



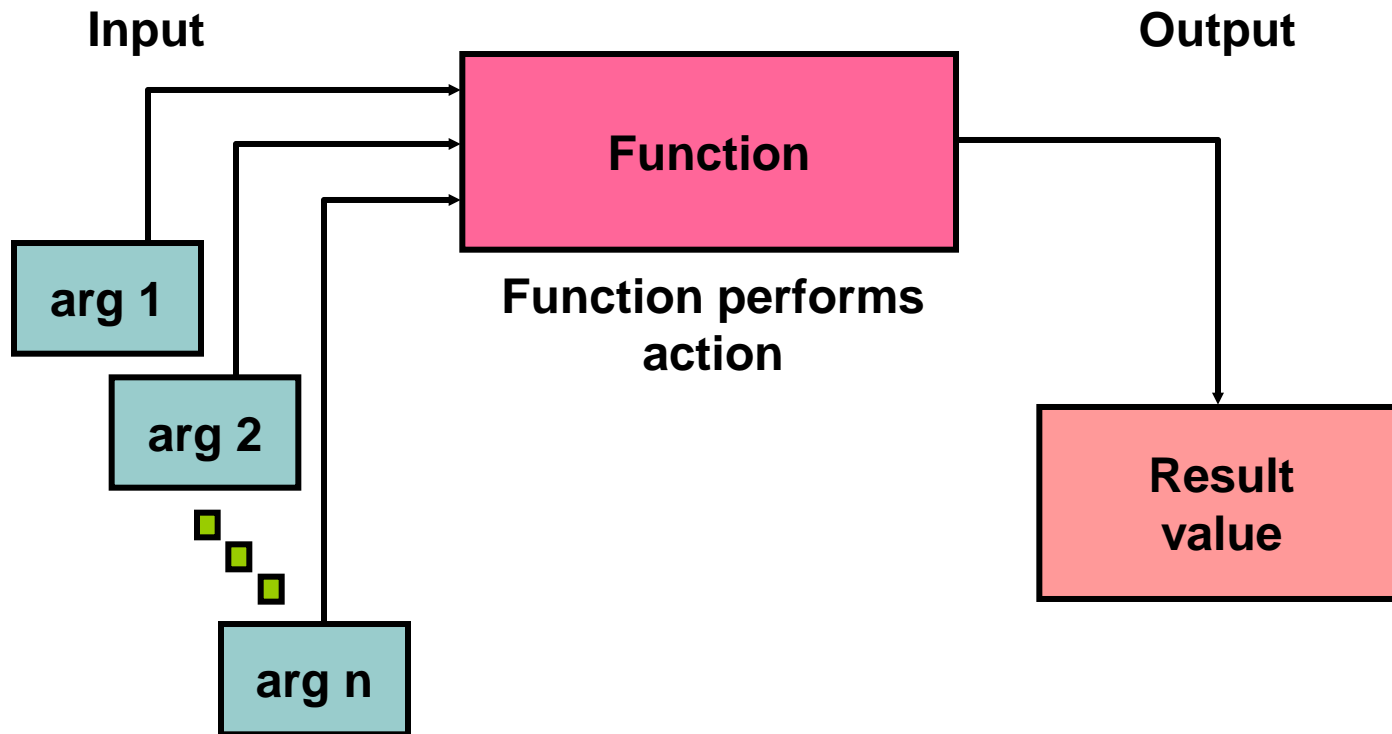
Using Single-Row Functions to Customize Output

Objectives

After completing this lesson, you should be able to do the following:

- **Describe various types of functions that are available in SQL**
- **Use character, number, and date functions in `SELECT` statements**
- **Describe the use of conversion functions**

SQL Functions

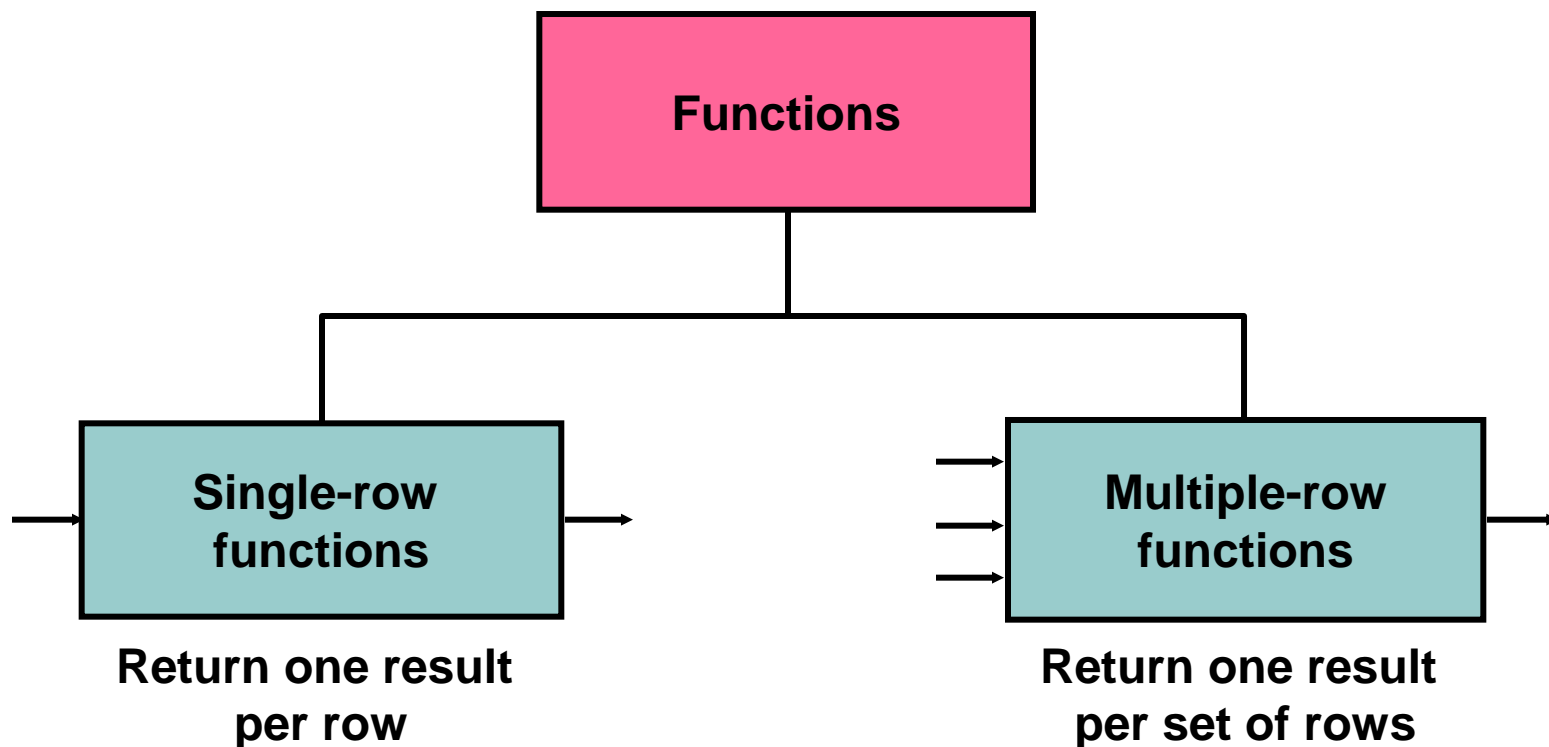


Two Types of SQL Functions

Supported SQL functions change from DBMS to DBMS!

MySql functions - https://www.w3schools.com/sql/sql_ref_mysql.asp

SQL Server functions - https://www.w3schools.com/sql/sql_ref_sqlserver.asp



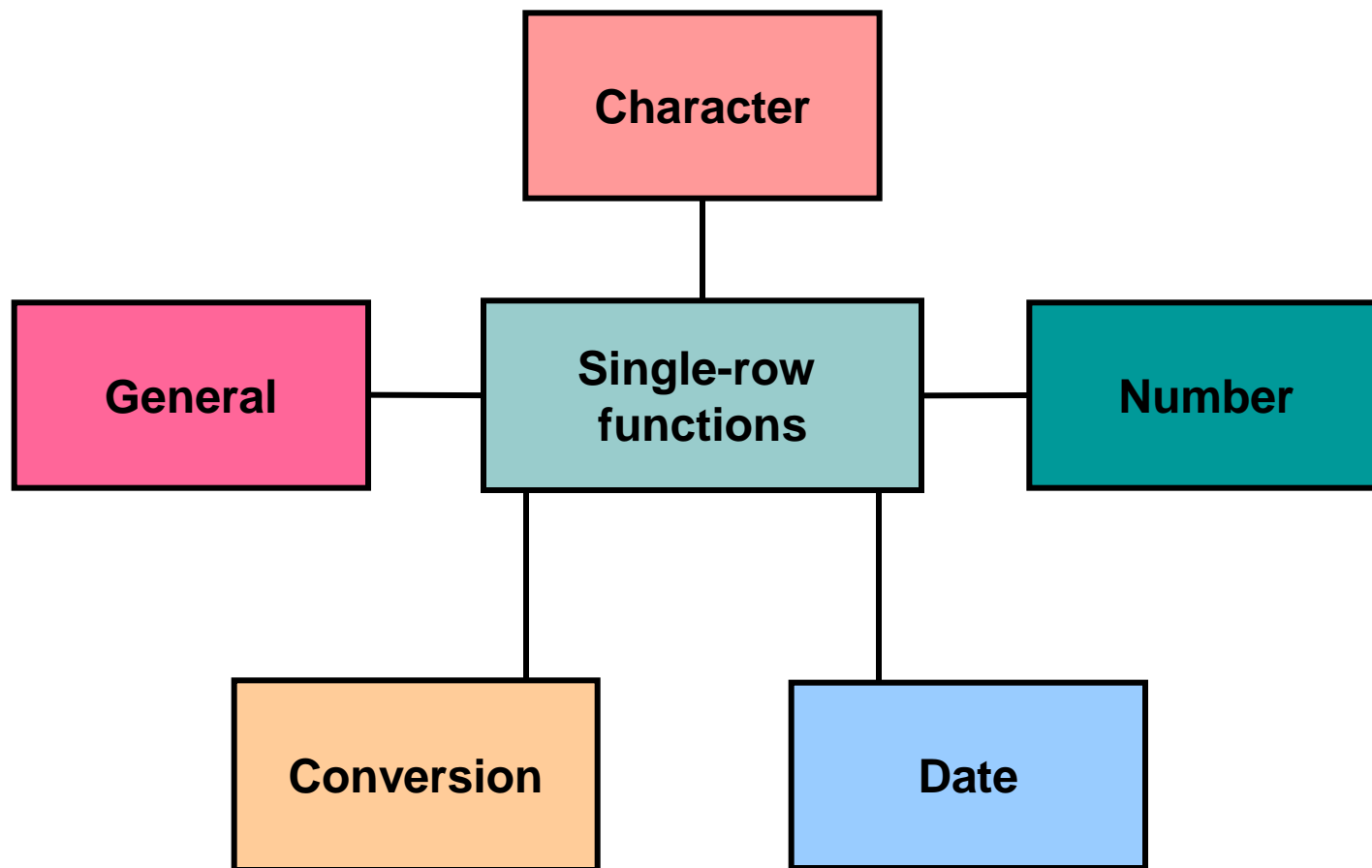
Single-Row Functions

Single-row functions:

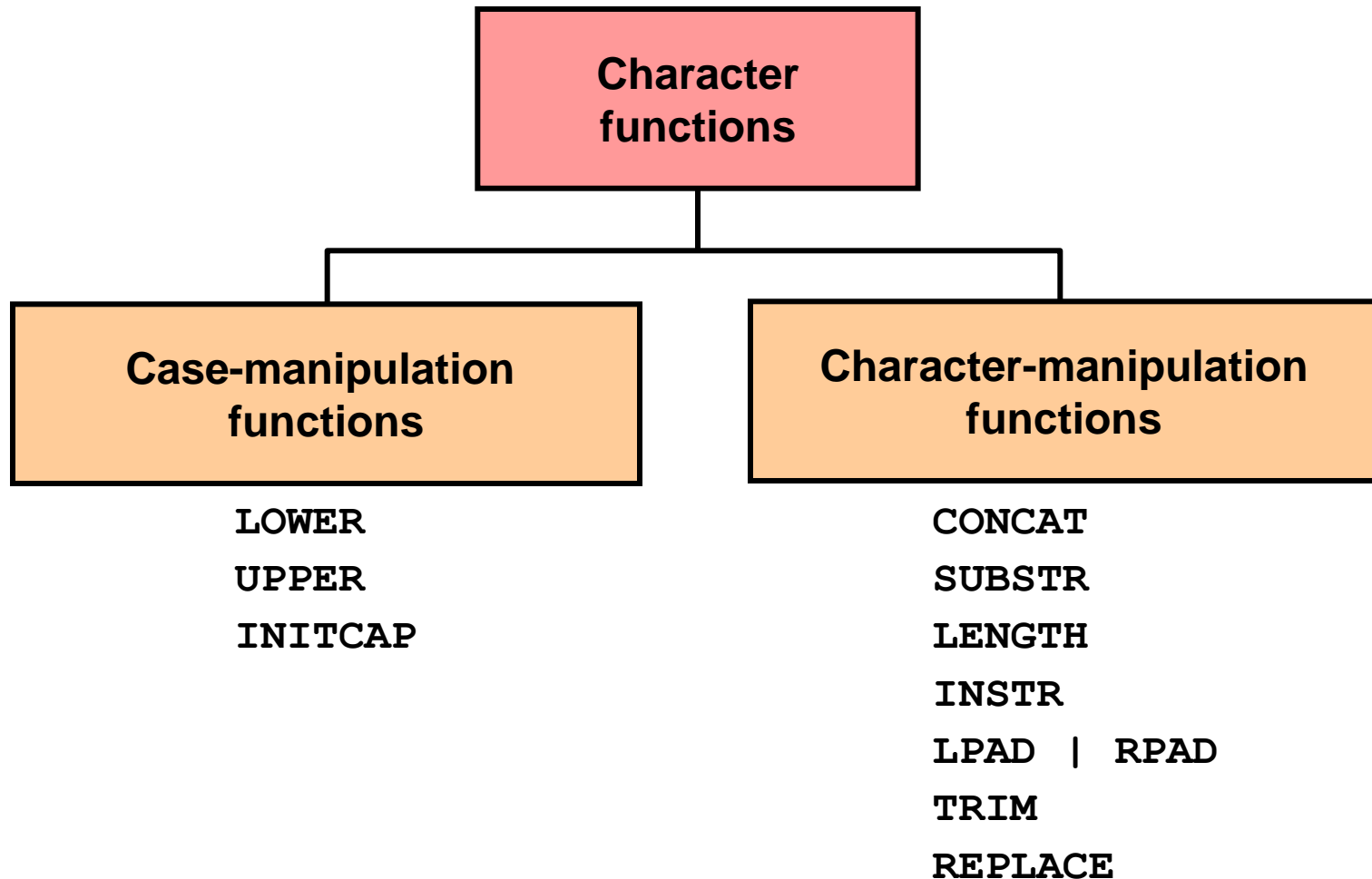
- Manipulate data items
- Accept arguments and return one value
- Act on each row that is returned
- Return one result per row
- May modify the data type
- Can be nested
- Accept arguments that can be a column or an expression

```
function_name [(arg1, arg2, ...)]
```

Single-Row Functions



Character Functions



Case-Manipulation Functions

These functions convert case for character strings:

Function	Result
<code>LOWER('SQL Course')</code>	<code>sql course</code>
<code>UPPER('SQL Course')</code>	<code>SQL COURSE</code>
<code>INITCAP('SQL Course')</code>	<code>Sql Course</code>

Using Case-Manipulation Functions

Display the employee number, name, and department number for employee Higgins:

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

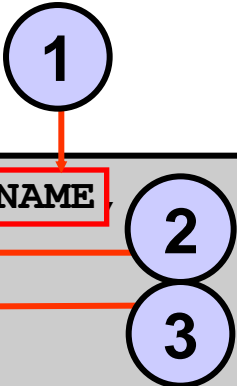
Character-Manipulation Functions

These functions manipulate character strings:


Function	Result
<code>CONCAT('Hello', 'World')</code>	HelloWorld
<code>SUBSTR('HelloWorld',1,5)</code>	Hello
<code>LENGTH('HelloWorld')</code>	10
<code>INSTR('HelloWorld', 'W')</code>	6
<code>LPAD(salary,10,'*')</code>	*****24000
<code>RPAD(salary, 10, '*')</code>	24000*****
<code>REPLACE('JACK and JUE','J','BL')</code>	BLACK and BLUE
<code>TRIM('H' FROM 'HelloWorld')</code> oracle	elloWorld
<code>TRIM('SQL Tutorial')</code> MySQL	SQL Tutorial

Using the Character-Manipulation Functions

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH(last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, length(job_id)-2, length(job_id)) = 'REP';
```



EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2



Now review all Single Row Functions Supported by MySQL

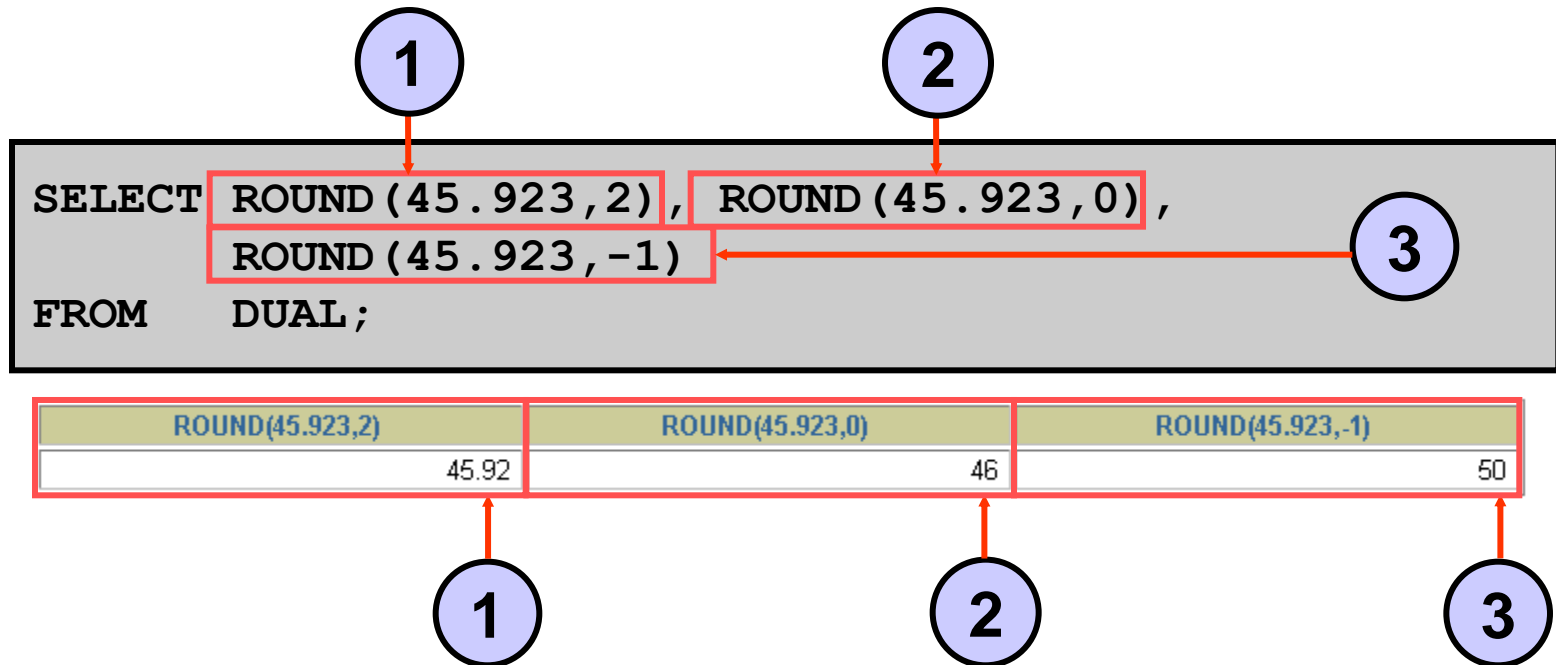
https://www.w3schools.com/sql/sql_ref_mysql.asp

Number Functions

- **ROUND:** Rounds value to specified decimal
- **TRUNC:** Truncates value to specified decimal
- **MOD:** Returns remainder of division

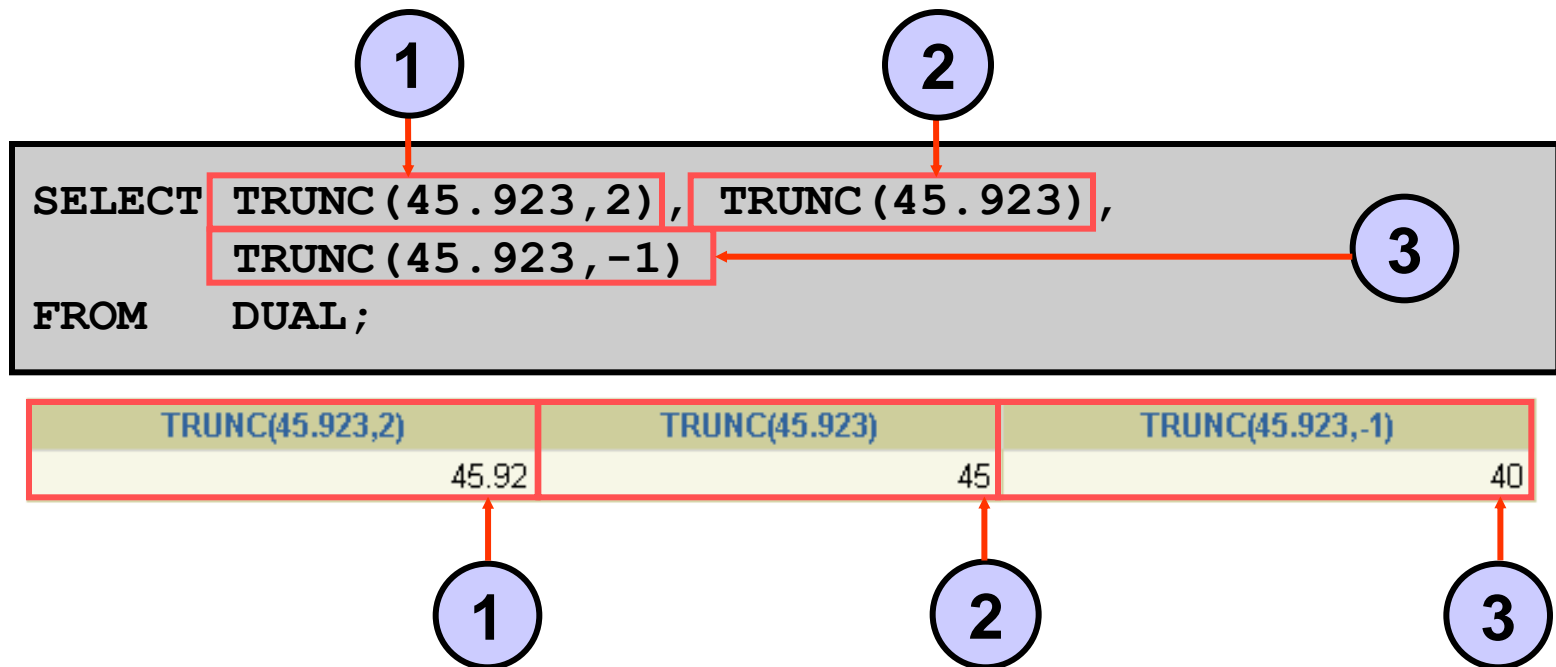
Function	Result
ROUND (45 . 926 , 2)	45 . 93
TRUNC (45 . 926 , 2)	45 . 92
MOD (1600 , 300)	100

Using the ROUND Function



DUAL is a dummy table that you can use to view results from functions and calculations.

Using the TRUNC Function



Using the MOD Function

For all employees with job title of Sales Representative, calculate the remainder of the salary after it is divided by 5,000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

Working with Dates

- The Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds.
- The default date display format is DD-MON-RR.
 - Enables you to store 21st-century dates in the 20th century by specifying only the last two digits of the year
 - Enables you to store 20th-century dates in the 21st century in the same way

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date < '01-FEB-88';
```

LAST_NAME	HIRE_DATE
King	17-JUN-87
Whalen	17-SEP-87

Using the SYSDATE Function

SYSDATE is a function that returns:

- **Date**
- **Time**

In MySQL, we use sysdate()

```
SELECT sysdate()  
FROM   customer;
```

`sysdate()`

2023-12-04 09:58:04

Arithmetic with Dates

- **Add or subtract a number to or from a date for a resultant date value.**
- **Subtract two dates to find the number of days between those dates.**
- **Add hours to a date by dividing the number of hours by 24.**

Using Arithmetic Operators with Dates

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM employees  
WHERE department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

Note: Sysdate contains date and time! What if you do not have time information in your table?

Orders

OrderID	CustomerID	ProductID	Quantity	OrderDate
1000	5	1	1	2023-03-20
1001	3	4	3	2023-03-21
1002	2	3	2	2023-03-22
1003	1	2	2	2023-03-23
1004	1	2	1	2023-03-23

Prefer DateDIFF function for MySQL!

```
SELECT orderid, DATEDIFF(current_Date(),orderDate)/30  
AS Months FROM orders;
```

orderid	Months
1000	8.6333
1001	8.6000
1002	8.5667
1003	8.5333
1004	8.5333

OrderID	CustomerID	ProductID	Quantity	OrderDate
1000	5	1	1	2023-03-20
1001	3	4	3	2023-03-21
1002	2	3	2	2023-03-22
1003	1	2	2	2023-03-23
1004	1	2	1	2023-03-23

```
SELECT orderid, DATEDIFF(current_Date(),orderDate)/7
AS Weeks FROM orders;
```

orderid	Weeks
1000	37.0000
1001	36.8571
1002	36.7143
1003	36.5714
1004	36.5714

Date Functions

Function	Result
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
ROUND	Round date
TRUNC	Truncate date

Using Date Functions

Function	Result
MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94')	19.6774194
ADD_MONTHS ('11-JAN-94', 6)	'11-JUL-94'
NEXT_DAY ('01-SEP-95', 'FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

Using Date Functions

Assume SYSDATE = '25-JUL-03':

Function	Result
ROUND (SYSDATE , 'MONTH')	01-AUG-03
ROUND (SYSDATE , 'YEAR')	01-JAN-04
TRUNC (SYSDATE , 'MONTH')	01-JUL-03
TRUNC (SYSDATE , 'YEAR')	01-JAN-03

MySQL Date Examples (ADDDATE)

```
SELECT orderid, orderDate,  
ADDDATE(orderDate, INTERVAL 10 DAY) as NEWDate  
FROM    orders;
```

orderid	orderDate	NEWDate
1000	2023-03-20	2023-03-30
1001	2023-03-21	2023-03-31
1002	2023-03-22	2023-04-01
1003	2023-03-23	2023-04-02
1004	2023-03-23	2023-04-02

MySQL Date Examples (SUBDATE)

```
SELECT orderid, orderDate,  
SUBDATE(orderDate, INTERVAL 10 DAY) as NEWDate  
FROM    orders;
```

orderid	orderDate	NEWDate
1000	2023-03-20	2023-03-10
1001	2023-03-21	2023-03-11
1002	2023-03-22	2023-03-12
1003	2023-03-23	2023-03-13
1004	2023-03-23	2023-03-13

MySQL Date Examples (Year, Month, Day, Weekofyear)

```
SELECT orderid, orderDate,  
YEAR(orderDate) as year,  
Month(orderDate) as month, DAY(orderDate) as day,  
WEEKOFYEAR(orderDate) as weekofyear  
FROM    orders;
```

orderid	orderDate	year	month	day	weekofyear
1000	2023-03-20	2023	3	20	12
1001	2023-03-21	2023	3	21	12
1002	2023-03-22	2023	3	22	12
1003	2023-03-23	2023	3	23	12
1004	2023-03-23	2023	3	23	12

Now review all Date Functions Supported by MySQL

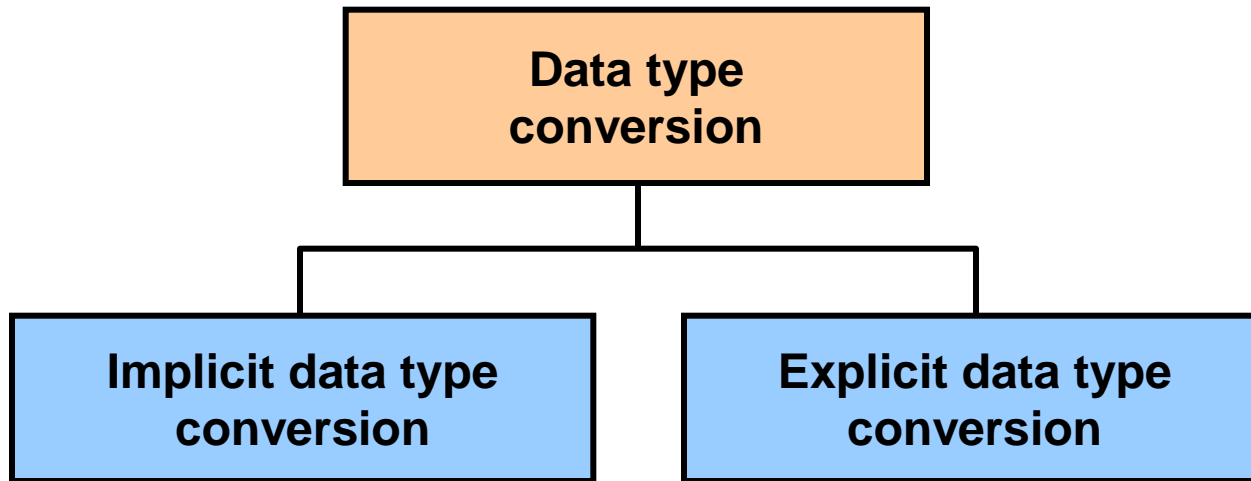
https://www.w3schools.com/sql/sql_ref_mysql.asp

Practice 3: Overview of Part 1

This practice covers the following topics:

- **Writing a query that displays the current date**
- **Creating queries that require the use of numeric, character, and date functions**
- **Performing calculations of years and months of service for an employee**

Conversion Functions



Implicit Data Type Conversion

For assignments, the Oracle server can automatically convert the following:

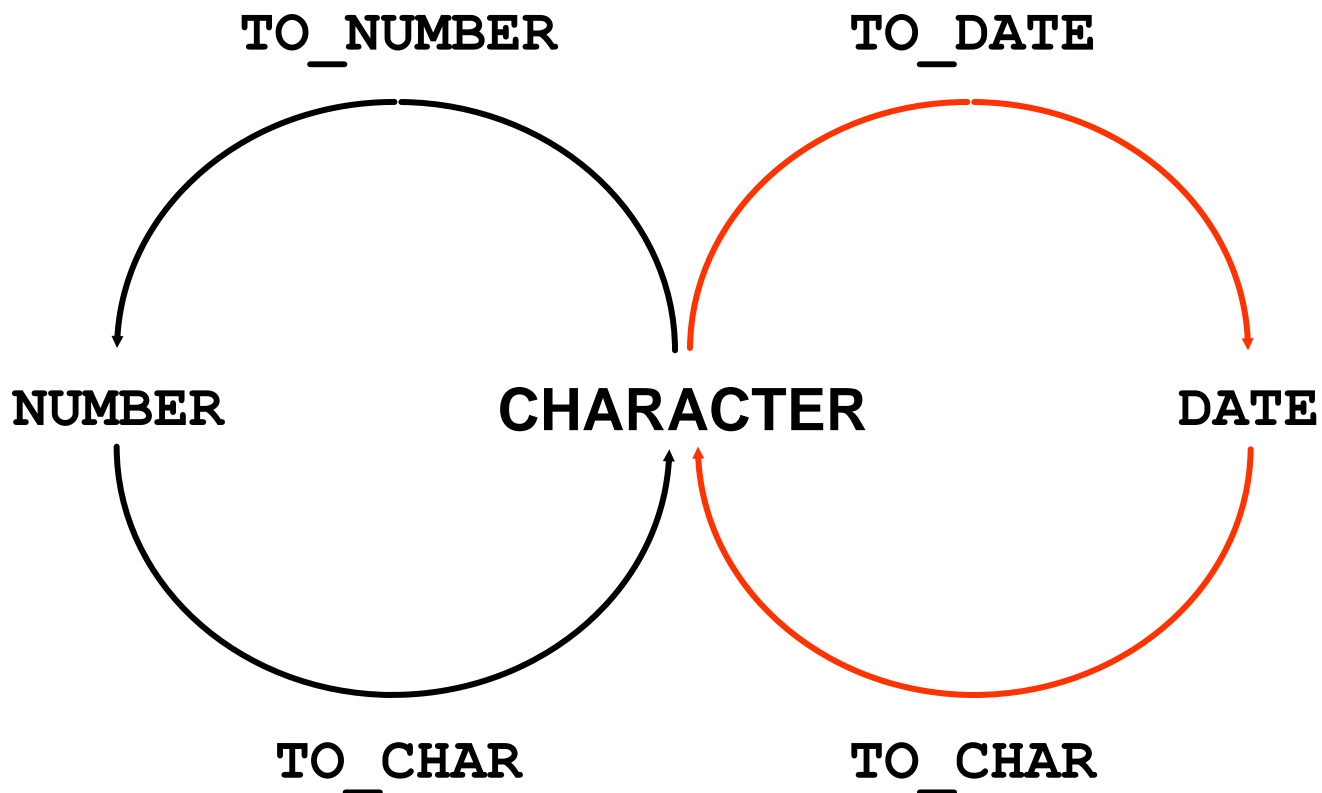
From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Implicit Data Type Conversion

For expression evaluation, the Oracle Server can automatically convert the following:

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

Explicit Data Type Conversion



Using the TO_CHAR Function with Dates

```
TO_CHAR(date, 'format_model')  

```

The format model:

- Must be enclosed by single quotation marks
- Is case-sensitive
- Can include any valid date format element
- Has an `fm` element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

Elements of the Date Format Model

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

Elements of the Date Format Model

- Time elements format the time portion of the date:

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

- Add character strings by enclosing them in double quotation marks:

DD "of" MONTH	12 of OCTOBER
---------------	---------------

- Number suffixes spell out numbers:

ddspth	fourteenth
--------	------------

Using the TO_CHAR Function with Dates

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
       AS HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999

...

20 rows selected.

Using the TO_CHAR Function with Numbers

```
TO_CHAR(number, 'format_model') ddspth
```

These are some of the format elements that you can use with the TO_CHAR function to display a number value as a character:

Element	Result
9	Represents a number
0	Forces a zero to be displayed
\$	Places a floating dollar sign
£	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as thousands indicator

Using the TO_CHAR Function with Numbers

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM   employees  
WHERE  last_name = 'Ernst';
```

SALARY
\$6,000.00

Using the TO_NUMBER and TO_DATE Functions

- Convert a character string to a number format using the TO_NUMBER function:

```
TO_NUMBER(char[, 'format_model'])
```

- Convert a character string to a date format using the TO_DATE function:

```
TO_DATE(char[, 'format_model'])
```

- These functions have an **fx** modifier. This modifier specifies the exact matching for the character argument and date format model of a TO_DATE function.

RR Date Format

Current Year	Specified Date	RR Format	YY Format
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		If the specified two-digit year is:	
		0–49	50–99
If two digits of the current year are:	0–49	The return date is in the current century	The return date is in the century before the current one
	50–99	The return date is in the century after the current one	The return date is in the current century

Example of RR Date Format

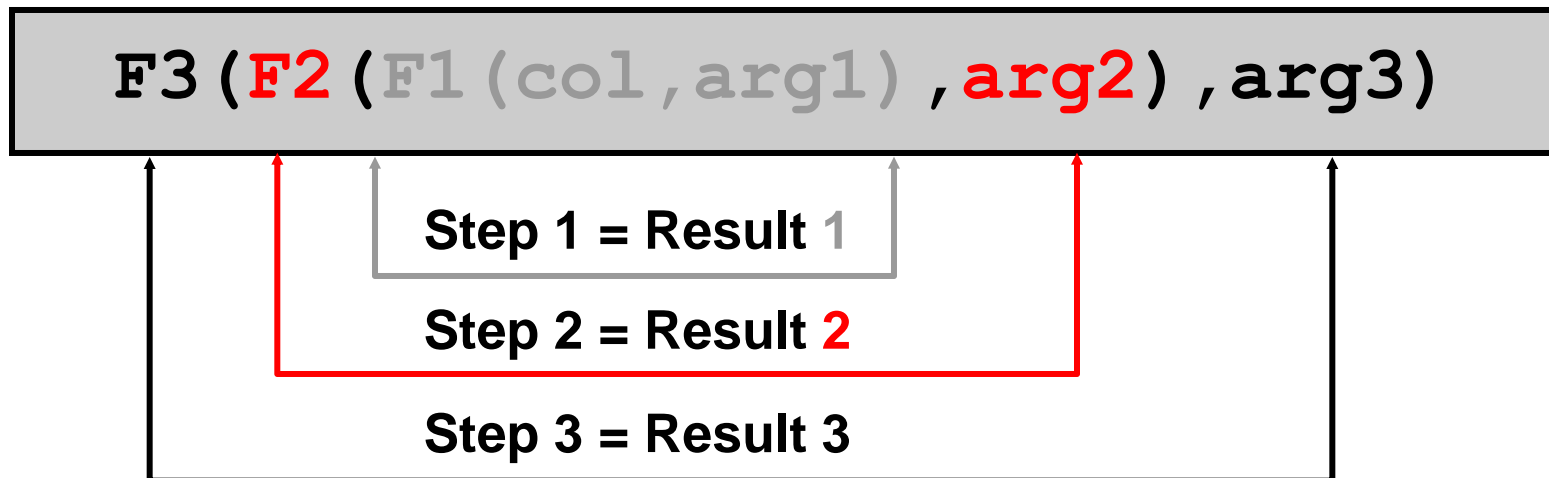
To find employees hired prior to 1990, use the RR date format, which produces the same results whether the command is run in 1999 or now:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')  
FROM employees  
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIR
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

Nesting Functions

- Single-row functions can be nested to any level.
- Nested functions are evaluated from deepest level to the least deep level.



Nesting Functions

```
SELECT last name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
FROM   employees  
WHERE  department_id = 60;
```

LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8
Hunold	HUNOLD_US
Ernst	ERNST_US
Lorentz	LORENTZ_US