**More Single Row Functions**

**Please be aware that MS Access ‘Date’ outputs are formatted slightly differently ‘01/02/2020’ compared to Oracle ’01-FEB-20’**

**Aims of this Section**

In this section we extend our discussion of functions by covering DATE functions, conversion functions, and functions that accept any data as input.

**Date Functions**

Oracle stores dates in an internal numeric format, representing:

* Century
* Year
* Month
* Day
* Hours
* Minutes
* Seconds

The default display/input for any date is *DD-MON-YY*. Oracle dates can range between *1st Jan 4712 BC* and *31st Dec 4712 AD*.

SYSDATE is a pseudo-column that returns the current date and time. You can use SYSDATE just as you would use any other column name. For example, you can display the current date by selecting SYSDATE from a table. It is customary to select SYSDATE from a dummy table called DUAL. The DUAL table is owned by the SYSTEM and may be accessed by all users. It contains one column, DUMMY, and one row with a value 'X'. The DUAL table is useful when you want to return a value once only - for instance, the value of a constant, pseudo-column or expression that is not derived from a table with 'user' data.

To display the current date:

SELECT SYSDATE

FROM DUAL;

**MS Access:**

SELECT DATE() as Currentdate; (Only current Date)

or

SELECT NOW() as Currentdate; (Includes current Date and Time)

**Using Arithmetic Operators**

Owing to the fact that the DATE is stored as a number, it is possible to perform calculations with dates using arithmetic operators such as addition and subtraction. You can add and subtract number constants, as well as other dates, from dates.

The operations you may perform are :

date + number adds a number of days to a date, producing a date

date - number subtracts a number of days from a date, producing a date

date - date subtracts one date from another, producing a number of days

date + adds a number of hours to a date producing a date

number/24

SELECT HIREDATE, HIREDATE + 7, SYSDATE - HIREDATE

FROM EMP

WHERE HIREDATE LIKE '%JUN%' ;

**HIREDATE HIREDATE + 7 SYSDATE - HIREDATE**

**--------- ------------ ------------------**

**13-JUN-13 20-JUN-13 1982.70628**

**11-JUN-14 18-JUN-14 1618.70628**

**04-JUN-14 11-JUN-14 1625.70628**

**04-JUN-14 11-JUN-14 1625.70628**

(BE aware the system date is constantly changing so the printed output may not reflect your output)

Subtracting SYSDATE from the HIREDATE column of the EMP table returns the *number of days* since each employee was hired.

**MS Access**

The DateAdd () function adds a time/date interval to a date and then returns the date.

DateAdd (interval, number, date)

|  |  |
| --- | --- |
| Parameter | Description |
| interval | Required. The time/date interval to add. Can be one of the following values:  yyyy = Year  q = Quarter  m = month  y = Day of the year  d = Day  w = Weekday  ww = Week  h = hour  n = Minute  s = Second |
| number | Required. The number of interval to add to date. Can be positive (to get dates in the future) or negative (to get dates in the past) |
| date | Required. The date to which the interval should be added |

SELECT HIREDATE,

DATEADD("d",7, HIREDATE),

DATEDIFF(“d”, HIREDATE, DATE())

FROM EMP

WHERE HIREDATE LIKE '\*13\*' ;

The DATEDIFF () function returns the difference between two dates. Subtracting DATE () from the HIREDATE column of the EMP table returns the *number of days* since each employee was hired.

DateDiff (*datepart*, *date1*, *date2*, *firstdayofweek*, *firstweekofyear*)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *datepart* | Required. The part to return. Can be one of the following values:  yyyy = Year  q = Quarter  m = month  y = Day of the year  d = Day  w = Weekday  ww = Week  h = hour  n = Minute  s = Second |
| *date1* and *date2* | Required. The two dates to calculate the difference between |
| *firstdayofweek* | Optional. Specifies the first day of the week. Can be one of the following values:  0 = Use the NLS API setting  1 = Sunday (this is default)  2 = Monday  3 = Tuesday  4 = Wednesday  5 = Thursday  6 = Friday  7 = Saturday |
| *firstdayofyear* | Optional. Specifies the first week of the year. Can be one of the following values:   * 0 = Use the NLS API setting * 1 = Use the first week that includes Jan 1st (default) * 2 = Use the first week in the year that has at least 4 days * 3 = Use the first full week of the year |

MONTHS\_BETWEEN finds the number of months between *date1* and (date1, date2) *date2*.

The result can be positive or negative. If date1 is later than date2, the result is positive; if date1 is earlier than date2, the result is negative.

SELECT MONTHS\_BETWEEN(SYSDATE, HIREDATE)

FROM EMP

WHERE MONTHS\_BETWEEN(SYSDATE, HIREDATE) > 59 ;

**MONTHS\_BETWEEN(SYSDATE,HIREDATE)**

**--------------------------------**

**65.0873622**

**63**

**60.5067171**

**59.3454267**

**59.3454267**

**59.8292977**

For MS Access use the DATEDIFF again:

SELECT HIREDATE, DATEDIFF ("m", HIREDATE, DATE ())

FROM EMP

WHERE DATEDIFF ("m", HIREDATE, DATE ()) >59;

Note that in examples where one may be using dates other than column name or system date, single quotes are included, e. g. MONTHS\_BETWEEN( '01-JAN-88 ', '02-AUG-90 ').

ADD\_MONTHS(*date, n* ) adds *n* number of calendar months to date.

*n* must be an integer and can be negative.

SELECT HIREDATE, ADD\_MONTHS(HIREDATE, 3),

ADD\_MONTHS(HIREDATE, -3)

FROM EMP

WHERE DEPTNO = 20;

**HIREDATE ADD\_MONTHS(HIREDATE, 3) ADD\_MONTHS(HIREDATE, -3)**

**--------- ----------------------- ------------------------**

**13-JUN-13 13-SEP-13 13-MAR-13**

**31-OCT-13 31-JAN-14 31-JUL-13**

**05-MAR-14 05-JUN-14 05-DEC-13**

**04-JUN-14 04-SEP-14 04-MAR-14**

**05-DEC-13 05-MAR-14 05-SEP-13**

**MS Access**

Use DATEADD: SELECT HIREDATE, DATEADD ("m", 3, HIREDATE) as 'HireDate + 3', DATEADD ("m",-3, HIREDATE) as 'Hire date -3'

FROM EMP

WHERE DEPTNO = 20;

NEXT\_DAY( *date1, char* ) Date of the next day of the week (char) following *date1*. *Char* may be

a number representing a day, or character string.

(Not available in MS Access)

SELECT HIREDATE, NEXT\_DAY(HIREDATE, 6),

NEXT\_DAY(HIREDATE, 'FRIDAY' )

FROM EMP

WHERE DEPTNO = 10;

**HIREDATE NEXT\_DAY(HIREDATE,6) NEXT\_DAY(HIREDATE, 'FRIDAY')**

**--------- -------------------- ----------------------------**

**14-MAY-14 18-MAY-14 18-MAY-14**

**09-JUL-14 13-JUL-14 13-JUL-14**

**21-NOV-13 25-NOV-13 25-NOV-13**

LAST\_DAY(*date1* ) finds the date of the last day of the month that contains *date1.*

(Not available in MS Access)

SELECT HIREDATE, LAST\_DAY(HIREDATE), LAST\_DAY( '14-FEB-88 ')

FROM EMP

WHERE DEPTNO = 20 ;

**HIREDATE LAST\_DAY(HIREDATE) LAST\_DAY('14-FEB-88')**

**--------- ------------------ ---------------------**

**17-DEC-80 31-DEC-80 29-FEB-88**

**02-APR-81 30-APR-81 29-FEB-88**

**09-DEC-82 31-DEC-82 29-FEB-88**

**12-JAN-03 31-JAN-03 29-FEB-88**

**03-DEC-81 31-DEC-81 29-FEB-88**

**The Round Function can be applied to dates**

(Not available in MS Access)

ROUND( *date1* ) returns date1 with the time set to 12:00AM (midnight). Useful when comparing dates with different times

ROUND( *date1, ' MONTH'* ) returns the first of the month containing date1 *if date1 is in the first*

*half of the month* ; otherwise returns the first of the *following m*onth.

ROUND( *date1, 'YEAR '* ) returns the first day of the year containing date1 *if*

*date1 is in the first half of the year* ; otherwise returns

the first of the *following*  year

SELECT SYSDATE,

ROUND(SYSDATE, 'MONTH'),ROUND(SYSDATE, 'YEAR')

FROM DUAL;

**SYSDATE ROUND(SYSDATE,'MONTH') ROUND(SYSDATE,'YEAR')**

**--------- ---------------------- ---------------------**

**11-JUL-11 01-JUL-11 01-JAN-12**

TRUNC (*date1, ‘char’)* finds the date of the first day of the month contained in *date1* when *char* = ' MONTH'. If *char* = 'YEAR', it finds the date of the first day of the year containing *date1.*

SELECT SYSDATE,

TRUNC(SYSDATE, 'MONTH'), TRUNC(SYSDATE, 'YEAR')

FROM DUAL;

**SYSDATE TRUNC(SYSDATE,'MONTH') TRUNC(SYSDATE,'YEAR')**

**--------- ---------------------- --------------------**

**11-JUL-11 01-JUL-11 01-JAN-11**

**Conversion Functions**

SQL provides a number of functions to control data type conversions. These conversion functions convert a value from one data type into another.

TO\_CHAR(*number* | *date,* [' fmt']) converts number or date to character format *fmt*

TO\_NUMBER (char) converts *char* , which contains a number to a NUMBER

TO\_DATE (*'char', 'fmt'* ) converts the *char* value representingdate, into a date value according to *fmt* specified. If fmt is omitted format is DD-MON-YY.

**TO\_CHAR**

The TO\_CHAR function is frequently used to change a date format from the default to an alternative display format.

TO\_CHAR(date, 'date picture') specifies that date is to be converted to a new format on output

To convert the current date from the default format (DD-MON-YY) to a new 'date picture', enter:

SELECT TO\_CHAR(SYSDATE, 'DAY, DDTH MONTH YYYY' )

FROM DUAL;

**TO\_CHAR(SYSDATE,'DAY, DDTH MONTH YYYY')**

**----------------------------------------**

**TUESDAY , 05TH SEPTEMBER 1989**

Note that the 'date picture' must be embedded within single quotes, can include any of the formats listed below, and the column and 'date picture' must be separated by a comma. Furthermore, DAY and MONTH in the output are automatically padded with blanks to a length of 9.

To remove the blank padding use the FM (Fill Mode) prefix:

SELECT TO\_CHAR(SYSDATE, 'fmDAY, ddTH MONTH YYYY' )

FROM DUAL;

**TO\_CHAR(SYSDATE,'fmDAY, DDTH MONTH YYYY')**

**-------------------------------------------**

**TUESDAY, 5TH SEPTEMBER 1989**

Note also that FM can be used to suppress leading zeros for the *ddth* format, as above where 05TH is changed to 5th. The case in which the 'date picture' is entered is the case in which it will be displayed.

TO\_CHAR can also be used to extract the time of day only, and display it in a specified format:

SELECT TO\_CHAR(SYSDATE, 'HH: MI: SS' )

FROM DUAL ;

**TO\_CHAR(SYSDATE,'HH:MI:SS)**

**--------------------------**

**08:16:24**

**Date Formats**

The elements of a date format model are:

**Picture Meaning**

SCC or CC Century; 'S' prefixes "BC" dates with ' - '

YYYY or SYYYY Year; 'S' prefixes "BC" dates with ' - '

YYY or YY or Y Last 3, 2, or 1 digit(s) of year

Y, YYY Year with comma in this position

SYEAR or YEAR Year, spelled out: 'S' prefixes "BC" dates with ' - '

BC or AD BC / AD indicator

B.C. or A. D. BC / AD indicator with periods

Q Quarter of year

MM Month

MONTH Name of month, padded with blanks to length of 9 characters

MON Name of month, 3 letter abbreviation

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 length of 9 characters

DY Name of day, 3 letter abbreviation

J Julian day; the number of days since December 31, 4713 BC

AM or PM Meridian indicator

A. M. or P. M. Meridian indicator with periods

HH or HH12 Hour of day (1 - 12)

HH24 Hour of day (0 - 23)

MI Minute

SS or SSSS Seconds or seconds past midnight (0-86399) respectively

" ... " Quoted string is reproduced in the result

**The prefix below may be added to the codes above:**

fm 'Fill Mode'. Prefixed to MONTH or DAY, suppresses blank padding, leaving a variable-length result, FM will suppress leading zeros for the ddth format. Not meaningful with other codes. A second occurrence of FM turns blanks on again

**The suffixes below may be added to the codes above:**

TH Ordinal number (e. g. " DDTH" for " 4TH")

SP Spelled out number (e. g. "DDSP" for " FOUR")

SPTH or thsp Spelled out ordinal numbers (e.g. " DDSPTH" for "FOURTH")

NB : The codes are case sensitive and will affect display of date elements :

DAY MONDAY

Day Monday

Month October

ddth 14th

ddTh 14Th

**MS Access use SELECT FORMAT (DATE (), "Long Date", 2);**

Returns: 02 September 2020

The Format () function formats a date value with the specified format.

Format (*value*, *format*, *firstdayofweek*, *firstweekofyear*)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *value* | Required. The date value to format |
| *format* | Optional. The format to use.   |  |  | | --- | --- | | **Format** | **Description** | | General Date | Shows *value* as a date based on the system settings | | Long Date | Shows *value* as a date based on the system's long date settings | | Medium Date | Shows *value* as a date based on the system's medium date settings | | Short Date | Shows *value* as a date based on the system's short date settings | | Long Time | Shows *value* as a date based on the system's long time settings | | Medium Time | Shows *value* as a date based on the system's medium time settings | | Short Time | Shows *value* as a date based on the system's short time settings | |
| firstdayofweek | Optional. Specifies the first day of the week. Can be one of the following values:  0 = Use the NLS API setting  1 = Sunday (this is default)  2 = Monday  3 = Tuesday  4 = Wednesday  5 = Thursday  6 = Friday  7 = Saturday |
| firstdayofyear | Optional. Specifies the first week of the year. Can be one of the following values:  0 = Use the NLS API setting  1 = Use the first week that includes Jan 1st (default)  2 = Use the first week in the year that has at least 4 days  3 = Use the first full week of the year |

**TO\_NUMBER**

In the following example, the TO\_NUMBER function is used to transform a number stored as a character to number datatype:

SELECT EMPNO, ENAME, JOB, SAL

FROM EMP

WHERE SAL > TO\_NUMBER(' 1500');

**EMPNO ENAME JOB SAL**

**----- ----- --------- ----**

**7499 ALLEN SALESMAN 1600**

**7566 JONES MANAGER 2975**

**7698 BLAKE MANAGER 2850**

**7782 CLARK MANAGER 2450**

**7788 SCOTT ANALYST 3000**

**7839 KING PRESIDENT 5000**

**7902 FORD ANALYST 3000**

**MS Access**

Use the VAL () function which reads a String and returns the numbers found in the String

**TO\_DATE**

To show all employees hired on June 4, 2004 (non-default format), we can use the TO\_DATE function:

SELECT EMPNO, ENAME, HIREDATE

FROM EMP

WHERE HIREDATE=TO\_DATE('June 4,2014','Month dd,yyyy');

**EMPNO ENAME HIREDATE**

**----- ------ ---------**

**7844 TURNER 04-JUN-14**

**7876 ADAMS 04-JUN-14**

The constant is converted to a date and then compared to the HIREDATE value.

**MS Access**: To convert a String to a date use the DATEVALUE () function. DATEVALUE (string\_date).

Example: SELECT DATEVALUE (“September 02, 2020”); Returns 9/02/2020

**Functions That Accept Any Data Type As Input**

**DECODE**

DECODE is perhaps the most powerful SQL function. It facilitates conditional enquiries by doing the work of a 'case' or 'if-then-else' command.

DECODE (col / expression, search1, result1, [search2, result2, ...,]

default)

*Col / expression* is compared to each *search* value and returns the *result* if *col/expression* equals the search value. If no match is made, the DECODE function returns the *default* value.

**Arguments**

DECODE must have a minimum of 4 parameters or arguments

* + - *COL / EXPRESSION* - the column name or expression to be evaluated
    - *SEARCH1* - the first value to be tested
    - *RESULT1* - the value to be returned if search1 is matched
    - *SEARCH1* and *RESULT1* may be repeated as many times as is necessary - [ *SEARCH2, RESULT2,* etc.,]
    - *DEFAULT* - the value to be returned if no match is made

Note:

* + - *col / expression may be any data type*
    - *the value returned is forced to the same data type as the third argument (result1 )*

The following example decodes the job types MANAGER and CLERK only; no others are tested for they become the default of UNDEFINED.

SELECT ENAME, JOB, DECODE(JOB, 'CLERK', 'WORKER', 'MANAGER', 'BOSS',

'UNDEFINED') DECODED\_JOB

FROM EMP;

To display bonus percentages dependent upon salary grades, enter:

SELECT GRADE,

DECODE (GRADE, '1','15%','2','10%','3','8%','5%') BONUS

FROM SALGRADE;

**GRADE BONUS**

**----- -----**

**1 15%**

**2 10%**

**3 8%**

**4 5%**

**5 5%**

**MS Access** use the SWITCH () function

The MS Access SWITCH() function is comparable to the Oracle DECODE.

It evaluates a list of expressions and returns the corresponding value for the first expression in the list that is TRUE.

**Switch(** ***expr-1, value-1***[***, expr-2, value-2*** ] … [***, expr-n, value-n*** ] **)**

The **Switch** function syntax has these parameters or arguments:

|  |  |
| --- | --- |
| **Argument** | **Description** |
| ***expr*** | Required. Variant expression you want to evaluate.  The Switch is looking for the first expression to be **True** |
| ***value*** | Required. Value or expression to be returned if the corresponding expression is **True** |

The Switch returns any datatype such as String, Numeric, Date etc

MS Access evaluates all of the expressions passed to the Switch, so if any generate an error, the function will generate an error.

SELECT GRADE,

SWITCH (

GRADE = 1,'15%',

GRADE = 2, '10%',

GRADE = 3, '8%',

TRUE, '5%’) as BONUS

FROM SALGRADE;

NVL (*col* |*value, val*) converts a NULL value to *val*. Types must match

SELECT SAL\*12 + NVL (COMM, 0), NVL (COMM, 1000),

SAL\*12 + NVL (COMM, 1000)

FROM EMP

WHERE DEPTNO = 10;

**SAL\*12+NVL(COMM,0) NVL(COMM,1000) SAL\*12+NVL(COMM,1000)**

**------------------ -------------- ---------------------**

**29400 1000 30400**

**60000 1000 61000**

**15600 1000 16600**

**MS Access** use Nz () function:

SELECT SAL\*12 + Nz (COMM,0), Nz(COMM,1000), SAL\*12 + Nz (COMM,1000)

FROM EMP

WHERE DEPTNO = 10;

(Not available in MS Access)

GREATEST (col | value1, col | value2 ...) returns the greatest of the list of values. All *col* |

*value2* 's are converted to the *col* |*value1* type before comparison

SELECT GREATEST(1000, 2000), GREATEST(SAL, COMM)

FROM EMP

WHERE DEPTNO = 30;

**GREATEST(1000,2000) GREATEST(SAL,COMM)**

**------------------- ------------------**

**2000 1600**

**2000 1250**

**2000 1400**

**2000**

**2000 1500**

**2000**

(Not available in MS Access)

LEAST(col | value1, col | value2, ...) returns the least of the list of values. All *col* |*value2* 's are converted to the *col* |*value1* type before comparison

SELECT LEAST (1000, 2000), LEAST (SAL, COMM)

FROM EMP

WHERE DEPTNO = 30;

**LEAST(1000,2000) LEAST(SAL, COMM)**

**---------------- ----------------**

**1000 300**

**1000 500**

**1000 1250**

**1000**

**1000 0**

**1000**

(Not available in MS Access)

VSIZE (*col* |*value*) returns the number of bytes in ORACLE' s internal representation of *col* |*value*

SELECT DEPTNO, VSIZE(DEPTNO),VSIZE(HIREDATE), VSIZE(SAL),

VSIZE(ENAME)

FROM EMP

WHERE DEPTNO = 10;

**DEPTNO VSIZE(DEPTNO) VSIZE(HIREDATE) VSIZE(SAL) VSIZE(ENAME)**

**------ ------------ --------------- ---------- -----------**

**10 2 7 3 5**

**10 2 7 2 4**

**10 2 7 2 6**

**Nested Functions Revisited**

We have already seen that functions can be nested to any level, and that inner functions within the nesting are evaluated first, working to the outer function last.

Remember to keep track of the number of open and closed parentheses. Here are some more examples of nested functions:

SELECT ENAME, NVL(TO\_CHAR(MGR), ' UNMANAGEABLE’)

FROM EMP

WHERE MGR IS NULL;

**MS Access**

SELECT ENAME, Nz (MGR, 'UNMANAGEABLE')

FROM EMP

WHERE MGR IS NULL;

**ENAME NVL(TO\_CHAR(MGR),'UNMANAGEABLE')**

**----- ----------------------------------**

**KING UNMANAGEABLE**

1. MGR column is first converted to a character with the TO\_CHAR function

2. NVL function replaces a NULL mgr with the character string UNMANAGEABLE.

Display the date of the Friday that is two months from today in the format Day dd Month YYYY.

SELECT SYSDATE,

TO\_CHAR(NEXT\_DAY(ADD\_MONTHS(SYSDATE, 2), 'FRIDAY' ) ,'Day dd Month YYYY’)

FROM DUAL;

**SYSDATE TO\_CHAR(NEXT\_DAY(ADD\_MONTHS(SYSDATE,2),'FRIDAY'),**

**'Day dd Month YYYY')**

**--------- -------------------------------------------------**

**11-JUL-11 Friday 16 September 2011**

1. The ADD\_MONTHS function adds two months to the current date (Dec).

2. The NEXT\_DAY function finds the Friday two months from SYSDATE.

3. The TO\_CHAR function converts the date column to CHAR datatype in order to display a non-default date format - Day dd Month YYYY.

**Conversion and Date Function Tutorial**

**Please be aware that MS Access ‘Date’ outputs are formatted slightly differently ‘01/02/2020’ compared to Oracle ’01-FEB-20’**

This exercise covers the single row functions discussed in this section, as well as revising some of the functions in the previous sections.

**Workshop**

1. Display name and total renumeration for all employees.

**RENUMERATION**

**- - - - - - - - - - - - - -**

**9600**

**19500**

**15500**

**35700**

**16400**

**34200**

**29400**

**36000**

**60000**

**18000**

**13200**

**11400**

**36000**

**15600**

**ENAME**

**- - - - - - -**

**SMITH**

**ALLEN**

**WARD**

**JONES**

**MARTIN**

**BLAKE**

**CLARK**

**SCOTT**

**KING**

**TURNER**

**ADAMS**

**JAMES**

**FORD**

**MILLER**

**(14 rows selected)**

2. ORACLE USERS: Display each employees name and hiredate from dept. 20.

**ENAME DATE\_HIRED**

**------------ ----------------------**

**SMITH June, Thirteenth 2013**

**JONES October, Thirty-First 2013**

**SCOTT March, Fifth 2014**

**ADAMS June, Fourth 2014**

**FORD December, Fifth 2013**

MS ACCESS USERS: Display each employees name and hiredate from dept. 20.

**ENAME DATE\_HIRED**

**------------ ----------------------**

**SMITH 13 June 2013**

**JONES 31 October 2013**

**SCOTT 05 March 2014**

**ADAMS 04 June 2014**

**FORD 05 December 2013**

3. Display each employee name with hiredate, and salary review date. Assume review date is one year after hiredate. Order the output in ascending review date order.

**ENAME HIREDATE REVIEW**

**------- --------- ---------**

**SMITH 13-JUN-13 13-JUN-14**

**ALLEN 15-AUG-13 15-AUG-14**

**JONES 31-OCT-13 31-OCT-14**

**MILLER 21-NOV-13 21-NOV-14**

**MARTIN 05-DEC-13 05-DEC-14**

**FORD 05-DEC-13 05-DEC-14**

**SCOTT 05-MAR-14 05-MAR-15**

**WARD 26-MAR-14 26-MAR-15**

**CLARK 14-MAY-14 14-MAY-15**

**TURNER 04-JUN-14 04-JUN-15**

**ADAMS 04-JUN-14 04-JUN-15**

**BLAKE 11-JUN-14 11-JUN-15**

**KING 09-JUL-14 09-JUL-15**

**JAMES 23-JUL-14 23-JUL-15**

4. (CHALLENGE) Print a list of employees displaying just salary if more than 1500. If exactly 1500 display 'On target', if less than 1500 display 'Below 1500'.

Hint: Oracle User will need DECODE, SIGN and ORDER BY

Hint: Access Users will need SWITCH, SGN and ORDER BY

**ENAME SALARY**

**------- ----------**

**ADAMS Below 1500**

**ALLEN 1600**

**BLAKE 2850**

**CLARK 2450**

**FORD 3000**

**JAMES Below 1500**

**JONES 2975**

**KING 5000**

**MARTIN Below 1500**

**MILLER Below 1500**

**SCOTT 3000**

**SMITH Below 1500**

**TURNER On target**

**WARD Below 1500**