upGrad

PLACEMENT REFRESHER PROGRAM

Session 5 - SQL 3

Advanced Functions & Filtering

By
Ritesh Kumar Pandey

Agenda

- Conditional Functions / Filter
- Mathematical Functions
- Character Functions
- Date Functions
- Window Functions

Conditional Functions

CASE

If else …. then

All of the expressions(expr, comparison_expr, return_expr) must be of the same data type.

Syntax:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
        [WHEN comparison_expr2 THEN return_expr2
         .
         .
         WHEN comparison_exprn THEN return_exprn
         ELSE else_expr]
END
```

Conditional Functions

Count the total number of students in each department and the number of student who score more than 50 marks

Conditional Functions

Count the total number of students in each department and the number of student who score more than 50 marks

```
select department,
count(*) as "Number of Students",
count(CASE WHEN marks > 50 THEN 1 ELSE NULL END) as "Number
of Students scoring more than 50 Marks"
from student_info group by department;
```

Conditional Functions

Count the total number of students in each department and the number of student who score more than 50 marks

```
select department,
count(*) as "Number of Students",
count(*) filter(where marks > 50) as "Number of Students scoring more than 50 Marks"
from student_info group by department;
```

Conditional Functions

Based on the given marks, print the id, name, marks and grade of students. If marks is greater than 90 then grade is A, if marks is greater than or equal to 70 then grade is B, if marks is greater than or equal to 50, then grade is C or else grade is D.

Conditional Functions

Based on the given marks, print the id, name, marks and grade of students. If marks is greater than 90 then grade is A, if marks is greater than or equal to 70 then grade is B, if marks is greater than or equal to 50, then grade is C or else grade is D.

```
select student_id, student_name, marks,
case
    when marks >=90 then 'A'
    when marks >=70 then 'B'
    when marks >=50 then 'C'
else 'D'
end 'Grade'
from student_info;
```

Conditional Functions

If marks is greater than 60 then the student is passed else the student is failed. Find out the total number of passed and failed students.

Conditional Functions

If marks is greater than 60 then the student is passed else the student is failed. Find out the total number of passed and failed students.

```
SELECT
  CASE
    WHEN marks >= 60
      THEN "passed"
    ELSE "failed"
  END AS result,
  COUNT(*) AS number_of_students
FROM student_info
GROUP BY result;
```

Mathematical Functions

- POWER(x, y)
- SQUARE (x)
- ROUND(x, y)
- PI()
- SQRT(x)
- CEILING(x)
- FLOOR(x)

- select power(5, 2) = 25
- select square(5) = 25
- select round(25.3324, 2) = 25.33
- select pi() = 3.141593
- select sqrt(25) = 5
- select ceiling(25.33) = 26
- select floor(25.33) = 25

Character Functions

upGrad

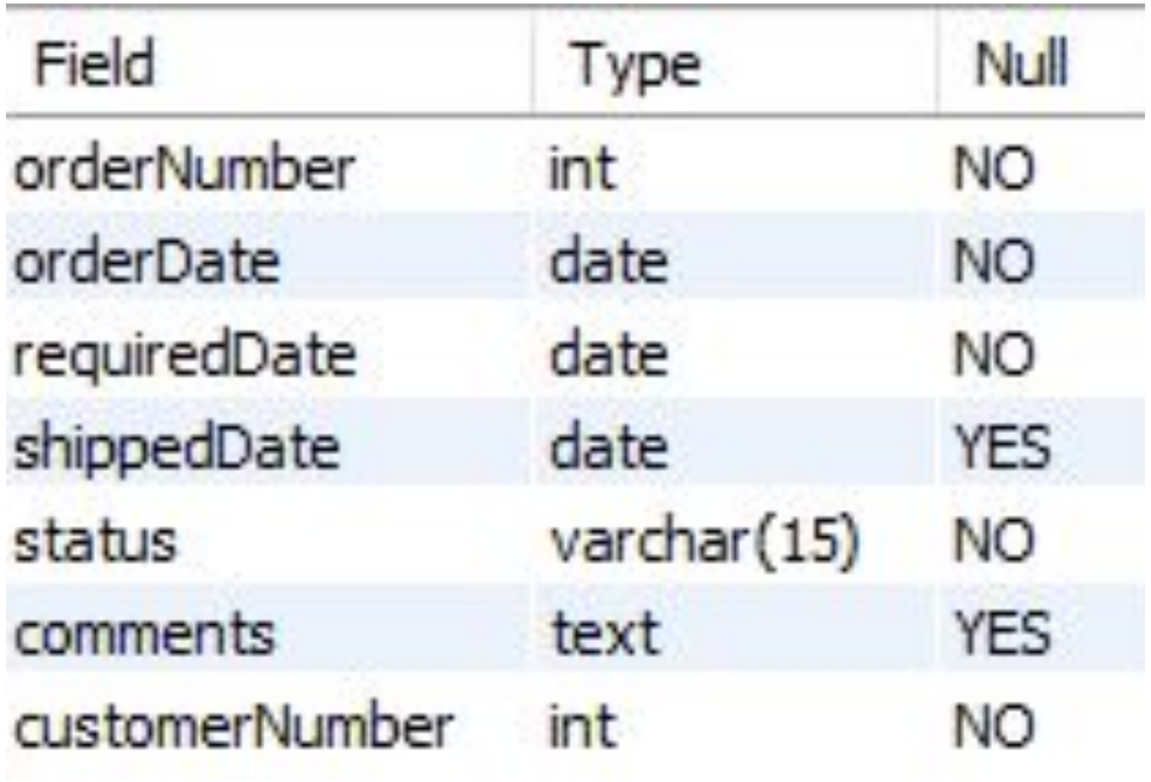
- select lower('UPGRAD'); = upgrad
- select upper('upgrad'); = UPGRAD
- select concat('up', 'Grad'); = upGrad
- select length('upgrad'); = 6
- select substr('upgrad', 3, 2); = gr
- select lpad('grad', 6, 'up'); = upgrad
- select rpad('up', 6, 'grad'); = upgrad
- select trim('x' from 'upgradx'); = upgrad
- select replace('upgrax', 'x', 'd'); = upgrad

Date Functions

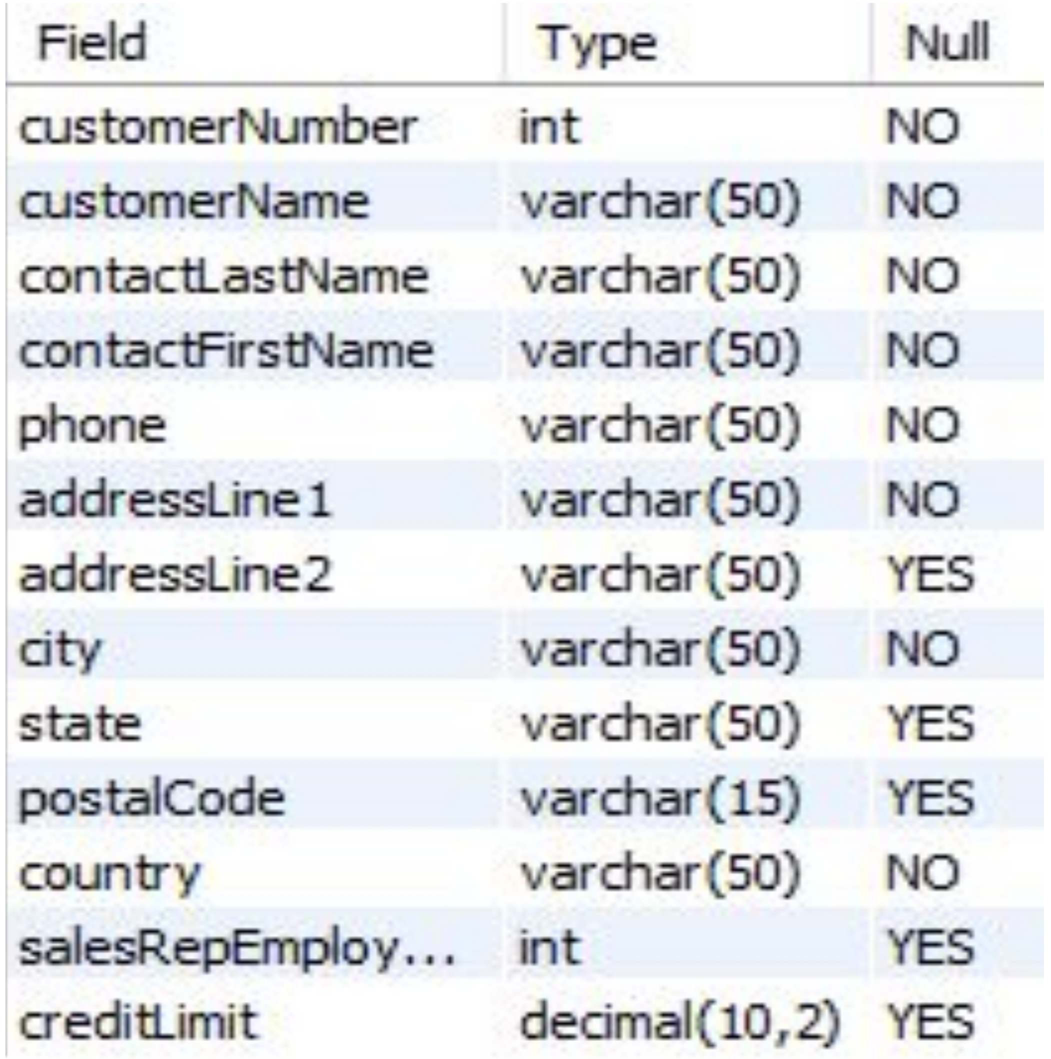
- select now();
- select curdate();
- select curtime();
- select date('1996-09-26 16:44:15.581');
- select extract(day from '1996-09-26 16:44:15.581');
- select date_add('1996-09-26 16:44:15.581', interval 1 month);
- select date_sub('1996-09-26 16:44:15.581', interval 1 month);
- select datediff('2017-01-13','2017-01-03');

Date Functions - Questions

You’ve two tables “**orders**” and “**customers**” whose schema is as follows:



Field	Type	Null
orderNumber	int	NO
orderDate	date	NO
requiredDate	date	NO
shippedDate	date	YES
status	varchar(15)	NO
comments	text	YES
customerNumber	int	NO



Field	Type	Null
customerNumber	int	NO
customerName	varchar(50)	NO
contactLastName	varchar(50)	NO
contactFirstName	varchar(50)	NO
phone	varchar(50)	NO
addressLine1	varchar(50)	NO
addressLine2	varchar(50)	YES
city	varchar(50)	NO
state	varchar(50)	YES
postalCode	varchar(15)	YES
country	varchar(50)	NO
salesRepEmploy...	int	YES
creditLimit	decimal(10,2)	YES

Date Functions - Questions

Answer the following questions based on the given information:

1. Find the number of days taken to ship a product from the date of order.
2. Find out the maximum days required to ship a product.
3. Find the customerNumber whose order shipping took longest time.
4. Find the customer Name whose order shipping took longest time.

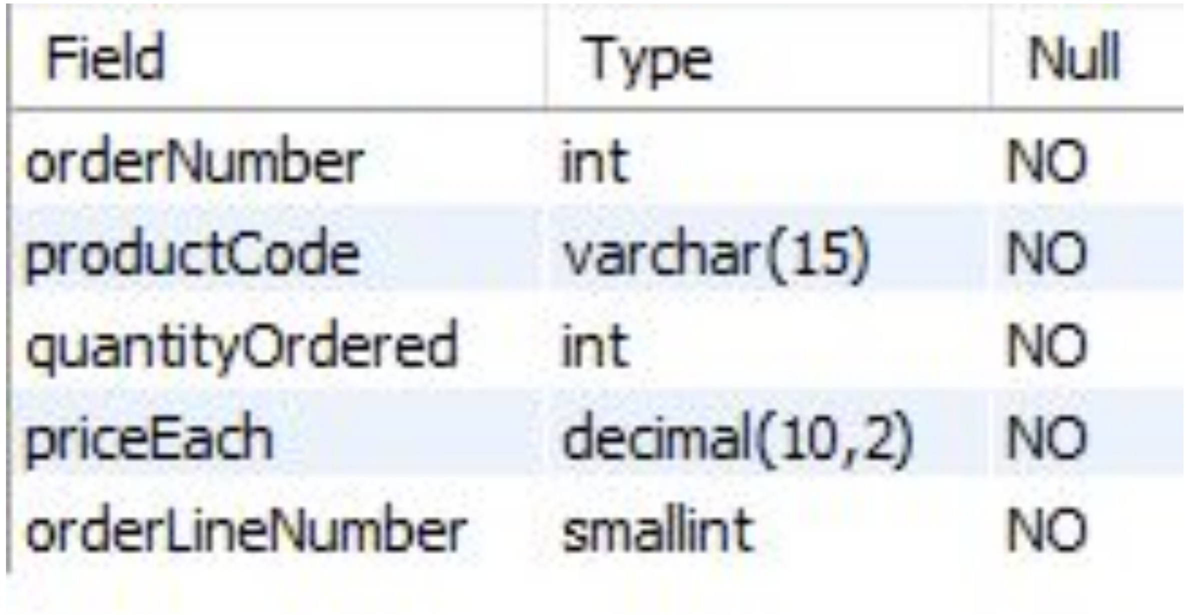
Date Functions - Questions

upGrad

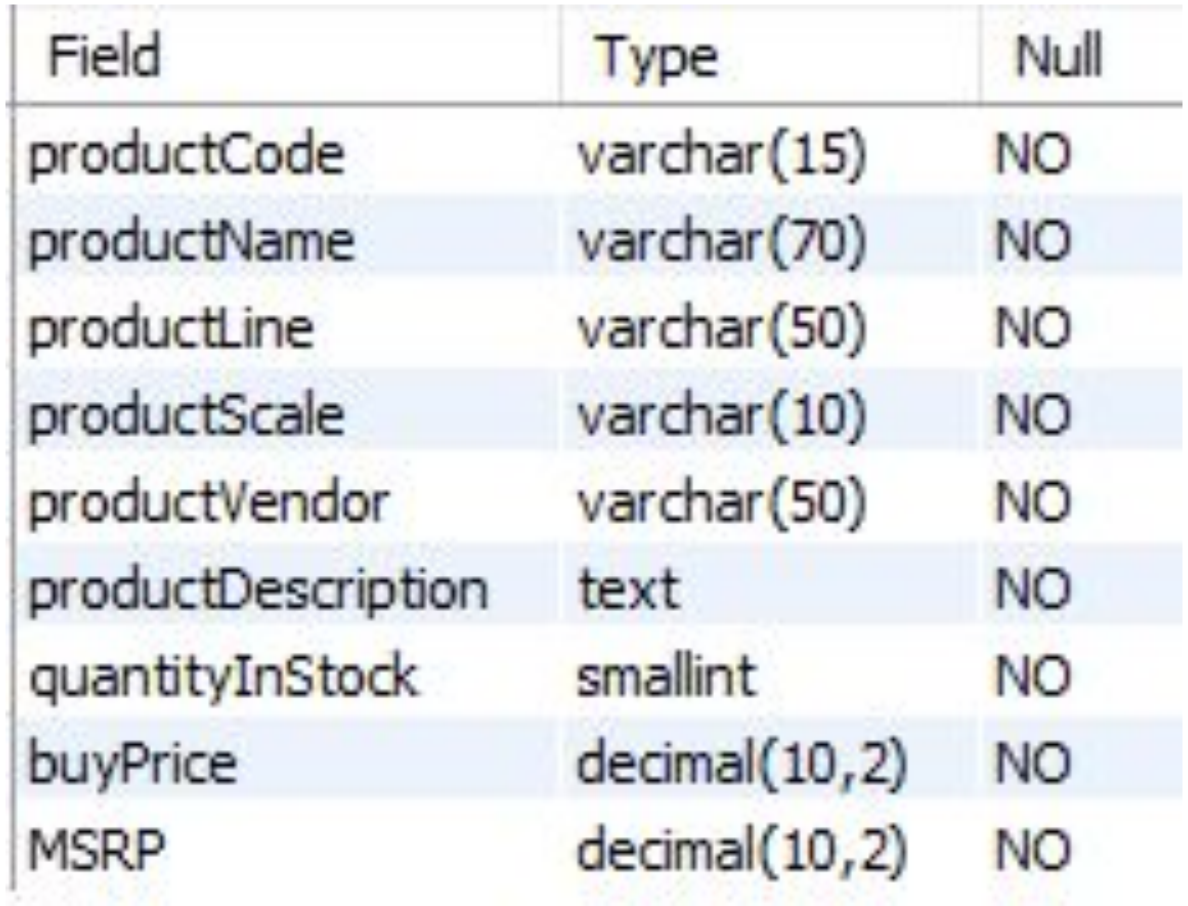
1. select datediff(shippedDate, orderDate) from orders;
2. select max(datediff(shippedDate, orderDate)) from orders;
3. select customerNumber from orders order by datediff(shippedDate, orderDate) desc limit 1;
4. select customerName from customers where customerNumber = (select customerNumber from orders order by datediff(shippedDate, orderDate) desc limit 1);

Date Functions - Questions

You’ve two tables “**orderdetails**” and “**products**” whose schema is as follows:



Field	Type	Null
orderNumber	int	NO
productCode	varchar(15)	NO
quantityOrdered	int	NO
priceEach	decimal(10,2)	NO
orderLineNumber	smallint	NO



Field	Type	Null
productCode	varchar(15)	NO
productName	varchar(70)	NO
productLine	varchar(50)	NO
productScale	varchar(10)	NO
productVendor	varchar(50)	NO
productDescription	text	NO
quantityInStock	smallint	NO
buyPrice	decimal(10,2)	NO
MSRP	decimal(10,2)	NO

Date Functions - Questions

Write a query to find out the name of the customer, phone number, product, purchase amount whose shipping took the longest time.

Date Functions - Questions

upGrad

```
select customerName, phone from customers where
customerNumber = (select customerNumber from orders order by
datediff(shippedDate, orderDate) desc limit 1);
```

```
select productName, productCode, (quantityOrdered*priceEach)
as 'Purchase Amount' from orderdetails natural join products
where orderNumber  = (select orderNumber from orders order by
datediff(shippedDate, orderDate) desc limit 1);
```

Window Functions

Window functions applies aggregate and ranking functions over a particular window (set of rows). OVER clause is used with window functions to define that window. OVER clause does two things :

- Partitions rows into form set of rows. (PARTITION BY clause is used)
- Orders rows within those partitions into a particular order. (ORDER BY clause is used)

Note: If partitions aren’t done, then ORDER BY orders all rows of table.

Window Functions

Syntax:

```
SELECT coulmn_name1, window_function(cloumn_name2)
 OVER([PARTITION BY column_name1] [ORDER BY column_name3])
AS new_column FROM table_name;
```

window_function= any aggregate or ranking function
column_name1= column to be selected
coulmn_name2= column on which window function is to be applied
column_name3= column on whose basis partition of rows is to be done
new_column= Name of new column
table_name= Name of table

Window Functions

Ranking functions are, RANK(), DENSE_RANK(), ROW_NUMBER()

- **RANK()** – It assigns rank to all the rows within every partition. Rank is assigned such that rank 1 given to the first row and rows having same value are assigned same rank. For the next rank after two same rank values, one rank value will be skipped.
- **DENSE_RANK()** – It assigns rank to each row within partition. Just like rank function first row is assigned rank 1 and rows having same value have same rank. The difference between RANK() and DENSE_RANK() is that in DENSE_RANK(), for the next rank after two same rank, consecutive integer is used, no rank is skipped.
- **ROW_NUMBER()** – It assigns consecutive integers to all the rows within partition. Within a partition, no two rows can have same row number.

Note –
ORDER BY() should be specified compulsorily while using rank window functions.

Window Functions

Assign row number, rank and dense rank to each student based on their department and marks;

Window Functions

upGrad

Assign row number, rank and dense rank to each student based on their department and marks;

```
SELECT
ROW_NUMBER() OVER (PARTITION BY department ORDER BY marks DESC) AS
student_row_number,
student_name, department, marks,
RANK() OVER (PARTITION BY department ORDER BY marks DESC) AS
student_rank,
DENSE_RANK() OVER (PARTITION BY department ORDER BY marks DESC) AS
student_dense_rank
FROM student_info;
```

upGrad

THANK YOU