SELECT Orders.OrderID, persons.FirstName  
FROM Orders  
INNER JOIN persons ON Orders.PersonID = Persons.ID;

To work with inner join in SQL you should have table relation using primary key and foreign key.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Select Orders.OrderID, persons.FirstName  
FROM  Persons,orders

Where Orders.PersonID = Persons.ID;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rounding the dates in SQL;

SELECT date\_format(now(),'%Y-%m'); -- round to the month

SELECT date\_format(now(),'%Y-%m-%d'); -- round to the day

SELECT date\_format(now(),'%Y-%m-%d %H'); -- round to the hour

SELECT date\_format(now(),'%Y-%m-%d %H:%i'); -- round to the minute

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

select dayofmonth(hire\_date) as day, month(hire\_date) as month, year(hire\_date) as year from employee;

Write a query that display all employee who joined in march;

Select datediff(now() , hire\_date) as experience from employee;

select date\_add(hire\_date,interval 11 month) from employee;

select date\_sub(hire\_date,interval 11 month) from employee;

Write an SQL query which lists the hire dates of all employees along the date of their first work appraisal (One year from the hiredate)

The function LAST\_DAY returns the date of the last day of the month given in a date.

The syntax is LAST\_DAY(date\_value).

following query lists all sales transactions that were made in the last 20 days of a month:

SELECT \*FROM SALES WHERE SALE\_DATE >= LAST\_DAY(SALE\_DATE)-20;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Single row numeric functions.Numeric functions take one numeric parameter and return one value.

Function Description

ABS Returns the absolute values of a numberSyntax: ABS(numeric\_value)

ROUND

Rounds a value to a specified precision(number of digits)

Syntax: ROUND(numeric\_value, p) where p = precision

TRUNCATET runcates a value to a specified precision(number of digits)

Syntax: TRUNC(numeric\_value, p) where p = precision

MODReturns the remainder of division

Syntax: MOD(m.n) where m is divided by n

The following example displays the individual LINE\_PRICE from the sales line table, rounded to one and zero places and truncated where the quantity of tickets purchased on that line is greater than 2

SELECT LINE\_PRICE, ROUND(LINE\_PRICE,1) AS “LINE\_PRICE1”,ROUND(LINE\_PRICE,0) AS “LINE\_PRICE1”,TRUNCATE(LINE\_PRICE,0) AS “TRUNCATED VALUE”FROM SALES\_LINEWHERE LINE\_QTY > 2;

String manipulation functions are amongst the most-used functions in

CONCATConcatenates data from two different character columns and returns a single column.

Syntax: CONCAT(strg\_value, strg\_value)

UPPER/LOWER Returns a string in all capital or all lowercase letters

Syntax: UPPER(strg\_value) ,

LOWER(strg\_value)

SUBSTRReturns a substring or part of a given string parameter

Syntax: SUBSTR(strg\_value, p, l) where p = start position and l = length of characters

LENGTH Returns the number of characters in a string value

Syntax: LENGTH(strg\_value)

SELECT CONCAT(EMP\_LNAME ,EMP\_FNAME) AS NAMEFROM EMPLOYEE;

UPPER/LOWERThe following query lists all employee last names in all capital letters and all first names in all lowercase letters.

SELECT CONCAT(UPPER(EMP\_LNAME),LOWER(EMP\_FNAME)) AS NAME

The following example lists the first three characters of all the employees’ first name.

SELECT EMP\_PHONE, SUBSTR(EMP\_FNAME,1,3)FROM EMPLOYEE;

**String Functions Exercise:**  
Write a query which generates a list of employee user IDs, using the day of the month they were born and the first six characters of last name in UPPER case.

SELECT ATTRACT\_NAME, LENGTH(ATTRACT\_NAME) AS NAMESIZEFROM ATTRACTIONORDER BY NAMESIZE DESC;

Conversion FunctionsConversion functions allow you to take a value of a given data type and convert it to the equivalent value in another data type. In MySQL, some conversions occur implicitly. For example, MySQL automatically converts numbers to strings when needed, and vice versa.So if you enter the following query:SELECT 10 + ‘10’MySQL would give you an answer of 20 as it would automatically convert the string containing ‘10’ into the number 10

[22](https://slideplayer.com/slide/13724066/85/images/22/Conversion+Functions.jpg) Conversion FunctionsIf you want to explicitly convert a number to a string then you can use either the CAST or CONCAT function. However MySQL 5.0 recommends only the CAST function is usedExample:SELECT 10, CAST(10 AS CHAR);

[23](https://slideplayer.com/slide/13724066/85/images/23/Conversion+Functions+IFNULL%28expr1%2Cexpr2%29+IFNULL.jpg) **Conversion Functions IFNULL(expr1,expr2) IFNULL**  
The IFNULL function lets you substitute a value when a null value is encountered in the results of a query. The syntax is:IFNULL(expr1,expr2)If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2. It is equivalent to Oracle’s NVL function.It is useful for avoiding errors caused by incorrect calculation when one of the arguments is null. IFNULL(Expe1, 0)

Case 1:

Case 2:

SELECT OrderID, Quantity,

CASE WHEN Quantity > 30 THEN 'The quantity is greater than 30'

WHEN Quantity = 30 THEN 'The quantity is 30'

ELSE 'The quantity is under 30'

END AS QuantityText

FROM OrderDetails

Select OrderID, PersonID,

CASE when OrderNumber > 24 THEN ‘Quantity is greater than 24’

ELSE ‘ Quanitiy is less than 5’ END as BulkOrders from orders;