# SELECT Statement

The select statement lets you make queries on tables.

Syntax

Select <column name>[,<column name,….]

From <table name>;



* Selecting columns from tables

Syntax

Select <column name>[,<column name,….]

From <table name>;

eg:To select columns Ename and Gender from Employee table.

Select ename,gender from employee;

* Selecting all columns: using \* operator

eg: select \* from employee;

* Reordering Columns in query results : Result can be obtained in any order

select gross,ecode,ename from employee;

* Eliminating redundant data: DISTINCT key word eliminates redundant data

eg:SELECT Distinct(gender) from employee;

* Selecting from all the rows:ALL keyword can be used for selecting all the rows

eg:SELECT ALL gender from employee;

* View structure of a table

SHOW TABLES: lIsts all the tables in the database

DESC <tablename>(Describe <tblname>): shows the structure of table

eg: desc employee

* Performing simple calculation

eg: select 4\*6;

 Mysql also provides a dummy table called dual.

Dual table has one row and one column.

eg: select 6\*7 from dual;

 select curdate();

* Scalar expressions with selected field: We can perform numeric calculations on data

Eg:select ename,gross\*100 from employee

* Using column aliases: The columns you select in the query can be given a different name using As keyword.

select ecode as “Employee code” ,ename from employee;

* Handling Nulls: IFNULL() function can be used.

PET

|  |  |  |
| --- | --- | --- |
| Name | Birth | Death |
| Fluffy | 1993-03-09 | Null |
| Ruffle | 1991-03-09 | Null |
| Snowy | 1996-01-23 | 1997-05-23 |
| Buffy | 1995-03-19 | 1996-01-02 |

If you want to substitute value for null in death then

select name,IFNULL(Death,”Alive”) As “ Died on” from PET;

|  |  |
| --- | --- |
| Name | Died on  |
| Fluffy | Alive |
| Ruffle | Alive |
| Snowy | 1997-05-23 |
| Buffy | 1996-01-02 |

* Selecting specific rows- Where clause

Syntax

Select <column name>[,<column name,….]

From <table name>

Where <condition>;

eg: select ecode ,ename from employee where gross>30000;

* Relational operators (=,>,<,>=,<=,<>):

eg: select ename from employee where gender <> ‘M’;

 select ename from employee where gender=’F’;

* Logical operators :

Logical OR=’||’

Logical AND= &&

Logical Not= !

1. eg: select ecode,ename,grade

 from employee

 where( grade=’A1’ or grade=’A2’);

1. select ename from employee

where (not grade=’G1’);

1. select ename from employee

where (grade= ‘E4’ and gross < 9000);

* Condition based on range: using the between operator. It defines a range of values that column value must fall in to make its condition true. it includes both lower and upper value

eg:

Items table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Icode | Descp | Price | QOH | ROL | ROQ |
| 101 | Milk | 15.00 | 20 | 10 | 20 |
| 102 | Cake | 5.00 | 60 | 20 | 50 |
| 103 | Bread | 9.00 | 40 | 10 | 40 |
| 104 | Biscuit | 10.00 | 50 | 40 | 60 |
| 105 | Nmakeen | 15.00 | 100 | 50 | 70 |
| 106 | Creamroll | 7.00 | 10 | 20 | 30 |

1. Select icode,descp,QOH

from items

where QOH between 30 and 50;

1. select icode,descp from items

where ROL NOT between 100 and 1000;

1. Write a query to display item descp and code whose price are in he range 10-20
2. Write a query to display item descp and code whose ROQ is not between 30 and 50
* Condition based on List : using IN/NOT IN operator

Select Icode from Items where Descp in (‘Milk’,’Cake’);

Select Icode from Items where Descp NOT in (‘Milk’);

* Condition based on Pattern Matches- using LIKE keyword
	+ Percent ( %): matches any string
	+ underscore(\_): matches a character

eg: select ecode,ename from employees where ename like “A%”;

 select ecode,ename from employees where ename like “%y”;

 select ecode,ename from employees where ename like “%la”;

 select ecode,ename from employees where ename like “\_a%”;

* Searching for NULL : using IS NULL in Where clause

Select name from PET where death is null;

* Sorting Results –Using Order by clause

Syntax:

Select <column name>[,<column name,….]

From <table name>

Where <condition>

Order by [ASC/DESC] <columnname>;

# Aggregate Functions

An aggregate function performs a calculation on a set of values, and returns a single value.

### ****COUNT**** Function

The COUNT function returns the total number of values in the specified field. It works on both numeric and non-numeric data types. **All aggregate functions by default exclude nulls values before working on the data.**

COUNT (\*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (\*) also considers Nulls and duplicates.

The table shown below shows data in movierentals table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **reference\_ number** | **transaction\_ date** | **return\_date**  | **membership\_ number** | **movie\_id** | **movie\_ returned** |
| 11 | 20-06-2012 | NULL | 1 | 1 | 0 |
| 12 | 22-06-2012 | 25-06-2012 | 1 | 2 | 0 |
| 13 | 22-06-2012 | 25-06-2012 | 3 | 2 | 0 |
| 14 | 21-06-2012 | 24-06-2012 | 2 | 2 | 0 |
| 15 | 23-06-2012 | NULL | 3 | 3 | 0 |

Let's suppose that we want to get the number of times that the movie with id 2 has been rented out

SELECT COUNT(`movie\_id`) FROM `movierentals` WHERE `movie\_id` = 2;

SELECT COUNT(DISTINCT `movie\_id`) FROM `movierentals`;

### ****MIN**** function

The MIN function **returns the smallest value in the specified table field**.

SELECT MIN(`transaction\_date`) FROM `movierental`;

### MAX function

It **returns the largest value from the specified table field**.

SELECT MAX(`year\_released`) FROM `movies`;

### ****SUM**** function

**SUM works on numeric fields only**. **Null values are excluded from the result returned.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **payment\_ id** | **membership\_ number** | **payment\_ date** | **description** | **amount\_ paid** | **external\_ reference \_number** |
| 1 | 1 | 23-07-2012 | Movie rental payment | 2500 | 11 |
| 2 | 1 | 25-07-2012 | Movie rental payment | 2000 | 12 |
| 3 | 3 | 30-07-2012 | Movie rental payment | 6000 | NULL |

SELECT SUM(`amount\_paid`) FROM `payments`;

### ****AVG**** function

AVG function **returns the average of the values in a specified column**

SELECT AVG(`amount\_paid`) FROM `payments`;

**REFERENTIAL INTEGRITY**

- A referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related

tables are valid, and that users don’t accidentally delete or change related data. This integrity is ensured by foreign

key.







|  |  |
| --- | --- |
| Select,Insert,Update,Delete | DML |
| Create,Alter,Drop | DDL |
|  |  |