# Appendix H: Data Type Functions

I have used the following sources:

- Oracle8 SQL Reference, Oracle Corporation, 1997, Part No. A58225-01.
- Date/Darwen: A Guide to the SQL Standard, Fourth Edition, Addison-Wesley, 1997.
- Sunderraman: Oracle Programming, A Primer. Addison-Wesley, 1999.
- Stefan Brass: Skript zur Vorlesung Informationssysteme (in German), Universität Hildesheim, 1996/97.
- van der Lans: SQL, Der ISO-Standard (in German, there is an English version), Hanser, 1990.
- Don Chamberlin: A Complete Guide to DB2 Universal Database. Morgan Kaufmann, 1998.
- Microsoft SQL Server Books Online: Accessing and Changing Data.





## Overview

1. Data Type Functions in Oracle
2. Data Type Functions in SQL-92
3. Data Type Functions in DB2
4. Data Type Functions on SQL Server





## Functions for Numbers (1)

- `+`, `-`, `*`, `/`: Addition etc.
- `ABS(`$x$`)`: Absolut value of $x$, e.g. `ABS(-2)=2`.
- `ACOS(`$x$`)`, `ASIN(`$x$`)`, `ATAN(`$x$`)`, `ATAN2(`$x$`,`$y$`)`: Arc cosine etc.
  The result is accurate to 30 decimal digits.
- `CEIL(`$x$`)`: Smallest integer $\geq x$, e.g. `CEIL(3.2)=4`.
- `COS(`$x$`)`, `SIN(`$x$`)`, `TAN(`$x$`)`: Cosine/Sine/Tangent.
  The angle is measured in radians, i.e. $180° = \pi = 3.14\ldots$.
- `COSH(`$x$`)`, `SINH(`$x$`)`, `TANH(`$x$`)`: Hyperbolic Cosine etc.
- `EXP(`$x$`)`: $e^x$, $e$ raised to the $x$ th power, $e = 2.718\ldots$.
- `FLOOR(`$x$`)`: Largest integer $\leq x$, e.g. `FLOOR(3.7)=3`.





## Functions for Numbers (2)

- `LN(`$x$`)`: Natural logarithm of $x$, i.e. $\log_e(x)$.
- `LOG(`$b$`, `$x$`)`: Logarithm of $x$ with base $b$, e.g. `LOG(2, 8)=3`.
- `MOD(`$m$`, `$n$`)`: Remainder of $m$ divided by $n$: `MOD(7, 3)=1`.
- `POWER(`$x$`, `$y$`)`: $x^y$, e.g. `POWER(3, 2)=9`.
- `ROUND(`$x$`)`: $x$ rounded to next integer, e.g. `ROUND(3.7)=4`.
- `ROUND(`$x$`, `$n$`)`: $x$ rounded to $n$ places right of the decimal point, e.g. `ROUND(2.66, 1)=2.7`.
- `SIGN(`$x$`)`: $-1$ if $x < 0$, $+1$ if $x > 0$, 0 if $x = 0$.
- `SQRT(`$x$`)`: $\sqrt{x}$ (square root of $x$), e.g. `SQRT(9)=3.0`.
- `TRUNC(`$x$`, `$n$`)`: Returns $x$ truncated to $n$ decimal places.

## Functions for Strings (1)

- $s_1$||$s_2$: $s_1$ concatenated with $s_2$, e.g. 'ab'||'c'='abc'.
  One can also write concat($s_1$,$s_2$) instead of $s_1$ || $s_2$.
- ASCII($s$): Returns the code of $s$ in the DB character set.
  E.g. ASCII('a')=97. The DB character set does not have to be "ASCII", so the name is a bit misleading.
- CHR($n$): Character with code $n$ in the DB character set.
  E.g. CHR(97)='a'.
- INSTR($s$,$x$): Returns position of first occurrence of $x$ in $s$.
  If $x$ does not occur in $s$, 0 is returned. INSTR($s$,$x$,$n$,$m$) searches $s$ only beginning at position $n$ and returns the position of the $m$-th occurrence of $x$ in $s$. If $n$ is negative, startposition and $m$-th match are counted from the end.





## Functions for Strings (2)

- INITCAP($s$): First letter of every word in uppercase.
  All other letters of $s$ are in lowercase in INITCAP($s$).
- LENGTH($s$): Number of characters in $s$.
  E.g. LENGTH('abc')=3.
- LOWER($s$): $s$ with all letters lowercase.
- LPAD($s$,$n$): $s$ is filled with spaces to length $n$ (on the left).
  LPAD($s$,$n$,$p$) allows to use any string $p$ instead of a space as filler. If $s$ is longer than $n$ it will be cutted to the right.
- LTRIM($s$): Returns $s$ with leading spaces removed.
  LTRIM($s$,$x$) allows to specify a set of characters $x$ (as a string) which should be removed (instead of spaces).





## Functions for Strings (3)

- REPLACE($s$,$x$,$y$): Replaces every occurrence of $x$ by $y$ in $s$.
- RPAD($s$,$n$): $s$ is filled with spaces to length $n$ (on the right).
  See LPAD for a possibility to specify the fill material.
- RTRIM($s$): Returns $s$ with spaces at the end removed.
  See LTRIM for how to specify the characters to be removed.
- SOUNDEX($s$): Phonetic representation of $s$.
  A condition like SOUNDEX(EName)=SOUNDEX('...') might help to find employees with a similar name to the one given.
- SUBSTR($s$,$m$): Returns suffix of $s$ beginning at position $m$.
  E.g. SUBSTR('abc',2)='bc'. See also next page.





## Functions for Strings (4)

- SUBSTR($s$,$m$,$n$): Returns a substring of $s$.
  The substring starts at position $m$ and is $n$ characters long (i.e. it ends at position $m+n-1$, or at the end of $s$ if $s$ is shorter). The first character has Position $1$. If $m$ is negative, the position is relative to the end of $s$.
- TRANSLATE($s$,$x$,$y$): Substitutes in $s$ every character occurring in $x$ by the corresponding character in $y$.
  If $y$ is shorter than $x$, the corresponding characters will be deleted. However, if $y$ is empty, the null value is returned.
- UPPER($s$): Returns $s$ with all letters uppercase.

## Functions for Date Values (1)

- $d$ + $n$: Returns date $d$ plus $n$ days.
  $n$ does not have to be a whole number, e.g. with $d + 10/1440$ you can add 10 minutes.
- $d$ - $n$: Returns date $d$ minus $n$ days.
- $d_1$ - $d_2$: Number of days between $d_2$ and $d_1$.
- `ADD_MONTHS(`$d$`,`$n$`)`: Returns date $d$ plus $n$ months.
- `LAST_DAY(`$d$`)`: Returns the last day of the month which contains $d$.

  E.g. `LAST_DAY(SYSDATE) - SYSDATE` gives the number of days left in the current month.





## Functions for Date Values (2)

- `MONTHS_BETWEEN(`$d_1$`,`$d_2$`)`: Returns the number of months between $d_1$ and $d_2$.

  The result can be fractional if $d_1$ and $d_2$ are not the same day in different months or both the last day of the month.
- `NEW_TIME(`$d$`,`$z_1$`,`$z_2$`)`: Returns date and time in time zone $z_2$ corresonding to $d$ in time zone $z_1$.

  E.g. `NEW_TIME(`$d$`,'EST','PST')` returns the result of converting $d$ from Eastern Time into Pacific Time.
- `NEXT_DAY(`$d$`,`$w$`)`: Returns the date of the first weekday $w$ after Date $d$.

  E.g. if `SYSDATE` is 30-MAR-99, which is a Tuesday, then `NEXT_DAY(SYSDATE,'WED')` returns 31-MAR-99.





## Functions for Date Values (3)

- `ROUND(`$d$`)`: Round $d$ to the nearest beginning of a day.

  Remember that although the type is called “`DATE`” it is really a timestamp which contains date and time.
- `ROUND(`$d$`,`$u$`)`: Round $d$ to unit $u$.

  A format string is used for defining the unit, e.g. `'YYYY'` for year, `'MON'` for month, `'HH'` for hour.
- `SYSDATE`: Current date (and time).
- `TRUNC(`$d$`)`: Take the same day as $d$ but set time to $0^{00}$.
- `TRUNC(`$d$`,`$u$`)`: Round down to the unit $u$.

  Units are as in `ROUND`.





## Type Conversion Functions (1)

- Oracle tries to make types always fitting by automatically converting values in other types whenever possible.
- E.g. the following would give 128 and not an error:

```
SELECT '0123' + 5
FROM   DUAL
```

  “`DUAL`” is an Oracle system table which contains a single row. You can evaluate arithmetic expressions in this way.
- It is, however, much clearer to request the type conversion explicitly.
- You need an explict conversion to see the time component of date values.

# Type Conversion Functions (2)

- `TO_CHAR(`$d$`,`$f$`)`: Conversion of date to string (format $f$).

  Some examples for formats and results are:

| Format $f$ | Result |
|---|---|
| 'Month DD, YYYY' | March 30, 1999 |
| 'DY, MM/DD/YY' | TUE, 03/30/99 |
| 'DD.MM.YY HH24:MI:SS' | 30.03.99 18:49:28 |
| 'DD-MON-YY, HH12:MI AM' | 30-MAR-99, 06:49 PM |

  If you write `TO_CHAR(`$d$`)`, the default format is used which depends on the national language settings ('DD-MON-YY').

- `TO_DATE(`$s$`,`$f$`)`: Conversion of a string to `DATE` (date+time).

  `TO_DATE(`$s$`)` is also possible. For the format $f$ see above.





# Type Conversion Functions (3)

- `TO_CHAR(`$x$`,`$f$`)`: Conversion of number to string (format $f$).

  Some examples for formats and results are:

| Format $f$ | Number $x$ | Result |
|---|---|---|
| '99.99' | 0 | '   .00' |
| '90.99' | 0 | ' 0.00' |
| '90.99MI' | -3 | ' 3.00-' |
| '999,999.99' | 1000 | '    1,000.00' |

  Without a format, i.e. `TO_CHAR(`$x$`)`, all digits of $x$ are printed. Try `SELECT TO_CHAR(ASIN(1)*2) FROM DUAL`.

- `TO_NUMBER(`$s$`,`$f$`)`: Conversion of a string to a number.

  `TO_NUMBER(`$s$`)` is also possible. For the format $f$ see above.





# Type Conversion Functions (4)

- `DUMP(`$x$`)`: Convert any Oracle data value into a string showing the internal representation.

  E.g. `DUMP('abc')` gives `'Typ=96 Len=3:  97,98,99'` and `DUMP(1)` gives `'Typ=2 Len=2:  193,2'`. You can request hexadecimal notation for the bytes, e.g. `DUMP(1,16)`.

- `VSIZE(`$x$`)`: Number of bytes in internal representation of $x$.
- `HEXTORAW(`$s$`)`: Convert a string containing hexadecimal digits into the corresponding `RAW` value (byte string).
- `RAWTOHEX(`$r$`)`: Convert a raw value into a string.
- There are further conversion functions for the type `ROWID` (not discussed yet) and for converting character sets.





# Conditional Expressions

- `GREATEST(`$x_1$`, ..., `$x_n$`)`: Maximum of $\{x_1, \ldots, x_n\}$.

  The name `MAX` is already used for an aggregation function.

- `LEAST(`$x_1$`, ..., `$x_n$`)`: Minimum of $\{x_1, \ldots, x_n\}$.
- `NVL(`$x_1$`,`$x_2$`)`: If $x_1$ is not null, $x_1$ is returned, otherwise $x_2$.

  E.g. if you want to treat a null value in the attribute $A$ like the number 0, you write `NVL(`$A$`,0)`.

- `DECODE(`$v$`,`$x_1$`,`$y_1$`,...,`$x_n$`,`$y_n$`,`$y$`)`: If $v$ is one of the $x_i$, the corresponding $y_i$ is returned. Otherwise $y$ is returned.

  E.g. `DECODE(DEPTNO,1,'Research',2,'Sales','Other')` returns `'Research'` if `DEPTNO=1` or `'Sales'` if `DEPTNO=2`, or `'Other'` otherwise. If you don't specify $y$, a null value is returned if no $x_i$ matches.

## Other Functions

- SYSDATE: Returns the current date and time.
- UID: Returns an integer that identifies the current user.
- USER: Returns name of current user.
- USERENV('LANGUAGE'): Returns language, territory, and database character set, e.g. AMERICAN_AMERICA.US7ASCII.
  There are other possible parameters to USERENV.
- There are more functions for object types.
- You can write your own functions in PL/SQL.





## Overview

1. Data Type Functions in Oracle
2. Data Type Functions in SQL-92
3. Data Type Functions in DB2
4. Data Type Functions on SQL Server





## General Remark

- In the SQL-86 standard, the only data type functions available were +, −, *, and /.
- So you can expect these in any DBMS.
- But other functions differ greatly from DBMS to DBMS. If you use more than +, −, *, and /, you lose portability.
- The SQL-92 standard contains still less functions than Oracle.
- Also, the notation is quite different. E.g. in SQL-92 you have to write “SUBSTRING($s$ FROM $m$ FOR $n$)” instead of “SUBSTR($s$,$m$,$n$)” in Oracle.
- SQL-92 contains the string concatenation operator ||.





## Data Type Functions in SQL-92 (1)

- +, −, *, /
  + and − can also be used to a date/time value and an interval, − can be used to compute and interval from two date/time values, and * and / can be used on intervals and numbers.
- $s_1$ || $s_2$
- BIT_LENGTH($b$)
- CASE WHEN $x_1$ THEN $y_1$ WHEN $x_2$ THEN $y_2$ ... ELSE $y$ END
- CAST($x$ AS ⟨Data Type⟩)

## Data Type Functions in SQL-92 (2)

- `CHARACTER_LENGTH(`$s$`)`
  `CHAR_LENGTH(`$s$`)` can be written instead.
- `COALESCE(`$x_1$`,` $x_2$`, ...,` $x_n$`)`
  Returns the first among its arguments which is non-null.
- `CURRENT_DATE`
- `CURRENT_TIME`
- `CURRENT_TIMESTAMP`
- `CURRENT_USER`
- `EXTRACT(YEAR FROM` $d$`)`
  Instead of year, also `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND`, `TIMEZONE_HOUR`, `TIMEZONE_MINUTE` can be used.





## Data Type Functions in SQL-92 (3)

- `LOWER(`$s$`)`
- `NULLIF(`$x$`,` $y$`)`
  Returns x unless $x = y$. For $x = y$ it returns null.
  E.g. useful to exclude a certain value from aggregations.
- `OCTET_LENGTH(`$s$`)`
- `POSITION(`$s_1$ `IN` $s_2$`)`
- `SESSION_USER`
- `SUBSTRING(`$s$ `FROM` $m$ `FOR` $n$`)`
- `SYSTEM_USER`





## Data Type Functions in SQL-92 (4)

- `TRIM(LEADING` $c$ `FROM` $s$`)`
  Instead of `LEADING`, also `TRAILING` and `BOTH` can be used.
  `TRIM(`$s$`)` is equivalent to `TRIM(BOTH ' ' FROM` $s$`)`.
- `UPPER(`$s$`)`
- `USER`
- Plus functions for the national language support.





## Overview

1. Data Type Functions in Oracle
2. Data Type Functions in SQL-92
3. Data Type Functions in DB2
4. Data Type Functions on SQL Server

## Functions for Numbers (1)

- +, −, ∗, /: Addition etc.
- ABS($x$): Absolut value of $x$, e.g. ABS(-2)=2.
  ABSVAL($x$) is a synonym for ABS($x$).
- ACOS($x$), ASIN($x$), ATAN($x$), ATAN2($x$,$y$): Arc cosine etc.
- CEIL($x$): Smallest whole number $\geq x$, e.g. CEIL(3.2)=4.0.
- COS($x$), SIN($x$), TAN($x$), COT($x$): Cosine/Sine/Tangent/...
  The angle is measured in radians, i.e. $180° = \pi = 3.14\ldots$.
- DEGREES($x$): Converts radians to degrees.
  RADIANS($x$) converts degrees to radians.
- EXP($x$): $e^x$, $e$ raised to the $x$ th power, $e = 2.718\ldots$.
- FLOOR($x$): Largest whole number $\leq x$, e.g. FLOOR(3.7)=3.0.





## Functions for Numbers (2)

- LN($x$): Natural logarithm of $x$, i.e. $\log_e(x)$.
  LOG($x$) is a synonym.
- LOG10($x$): Logarithm of $x$ with base 10.
- MOD($m$, $n$): Remainder of $m$ divided by $n$: MOD(7, 3)=1.
- POWER($x$, $y$): $x^y$, e.g. POWER(3, 2)=9.
- RAND(): Returns a random floating-point number between 0 and 1.
  RAND($i$) uses an integer $i$ as seed value.
- ROUND($x$, $n$): $x$ rounded to $n$ places right of the decimal point, e.g. ROUND(2.66, 1)=2.7.
  $n$ can be negative, e.g. ROUND(127,-1)=130.





## Functions for Numbers (3)

- SIGN($x$): $-1$ if $x < 0$, $+1$ if $x > 0$, 0 if $x = 0$.
- SQRT($x$): $\sqrt{x}$ (square root of $x$), e.g. SQRT(9)=3.0.
- TRUNC($x$, $n$): Returns $x$ truncated to $n$ decimal places to the right of the decimal point.
  $n$ can be negative. TRUNCATE($x$, $n$) is a synonym.





## Functions for Strings (1)

- $s_1$||$s_2$: $s_1$ concatenated with $s_2$, e.g. 'ab'||'c'='abc'.
  One can also write concat($s_1$,$s_2$) instead of $s_1$ || $s_2$.
- ASCII($s$): Returns the ASCII code of the first character of $s$.
- CHR($n$): Returns the character with ASCII code $n$.
- DIFFERENCE($s_1$,$s_2$): Returns an integer from 0 to 4 measuring the difference between the sounds of the two strings (4 is perfect match).
- GENERATE_UNIQUE(): Returns a binary string of length 13, different from every other invocation of the same function.
- INSERT($s_1$,$n$,$l$,$s_2$): Returns a string computed from $s_1$ by replacing the substring of length $l$ starting at position $n$ by $s_2$.

## Functions for Strings (2)

- `LCASE(`$s$`)`: Returns a copy of $s$ with all letters lowercase.
- `LEFT(`$s$`,`$n$`)`: Prefix of $s$ of length $n$ (leftmost characters).
- `LENGTH(`$s$`)`: Number of characters in $s$.
  E.g. `LENGTH('abc')`=3.
- `LOCATE(`$s_1$`,`$s_2$`)`: Returns the position of the first occurrence of the substring $s_2$ inside $s_1$.
  Positions are counted from 1, the result 0 means that $s_2$ was not found in $s_1$. It is possible to specify a starting position for the search: `LOCATE(`$s_1$`,`$s_2$`,`$n$`)`.
  `POSSTR(`$s_1$`,`$s_2$`)` is a synonym for `LOCATE(`$s_1$`,`$s_2$`)`.
- `LTRIM(`$s$`)`: Returns a copy of $s$ with leading spaces removed.





## Functions for Strings (3)

- `REPEAT(`$s$`,`$n$`)`: Returns a string consisting of $n$ repetitions of $s$.
- `REPLACE(`$s$`,`$f$`,`$t$`)`: Replaces every occurrence of $f$ by $t$ in $s$.
- `RIGHT(`$s$`,`$n$`)`: Returns the rightmost $n$ characters of $s$.
- `RTRIM(`$s$`)`: Returns $s$ with spaces at the end removed.
- `SOUNDEX(`$s$`)`: Phonetic representation of $s$ (in four characters).
  A condition like `SOUNDEX(EName)=SOUNDEX('...')` might help to find employees with a similar name to the one given.
- `SPACE(`$n$`)`: Returns a string consisting of $n$ spaces.
- `SUBSTR(`$s$`,`$m$`)`: Returns suffix of $s$ beginning at position $m$.
  E.g. `SUBSTR('abc',2)='bc'`. See also next page.





## Functions for Strings (4)

- `SUBSTR(`$s$`,`$m$`,`$n$`)`: Returns a substring of $s$.
  The substring starts at position $m$ and is $n$ characters long. The result is filled with spaces if $s$ is shorter than $m+n-1$ characters.
- `TRANSLATE(`$s$`,`$t$`,`$f$`)`: Substitutes in $s$ every character occurring in $f$ by the corresponding character in $t$.
  If $t$ (to-string) is shorter than $f$ (from-string), the corresponding characters are replaced by spaces (or a pad-character which can be specified as fourth argument).
- `UCASE(`$s$`)`: Returns $s$ with all letters uppercase.
  `TRANSLATE(`$s$`)` is a synonym.





## Functions for Date Values (1)

- $d+x$: You can add a date duration to a date, a time duration to a time, a timestamp duration to a timestamp.
- $d-x$: You can subtract a duration from a date, time, or timestamp value.
- $d_1-d_2$: You can also subtract two dates/time/timestamp values from each other to get a duration.
- `DAY(`$d$`)`: Returns the day portion of a date or timestamp.
- `DAYNAME(`$d$`)`: Returns the weekday name of a date/timestamp.
- `DAYOFWEEK(`$d$`)`: Returns the number of the weekday.
  E.g. 1 means Sunday, 2 means Monday, and so on.
- `DAYOFYEAR(`$d$`)`: Returns the day of the year (1..366).

## Functions for Date Values (2)

- `DAYS(`$d$`)`: Returns the number of days since January 1, 0001.
- `HOUR(`$t$`)`: Returns the hour part of a time or timestamp value.
- `JULIAN_DAY(`$d$`)`: Returns the number of days since January 1, 4712 B.C.
- `MICROSECOND(`$t$`)`: Extracts the microsecond part from a timestamp.
- `MIDNIGHT_SECONDS`: Returns the number of seconds since midnight for a given time or timestamp value.
- `MINUTE(`$t$`)`: Returns the minute part of a time/timestamp.





## Functions for Date Values (3)

- `MONTH(`$d$`)`: Extracts the month part from a date/timestamp.
- `MONTHNAME(`$d$`)`: Returns the name of the month part of a date/timestamp.
- `QUARTER(`$d$`)`: Returns the quarter (`1..4`) of a date/timestamp.
- `SECOND(`$t$`)`: Extracts the second part from a time/timestamp.
- `TIMESTAMPDIFF(`$u$`,`$x$`)`: Rounds the difference of two timestamp values passed in the argument $x$ to the unit $u$.
- `WEEK(`$d$`)`: Returns the week of the year (`1..54`).
  Each week starts on Sunday.
- `YEAR(`$d$`)`: Extracts the year part of a date/timestamp.





## Type Conversion Functions (1)

- `CAST(`$x$ `AS` $t$`)`: Converts $x$ to type $t$.
- `CAST(NULL AS` $t$`)`: Produces a null value of type $t$.
- `BLOB(`$x$`)`: Result of converting $x$ into a binary large object.
  `BLOB(`$x$`,`$n$`)` produces a BLOB of max. $n$ bytes (truncated if necessary).
- `CHAR(`$x$`)`: Converts $x$ into a string.
  `CHAR(`$x$`,`$n$`)`: returns a string of length $n$ (blank-padded or truncated). The character used for the decimal point can be specified in `CHAR(`$n$`,`$c$`)`.
- `CLOB(`$x$`)`, `CLOB(`$x$`,`$n$`)`: Character large object.





## Type Conversion Functions (2)

- `DATE(`$x$`)`: Converts $x$ to a date value.
  $x$ must be a string in a format which DB2 understands or an integer (number of days since January 1, 0001).
- `DBCLOB(`$x$`)`, `DBCLOB(`$x$`,`$n$`)`: Double byte character large object.
- `DIGITS(`$x$`)`: A string with the digits in the decimal representation of the number (without sign or decimal point).
- `DOUBLE(`$x$`)`, `DOUBLE_PRECISION(`$x$`)`, `FLOAT(`$x$`)`: Conversion to floating-point number.
- `GRAPHIC(`$x$`)`, `GRAPHIC(`$x$`,`$n$`)`: Conversion to double-byte string.

## Type Conversion Functions (3)

- `HEX(`$x$`)`: Hexadecimal representation of the internal storage format of $x$.
- `INTEGER(`$x$`)`, `INT(`$x$`)`: Conversion of $x$ to an integer.
- `LONG_VARCHAR(`$x$`)`: Converts $x$ to type `LONG VARCHAR`.
- `LONG_VARGRAPHIC(`$x$`)`: Converts $x$ to type `LONG VARGRAPHIC`.
- `SMALLINT(`$x$`)`: Converts $x$ to type `SMALLINT`.
- `TIME(`$x$`)`: Converts $x$ to type `TIME`.
- `TIMESTAMP(`$x$`)`: Converts $x$ to type `TIMESTAMP`.





## Type Conversion Functions (4)

- `TIMESTAMP_ISO(`$x$`)`: Converts $x$ to type `TIMESTAMP`.
  If $x$ is a date, the time part will be `00:00`. If $x$ is a time, the date part will be the current date.
- `VARCHAR(`$x$`)`: Converts $x$ to type `VARCHAR`.
- `VARGRAPHIC(`$x$`)`: Converts $x$ to type `VARGRAPHIC`.





## Conditional Expressions

- `COALESCE(`$x_1$`,`$x_2$`,...)`: Returns the first argument which is not null.
  `VALUE` is a synonym for `COALESCE`.
- `NULLIF(`$x$`,`$y$`)`: Returns null if $x = y$, otherwise $x$.
- `CASE` $x$ `WHEN` $y_1$ `THEN` $z_1$
  `WHEN` $y_2$ `THEN` $z_2$
  ...
  `ELSE` $z$
  `END`
  Returns $z_i$ if $x = y_i$.