

SQL Functions

Numeric Functions

Date Functions

TO_CHAR function

Conversion Functions

Numeric Functions

- ROUND
- DECIMAL
- TRUNC
- MOD
- CEILING
- FLOOR
- ABS

ROUND & DECIMAL Functions

- The **ROUND** function rounds a numeric value to a specified number of decimals
- The **DECIMAL** function returns a decimal representation of a number

ROUND Function

- Used with both numbers and dates
- Mainly used to round numbers to a specified number of decimal places
- Can be used to round numbers to the left of the decimal point
- If the number of decimal places is not specified or is zero, the number will round to no decimal places

```
ROUND(column|expression, decimal places)
```

```
VALUES ROUND(45.956);      -- Returns 46.000
```

```
VALUES ROUND(45.956, 0);   -- Returns 46.000
```

ROUND Function

- If the number of decimal places is a **positive** number, the number is rounded to that number of decimal places

```
VALUES ROUND(45.456, 0); -- Returns 45.000
VALUES ROUND(45.956, 0); -- Returns 46.000
VALUES ROUND(45.446, 1); -- Returns 45.400
VALUES ROUND(45.456, 1); -- Returns 45.500
VALUES ROUND(45.956, 1); -- Returns 46.000
VALUES ROUND(45.444, 2); -- Returns 45.440
VALUES ROUND(45.446, 2); -- Returns 45.450
```

ROUND Function

- If the number of decimal places is a **negative** number, numbers to the left of the decimal are rounded

```
VALUES ROUND(45.456, 0); -- Returns 45.000
VALUES ROUND(45.956, 0); -- Returns 46.000
VALUES ROUND(44.456, -1); -- Returns 40.000
VALUES ROUND(45.456, -1); -- Returns 50.000
VALUES ROUND(45.956, -1); -- Returns 50.000
VALUES ROUND(45.456, -2); -- Returns 00.000
VALUES ROUND(45.956, -2); -- Returns 00.000
VALUES ROUND(55.456, -2); -- Returns 100.000
```

DECIMAL Function

- DECIMAL(numeric_value), precision, scale)
 - Precision –number of digits in the result number
 - Scale – number of positions after the decimal point

```
VALUES DECIMAL(45.456, 5, 0); -- Returns 45
VALUES DECIMAL(45.556, 5, 0); -- Returns 45
VALUES DECIMAL(45.456, 5, 1); -- Returns 45.4
VALUES DECIMAL(45.556, 5, 1); -- Returns 45.5
VALUES DECIMAL(45.446, 5, 2); -- Returns 45.44
VALUES DECIMAL(45.456, 5, 2); -- Returns 45.45
VALUES DECIMAL(45.4567, 5, 3); -- Returns 45.456
```

ROUND & DECIMAL Functions

- DECIMAL(numeric_value), precision, scale)

```
VALUES ROUND(45.456 * 1.8654);      -- Returns 85.0000000
VALUES ROUND(45.456 * 1.8654, 2);    -- Returns 84.7900000

VALUES DECIMAL( ROUND(45.456 * 1.8654, 2), 5, 2);  -- Returns 84.79
```


Example 10-21
ROUND/DECIMAL
Functions

Retrieve customer_name and discount. Calculate a 2.575% increase on the current discount. Calculate the new discount by multiplying discount by 1.2575. Round the results to three decimal places and convert the results to a decimal number with three decimal places.

Data Set ds9_21

ID	CUSTOMER_NAME	DISCOUNT
101	Bargains Galore	0.070
102	RedHot Discount	0.100
103	NewTown Deals	0.035
104	Ajax Sales	0.100
105	BigBox Direct	0.055

SQL Statement

```
SELECT customer_name AS "Customer",
       discount AS "Old Discount",
       discount * 1.2575 AS "New Discount",
       DECIMAL (ROUND(discount * 1.2575, 3), 3,3) AS "New Discount2"
FROM ds9_21;
```

Result Set

Customer	Old Discount	New Discount	New Discount2
Bargains Galore	0.070	0.0880250	0.088
RedHot Discount	0.100	0.1257500	0.126
NewTown Deals	0.035	0.0440125	0.044
Ajax Sales	0.100	0.1257500	0.126
BigBox Direct	0.055	0.0691625	0.069

TRUNCATE or TRUNC Function

- Truncate a numeric value to a specific position
- Can be used with both numbers and dates
- If the number of decimal places is not specified or is specified as zero, the specified number defaults to zero
- Does not round the number. It simply terminates the number at a given point

`TRUNC (column|expression, decimal_places)`

VALUES	TRUNCATE (29.45);	-- Returns	29.00
VALUES	TRUNCATE (29.95);	-- Returns	29.00
VALUES	TRUNCATE (29.45, 0);	-- Returns	29.00
VALUES	TRUNCATE (29.95, 0);	-- Returns	29.00
VALUES	TRUNCATE (29.45, 1);	-- Returns	29.40
VALUES	TRUNCATE (29.95, 1);	-- Returns	29.90
VALUES	TRUNCATE (29.45, -1);	-- Returns	20.00
VALUES	TRUNCATE (29.95, -1);	-- Returns	20.00

MOD Function

MOD (dividend, divider)

- The remainder of one value divided by another value
- For example, the MOD of 5 divided by 2 = 1
- VALUES MOD(1600, 500);

MOD(1600,500)
100

MOD Function

- Can be used to determine whether a value is odd or even
- If a value is divided by 2 and there is no remainder, the number must be an even number
- Next slide

Example 10-22: MOD function

```
SELECT
    MOD(100, 2)  AS "Even Number",
    MOD(101, 2)  AS "Odd Number"
FROM SYSIBM.SYSDUMMY1;
```

Results

Even Number	Odd Number
0	1

MOD Function

- Can be used to convert ounces to pounds
- Next slide

Example 10-26: MOD function

Convert 235 ounces to pounds

```
SELECT TRUNCATE(235/16, 0) AS LBS, MOD(235, 16) AS OZ  
FROM SYSIBM.SYSDUMMY1;
```

Results

LBS	OZ
---	--
14	11

Ceiling Function

- Returns the **smallest** integer value **greater than** or **equal** to a *number* or *numeric-expression*

Example 10-27: The CEILING function

```
SELECT CEILING( 22.45)    AS "CEIL1",  
       CEILING( 22.5 )   AS "CEIL2",  
       CEILING( 22.972 ) AS "CEIL3",  
       CEILING( -22.45 ) AS "CEIL4",  
       CEILING( -22.5 )  AS "CEIL5",  
       CEILING( -22.972 ) AS "CEIL6"  
FROM SYSIBM.SYSDUMMY1;
```

Results

CEIL1	CEIL2	CEIL3	CEIL4	CEIL5	CEIL6
-----	-----	-----	-----	-----	-----
23	23	23	-22	-22	-22

FLOOR Function

- Returns the **largest** integer value **less than** or **equal** to a *number* or *numeric-expression*

Example 10-28: The FLOOR function

```
SELECT FLOOR( 22.45 ) AS "FLOOR1",  
       FLOOR( 22.5 ) AS "FLOOR2",  
       FLOOR( 22.972 ) AS "FLOOR3",  
       FLOOR( -22.45 ) AS "FLOOR4",  
       FLOOR( -22.5 ) AS "FLOOR5",  
       FLOOR( -22.972 ) AS "FLOOR6"  
FROM SYSIBM.SYSDUMMY1;
```

Results

FLOOR1	FLOOR2	FLOOR3	FLOOR4	FLOOR5	FLOOR6
-----	-----	-----	-----	-----	-----
22	22	22	-23	-23	-23

ABS Function

- Returns the absolute, or positive value of the numeric values supplied by the argument

Example 10-29: The ABS function

```
SELECT ABS(-24)      AS ABS1,  
       ABS(-24.45)   AS ABS2,  
       ABS(-24.95)   AS ABS3  
FROM SYSIBM.SYSDUMMY1;
```

Results

ABS1	ABS2	ABS3
24	24.45	24.95

Date Functions

Date Format

- The default format for dates is *ISO
 - YYYY-MM-DD -- that is, 2021-01-15
- Dates are stored internally with a numeric format representing century, year, month, day, hour, minute, and second

Format name	Abbreviation	Format	Example
International Standards Organization (*ISO)	ISO	yyyy-mm-dd	'2019-04-07'
IBM USA Standard (*USA)	USA	mm/dd/yyyy	'04/07/2019'
IBM European Standard (*EUR)	EUR	dd.mm.yyyy	'07.04.2019'
Japanese Industrial Standard Christian era (*JIS)	JIS	yyyy-mm-dd	'2019-04-07'

CURRENT_DATE

- CURRENT_DATE is a date function that returns the current date
- CURRENT DATE without underscore will work with DB2

```
SELECT CURRENT_DATE  
FROM SYSIBM.SYSDUMMY1;
```

```
VALUES CURRENT_DATE;
```

```
00001
```

```
-----
```

```
2019-11-05
```

Working with Dates - DAYS

- Add 60 days to hire_date

```
SELECT last_name, hire_date,  
       hire_date + 60 DAYS AS first_review  
FROM employees  
ORDER BY first_review;
```

LAST_NAME	HIRE_DATE	FIRST_REVIEW
King	1987-06-17	1987-08-16
Whalen	1987-09-17	1987-11-16
Kochhar	1989-09-21	1989-11-20
Hunold	1990-01-03	1990-03-04
Ernst	1991-05-21	1991-07-20
De Haan	1993-01-13	1993-03-14
Higgins	1994-06-07	1994-08-06
Gietz	1994-06-07	1994-08-06

Working with Dates - DAYS

- Number of days since employee was hired

```
SELECT last_name, hire_date,  
       DAYS(CURRENT_DATE) - DAYS(hire_date) AS tenure  
FROM employees  
ORDER BY tenure DESC;
```

LAST_NAME	HIRE_DATE	TENURE
King	1987-06-17	11829
Whalen	1987-09-17	11737
Kochhar	1989-09-21	11002
Hunold	1990-01-03	10898
Ernst	1991-05-21	10395
De Haan	1993-01-13	9792
Higgins	1994-06-07	9282

Example 10-30

Calculate the number of days it took to make shipments by subtracting the order date from the ship date. Do not include shipments that have not been shipped; that is, the ship date is NULL.

Data Set ds9_30

ID	ORDER_DATE	SHIP_DATE	
--	-----	-----	
1	2020-01-16	2020-01-30	
2	2020-02-05	2020-03-22	
3	2020-03-20	NULL	
4	2020-03-22	2019-12-28	-- ship_date entered incorrectly

SQL Statement

```
SELECT id,  
       order_date,  
       ship_date,  
       DAYS( ship_date ) - DAYS( order_date ) AS "Days To Ship"  
FROM   ds9_30  
WHERE  ship_date IS NOT NULL;
```

Result Set

ID	ORDER_DATE	SHIP_DATE	Days to Ship
--	-----	-----	-----
1	2020-01-16	2020-01-30	14
2	2020-02-05	2020-03-22	46
4	2020-03-22	2019-12-28	-85

Example 10-31 Use the DAYS, MONTHS, and YEAR functions to calculate future dates from hire_date.

Data Set ds9_31

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name AS name,  
       hire_date + 14 DAYS AS start_date,  
       hire_date + 3 MONTHS AS first_review,  
       hire_date + 1 YEAR AS yearly_review,  
       hire_date + 1 YEAR + 2 MONTHS + 14 DAYS AS raise_date  
FROM   ds9_31;
```

Result Set

NAME	START_DATE	FIRST_REVIEW	YEARLY_REVIEW	RAISE_DATE
Lopez	2000-06-13	2000-08-30	2001-05-30	2001-08-13
Brown	2001-03-06	2001-05-20	2002-02-20	2002-05-04
Patel	2001-06-29	2001-09-15	2002-06-15	2002-08-29
Moore	2002-08-31	2002-11-17	2003-08-17	2003-10-31
Fang	2002-09-29	2002-12-15	2003-09-15	2003-11-29

Working with Dates

- Calculate number of weeks of service

Example 10-32

Calculated the number of weeks of employment. Assume CURRENT_DATE is 2019-06-15

Data Set ds9_32

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name AS name,  
       hire_date,  
       ( DAYS(CURRENT_DATE) - DAYS(hire_date) ) / 7 AS nbr_weeks  
FROM   ds9_32;
```

Result Set

NAME	HIRE_DATE	NBR_WEEKS
Lopez	2000-05-30	993
Brown	2001-02-20	955
Patel	2001-06-15	939
Moore	2002-08-17	878
Fang	2002-09-15	873

Example 10-33

Calculated the number of weeks of employment to two decimals. Assume CURRENT_DATE is 2019-06-15

Data Set ds9_33

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT
  last_name AS name,
  hire_date,
  ( DAYS(CURRENT_DATE) - DAYS(hire_date) ) / 7 AS nbr_weeks,
  DECIMAL((DAYS(CURRENT_DATE) - DAYS(hire_date)) / 7.00, 7, 2)
  AS nbr_weeks2
FROM   ds9_33;
```

Result Set

NAME	HIRE_DATE	NBR_WEEKS	NBR_WEEKS2
Lopez	2000-05-30	993	993.57
Brown	2001-02-20	955	955.57
Patel	2001-06-15	939	939.14
Moore	2002-08-17	878	878.00
Fang	2002-09-15	873	873.85

Example 10-34

Calculated the number of years of employment to two decimals. Use the number 365.2425 to represent the average number of days in a year Assume CURRENT_DATE is 2019-06-15

Data Set ds9_34

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT
  last_name AS name,
  hire_date,
  DECIMAL((DAYS(CURRENT_DATE) - DAYS(hire_date)) / 365.2425, 5,2)
    AS years
FROM   ds9_34;
```

Result Set

NAME	HIRE_DATE	YEARS
Lopez	2000-05-30	19.04
Brown	2001-02-20	18.31
Patel	2001-06-15	17.99
Moore	2002-08-17	16.82
Fang	2002-09-15	16.74

Date Functions

- All these date functions return a value with a DATE data type except the MONTHS_BETWEEN function, which returns a numeric value

Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month

MONTHS_BETWEEN Function

- Returns a numeric value of the number of months between two dates
- If the first date is earlier than the second date, a negative number is returned

Example 10-35

Use the MONTHS_BETWEEN function to calculate the months of employment. Assume CURRENT_DATE is 2019-06-21.

Data Set ds9_35

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       MONTHS_BETWEEN(CURRENT_DATE, hire_date) AS nbr_months  
FROM ds9_35;
```

Result Set

LAST_NAME	HIRE_DATE	NBR_MONTHS
Lopez	2000-05-30	228.709677419354839
Brown	2001-02-20	220.032258064516129
Patel	2001-06-15	216.193548387096774
Moore	2002-08-17	202.129032258064516
Fang	2002-09-15	201.193548387096774

Example 10-36

Examples of using the ROUND and DECIMAL functions with the MONTHS_BETWEEN function to calculate the months of employment to two decimals. Assume CURRENT_DATE is 2019-06-21.

Data Set ds9_36

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
---	-----	-----	-----
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```

SELECT last_name,
       hire_date,
       MONTHS_BETWEEN(CURRENT_DATE, hire_date) AS nbr_months,
       ROUND(MONTHS_BETWEEN(CURRENT_DATE, hire_date), 2) AS monthsR,
       DECIMAL( ROUND(MONTHS_BETWEEN(CURRENT_DATE, hire_date), 2), 5,2)
         AS monthsR2
FROM ds9_36;

```

Result Set

LAST_NAME	HIRE_DATE	NBR_MONTHS	MONTHSR	MONTHSR2
-----	-----	-----	-----	-----
Lopez	2000-05-30	228.709677419354839	228.710000000000000	228.71
Brown	2001-02-20	220.032258064516129	220.030000000000000	220.03
Patel	2001-06-15	216.193548387096774	216.190000000000000	216.19
Moore	2002-08-17	202.129032258064516	202.130000000000000	202.13
Fang	2002-09-15	201.193548387096774	201.190000000000000	201.19

Example 10-37

Use the MONTHS_BETWEEN function to calculate the number of years between two dates. Assume CURRENT_DATE is 2019-06-21.

Data Set ds9_37

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       MONTHS_BETWEEN(CURRENT_DATE, hire_date) / 12 AS nbr_years  
FROM ds9_37;
```

Result Set

LAST_NAME	HIRE_DATE	NBR_YEARS
Lopez	2000-05-30	19.059139784946236
Brown	2001-02-20	18.336021505376344
Patel	2001-06-15	18.016129032258064
Moore	2002-08-17	16.844086021505376
Fang	2002-09-15	16.766129032258064

ADD_MONTHS Function

- Add a specified number of months to a date
- Returns a date
- If the number supplied is negative, the function will subtract that number of months from the date argument

Example 10-38

Use the ADD_MONTHS function to add months to a date.

Data Set ds9_38

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       ADD_MONTHS(hire_date, 6) AS review_date  
FROM ds9_38;
```

Result Set

LAST_NAME	HIRE_DATE	REVIEW_DATE
Lopez	2000-05-30	2000-11-30
Brown	2001-02-20	2001-08-20
Patel	2001-06-15	2001-12-15
Moore	2002-08-17	2003-02-17
Fang	2002-09-15	2003-03-15

+ MONTHS

- Add a specified number of months to a date
- Returns a date
- If the number supplied is negative, the function will subtract that number of months from the date argument

Example 10-39

Use the ADD_MONTHS function to add months to a date.

Data Set ds9_39

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       hire_date + 6 MONTHS AS review_date  
FROM ds9_39;
```

Result Set

LAST_NAME	HIRE_DATE	REVIEW_DATE
Lopez	2000-05-30	2000-11-30
Brown	2001-02-20	2001-08-20
Patel	2001-06-15	2001-12-15
Moore	2002-08-17	2003-02-17
Fang	2002-09-15	2003-03-15

NEXT_DAY Function

- Determines the next occurrence of a specified day of the week after a given date

Example 10-40

Use the NEXT_DAY function to determine the start day (the first Monday after the hire date).

Data Set ds9_40

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       NEXT_DAY(hire_date, 'MONDAY') AS start_date  
FROM ds9_40;
```

Result Set

LAST_NAME	HIRE_DATE	START_DATE
Lopez	2000-05-30	2000-06-05
Brown	2001-02-20	2001-02-26
Patel	2001-06-15	2001-06-18
Moore	2002-08-17	2002-08-19
Fang	2002-09-15	2002-09-16

LAST_DAY Function

- Determines the last day of the month from a given date

Example 10-41

Payday is always the last day of the month. Use the LAST_DAY function to determine the last day of the month which is the first payday

Data Set ds9_41

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       LAST_DAY(hire_date) AS first_pay  
FROM ds9_41;
```

Result Set

LAST_NAME	HIRE_DATE	FIRST_PAY
Lopez	2000-05-30	2000-05-31
Brown	2001-02-20	2001-02-28
Patel	2001-06-15	2001-06-30
Moore	2002-08-17	2002-08-31
Fang	2002-09-15	2002-09-30

LAST_DAY + 1

- Determine first day of next month from a given date

Example 10-42

Return hire date and job review date. The job review date is calculated as the first day of the month following the month of the hire date using the `LAST_DAY + 1` function.

Data Set ds9_42

ID	FIRST_NAME	LAST_NAME	HIRE_DATE
201	Martina	Lopez	2000-05-30
202	Avery	Brown	2001-02-20
203	Chetan	Patel	2001-06-15
204	Logan	Moore	2002-08-17
205	Wang	Fang	2002-09-15

SQL Statement

```
SELECT last_name,  
       hire_date,  
       LAST_DAY(hire_date) + 1 DAY AS job_review  
FROM ds9_42;
```

Result Set

LAST_NAME	HIRE_DATE	JOB_REVIEW
Lopez	2000-05-30	2000-06-01
Brown	2001-02-20	2001-03-01
Patel	2001-06-15	2001-07-01
Moore	2002-08-17	2002-09-01
Fang	2002-09-15	2002-10-01

Nested Example

```
SELECT employee_id, hire_date,  
       DECIMAL( ( MONTHS_BETWEEN(CURRENT_DATE, hire_date) /12 ), 4,2) AS years_of_service,  
       ADD_MONTHS (hire_date, 6) AS review_date,  
       NEXT_DAY(hire_date, 'FRIDAY') AS NEXT_FRIDAY,  
       LAST_DAY(hire_date) AS LAST_DAY  
FROM employees  
WHERE MONTHS_BETWEEN (CURRENT_DATE, hire_date) > 36;
```

EMPLOYEE_ID	HIRE_DATE	YEARS_OF_SERVICE	REVIEW_DATE	NEXT_FRIDAY	LAST_DAY
100	06/17/87	30.65	12/17/87	06/19/87	06/30/87
101	09/21/89	28.39	03/21/90	09/22/89	09/30/89
102	01/13/93	25.08	07/13/93	01/15/93	01/31/93
200	09/17/87	30.40	03/17/88	09/18/87	09/30/87

Conversion Functions

Format Dates with TO_CHAR Function

- Convert dates stored in the default YYYY-MM-DD format to a formatted "**character**" string
- The '**format model**' must be enclosed in single quotations (') and is case-sensitive
- TO_CHAR (date column name, 'format model')
- Separate the date value from the 'format model' with a comma

<code>CURRENT_DATE;</code>	2019-02-21
<code>TO_CHAR(CURRENT_DATE, 'mm/dd/yyyy')</code>	06/21/2019
<code>TO_CHAR(CURRENT_DATE, 'Mon dd, yyyy')</code>	Jun 21, 2019
<code>TO_CHAR(CURRENT_DATE, 'month DD, YYYY')</code>	june 21, 2019
<code>TO_CHAR(CURRENT_DATE, 'Month DD, YYYY')</code>	June 21, 2019
<code>TO_CHAR(CURRENT_DATE, 'MONTH dd, yyyy')</code>	JUNE 21, 2019
<code>TO_CHAR(CURRENT_DATE, 'DAY, Month DD, YYYY')</code>	FRIDAY, June 21, 2019
<code>TO_CHAR(CURRENT_DATE, 'Day, Month DD, YYYY')</code>	Friday, June 21, 2019

TO_CHAR (Format Dates)

Displaying different format models

```
SELECT TO_CHAR(CURRENT_DATE, 'month DD, YYYY') AS date1,  
       TO_CHAR(CURRENT_DATE, 'Month DD, YYYY') AS date2,  
       TO_CHAR(CURRENT_DATE, 'MONTH dd, yyyy') AS date3  
FROM SYSIBM.SYSDUMMY1;
```

DATE1	DATE2	DATE3
february 13, 2018	February 13, 2018	FEBRUARY 13, 2018

TO_CHAR (Format Dates)

```
SELECT TO_CHAR(CURRENT_DATE, 'Mon dd, yyyy')           AS date1,  
       TO_CHAR(CURRENT_DATE, 'DAY, Month DD, YYYY') AS date2,  
       TO_CHAR(CURRENT_DATE, 'Day, Month DD, YYYY') AS date3  
FROM SYSIBM.SYSDUMMY1;
```

DATE1	DATE2	DATE3
Feb 13, 2018	TUESDAY, February 13, 2018	Tuesday, February 13, 2018

Format Numeric Values with TO_CHAR Function

- Numbers stored in the database have no formatting
 - No currency signs/symbols, no commas, no decimals or other formatting
- TO_CHAR converts numbers to a formatted character string
 - To add formatting, first need to convert the number to a character format:
 - `TO_CHAR(number, 'format model')`

Format Numeric Values with TO_CHAR Function

```
SELECT TO_CHAR(8000, '99,999')           AS FMT1,  
       TO_CHAR(8000, '$99,999.99')      AS FMT2,  
       TO_CHAR(0100.55, '000999.99')    AS FMT3,  
       TO_CHAR(050.00, '099.99')        AS FMT4,  
       TO_CHAR(00.00, '990.99')         AS FMT5,  
       TO_CHAR(-123.45, '990.99')       AS FMT6  
FROM SYSIBM.SYSDUMMY1;
```

FMT1	FMT2	FMT3	FMT4	FMT5	FMT6
8,000	\$ 8,000.00	000100.55	050.00	0.00	-123.45

How Functions are Evaluated

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'),  
              'Day, Month DD, YYYY') AS "Next Evaluation"  
FROM employees  
WHERE employee_id=100;
```

- Step 1: Six months is added to hire_date
- Step 2: The first Friday following the future day is determined
- Step 3: The default date format will be formatted to read and display the Friday in a format similar to: Friday, December 18, 1987, and will appear under the column name "Next Evaluation."

```
Next Evaluation  
-----  
Friday, December 18, 1987
```

Conditional Functions

CASE

COALESCE

CASE Expression

- Does the work of an IF-THEN-ELSE statement
- Data types of the CASE, WHEN, and ELSE expressions must be the same
- Returns the first value that satisfies the condition

Example 10-43 Use the CASE function to evaluate the credit limit column and categorize into four categories according to the value.

Data Set ds9_43

ID	CUSTOMER_NAME	CREDIT_LIMIT
---	-----	-----
101	Bargains Galore	125000
102	RedHot Discount	35500
103	NewTown Deals	95500
104	Ajax Sales	150500
105	BigBox Direct	145000
106	Mainstreet Inc	200000
107	Riverside Mfg	42750
108	Cube Industries	105250
109	LowCost Shops	155000
110	Sterling Mfg	180000

SQL Statement

```
SELECT id,  
       credit_limit,  
       CASE  
         WHEN credit_limit <= 40000 THEN 'No Category'  
         WHEN credit_limit <= 100000 THEN 'Bronze'  
         WHEN credit_limit <= 150000 THEN 'Silver'  
         WHEN credit_limit > 150000 THEN 'Gold'  
       END AS category  
FROM ds9_43  
ORDER BY credit_limit DESC;
```

Result Set

ID	CREDIT_LIMIT	CATEGORY
---	-----	-----
106	200000	Gold
110	180000	Gold
109	155000	Gold
104	150500	Gold
105	145000	Silver
101	125000	Silver
108	105250	Silver
103	95500	Bronze
107	42750	Bronze
102	35500	No Category

CASE Expression

- Using the optional ELSE with a CASE Structure

Example 10-44 Use the CASE function to evaluate the credit limit column and categorize into two categories according to credit limit. Use the ELSE clause to group all others into one category.

Data Set ds9_44

ID	CUSTOMER_NAME	CREDIT_LIMIT
101	Bargains Galore	125000
102	RedHot Discount	35500
103	NewTown Deals	95500
104	Ajax Sales	150500
105	BigBox Direct	145000
106	Mainstreet Inc	200000
107	Riverside Mfg	42750
108	Cube Industries	105250
109	LowCost Shops	155000
110	Sterling Mfg	180000

SQL Statement

```

SELECT id,
       credit_limit,
       CASE
         WHEN credit_limit > 150000 THEN 'Valued customer'
         WHEN credit_limit <= 40000 THEN 'Evaluate credit'
         ELSE
           'Good standing'
       END AS category
FROM ds9_44
ORDER BY credit_limit DESC;

```

Result Set

ID	CREDIT_LIMIT	CATEGORY
106	200000	Valued customer
110	180000	Valued customer
109	155000	Valued customer
104	150500	Valued customer
105	145000	Good standing
101	125000	Good standing
108	105250	Good standing
103	95500	Good standing
107	42750	Good standing
102	35500	Evaluate credit

CASE Expression

- Using a Selector with a CASE Structure

Example 10-45

Use CASE function to evaluate city and return ship state.

Data Set ds9_45

ID	CUSTOMER_NAME	CITY
---	-----	-----
101	Bargains Galore	Detroit
102	RedHot Discount	San Diego
103	NewTown Deals	Dallas
104	Ajax Sales	Detroit
105	BigBox Direct	Chicago
106	Mainstreet Inc	Dallas
107	Riverside Mfg	Chicago
108	Cube Industries	Dallas
109	LowCost Shops	San Diego
110	Sterling Mfg	Chicago

SQL Statement

```
SELECT id,
       customer_name,
       CASE city
         WHEN 'Chicago' THEN 'Illinois'
         WHEN 'Dallas'  THEN 'Texas'
         WHEN 'Detroit' THEN 'Michigan'
         WHEN 'San Diego' THEN 'California'
       END AS ship_state
FROM ds9_45
ORDER BY ship_state;
```

Result Set

ID	CUSTOMER_NAME	SHIP_STATE
---	-----	-----
102	RedHot Discount	California
109	LowCost Shops	California
105	BigBox Direct	Illinois
107	Riverside Mfg	Illinois
110	Sterling Mfg	Illinois
101	Bargains Galore	Michigan
104	Ajax Sales	Michigan
103	NewTown Deals	Texas
106	Mainstreet Inc	Texas
108	Cube Industries	Texas

CASE Expression

- Generate a new column based on calculations

Example 10-46

Calculate a new credit limit based on city. If the city is San Diego, increase the credit limit by 2.5 percent; otherwise, increase credit limit by 1.25 percent.

Data Set ds9_46

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
---	-----	-----	-----
101	Bargains Galore	Detroit	125000
102	RedHot Discount	San Diego	35500
103	NewTown Deals	Dallas	95500
104	Ajax Sales	Detroit	150500
105	BigBox Direct	Chicago	145000
106	Mainstreet Inc	Dallas	200000
107	Riverside Mfg	Chicago	42750
108	Cube Industries	Dallas	105250
109	LowCost Shops	San Diego	155000
110	Sterling Mfg	Chicago	180000

SQL Statement

```

SELECT id,
       customer_name,
       city,
       credit_limit,
       CASE
         WHEN city = 'San Diego' THEN DECIMAL( credit_limit * 1.025, 9,2 )
         ELSE                        DECIMAL( credit_limit * 1.0125, 9,2 )
       END AS new_credit_limit
FROM   ds9_46;

```

Result Set

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT	NEW_CREDIT_LIMIT
---	-----	-----	-----	-----
101	Bargains Galore	Detroit	125000	126562.50
102	RedHot Discount	San Diego	35500	36387.50
103	NewTown Deals	Dallas	95500	96693.75
104	Ajax Sales	Detroit	150500	152381.25
105	BigBox Direct	Chicago	145000	146812.50
106	Mainstreet Inc	Dallas	200000	202500.00
107	Riverside Mfg	Chicago	42750	43284.37
108	Cube Industries	Dallas	105250	106565.62
109	LowCost Shops	San Diego	155000	158875.00
110	Sterling Mfg	Chicago	180000	182250.00

CASE Expression

- Using a CASE expression in the WHERE clause

Example 10-47

Use the CASE function in the WHERE clause to set the value in which the WHERE clause uses in the selection.

Data Set ds9_47

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
---	-----	-----	-----
101	Bargains Galore	Detroit	125000
102	RedHot Discount	San Diego	35500
103	NewTown Deals	Dallas	95500
104	Ajax Sales	Detroit	150500
105	BigBox Direct	Chicago	145000
106	Mainstreet Inc	Dallas	200000
107	Riverside Mfg	Chicago	42750
108	Cube Industries	Dallas	105250
109	LowCost Shops	San Diego	155000
110	Sterling Mfg	Chicago	180000

SQL Statement

```

SELECT id,
       customer_name,
       city,
       credit_limit
FROM   ds9_47
WHERE  credit_limit >= CASE city
                        WHEN 'Chicago' THEN 148000
                        ELSE              108000
                        END
ORDER BY credit_limit;

```

Result Set

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
---	-----	-----	-----
101	Bargains Galore	Detroit	125000
104	Ajax Sales	Detroit	150500
109	LowCost Shops	San Diego	155000
110	Sterling Mfg	Chicago	180000
106	Mainstreet Inc	Dallas	200000

COALESCE Function

- Pronounced kow-uh-**les**
- Coalesce means "to come together"
- Is an SQL ANSI standard function
- Returns the first non-NULL value by substituting a value for a NULL value

COALESCE Function

- Evaluates the arguments in a list and returns the value of the first non-null value
- If the first expression is null, the function continues down the line until a **not null** expression is found
- The data types of the null value column and the new value must be the same

Example 10-48

Evaluate the salary and commission columns and use the COALESCE function to identify the first non-null value in each row.

Data Set ds9_48

ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION
201	Martina	Lopez	2500	1200
202	Avery	Brown	2250	NULL
203	Chetan	Patel	NULL	1460
204	Logan	Moore	3125	1450
205	Wang	Fang	NULL	NULL

SQL Statement

```
SELECT id,  
       last_name,  
       salary,  
       commission,  
       COALESCE(salary, commission, 0) coalesce_value  
FROM ds9_48;
```

Result Set

ID	LAST_NAME	SALARY	COMMISSION	COALESCE_VALUE
201	Lopez	2500	1200	2500
202	Brown	2250	NULL	2250
203	Patel	NULL	1460	1460
204	Moore	3125	1450	3125
205	Fang	NULL	NULL	0

COALESCE Function

- When a numeric calculation is performed with null, the result is null
- The COALESCE function converts the null value to a number before numeric calculations are done to avoid a null result

Example 10-49

Calculate bonus by multiplying salary by 3.25 percent. If salary is NULL, replace NULL with zero before the calculation is performed.

Data Set ds9_49

ID	FIRST_NAME	LAST_NAME	SALARY
201	Martina	Lopez	2500
202	Avery	Brown	2250
203	Chetan	Patel	NULL
204	Logan	Moore	3125
205	Wang	Fang	NULL

SQL Statement

```
SELECT id,  
       last_name,  
       salary,  
       DECIMAL( COALESCE(salary, 0) * .0325, 7,2 ) AS bonus  
FROM ds9_49;
```

Result Set

ID	LAST_NAME	SALARY	BONUS
201	Lopez	2500	81.25
202	Brown	2250	73.12
203	Patel	NULL	0.00
204	Moore	3125	101.56
205	Fang	NULL	0.00

COALESCE Function

- Using COALESCE function with character strings

Example 10-50

List the department for each employee. Use the COALESCE function to replace NULL values with 'Not assigned'

Data Set ds9_50

ID	FIRST_NAME	LAST_NAME	DEPARTMENT
---	-----	-----	-----
201	Martina	Lopez	Marketing
202	Avery	Brown	Sales
203	Chetan	Patel	NULL
204	Logan	Moore	Administration
205	Wang	Fang	NULL

SQL Statement

```
SELECT id,  
       last_name,  
       COALESCE(department, 'Not assigned') AS department  
FROM ds9_50;
```

Result Set

ID	LAST_NAME	DEPARTMENT
---	-----	-----
201	Lopez	Marketing
202	Brown	Sales
203	Patel	Not assigned
204	Moore	Administration
205	Fang	Not assigned

