Select and Aggregate Functions

The SELECT statement is used to retrieve data from a table. SELECT statement has many optional elements that we can use. The order of FROM, WHERE, GROUP BY, HAVING, ORDER BY and LIMIT has to be in the sequence mentioned. To select all columns in a table you can use asterisk (*) notation instead of listing all column names in the SELECT statement. For example, if you need to query all the columns in "employees" table, you can use the following query

```
mysql> SELECT * FROM employees;
```

SELECT statement also allows you to view partial data of a table by listing columns' name after the SELECT keyword. This is called **Projection**.

For example if you need to view only "firstname", "lastname" and "jobtitle" of employee in the "employees" table, you can use the following query

```
mysql> SELECT firstname, lastname, jobtitle FROM employees;
```

Where Clause

WHERE clause of the SELECT statement enables you to select particular rows which match its conditions or search criteria. You use WHERE clause to filter the records based on a certain conditions. For example, you can find the details of president of company by using the following query

```
mysql> SELECT firstname, lastname, email FROM employees WHERE jobtitle = ''president'';
```

Distinct Clause

With DISTINCT clause, you can eliminate the duplicate records while displaying the result. For example, to find how many job titles of all employees in the employees table, you use DISTINCT keyword in SELECT statement as follows

```
mysql> SELECT DISTINCT jobtitle FROM employees;
mysql> SELECT DISTINCT (jobtitle) FROM employees;
```

The DISTINCT clause can be applied with more than one column. In this case, the combinations of all columns are used to define the uniqueness of a record in the return result set.

For example, to get all cities and states of customers in the customers table, we can use the following query


```
mysql> SELECT DISTINCT city, state FROM customers;
```

ORDER BY Clause

The ORDER BY clause allows you to sort the result set on one or more columns in ascending or descending order. To sort the result set in ascending order you use ASC and in descending order you use DESC keywords. By default, the ORDER BY will sort the result set in ascending order. For example, to sort the name of employees by firstname, you can execute the following query

```
mysql>SELECT firstname, lastname, jobtitle FROM employees ORDER BY firstname;
```

LIMIT Clause

MySQL supports a feature called LIMIT to allow you to constrain the returned records with SELECT statement.

Let's say you have a database table with 10000 records and you want to get just first N records, you can use the following query

Syntax

```
mysql> SELECT * FROM table LIMIT N;
```

Example If you want to get the first five employees in the table employees, you can use the following query

```
mysql> SELECT firstname, lastname FROM employees LIMIT 5;
```

The MySQL LIMIT also allows you to get a range of records where you decide starting record number and how many records you want to retrieve.

Syntax

```
mysql> SELECT columns FROM table LIMIT S, N;
```


NOTE In the query above, S is the starting record index. MySQL specifies that the first record starts with 0. N is the number of records you want to select.

Example Now if you want to get five employees from employee number 10 you can use MySQL LIMIT with offset as follows

```
mysql> SELECT firstname, lastname FROM employees LIMIT 10, 5;
```

IN Operator

SQL IN allows you to select values which match any one of a list of values. The usage of SQL IN is as follows

Syntax

```
SELECT columnlist FROM tablename WHERE column IN (‘‘listitem1’’,‘‘listitem2’’);
```

Example Suppose if you want to find out all offices which are located in US and France, you can perform the following query

```
mysql> SELECT officeCode, city, phone FROM offices WHERE country IN (‘USA’, ‘France’);
```

Example To get all countries which are not located in USA or France, we can use NOT IN in the where clause as follows

```
mysql> SELECT officeCode, city, phone FROM offices WHERE country NOT IN (‘USA’, ‘France’);
```

LIKE Operator

MySQL provides LIKE operator in SQL standard. The MySQL LIKE operator is commonly used to select data based on patterns matching.
MySQL provides you two wildcard characters for using with LIKE

The Percentage (%) wildcard allows you to match any string of zero or more characters. Underscore (Example) allows you to match any single character.

Syntax


```
SELECT * from table where column LIKE PATTERN;
```

Example Suppose you want to search for employee in employees table who has first name starting with character 'a', you can do it as follows

```
mysql> SELECT * FROM employees WHERE firstname LIKE 'a%';
```

Functions

MySQL has many built-in functions for performing calculations on data as follows

Aggregrate Functions

Aggregate functions return a single value, calculated from values in a column.

AVG():

The AVG(): function returns the average value of a numeric column.

Syntax

```
SELECT AVG(columnname) FROM tablename;
```

Example

```
sql>SELECT AVG(SAL) from employees;
```

COUNT():

The COUNT(): function returns the number of rows that matches a specified criteria.

textbfSyntax

```
SELECT COUNT(columnname) FROM tablename;
```


The COUNT(columnname) function returns the number of values (NULL values will not be counted) of the specified column

textbfExample

```
mysql> SELECT COUNT(empno) FROM employees;
```

MAX():

The MAX(): function returns the largest value of the selected column.

Syntax

```
SELECT MAX(columnname) FROM tablename;
```

Example

```
sql>SELECT MAX(sal) FROM employees;
```

MIN():

The MIN(): function returns the smallest value of the selected column.

Syntax

```
SELECT MIN(columnname) FROM tablename;
```

Example

```
sql>SELECT MIN(sal) FROM employees;
```

SUM():

The SUM(): function returns the total sum of a numeric column.

Syntax

```
SELECT SUM(columnname) FROM tablename;
```

Example

```
sql>SELECT SUM(sal) FROM employees;
```


ROUND():

The ROUND(): function rounds a numeric field to the number of decimals specified.

Syntax

```
SELECT ROUND(columnname,number) FROM tablename;
```

Example

```
mysql> select ROUND(sal,2) from employees;
```

TRUNCATE():

The TRUNCATE(): truncates a numeric field to the number of decimals specified.

Syntax

```
SELECT TRUNCATE(columnname,number) FROM tablename;
```

Example

```
mysql> SELECT TRUNCATE(sal,2) from employees;
```

SCALAR Functions

UCASE():

Converts a field to upper case.

Syntax

```
SELECT UCASE(columnname) FROM tablename;
```

Example

```
mysql> SELECT UCASE(ename) from employees;
```


LCASE():

Converts a field to lower case.

Syntax

```
SELECT LCASE(columnname) FROM tablename;
```

Example

```
mysql> SELECT LCASE(ename) FROM employees;
```

LENGTH():

Returns of the length of the text field.

Syntax

```
SELECT LENGTH(columnname) FROM tablename;
```

Example

```
mysql> SELECT LCASE(ename) FROM employees;
```

MID():

Extracts character from a text field.

Syntax

```
SELECT MID(column,pos) from tablename;
```

Example

```
mysql> SELECT MID(ename,3) FROM emp WHERE ename='SANTOSH';
```

Note Above specified 3 is the starting character and ending character.

```
mysql> SELECT MID(ename,1,3) FROM emp WHERE ename='SANTOSH';
```

Note Above specified 1 is the starting character and 3 is the ending character.

REPLACE():

Returns the string with all occurrences of the string fromExamplestr replaced by the string toExamplestr. **Syntax**

```
SELECT REPLACE(str,from{Example}str,to_str) FROM tablename;
```

Example

```
\begin{verbatim} mysql> SELECT REPLACE(ename,'a','A') FROM employee;
```

DATE Functions

NOW():

Returns the current date and time.

Example

```
mysql> SELECT NOW(): FROM dual;
```

CURDATE():

Returns the current date.

Example

```
mysql> SELECT CURDATE(): FROM dual;
```

CURTIME():

Returns the current time.

Example

```
mysql> SELECT CURTIME(): FROM dual;
```

EXTRACT():

Extracts a single part of a date/time.

Example


```
mysql> SELECT EXTRACT(year FROM now():) FROM dual;
mysql> SELECT EXTRACT(month FROM now():) FROM dual;
mysql> SELECT EXTRACT(day FROM now():) FROM dual;
mysql> SELECT EXTRACT(hour FROM now():) FROM dual;
mysql> SELECT EXTRACT(minute FROM now():) FROM dual;
mysql> SELECT EXTRACT(second FROM now():) FROM dual;
```

DATE_ADD():

Adds a specified time interval to a date.

Example

```
mysql> SELECT DATE_ADD(NOW():,INTERVAL 10 DAY) FROM dual;
```

DATE_SUB():

Subtracts a specified time interval from a date.

Example

```
mysql> SELECT DATE_SUB(NOW():,INTERVAL 10 DAY) FROM dual;
```

DATEDIFF():

Returns the number of days between two dates.

Example

```
mysql> SELECT DATEDIFF(NOW():,'20130612') FROM dual;
mysql> SELECT DATEDIFF(NOW():,'20000612') FROM dual;
mysql> SELECT (DATEDIFF(NOW():,'20000612'))/365 FROM dual;
```


DATE_FORMAT():

Specifies the format for date and time.

Formats

Format	Description
%M	Month Name
%m	Month, in Numeric
%D	Day of Month Numeric(th)
%d	Day of Month numberic
%Y	4 Digits

Example

```
mysql> SELECT DATE_FORMAT(NOW():,'%d%m%y') FROM dual;
mysql> SELECT DATE_FORMAT(NOW():,'%D%M%Y') FROM dual;
```

ORDER BY Clause

The ORDER BY clause allows you to sort the result set on one or more columns in ascending or descending order. To sort the result set in ascending order you use ASC and in descending order you use DESC keywords. By default, the ORDER BY will sort the result set in ascending order. For example, to sort the name of employees by firstname, you can execute the following query:

```
mysql>SELECT firstname, lastname, jobtitle FROM employees ORDER BY firstname;
```

GROUP BY Clause

The MySQL GROUP BY clause is used with SQL SELECT statement to group selected records into a set of summary records by the one or more column's value or expression.
The MySQL GROUP BY clause must appear after the WHERE clause or FROM clause if WHERE clause is omitted of the SQL SELECT statement.

Syntax

```
SELECT columname(s) from tablename GROUP BY column(s);
```

Example

Let's say if you want to know how many orders in each status group you can use the COUNT function as follows:


```
mysql> SELECT status, count (*) FROM orders GROUP BY status;
```

If you want to see the result of the query above in the descending order, you can do it as follows:

```
mysql> SELECT status, count (*) FROM orders GROUP BY status DESC;
```

HAVING Clause

The HAVING clause is an optional part of and used only with the SQL SELECT statement. The HAVING clause specifies a filter condition for a group of record or an aggregate.

The HAVING is often used with GROUP BY clause. When using with GROUP BY clause, you can apply filter condition of the HAVING clause only to the columns appear in the GROUP BY clause.

NOTE If the GROUP BY clause is omitted, the HAVING clause will behave like a WHERE clause.

HAVING clause applies to groups as a whole while the WHERE clause applies to individual rows.

Example

What order has total value greater than 1000. In this case, you need to use the MySQL HAVING clause on aggregate to answer that question.

```
mysql> SELECT ordernumber, SUM (price) AS total FROM orderdetails
       GROUP BY ordernumber HAVING total > 1000;
```

