about employees who have more than 5 years of service. Display today's date along with their column information. Produce the same report but show today's date as 6 months from today.

4. Display all information about employees who are in **SALES**. If they have a null commission, print out **NO COMM** in place of the **NULL** value.

5. Retrieve the top ten employees by salary.