# Know Python Bytes

www.knowpythonbytes.blogspot.in

*Mrs. Payal Bhattacharjee, PGT(C.Sc.)*
*K V No.1 Kanchrapara*
*KVS-RO(Kolkata)*

## UNIT 3

## For XII CSc:

## Database Management

## UNIT 2

## For XII IP:

## Database Query using SQL

# CONTENTS

## ( Learning outcomes )

- Aggregate Functions

  MAX ( )

  MIN ( )

  SUM ( )

  AVG ( )

  COUNT ( ) / COUNT(*)

- GROUP BY clause
- GROUP BY – HAVING Clause

# TYPES OF FUNCTIONS IN MySQL

I. Single row ( or Scalar ) functions

II. Multiple row (or Group or Aggregate) functions

| Single row ( or Scalar ) functions | Multiple row (or Group or Aggregate) functions |
|---|---|
| It work with a single row at a time. | It work with data of multiple rows at a time and return aggregated value. |
| A single row function returns a result for every row of a queried table. | Group functions return single value. |
| Examples: day(), year() | Examples: sum(),max(),min(),avg(),count() |

## LEARNING MySQL QUERIES….
## (Practical)

## AGGREGATE FUNCTIONS / GROUP FUNCTIONS / MULTIPLE ROW FUNCTIONS

[Max(),Min(),Sum(),Avg(),count(),count(*)]

# Aggregate Functions / Group Functions/ Multiple row functions

- ❖ An aggregate function performs a calculation on multiple values
- ❖ Performs calculation on ( a group of rows and not on single row)
- ❖ Aggregate function returns a single value.

**SYNTAX**

**SELECT max/min/sum/avg/count**(<column name>)
**FROM <table name>;**

***Where is optional*

**DISTINCT** and **ALL** keywords are used with Group functions.

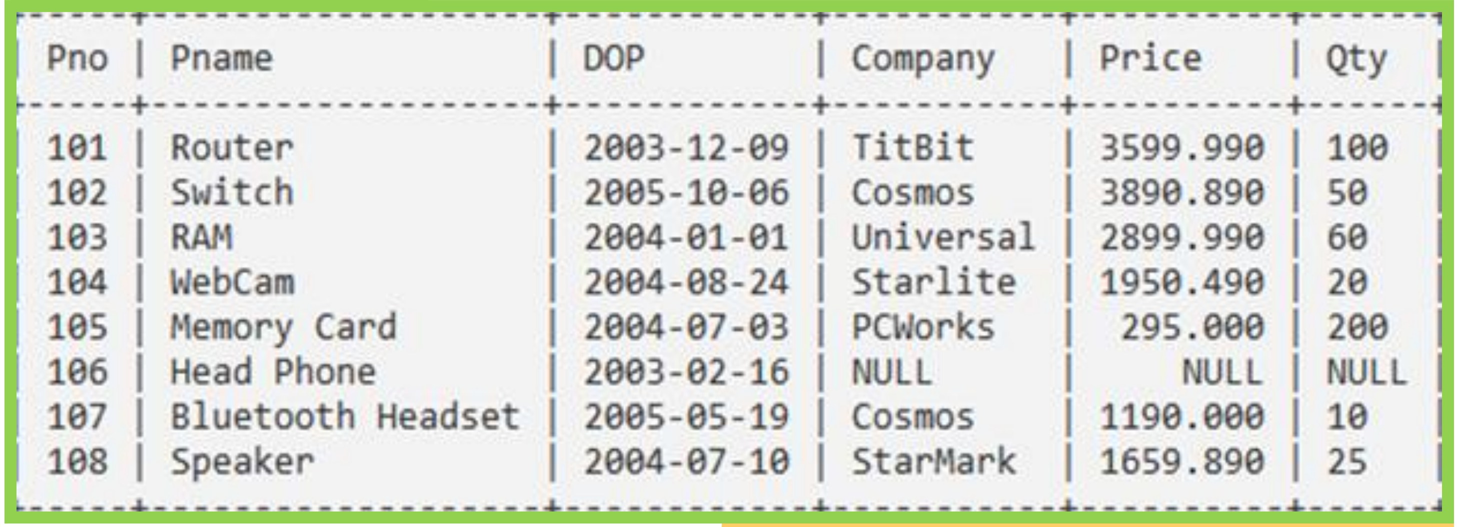
# Aggregate Functions: MAX()

## AGGREGATE FUNCTIONS

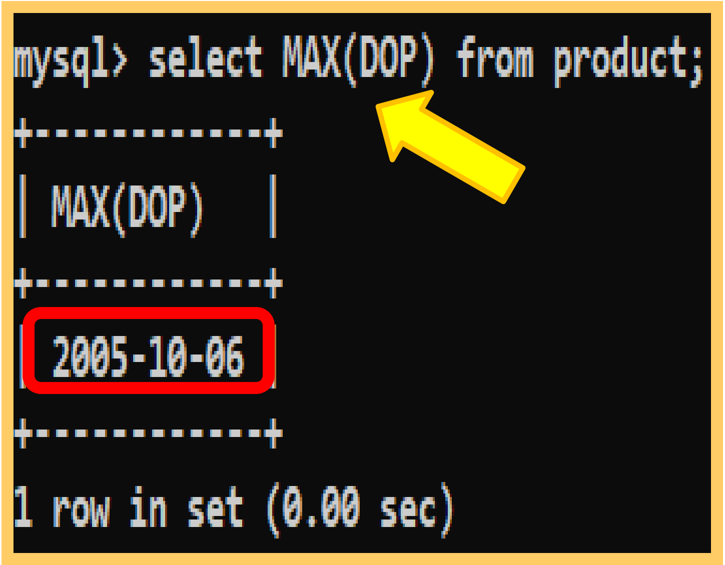
- **MAX ( )**
- MIN ( )
- SUM ( )
- AVG ( )
- COUNT ( )
- COUNT ( * )

**MAX( )** Max() function returns the maximum value from a given column or expressions

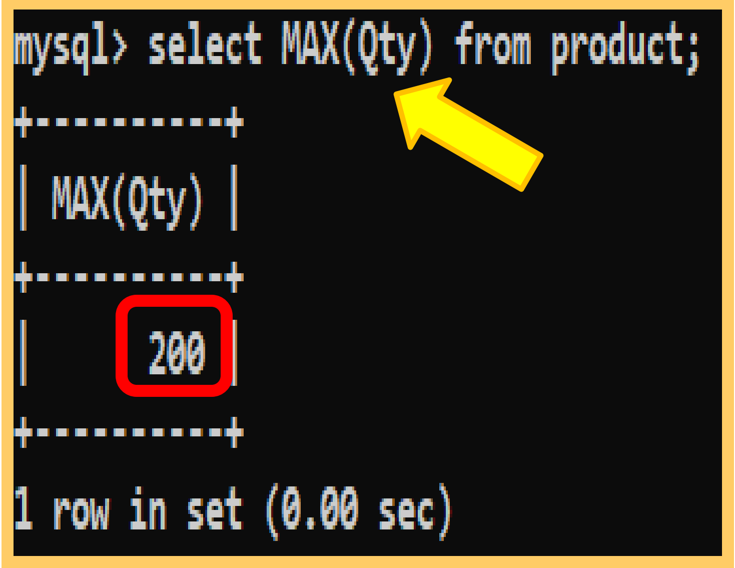
TABLE: **PRODUCT**



```
+-----+-------------------+------------+-----------+----------+------+
| Pno | Pname             | DOP        | Company   | Price    | Qty  |
+-----+-------------------+------------+-----------+----------+------+
| 101 | Router            | 2003-12-09 | TitBit    | 3599.990 | 100  |
| 102 | Switch            | 2005-10-06 | Cosmos    | 3890.890 | 50   |
| 103 | RAM               | 2004-01-01 | Universal | 2899.990 | 60   |
| 104 | WebCam            | 2004-08-24 | Starlite  | 1950.490 | 20   |
| 105 | Memory Card       | 2004-07-03 | PCWorks   |  295.000 | 200  |
| 106 | Head Phone        | 2003-02-16 | NULL      |     NULL | NULL |
| 107 | Bluetooth Headset | 2005-05-19 | Cosmos    | 1190.000 | 10   |
| 108 | Speaker           | 2004-07-10 | StarMark  | 1659.890 | 25   |
+-----+-------------------+------------+-----------+----------+------+
```



```
mysql> select MAX(DOP) from product;
+------------+
| MAX(DOP)   |
+------------+
| 2005-10-06 |
+------------+
1 row in set (0.00 sec)
```



```
mysql> select MAX(Qty) from product;
+----------+
| MAX(Qty) |
+----------+
|      200 |
+----------+
1 row in set (0.00 sec)
```

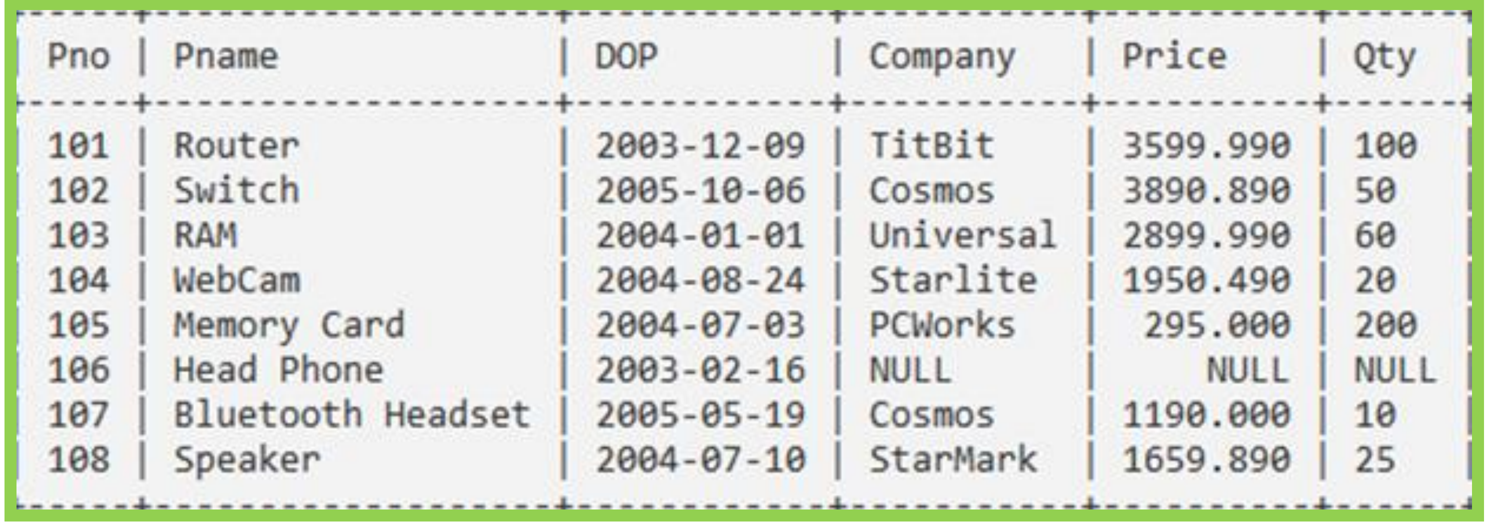
# Aggregate Functions: MIN()

**AGGREGATE FUNCTIONS**

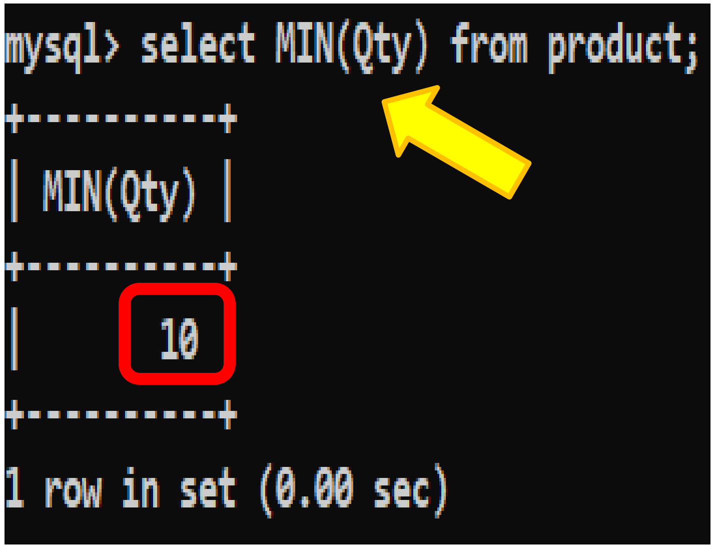
- MAX ( )
- MIN ( )
- SUM ( )
- AVG ( )
- COUNT ( )
- COUNT ( * )

**MIN( )** Min() function returns the minimum value from a given column or expressions

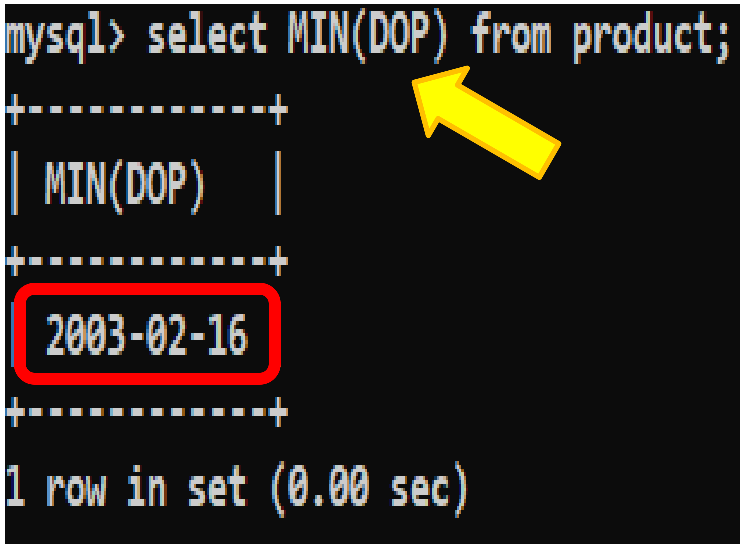
TABLE: **PRODUCT**



```
+-----+--------------------+------------+-----------+----------+------+
| Pno | Pname              | DOP        | Company   | Price    | Qty  |
+-----+--------------------+------------+-----------+----------+------+
| 101 | Router             | 2003-12-09 | TitBit    | 3599.990 | 100  |
| 102 | Switch             | 2005-10-06 | Cosmos    | 3890.890 | 50   |
| 103 | RAM                | 2004-01-01 | Universal | 2899.990 | 60   |
| 104 | WebCam             | 2004-08-24 | Starlite  | 1950.490 | 20   |
| 105 | Memory Card        | 2004-07-03 | PCWorks   |  295.000 | 200  |
| 106 | Head Phone         | 2003-02-16 | NULL      |     NULL | NULL |
| 107 | Bluetooth Headset  | 2005-05-19 | Cosmos    | 1190.000 | 10   |
| 108 | Speaker            | 2004-07-10 | StarMark  | 1659.890 | 25   |
+-----+--------------------+------------+-----------+----------+------+
```



```
mysql> select MIN(Qty) from product;
+----------+
| MIN(Qty) |
+----------+
|       10 |
+----------+
1 row in set (0.00 sec)
```



```
mysql> select MIN(DOP) from product;
+------------+
| MIN(DOP)   |
+------------+
| 2003-02-16 |
+------------+
1 row in set (0.00 sec)
```

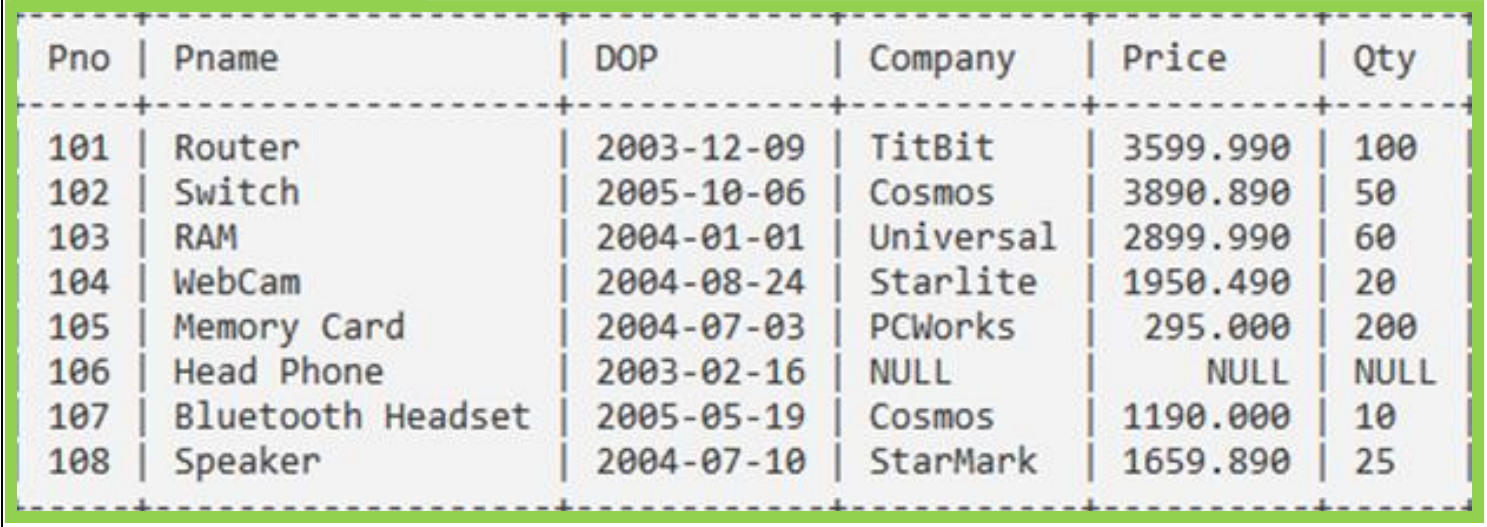
# Aggregate Functions: SUM()

## AGGREGATE FUNCTIONS

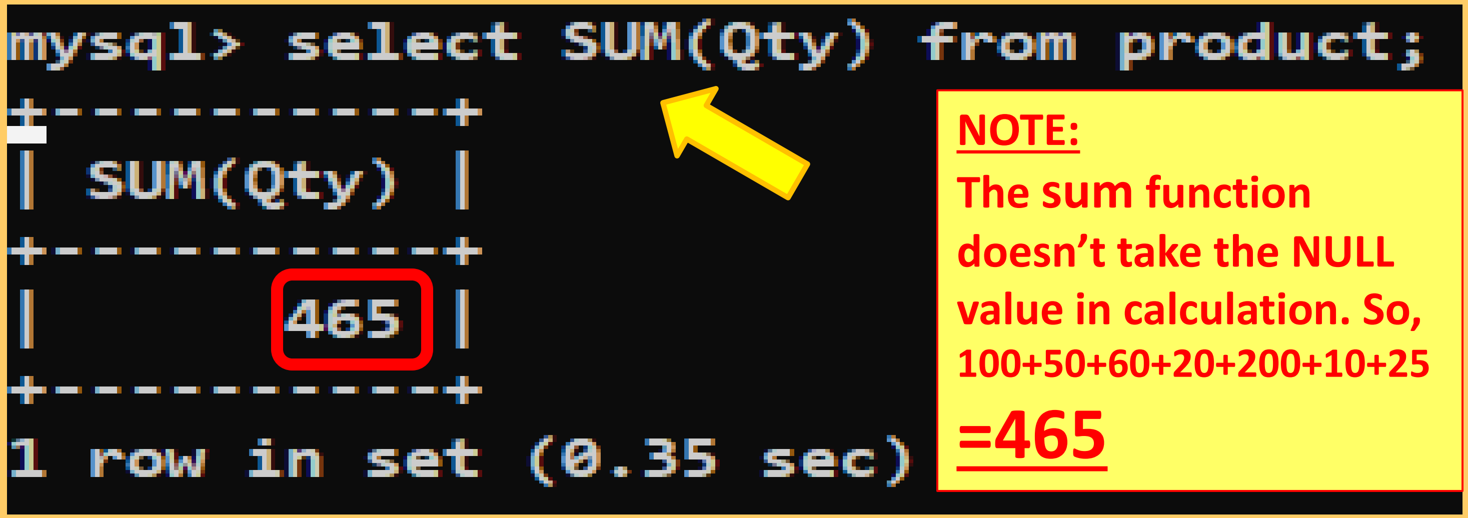
- MAX ( )
- MIN ( )
- **SUM ( )**
- AVG ( )
- COUNT ( )
- COUNT ( * )

**SUM( )** Sum() function returns the sum of values in the given parameter(input) column or expression .

TABLE: **PRODUCT**



```
+-----+-------------------+------------+-----------+----------+------+
| Pno | Pname             | DOP        | Company   | Price    | Qty  |
+-----+-------------------+------------+-----------+----------+------+
| 101 | Router            | 2003-12-09 | TitBit    | 3599.990 | 100  |
| 102 | Switch            | 2005-10-06 | Cosmos    | 3890.890 | 50   |
| 103 | RAM               | 2004-01-01 | Universal | 2899.990 | 60   |
| 104 | WebCam            | 2004-08-24 | Starlite  | 1950.490 | 20   |
| 105 | Memory Card       | 2004-07-03 | PCWorks   |  295.000 | 200  |
| 106 | Head Phone        | 2003-02-16 | NULL      |     NULL | NULL |
| 107 | Bluetooth Headset | 2005-05-19 | Cosmos    | 1190.000 | 10   |
| 108 | Speaker           | 2004-07-10 | StarMark  | 1659.890 | 25   |
+-----+-------------------+------------+-----------+----------+------+
```



```
mysql> select SUM(Qty) from product;
+----------+
| SUM(Qty) |
+----------+
|      465 |
+----------+
1 row in set (0.35 sec)
```

**NOTE:**
**The sum function doesn't take the NULL value in calculation. So, 100+50+60+20+200+10+25**
**=465**

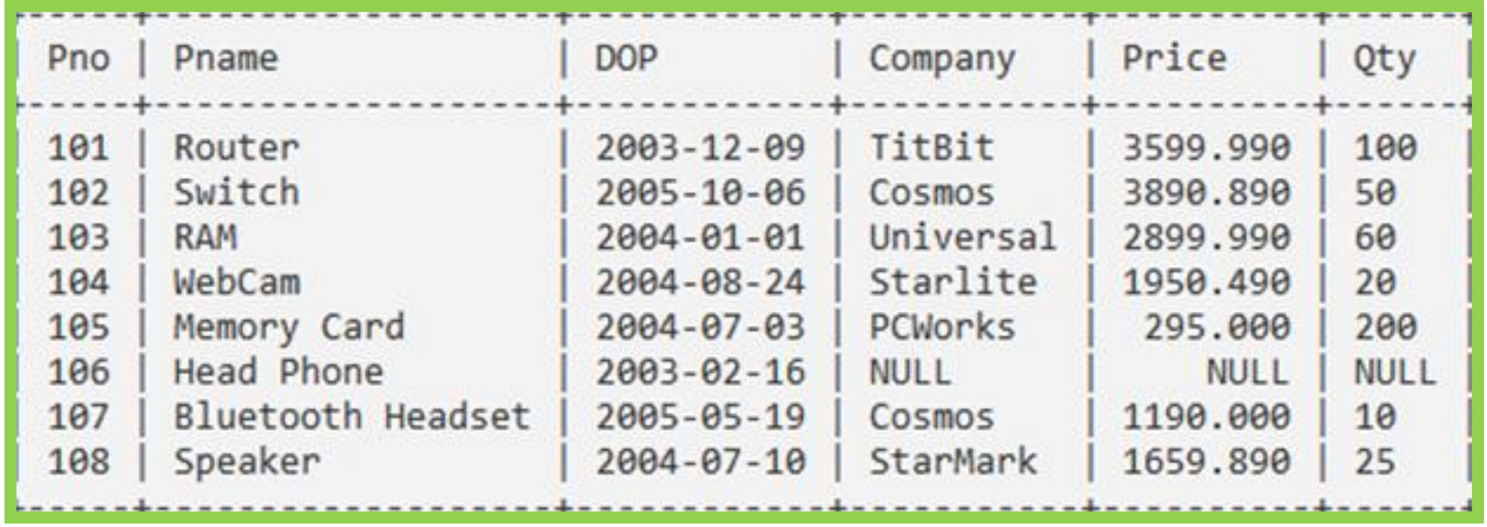
# Aggregate Functions: AVG()

## AGGREGATE FUNCTIONS

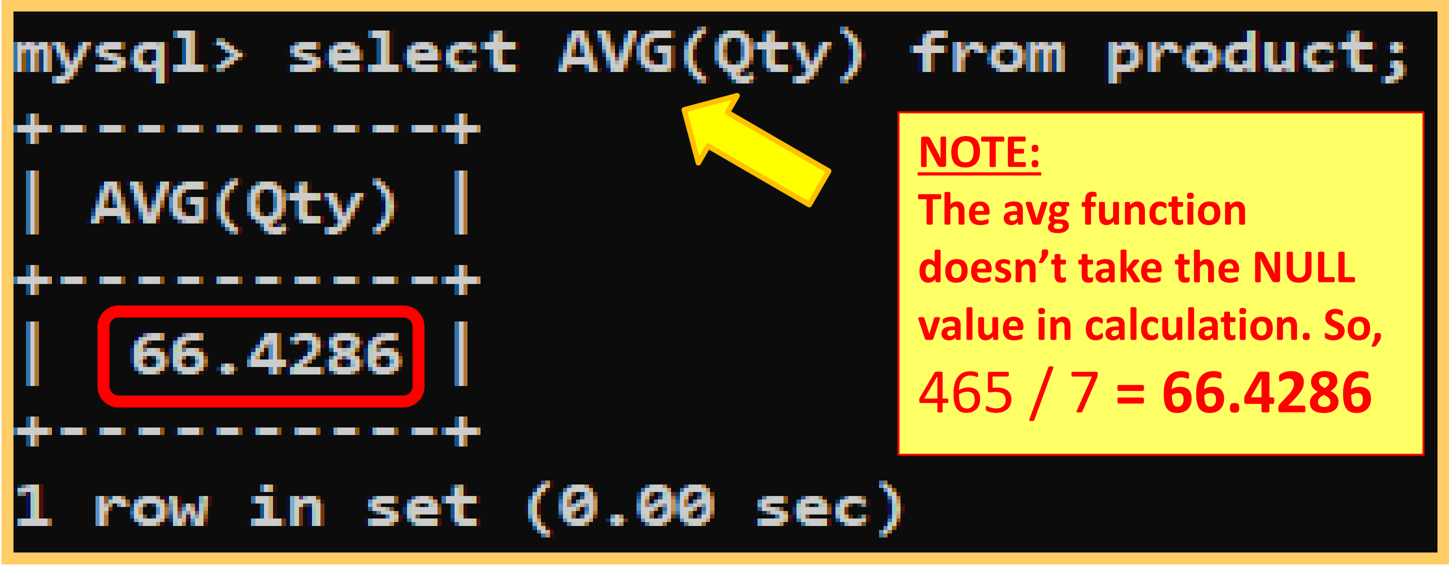
- MAX ( )
- MIN ( )
- SUM ( )
- **AVG ( )**
- COUNT ( )
- COUNT ( * )

**AVG( )** Avg() function returns the average value of the given parameter(input) value.

TABLE: **PRODUCT**



```
+-----+--------------------+------------+-----------+----------+------+
| Pno | Pname              | DOP        | Company   | Price    | Qty  |
+-----+--------------------+------------+-----------+----------+------+
| 101 | Router             | 2003-12-09 | TitBit    | 3599.990 | 100  |
| 102 | Switch             | 2005-10-06 | Cosmos    | 3890.890 | 50   |
| 103 | RAM                | 2004-01-01 | Universal | 2899.990 | 60   |
| 104 | WebCam             | 2004-08-24 | Starlite  | 1950.490 | 20   |
| 105 | Memory Card        | 2004-07-03 | PCWorks   |  295.000 | 200  |
| 106 | Head Phone         | 2003-02-16 | NULL      |     NULL | NULL |
| 107 | Bluetooth Headset  | 2005-05-19 | Cosmos    | 1190.000 | 10   |
| 108 | Speaker            | 2004-07-10 | StarMark  | 1659.890 | 25   |
+-----+--------------------+------------+-----------+----------+------+
```



```
mysql> select AVG(Qty) from product;
+----------+
| AVG(Qty) |
+----------+
|  66.4286 |
+----------+
1 row in set (0.00 sec)
```

**NOTE:**
**The avg function doesn't take the NULL value in calculation. So,**
465 / 7 = **66.4286**

# Aggregate Functions: COUNT( ) and COUNT(*)

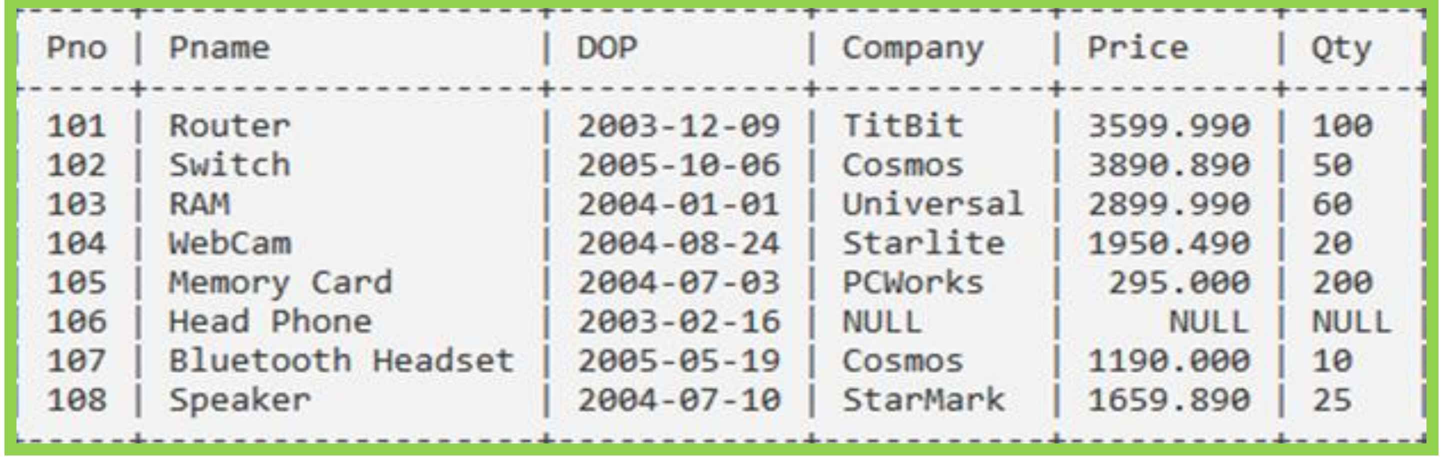
**AGGREGATE FUNCTIONS**

- MAX ( )
- MIN ( )
- SUM ( )
- AVG ( )
- COUNT ( )
- COUNT(*)

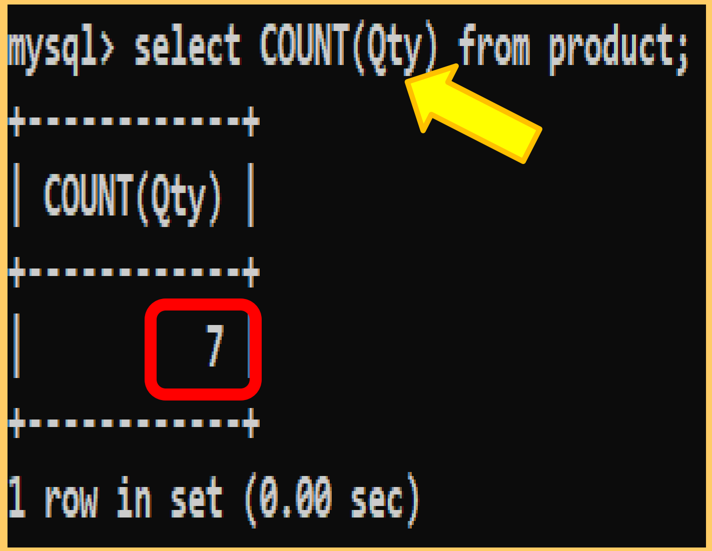
## COUNT( ) and COUNT(*) :

Count function returns the total no. of values/records under the specified column or expression.

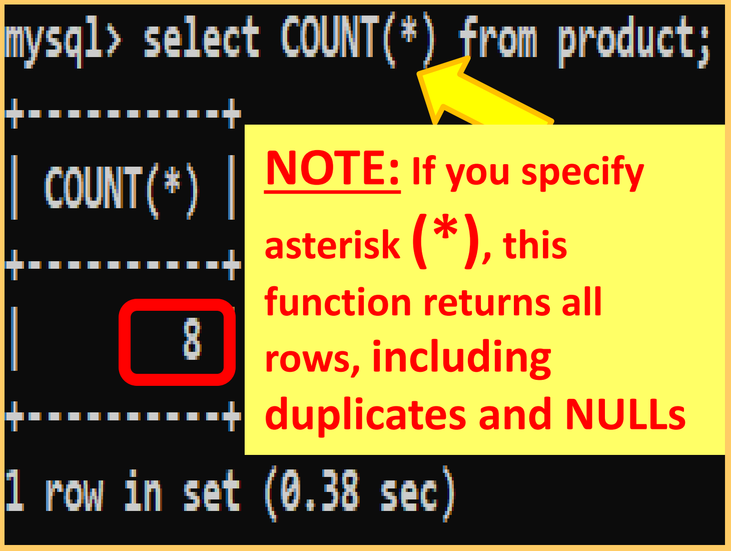
TABLE: **PRODUCT**



```
+-----+-------------------+------------+-----------+----------+------+
| Pno | Pname             | DOP        | Company   | Price    | Qty  |
+-----+-------------------+------------+-----------+----------+------+
| 101 | Router            | 2003-12-09 | TitBit    | 3599.990 | 100  |
| 102 | Switch            | 2005-10-06 | Cosmos    | 3890.890 | 50   |
| 103 | RAM               | 2004-01-01 | Universal | 2899.990 | 60   |
| 104 | WebCam            | 2004-08-24 | Starlite  | 1950.490 | 20   |
| 105 | Memory Card       | 2004-07-03 | PCWorks   |  295.000 | 200  |
| 106 | Head Phone        | 2003-02-16 | NULL      |     NULL | NULL |
| 107 | Bluetooth Headset | 2005-05-19 | Cosmos    | 1190.000 | 10   |
| 108 | Speaker           | 2004-07-10 | StarMark  | 1659.890 | 25   |
+-----+-------------------+------------+-----------+----------+------+
```



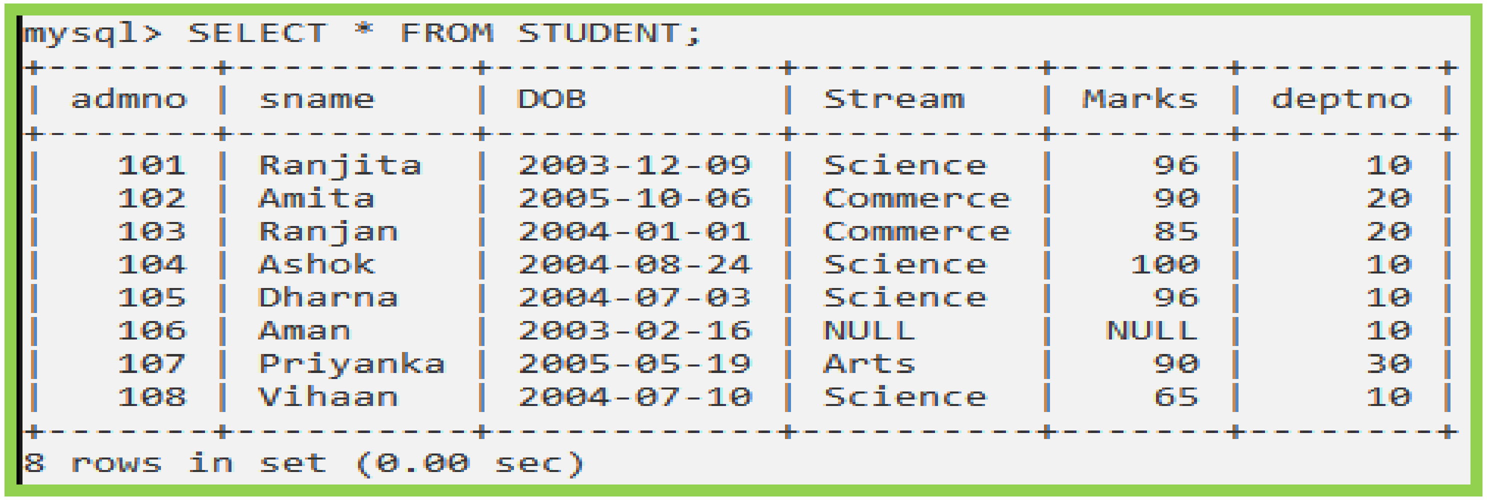
```
mysql> select COUNT(Qty) from product;
+------------+
| COUNT(Qty) |
+------------+
|          7 |
+------------+
1 row in set (0.00 sec)
```



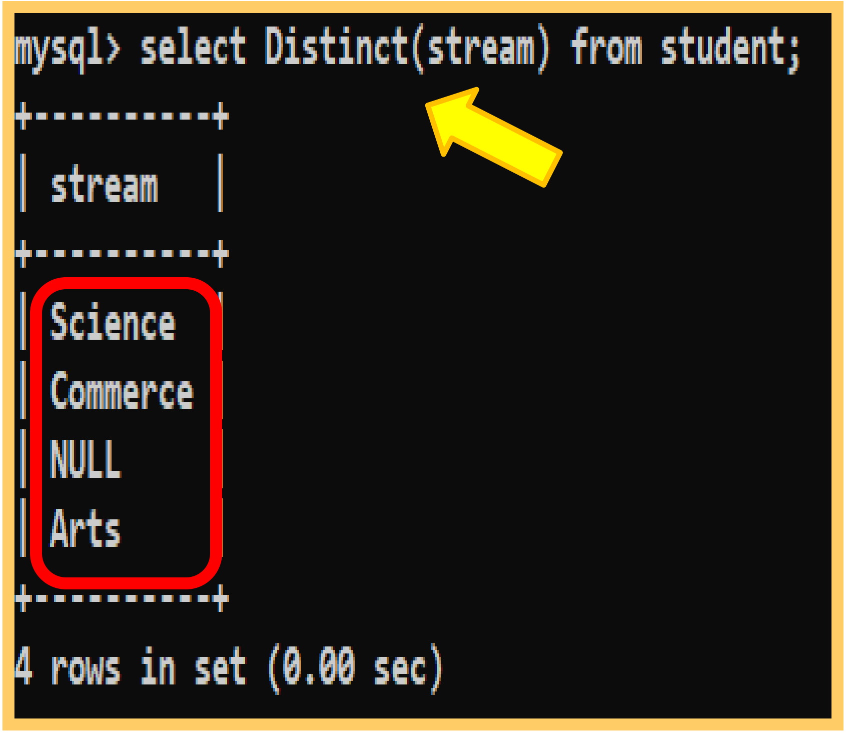
```
mysql> select COUNT(*) from product;
+----------+
| COUNT(*) |
+----------+
|        8 |
+----------+
1 row in set (0.38 sec)
```

**NOTE:** If you specify asterisk (*), this function returns all rows, including duplicates and NULLs

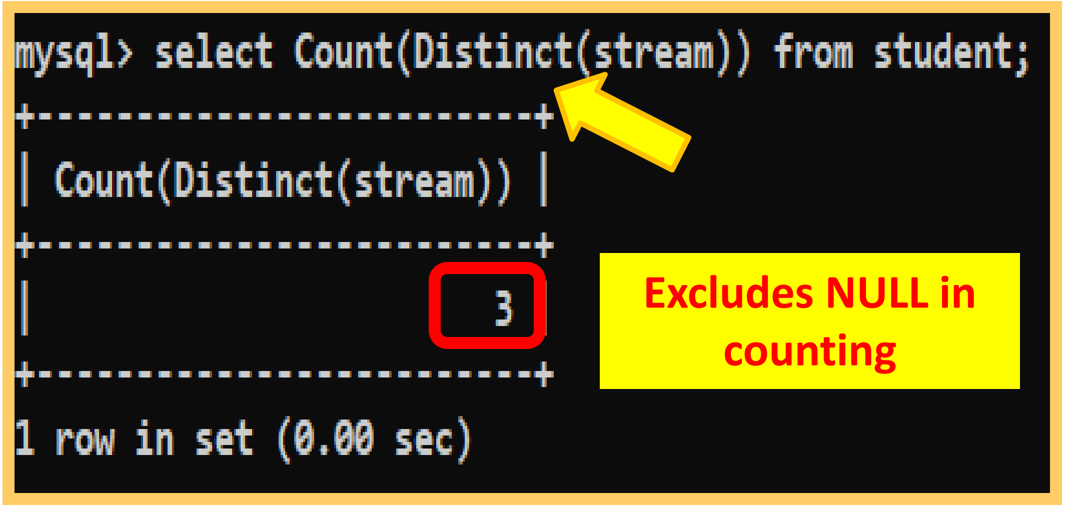
# More on Aggregate Functions: COUNT( ) with DISTINCT keywords



```
mysql> SELECT * FROM STUDENT;
+--------+----------+------------+----------+-------+--------+
| admno  | sname    | DOB        | Stream   | Marks | deptno |
+--------+----------+------------+----------+-------+--------+
|    101 | Ranjita  | 2003-12-09 | Science  |    96 |     10 |
|    102 | Amita    | 2005-10-06 | Commerce |    90 |     20 |
|    103 | Ranjan   | 2004-01-01 | Commerce |    85 |     20 |
|    104 | Ashok    | 2004-08-24 | Science  |   100 |     10 |
|    105 | Dharna   | 2004-07-03 | Science  |    96 |     10 |
|    106 | Aman     | 2003-02-16 | NULL     |  NULL |     10 |
|    107 | Priyanka | 2005-05-19 | Arts     |    90 |     30 |
|    108 | Vihaan   | 2004-07-10 | Science  |    65 |     10 |
+--------+----------+------------+----------+-------+--------+
8 rows in set (0.00 sec)
```



```
mysql> select Distinct(stream) from student;
+----------+
| stream   |
+----------+
| Science  |
| Commerce |
| NULL     |
| Arts     |
+----------+
4 rows in set (0.00 sec)
```

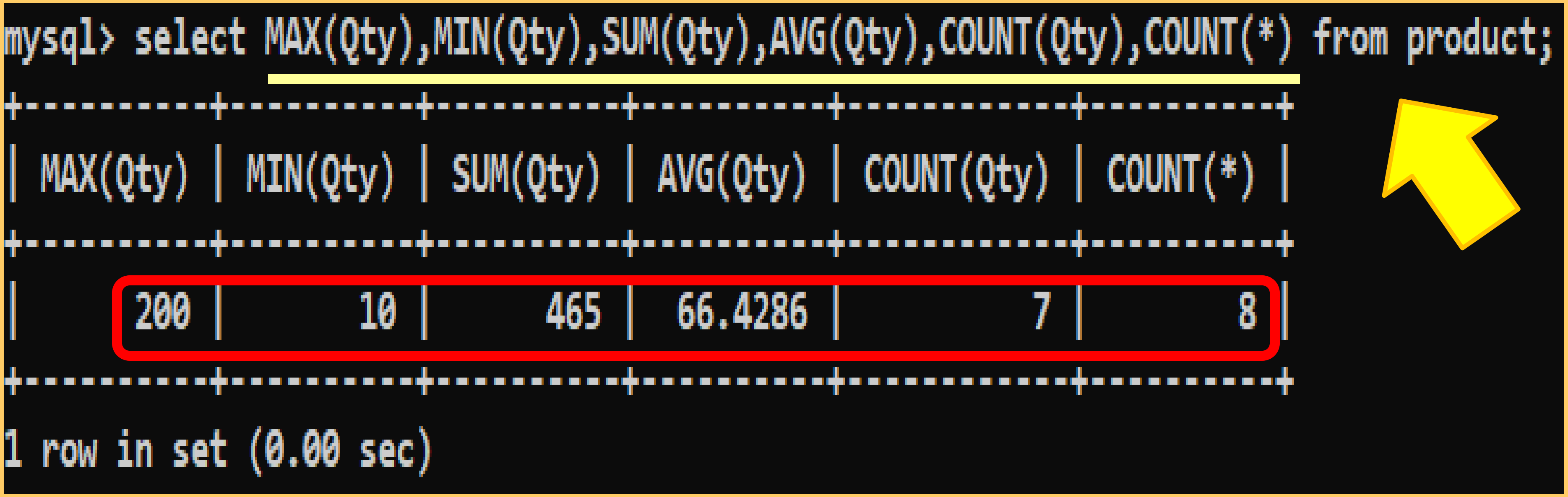


```
mysql> select Count(Distinct(stream)) from student;
+-------------------------+
| Count(Distinct(stream)) |
+-------------------------+
|                       3 |
+-------------------------+
1 row in set (0.00 sec)
```

Excludes NULL in counting

# AGGREGATE FUNCTIONS altogether

**NOTE:** The * is the only argument that includes NULLs when it is used only with COUNT (), functions other than COUNT disregard NULLs in any case.



```
mysql> select MAX(Qty),MIN(Qty),SUM(Qty),AVG(Qty),COUNT(Qty),COUNT(*) from product;
+----------+----------+----------+----------+------------+----------+
| MAX(Qty) | MIN(Qty) | SUM(Qty) | AVG(Qty) | COUNT(Qty) | COUNT(*) |
+----------+----------+----------+----------+------------+----------+
|      200 |       10 |      465 |  66.4286 |          7 |        8 |
+----------+----------+----------+----------+------------+----------+
1 row in set (0.00 sec)
```

# LEARNING MySQL QUERIES….
# (Practical)

- GROUP BY clause
- HAVING condition

# GROUP BY clause

- The Group By clause **combines all those records that have identical values in a particular field or a group of fields**. This grouping results into **one summary record per group** if group functions are used with it.

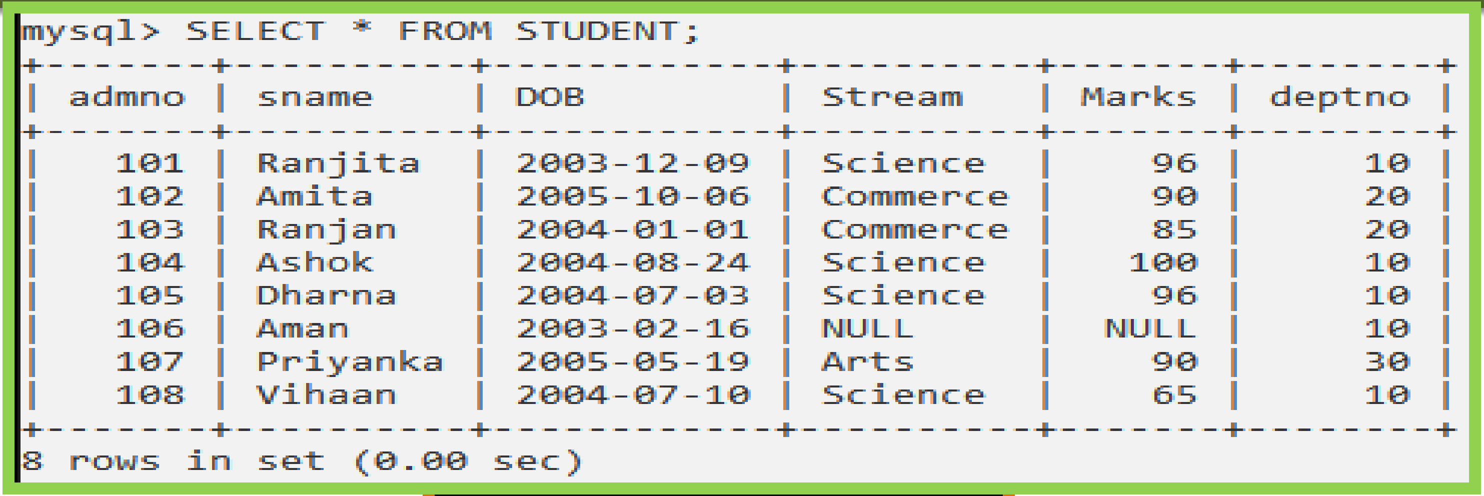
- Grouping can be done **by a column name, or with aggregate functions** in which case the aggregate produces a value for each group.

- All rows with a **NULL** in the column are treated as if **NULL** was another **value**. If a **grouping** column **contains null values**, all **null values** are considered equal, and they are put into a single **group**.

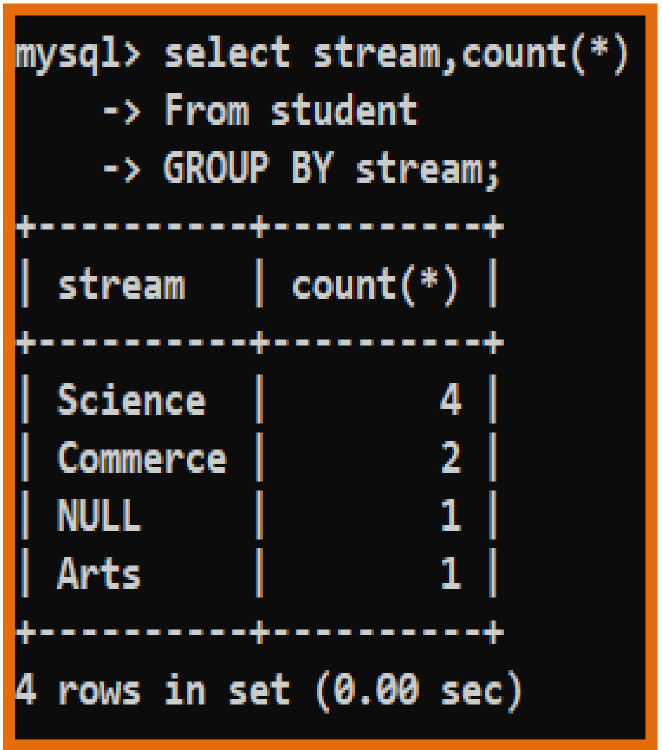
## SYNTAX

**SELECT <column name> [** *,<column name>,….***]**
**FROM <table name>**
**GROUP BY <coumn name>;**

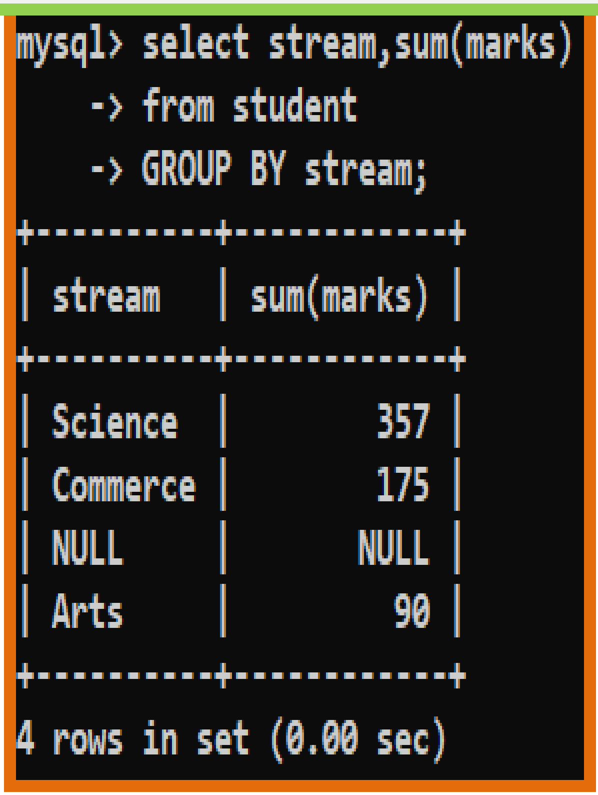
# GROUP BY QUERY examples



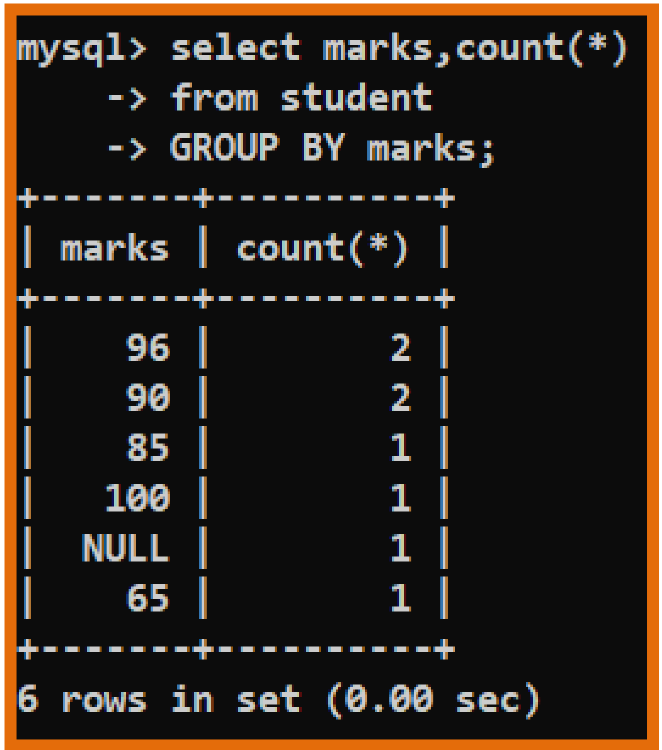
```
mysql> SELECT * FROM STUDENT;
+--------+----------+------------+----------+-------+--------+
| admno  | sname    | DOB        | Stream   | Marks | deptno |
+--------+----------+------------+----------+-------+--------+
|    101 | Ranjita  | 2003-12-09 | Science  |    96 |     10 |
|    102 | Amita    | 2005-10-06 | Commerce |    90 |     20 |
|    103 | Ranjan   | 2004-01-01 | Commerce |    85 |     20 |
|    104 | Ashok    | 2004-08-24 | Science  |   100 |     10 |
|    105 | Dharna   | 2004-07-03 | Science  |    96 |     10 |
|    106 | Aman     | 2003-02-16 | NULL     |  NULL |     10 |
|    107 | Priyanka | 2005-05-19 | Arts     |    90 |     30 |
|    108 | Vihaan   | 2004-07-10 | Science  |    65 |     10 |
+--------+----------+------------+----------+-------+--------+
8 rows in set (0.00 sec)
```



```
mysql> select stream,count(*)
    -> From student
    -> GROUP BY stream;
+----------+----------+
| stream   | count(*) |
+----------+----------+
| Science  |        4 |
| Commerce |        2 |
| NULL     |        1 |
| Arts     |        1 |
+----------+----------+
4 rows in set (0.00 sec)
```



```
mysql> select stream,sum(marks)
    -> from student
    -> GROUP BY stream;
+----------+------------+
| stream   | sum(marks) |
+----------+------------+
| Science  |        357 |
| Commerce |        175 |
| NULL     |       NULL |
| Arts     |         90 |
+----------+------------+
4 rows in set (0.00 sec)
```



```
mysql> select marks,count(*)
    -> from student
    -> GROUP BY marks;
+-------+----------+
| marks | count(*) |
+-------+----------+
|    96 |        2 |
|    90 |        2 |
|    85 |        1 |
|   100 |        1 |
|  NULL |        1 |
|    65 |        1 |
+-------+----------+
6 rows in set (0.00 sec)
```

# GROUP BY with HAVING condition

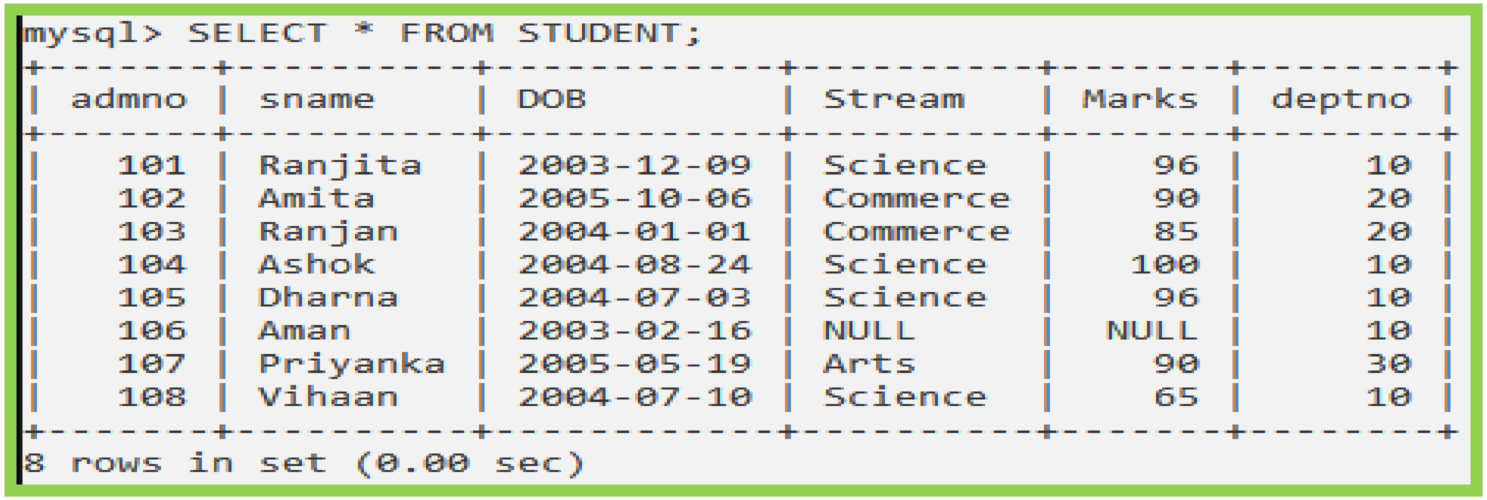
- The **Having clause** places conditions on groups and it can also include aggregate functions.

| WHERE clause | HAVING clause |
|---|---|
| **It places conditions on individual rows** | **It places conditions on groups.** |
| **It cannot include aggregate functions.** | **It can include aggregate functions like Sum(),Avg(),count() etc.** |
| Ex query:-<br>**Select * from student where marks>80;** | Ex query:-<br>**Select stream,count(*)**<br>**From student**<br>**Group by stream**<br>**Having count(*)>=2;** |

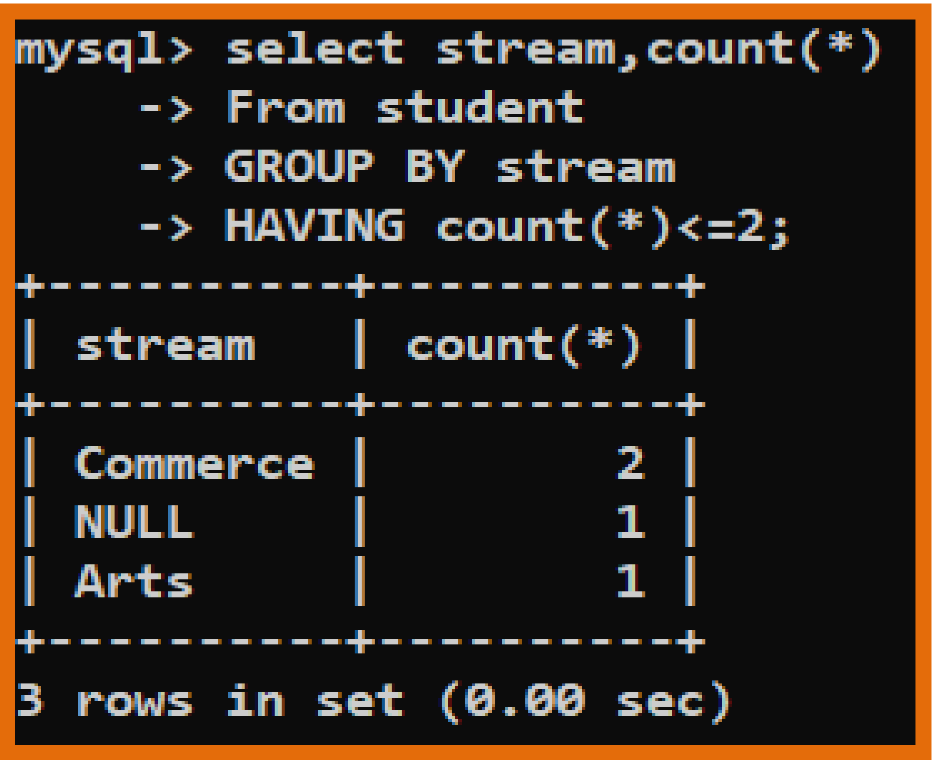
**SYNTAX**

**SELECT <column name> [** *,<column name>,….* **]**
**FROM <table name>**
**GROUP BY <coumn name>**
**HAVING** *condition* **;**

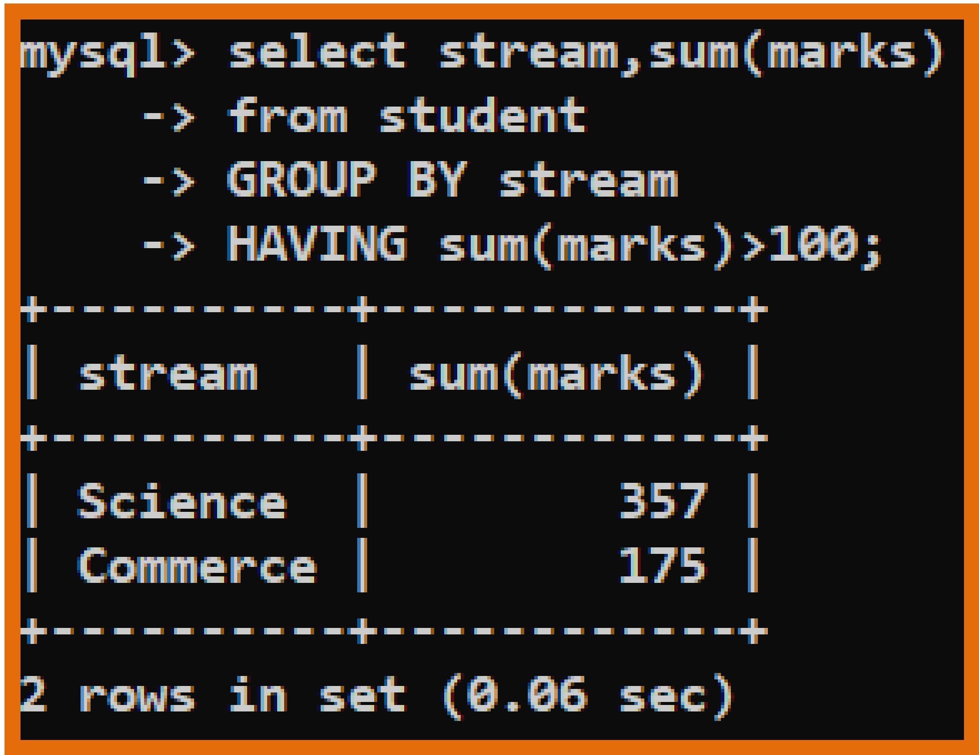
# GROUP BY with HAVING condition QUERY examples



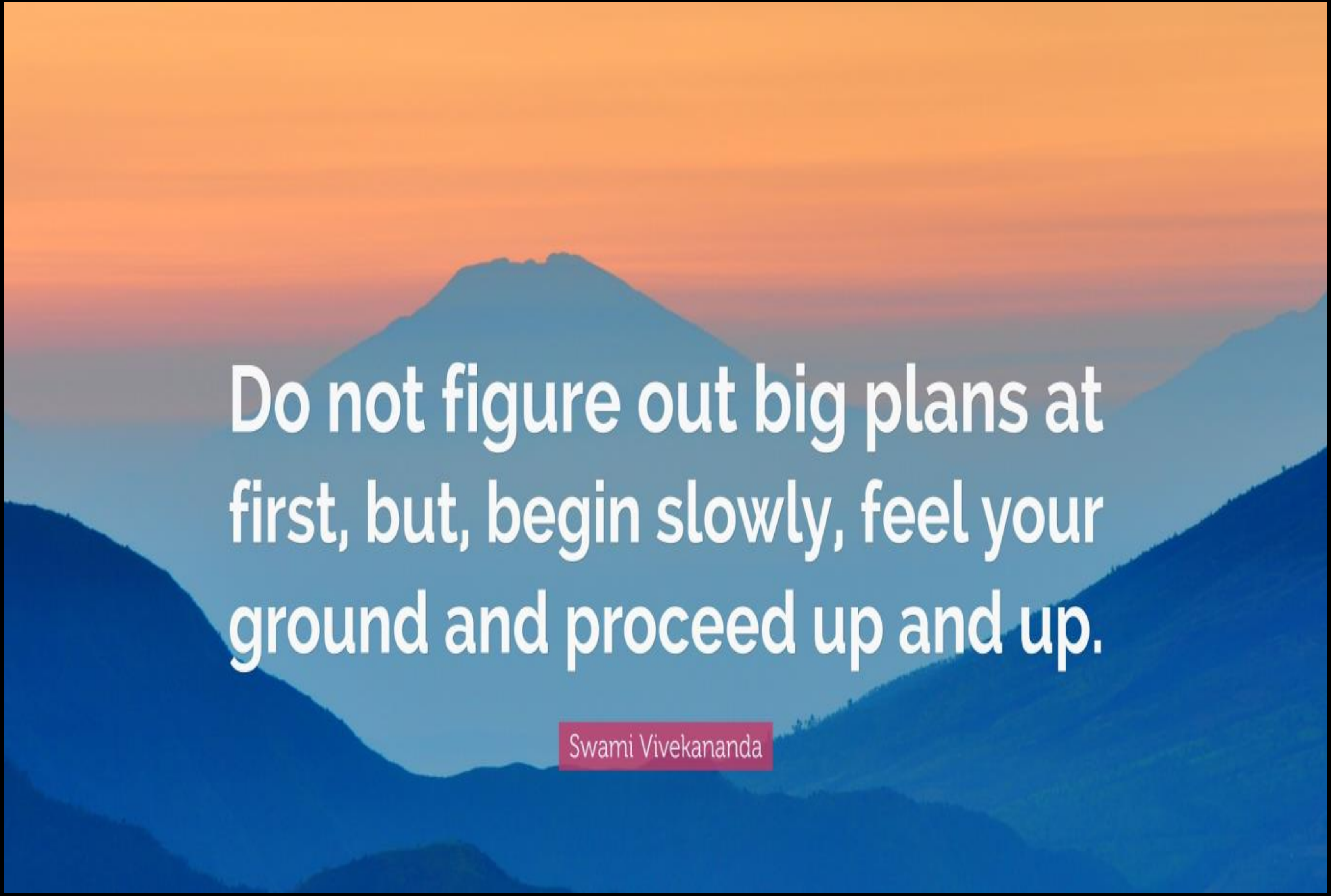
```
mysql> SELECT * FROM STUDENT;
+--------+----------+------------+----------+-------+--------+
| admno  | sname    | DOB        | Stream   | Marks | deptno |
+--------+----------+------------+----------+-------+--------+
|    101 | Ranjita  | 2003-12-09 | Science  |    96 |     10 |
|    102 | Amita    | 2005-10-06 | Commerce |    90 |     20 |
|    103 | Ranjan   | 2004-01-01 | Commerce |    85 |     20 |
|    104 | Ashok    | 2004-08-24 | Science  |   100 |     10 |
|    105 | Dharna   | 2004-07-03 | Science  |    96 |     10 |
|    106 | Aman     | 2003-02-16 | NULL     |  NULL |     10 |
|    107 | Priyanka | 2005-05-19 | Arts     |    90 |     30 |
|    108 | Vihaan   | 2004-07-10 | Science  |    65 |     10 |
+--------+----------+------------+----------+-------+--------+
8 rows in set (0.00 sec)
```



```
mysql> select stream,count(*)
    -> From student
    -> GROUP BY stream
    -> HAVING count(*)<=2;
+----------+----------+
| stream   | count(*) |
+----------+----------+
| Commerce |        2 |
| NULL     |        1 |
| Arts     |        1 |
+----------+----------+
3 rows in set (0.00 sec)
```



```
mysql> select stream,sum(marks)
    -> from student
    -> GROUP BY stream
    -> HAVING sum(marks)>100;
+----------+------------+
| stream   | sum(marks) |
+----------+------------+
| Science  |        357 |
| Commerce |        175 |
+----------+------------+
2 rows in set (0.06 sec)
```



Do not figure out big plans at first, but, begin slowly, feel your ground and proceed up and up.

Swami Vivekananda

*Stay safe. Stay aware. Stay healthy. Stay alert.*

THANK YOU FOR YOUR PATIENT HEARING ☺