MySQL

DDL and DML Statements

DDL:

DDL is short name of Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)

ALTER - alters the structure of the existing database

DROP - delete objects from the database

TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed.

COMMENT - add comments to the data dictionary
RENAME - rename an object

DML:

DML is short name of Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

SELECT - retrieve data from a database
INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - Delete all records from a database table
MERGE - UPSERT operation (insert or update)
CALL - call a PL/SQL or Java subprogram
EXPLAIN PLAN - interpretation of the data access path
LOCK TABLE - concurrency Control

MySQL SELECT:

MySQL SELECT statement is used to fetch data from a database table.

SYNTAX:

```
SELECT * FROM table_name

SELECT column_name(s) FROM table_name
```

MySQL WHERE:

The WHERE clause is used to filter records at the time of SELECT..

SYNTAX:

```
SELECT [*] FROM [Table_name]
WHERE [condition1] [AND [OR]]
[condition2]...
```

- WHERE clause can be used to apply various comma separated condition, in one or more tables.

- Using the WHERE clause to select the specified condition.

- Specific conditions using AND or OR operators.

- A WHERE clause can be used with DELETE or UPDATE.

SELECT without Table:

You can also issue SELECT without a table. For example, you can SELECT an expression or evaluate a built-in function.

```
mysql> SELECT 1+1;
+-----+
| 1+1 |
+-----+
|   2 |
+-----+


mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2012-10-24 22:13:29 |
+---------------------+
```

// Multiple columns

```
mysql> SELECT 1+1, NOW();
+-----+---------------------+
| 1+1 | NOW()               |
+-----+---------------------+
|   2 | 2012-10-24 22:16:34 |
+-----+---------------------+
```

PRODUCT TABLE

Database: product_DB
Table: product_TB

productID INT	productCode CHAR(3)	name VARCHAR(30)	quantity INT	price DECIMAL(10,2)
1001	PEN	Pen Red	5000	1.23
1002	PEN	Pen Blue	8000	1.25
1003	PEN	Pen Black	2000	1.25
1004	PEC	Pencil 2B	10000	0.48
1005	PEC	Pencil 2H	8000	0.49

```
-- List all rows for the specified columns

SELECT name, price FROM products;
+-----------+-------+
| name      | price |
+-----------+-------+
| Pen Red   |  1.23 |
| Pen Blue  |  1.25 |
| Pen Black |  1.25 |
| Pencil 2B |  0.48 |
| Pencil 2H |  0.49 |
+-----------+-------+
```

Comparison Operators:

For numbers (INT, DECIMAL, FLOAT), you could use comparison operators: '=' (equal to), '<>' or '!=' (not equal to), '>' (greater than), '<' (less than), '>=' (greater than or equal to), '<=' (less than or equal to), to compare two numbers. For example, price > 1.0, quantity <= 500.

mysql> SELECT name, price FROM products WHERE price < 1.0;

```
+-----------+-------+
| name      | price |
+-----------+-------+
| Pencil 2B |  0.48 |
| Pencil 2H |  0.49 |
+-----------+-------+
```

mysql> SELECT name, quantity FROM products WHERE quantity <= 2000;

```
+-----------+----------+
| name      | quantity |
+-----------+----------+
| Pen Black |     2000 |
+-----------+----------+
```

For strings, you could also use '=', '<>', '>', '<', '>=', '<=' to compare two strings (e.g., productCode = 'PEC'). The ordering of string depends on the so-called collation chosen. For example,

mysql> SELECT name, price FROM products WHERE productCode = 'PEN';

```
+-----------+-------+
| name      | price |
+-----------+-------+
| Pen Red   |  1.23 |
| Pen Blue  |  1.25 |
| Pen Black |  1.25 |
+-----------+-------+
```