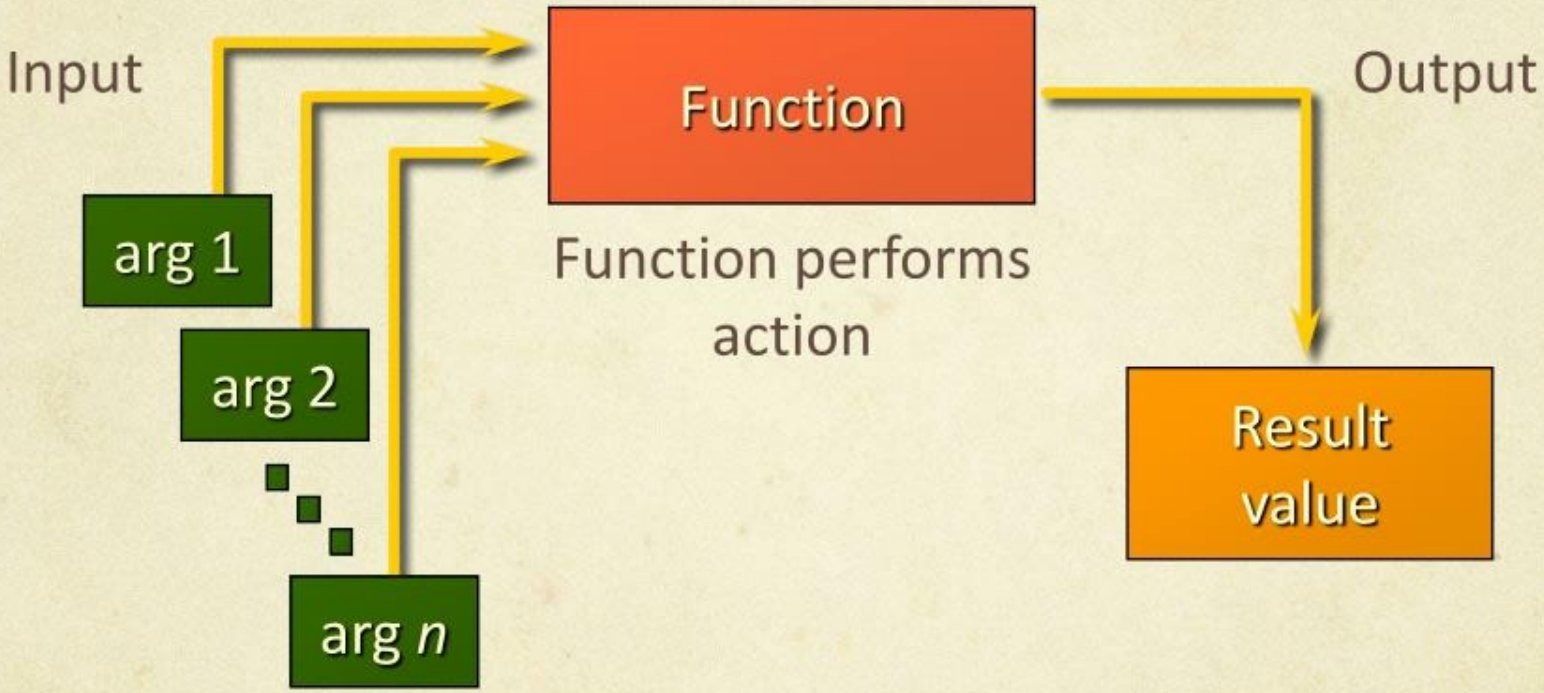
Single-Row Functions

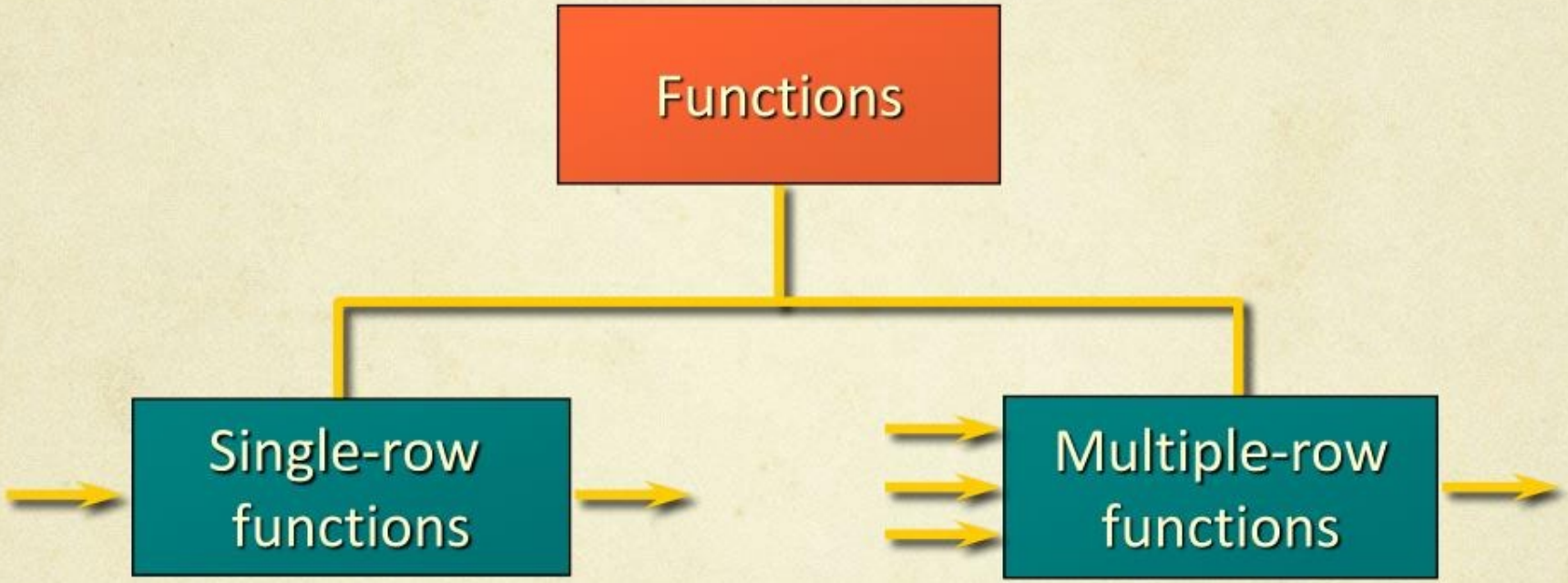
Objectives

- After completing this lesson, you should be able to do the following:
  - Describe various types of functions available in MySQL
  - Use character, number, and date functions in SELECT statements
  - Describe the use of conversion functions

MySQL Functions



Two Types of MySQL Functions

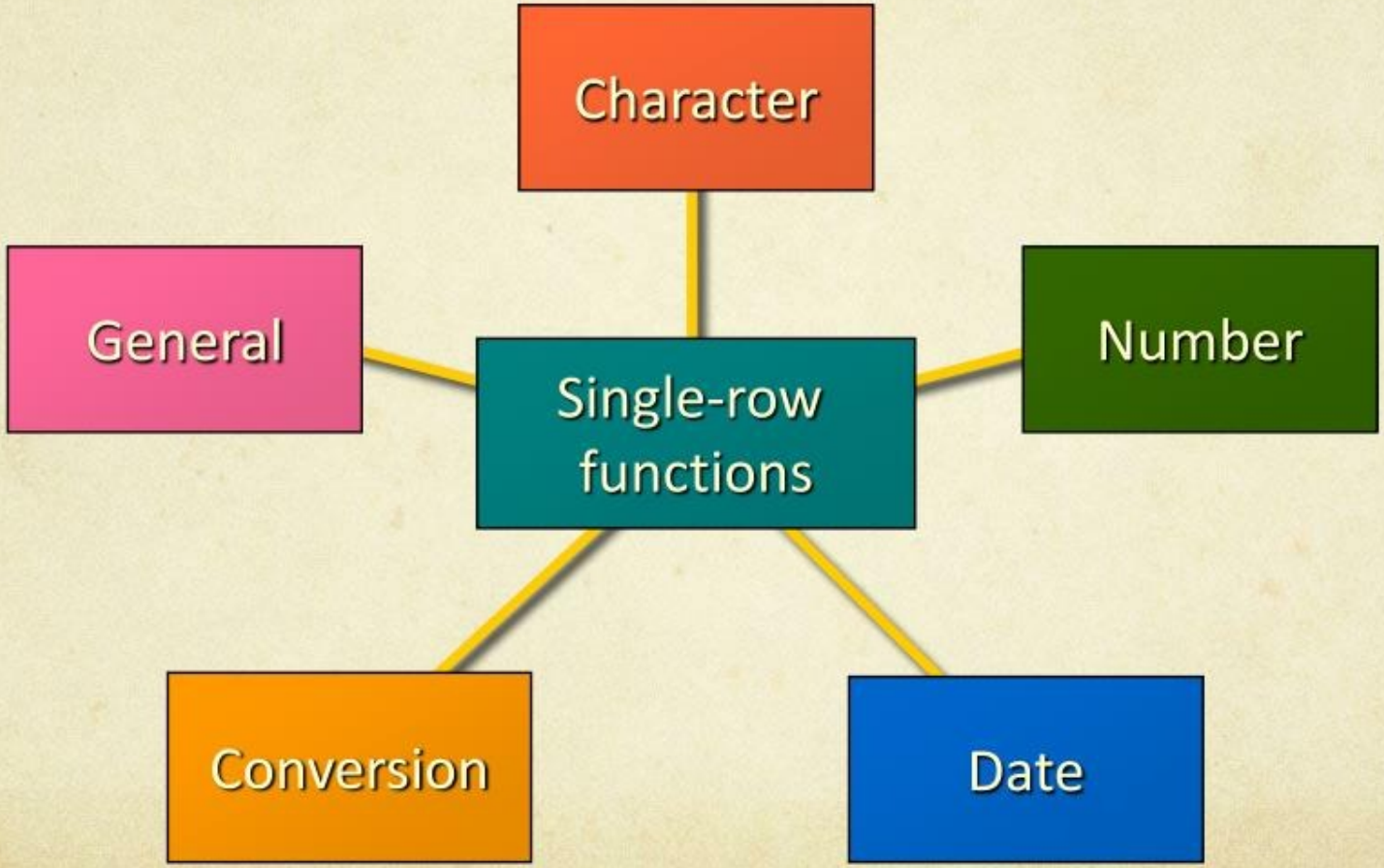


Single-Row Functions

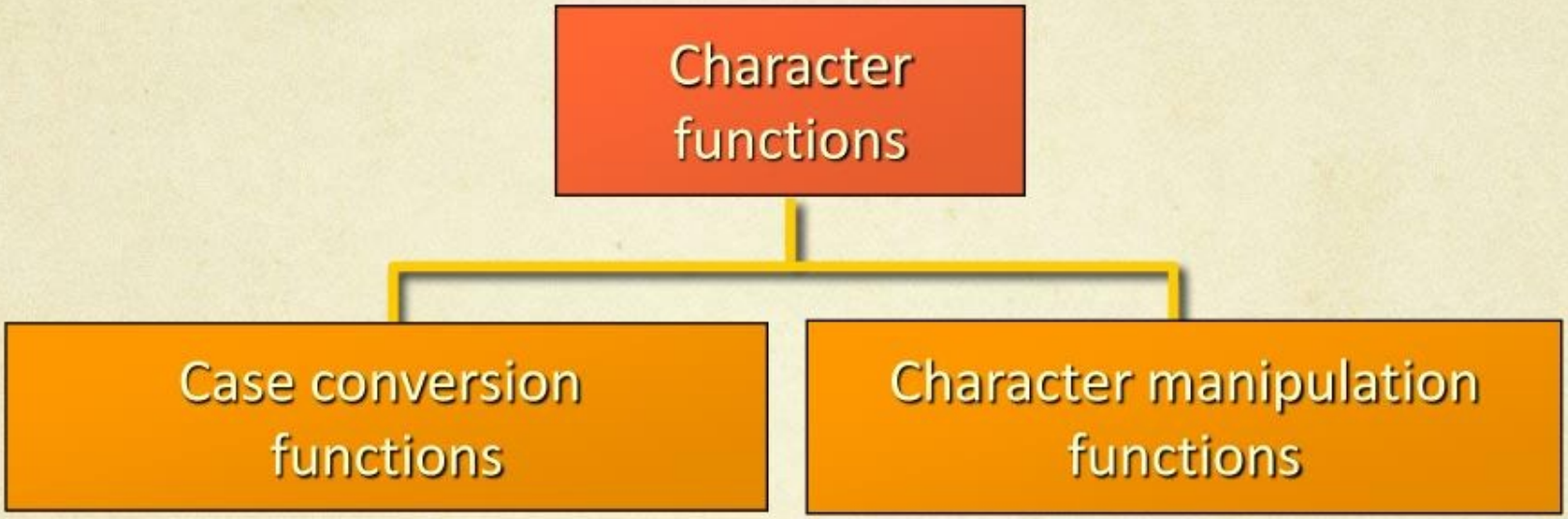
- Manipulate data items
- Accept arguments and return one value
- Act on each row returned
- Return one result per row
- May modify the datatype
- Can be nested

```
function_name (column|expression, [arg1, arg2,...])
```

Single-Row Functions



Character Functions



LOWER

UPPER

CONCAT

SUBSTR

LENGTH

INSTR

LPAD & RPAD

TRIM, LTRIM, & RTRIM

Case Conversion Functions

- Convert case for character strings

Function	Result
LOWER (‘MySQL Course’)	mysql course
UPPER (‘MySQL Course’)	MYSQL COURSE

Using Case Conversion Functions

- Display the employee number, name, and department number for employee Blake (All Caps in DB).

```
MySQL>SELECT  employee_nbr, name, dept_nbr
    ->FROM    employee
    ->WHERE   name = 'Blake';
no rows selected
```

```
MySQL>SELECT  employee_nbr, name, dept_nbr
    ->FROM    employee
    ->WHERE   name = UPPER('Blake');
```

```
employee_nbr name          dept_nbr
------------ ---------- ----------
        7698 BLAKE              30
```

Character Manipulation Functions

- Manipulate character strings

Function	Result
CONCAT('Good', 'String')	GoodString
SUBSTR('String',1,3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	******5000
TRIM('S' FROM 'SSMITH')	MITH
RTRIM('JONES ')	JONES

Using the Character Manipulation Functions

```
MySQL>SELECT name,  CONCAT (name, job),  LENGTH(name),
    ->        INSTR(name, 'a')
    ->FROM    employee
    ->WHERE   SUBSTR(job,1,5) = 'Sales';
```

name	CONCAT(name,job)	LENGTH(name)	INSTR(name,'a')
Martin	MartinSalesman	6	2
Allen	AllenSalesman	5	1
Turner	TurnerSalesman	6	0
Ward	WardSalesman	4	2

Number Functions

- ROUND: Rounds value to specified decimal

  ROUND(45.926, 2) → 45.93

- TRUNCATE: Truncates value to specified decimal

  TRUNCATE(45.926, 2) → 45.92

- MOD: Returns remainder of division

  MOD(1600, 300) → 100

Using the ROUND Function

```
MySQL>SELECT ROUND(45.923,2), ROUND(45.923,0),
    ->       ROUND(45.923,-1)
    ->FROM   DUAL;
```

```
ROUND(45.923,2) | ROUND(45.923,0) | ROUND(45.923,-1)
----------------|----------------|-----------------
          45.92 |             46 |              50
```

Using the TRUNCATE Function

```
MySQL>SELECT   TRUNCATE(45.923,2), TRUNCATE(45.923),
    ->         TRUNCATE(45.923,-1)
    ->FROM     DUAL;
```

TRUNCATE(45.923,2)	TRUNCATE(45.923)	TRUNCATE(45.923,-1)
45.92	45	40

Using the MOD Function

- Calculate the remainder of the ratio of salary to commission for all employees whose job title is salesman.

```
MySQL>SELECT  name, salary, commission, MOD(salary, commission)
    ->FROM    employee
    ->WHERE   job = 'Salesman';
```

name	Salary	Commission	MOD(Salary,Commission)
Martin	1250	1400	1250
Allen	1600	300	100
Turner	1500	0	1500
Ward	1250	500	250

Working with Dates

- MySQL stores dates in an internal numeric format: century, year, month, day, hours, minutes, seconds.
- The default date format is YYYY-MM-DD.
- NOW() is a function returning date and time.
- CURDATE() returns the current date.
- DUAL is a dummy table used to view NOW() or CURDATE().

Arithmetic with Dates

- DATE_ADD(date, INTERVAL expr unit),
  DATE_SUB(date, INTERVAL expr unit)
- These functions perform date arithmetic. The date argument specifies the starting date.
  - expr is an expression specifying the interval value to be added or subtracted from the starting date.
    - expr is a string; it may start with a "-" for negative intervals.
  - unit is a keyword indicating the units in which the expression should be interpreted.
- The INTERVAL keyword and the unit specifier are not case sensitive.

Arithmetic with Dates

- The following table shows the expected form of the expr argument for each unit value.

Unit Value	Expected expr Format
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES’
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS’

Arithmetic with Dates

- mysql> SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY) FROM dual;

  2008-02-02

- mysql> SELECT DATE_SUB('2008-01-02', INTERVAL 3 MONTH) FROM dual;

  2007-10-02

- mysql> SELECT DATE_ADD('2008-01-02', INTERVAL 3 MONTH) FROM dual;

  2008-04-02

Arithmetic with Dates

- DATEDIFF(expr1, expr2)
  - DATEDIFF() returns expr1 – expr2 expressed as a value in days from one date to the other.
  - expr1 and expr2 are date or date-and-time expressions.
  - Only the date parts of the values are used in the calculation.

Arithmetic with Dates

```
MySQL>SELECT name, DATEDIFF(CURDATE(), hire_date)/7 Weeks
    ->FROM   employee
    ->WHERE  dept_nbr = 10;
```

```
name          Weeks
--------- -----------
King       391.1429
Clark      423.7143
MILLER     402.1429
```

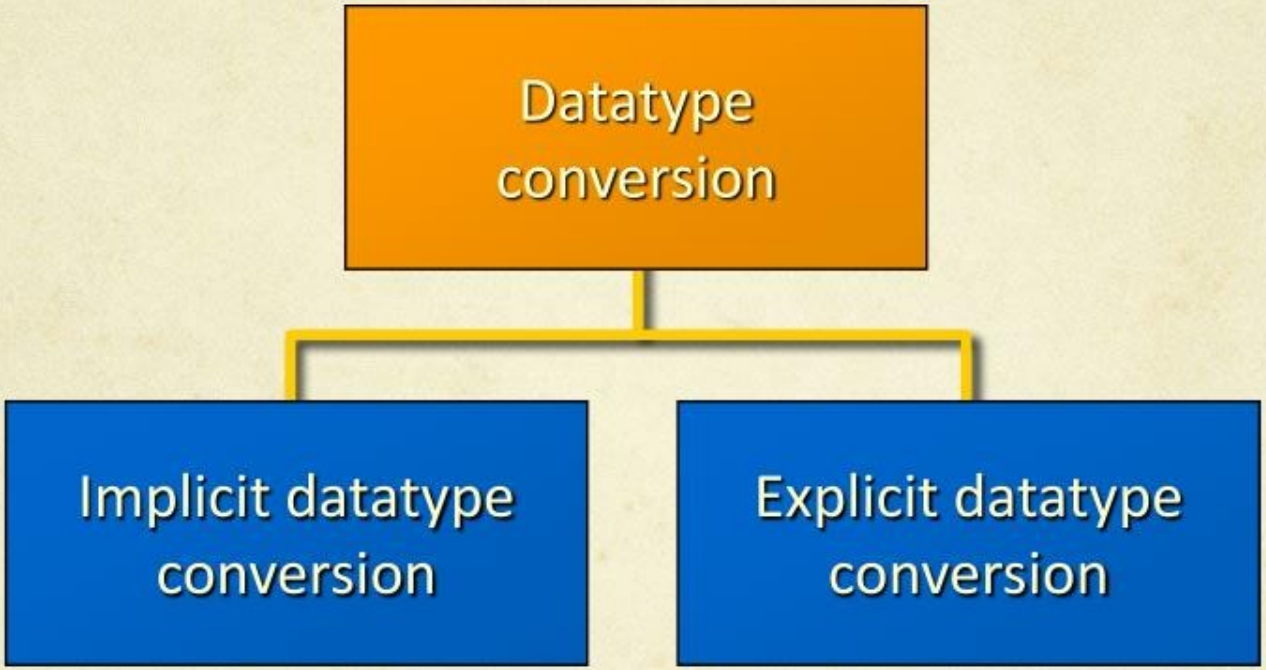
Date Functions

Function	Description
PERIOD_DIFF(*P1, P2*)	Returns the number of months between periods *P1* and *P2*. *P1* and *P2* should be in the format YYMM or YYYYMM.
EXTRACT(*unit* FROM *date*)	Extracts parts from the date rather than performing date arithmetic. Uses the same kinds of unit specifiers as DATE_ADD().
LAST_DAY(*date*)	Takes a date or datetime value and returns the corresponding value for the last day of the month.
DAYNAME(*date*)	Returns the name of the weekday for *date*

Using Date Functions

- PERIOD_DIFF(200812, 200706) → 18 Months
- EXTRACT(MONTH FROM '2008-02-15') → 2
- LAST_DAY('2008-02-15') → '2008-02-29 '
- DAYNAME('2008-02-15') → Friday

Conversion Functions



Implicit Datatype Conversion

- For assignments, the MySQL Server can automatically convert the following:

From	To
VARCHAR or CHAR	NUMBER
VARCHAR or CHAR	DATE
INTEGER & DECIMAL	VARCHAR
DATE	VARCHAR

Implicit Datatype Conversion

- For expression evaluation, the MySQL Server can automatically convert the following:

From	To
VARCHAR or CHAR	NUMBER
VARCHAR or CHAR	DATE

DATE_FORMAT Function with Dates

```
DATE_FORMAT(date, 'format')
```

- Formats the *date* value according to the *format* string.
- The format model:
  - Must be enclosed in single quotation marks and is case sensitive
  - Can be any valid date format element
  - Is separated from the date by a comma

Elements of Date Format Model

Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, …)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%j	Day of year (001..366)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss followed by AM or PM
%T	Time, 24-hour (hh:mm:ss)

Elements of Date Format Model

Specifier	Description
%U	Week (00..53), where Sunday is the first day of the week
%u	Week (00..53), where Monday is the first day of the week
%V	Week (01..53), where Sunday is the first day of the week; used with %X
%v	Week (01..53), where Monday is the first day of the week; used with %x
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)

Elements of Date Format Model

- Add character strings by including them between the quotation marks.

'%D of %M'	12th of OCTOBER

Using DATE_FORMAT Function with Dates

```
MySQL>SELECT  name,
    ->        DATE_FORMAT(hire_date,'%D of %M') "Hire Date"
    ->FROM    employee;
```

```
name        Hire Date
--------- -------------------
Smith       17th of December
Allen       20th of February
Ward        22nd of February
Jones       22nd of April
Martin      28th of September
Blake       1st of May
Clark       9th of June
Scott       9th of December
...
14 rows in set (0.00sec)
```

Using DATE_FORMAT Function with Dates

```
MySQL>SELECT  name,
    ->        DATE_FORMAT(hire_date,'%M %d,%Y') "Hire Date"
    ->FROM    employee;
```

```
name        Hire Date
--------- --------------------
Smith       December 17, 2009
Allen       February 20, 2001
Ward        February 22, 2001
Jones       April 02, 2001
Martin      September 28, 2001
Blake       May 01, 2001
Clark       June 09, 2001
Scott       December 09, 2001
...
14 rows in set (0.00 sec)
```

Using DATE_FORMAT Function with Dates

```
MySQL>SELECT  name,
    ->        DATE_FORMAT(hire_date,'%M %D %Y') "Hire Date"
    ->FROM    employee;
```

```
name        Hire Date
--------- --------------------
Smith       December 17th 2009
Allen       February 20th 2001
Ward        February 22nd 2001
Jones       April 2nd 2001
Martin      September 28th 2001
Blake       May 1st 2001
Clark       June 9th 2001
Scott       December 9th 2001
...
14 rows in set (0.00 sec)
```

FORMAT Function with Numbers

```
FORMAT(X, D)
```

- Formats the number *X* to a format like '#,###,###.##', rounded to *D* decimal places, and returns the result as a string. If *D* is 0, the result has no decimal point or fractional part.

Using FORMAT Function with Numbers

```
MySQL> SELECT CONCAT('$',FORMAT(salary,2)) Salary
    -> FROM   EMPLOYEE
    -> WHERE  name = 'Scott';
```

```
Salary
----------
$3,000.00
```

IFNULL Function

- Converts null to an actual value
  - Datatypes that can be used are date, character, and number.
  - Datatypes must match
    - IFNULL(comm,0)
    - IFNULL(hiredate,'01-JAN-97')
    - IFNULL(job,'No Job Yet')

IFNULL Function

- IFNULL() returns expr1; otherwise it returns expr2. IFNULL() returns a numeric or string value, depending on the context in which it is used.

```
MySQL>SELECT  IFNULL(1,0);
    -> 1
```

```
MySQL>SELECT  IFNULL(NULL,10);
    -> 10
```

```
MySQL>SELECT  IFNULL(1/0,10);
    -> 10
```

```
MySQL>SELECT  IFNULL(1/0,'yes');
    -> 'yes'
```

Using the IFNULL Function

```
MySQL>SELECT   name, salary, commission,
               (salary*12)+IFNULL(commission,0)
    ->FROM     employee;
```

```
name        Salary Commission (Salary*12)+IFNULL(Commission,0)
---------- ------ --------- -------------------------------
King          5000                                     60000
Blake         2850                                     34200
Clark         2450                                     29400
Jones         2975                                     35700
Martin        1250      1400                           16400
Allen         1600       300                           19500
...
14 rows selected.
```

CASE Control Flow Function

- Facilitates conditional inquiries by doing the work of a CASE or IF-THEN-ELSE statement

```
CASE value WHEN [compare_value] THEN result
[WHEN [compare_value] THEN result…] [ELSE result] END
```

```
CASE WHEN [condition] THEN result
[WHEN [condition] THEN result…] [ELSE result] END
```

- The first version returns the result where value = compare_value.
- The second version returns the result for the first condition that is true.
- If there was no matching result value, the result after ELSE is returned, or NULL if there is no ELSE part.

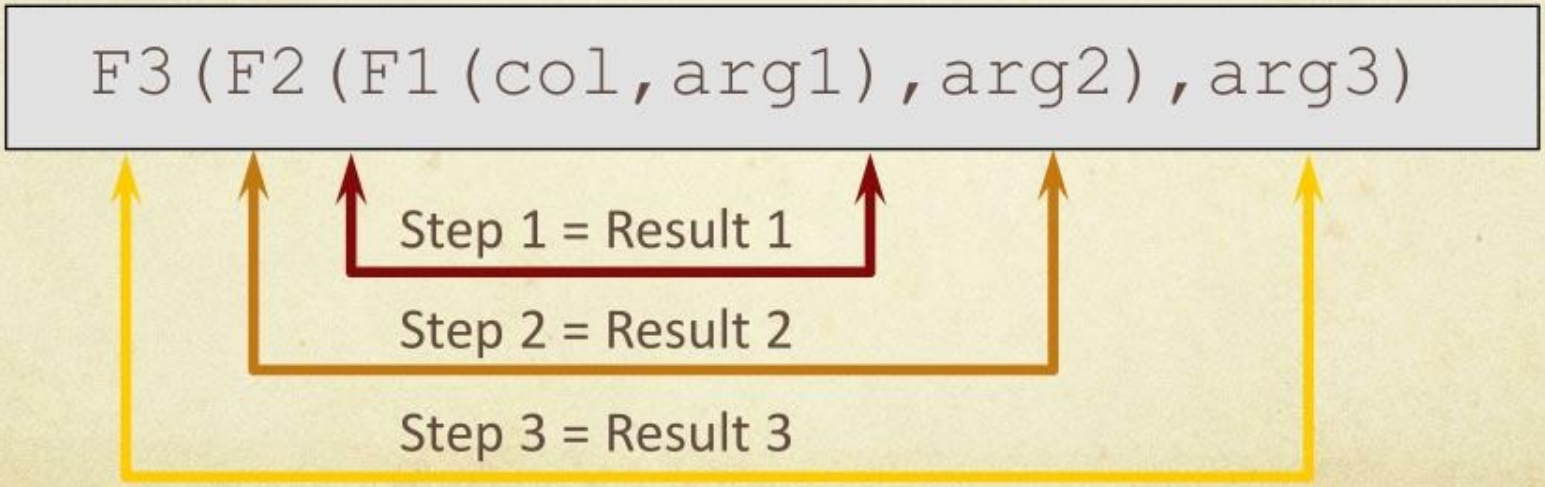
Using the CASE Function

```
MySQL>SELECT job, salary,
    ->        CASE job
    ->        WHEN 'Analyst' THEN salary*1.1
    ->        WHEN 'Clerk'    THEN salary*1.15
    ->        WHEN 'Manager' THEN salary*1.20
    ->        ELSE  salary END AS Revised_Salary
    ->FROM    employee;
```

```
job           salary Revised_Salary
--------- --------- ---------------
President       5000            5000
Manager         2850            3420
Manager         2450            2940
...
14 rows selected.
```

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

```
MySQL>SELECT  name,
    ->        IFNULL(TO_CHAR(mgr),'No Manager')
    ->FROM    employee
    ->WHERE   manager IS NULL;
```

```
name       IFNULL(TO_CHAR(MGR),'NOManager')
---------  --------------------------------
King        No Manager
```

Summary

- Use functions to do the following:
  - Perform calculations on data
  - Modify individual data items
  - Manipulate output for groups of rows
  - Alter date formats for display
  - Convert column datatypes