**MySQL Notes**

1. Mysql Quick Guide -

<https://www.tutorialspoint.com/mysql/mysql-quick-guide.htm>

2) Mysql complete guide (very IMP)

<https://www.guru99.com/mysql-tutorial.html>

3) MySQL and python -

<https://www.tutorialspoint.com/python/python_database_access.htm>

4) Install gui for MySQL -

**Steps -**

1. Install MySQL
2. Create database -

$ sudo mysqladmin -u root -p create NEW\_DATABASE\_NAME

Password for root : (Password for root)

Password : (Password for database)

1. Start Mysql -

$ mysql

1. mysql> show databases; ……….. This will show all databases
2. mysql> use NEW\_DATABASE\_NAME ……….. To open particular database
3. mysql> show tables;
4. Create table (table name - accounts)

Mysql > create table accounts(

-> accnt\_no INT NOT NULL AUTO\_INCREMENT,

-> first\_name VARCHAR(40) NOT NULL,

-> last\_name VARCHAR(40) NOT NULL,

-> phone\_no INT,

-> PRIMARY KEY(accnt\_no)

-> );

1. Give initial value = 1 to auto increment account number

mysql> ALTER TABLE accounts AUTO\_NCREMENT=1;

1. If you want to add new column to existing table -

mysql> ALTER TABLE accounts ADD balance INT NOT NULL;

1. **To add entry to table -**

mysql> INSERT INTO accounts

-> (first\_name, last\_name, phone\_no, balance)

-> VALUES

-> ('Parag', 'pan', 12, 0);

Same thing can be done using python code -

def add\_entry():

first\_name = raw\_input('Enter first name = ')  
 Last\_name = raw\_input('Enter last name = ')  
 phone\_no = int(raw\_input('Enter Phone number = '))  
   
 # prepare a cursor object using cursor() method  
 cursor = db.cursor()  
  
 # Prepare SQL query to INSERT a record into the database.  
 sql = "INSERT INTO accounts(first\_name, last\_name, phone\_no, balance)   
 VALUES ('%s', '%s', '%d', 0)" % (first\_name, Last\_name, phone\_no)  
  
 try:  
 # Execute the SQL command  
 cursor.execute(sql)  
 # Commit your changes in the database  
 db.commit()  
 except:  
 # Rollback in case there is any error  
 print("Table entry Insert exception occured")  
 db.rollback()

1. **Display all entries from table**

*Using mysql -*

mysql> SELECT \* FROM accounts;

*Using python code -*

def display(db):  
 print("Printing data")  
  
 cursor = db.cursor()  
  
 sql = "SELECT \* FROM accounts"  
  
 try:  
 # Execute the SQL command  
 cursor.execute(sql)  
 # Fetch all the rows in a list of lists.  
 results = cursor.fetchall()  
 for row in results:  
 accntno = row[0]  
 fname = row[1]  
 lname = row[2]  
 phn = row[3]  
 balance = row[4]  
 # Now print fetched result  
 print "Account no. = %d, fname = %s,lname = %s,phone no. = %d, balance = %d" % \  
 (accntno, fname, lname, phn, balance)  
 except:  
 print "Error: unable to fecth data"

1. **Modify particular entry from table -**

*Using mysql -*

mysql> UPDATE accounts SET first\_name = 'vicky' WHERE accnt\_no = 3;

**Commands -**

1. **SELECT**

SELECT [DISTINCT|ALL ] { \* | [fieldExpression [AS newName]} FROM tableName [alias] [WHERE condition][GROUP BY fieldName(s)] [HAVING condition] ORDER BY fieldName(s)

* **SELECT** is the SQL keyword that lets the database know that you want to retrieve data.
* **[DISTINCT | ALL]** are optional keywords that can be used to fine tune the results returned from the SQL SELECT statement. If nothing is specified then ALL is assumed as the default.
* **{\*| [fieldExpression [AS newName]}** at least one part must be specified, "\*" selected all the fields from the specified table name, fieldExpression performs some computations on the specified fields such as adding numbers or putting together two string fields into one.
* **FROM** tableName is mandatory and must contain at least one table, multiple tables must be separated using commas or joined using the JOIN keyword.
* **WHERE** condition is optional, it can be used to specify criteria in the result set returned from the query.
* **GROUP BY** is used to put together records that have the same field values.
* **HAVING** condition is used to specify criteria when working using the GROUP BY keyword.
* **ORDER BY** is used to specify the sort order of the result set.

More examples of SELECT : (IMP)

<https://www.guru99.com/select-statement.html>

1. **WHERE-**

**"WHERE"** is the keyword that restricts our select query result set and **"condition"** is the filter to be applied on the results. The filter could be a range, single value or sub query.

can be used with AND, OR, IN, NOT IN

<https://www.guru99.com/where-clause.html>

Where can be used with = , > , < , <>

= …. Equal to, > …. Greater than, <> …. Not equal to

1. **INSERT INTO -**

The INSERT command creates a new row in the table to store data.

<https://www.guru99.com/insert-into.html>

INSERT INTO `table\_name`(column\_1,column\_2,...) VALUES (value\_1,value\_2,...);

Explanation -

INSERT INTO `table\_name` is the command that tells MySQL server to add new row into a table named `table\_name`.

(column\_1,column\_2,...) specifies the columns to be updated in the new row

VALUES (value\_1,value\_2,...) specifies the values to be added into the new row

When supplying the data values to be inserted into the new table, the following should be considered while dealing with different data types.

* **String data types** - all the string values should be enclosed in single quotes.
* **Numeric data types** - all numeric values should be supplied directly without enclosing them in single or double quotes.
* **Date data types** - enclose date values in single quotes in the format 'YYYY-MM-DD'.

**If we are supplying values for ALL the columns in the table, then we can omit the columns from the insert query.**

Example:-

INSERT INTO `members` VALUES (9,'Howard Wolowitz','Male','1981-08-24','SouthPark','P.O. Box 4563', '0987786553', 'lwolowitz[at]email.me');

**The INSERT command can also be used to insert data into a table from another table. The basic syntax is as shown below.**

INSERT INTO table\_1 SELECT \* FROM table\_2;

1. **DELETE**

Delete command comes in handy to delete temporary or obsolete data from your database.The DELETE command can delete more than one row from a table in a single query.

DELETE FROM `table\_name` [WHERE condition];

DELETE FROM `table\_name` tells MySQL server to remove rows from the table ..

[WHERE condition] is optional and is used to put a filter that restricts the number of rows affected by the DELETE query.

*you cannot delete a single column for a table. You can delete an entire row.*

1. **UPDATE**

The Update command is used to modify rows in a table. The update command can be used to update a single field or multiple fields at the same time. It can also be used to update a table with values from another table .

UPDATE `table\_name` SET `column\_name` = `new\_value' [WHERE condition];

UPDATE `table\_name` is the command that tells MySQL to update the data in a table

SET `column\_name` = `new\_value' are the names and values of the fields to be affected by the update query. Note, when setting the update values, strings data types must be in single quotes. Numeric values do not need to be in quotation marks. Date data type must be in single quotes and in the format 'YYYY-MM-DD'.

[WHERE condition] is optional and can be used to put a filter that restricts the number of rows affected by the UPDATE query.

1. **ORDER BY - ASC / DESC**

The order by clause is used to sort the query result sets in either ascending or descending order. It is used in conjunction with the SELECT query. It has the following basic syntax.

SELECT statement... [WHERE condition | GROUP BY `field\_name(s)` HAVING condition] ORDER BY `field\_name(s)` [ASC | DESC];

**"SELECT statement..."** is the regular select query

**" | "** represents alternatives

**"[WHERE condition | GROUP BY `field\_name(s)` HAVING condition"** is the optional condition used to filter the query result sets.

**"ORDER BY"** performs the query result set sorting

**"[ASC | DESC]"** is the keyword used to sort result sets in either ascending or descending order. Note ***ASC*** is used as the default.

The DESC sort keyword has the following basic syntax.

SELECT {fieldName(s) | \*} FROM tableName(s) [WHERE condition] ORDER BY fieldname(s) ASC /DESC [LIMIT N]

**HERE**

**SELECT {fieldName(s) | \*} FROM tableName(s)** is the statement containing the fields and table(s) from which to get the result set from.

**[WHERE condition]** is optional but can be used to filter the data according to the given condition.

**ORDER BY** fieldname(s) is mandatory and is the field on which the sorting is to be performed. The DESC keyword specifies that the sorting is to be in descending order.

**[LIMIT]** is optional but can be used to limit the number of results returned from the query result set.

Both DESC and ASC work in conjunction with the ORDER BY keyword. They can also be used in combination with other keywords such as WHERE clause and LIMIT

1. **GROUP BY**

is used to group rows that have the same values.

<https://www.guru99.com/group-by.html>

SELECT statements... GROUP BY column\_name1[,column\_name2,...] [HAVING condition];

"SELECT statements..." is the standard SQL SELECT command query.

"GROUP BY *column\_name1*" is the clause that performs the grouping based on column\_name1.

"[,column\_name2,...]" is optional; represents other column names when the grouping is done on more than one column.

"[HAVING condition]" is optional; it is used to restrict the rows affected by the GROUP BY clause. It is similar to the WHERE clause.

1. **Wildcars (% , \_ )**

% -

Suppose we want to get all the movies that have the word "code" as part of the title, we would use the percentage wildcard to perform a pattern match on both sides of the word "code". Below is the SQL statement that can be used to achieve the desired results.

SELECT \* FROM movies WHERE title LIKE '%code%';

### **\_ underscore wildcard**

The underscore wildcard is used to **match exactly one character**. Let's suppose that we want to search for all the movies that were released in the years 200x where x is exactly one character that could be any value. We would use the underscore wild card to achieve that. The script below select all the movies that were released in the year "200x"

SELECT \* FROM movies WHERE year\_released LIKE '200\_';

### **Escape**

The ESCAPE keyword is used to **escape pattern matching characters** such as the (%) percentage and underscore (\_) if they form part of the data.

Let's suppose that we want to check for the string "67%" we can use;

LIKE '67#%%' ESCAPE '#';

If we want to search for the movie "67% Guilty", we can use the script shown below to do that.

SELECT \* FROM movies WHERE title LIKE '67#%%' ESCAPE '#';

Note the double "**%**%" in the LIKE clause, the first one in red "**%**" is treated as part of the string to be searched for. The other one is used to match any number of characters that follow.

The same query will also work if we use something like

SELECT \* FROM movies WHERE title LIKE '67=%%' ESCAPE '=';

1. **REGEXP**

Regular expressions provide a powerful and flexible pattern match that can help us implement power search utilities for our database systems.

<https://www.guru99.com/regular-expressions.html>

SELECT statements... WHERE fieldname REGEXP 'pattern';

**HERE -**

**"SELECT statements..."** is the standard SELECT statement

**"WHERE fieldname"** is the name of the column on which the regular expression is to be performed on.

**"REGEXP 'pattern'"** REGEXP is the regular expression operator and 'pattern' represents the pattern to be matched by REGEXP. **RLIKE** is the **synonym for REGEXP** and achieves the same results as REGEXP. To avoid confusing it with the LIKE operator, it **better to use REGEXP** instead.

SELECT \* FROM `movies` WHERE `title` REGEXP 'code';

The above query searches for all the movie titles that have the word code in them. It does not matter whether the "code" is at the beginning, middle or end of the title. As long as it is contained in the title then it will be considered.

Let's suppose that we want to search for movies that start with a, b, c or d , followed by any number of other characters, how would we go about to achieve that. We can use a regular expression together with the metacharacters to achieve our desired results.

SELECT \* FROM `movies` WHERE `title` REGEXP '^[abcd]';

SELECT \* FROM `movies` WHERE `title` REGEXP '^[^abcd]';

'^[^abcd]' the caret (^) means that the pattern match should be applied at the beginning and the charlist [^abcd] means that the movie titles starting with any of the enclosed characters is excluded from the result set.

1. **Functions**

MySQL comes bundled with a number of built in functions. Built in functions are simply functions come already implemented in the MySQL server.

Strings functions, Numeric functions, Date functions, [Aggregate functions](https://www.guru99.com/aggregate-functions.html),

MySQL also supports user defined functions that extend MySQL. User defined functions are functions that you can create using a programming language such as C, C++ etc. and then add them to MySQL server. Once added, they can be used just like any other function

Aggregate functions -

* MySQL supports all the five (5) ISO standard aggregate functions COUNT, SUM, AVG, MIN and MAX.
* SUM and AVG functions only work on numeric data.

1. **JOINS**

Joins help retrieving data from two or more database tables. The tables are mutually related using primary and foreign keys.

<https://www.guru99.com/joins.html>

**List of Keywords in SQL-**

1. SELECT
2. Concat
3. AS (alias)
4. LEFT - select few characters from string
5. WHERE
6. AND, OR, IN, NOT IN ……… (used with WHERE)
7. = , > , <> ………..(used with WHERE)
8. INSERT INTO
9. UPDATE
10. DELETE
11. ORDER BY
12. ASC
13. DESC
14. length ….. To get length of string ----> length(*variable\_name*)
15. LIMIT ……. To limit number results returned
16. GROUP BY
17. HAVING
18. COUNT ……… to count number of entries
19. MIN, MAX, SUM , AVG
20. ROUND ……. Rounding operation e.g. ---- ROUND(name, 1), rounding by 1 decimal
21. CEIL ………. Rounding off to upper value
22. DISTINCT …… to avoid duplicate results
23. LIKE ………. Can be used with wild cards ---- % and \_
24. NOT LIKE
25. ESCAPE

# REGEXP

1. NULL
2. AUTO\_INCREMENT (By default, the starting value for AUTO\_INCREMENT is 1, and it will increment by 1 for each new record. To let AUTO\_INCREMENT sequence start with another value , use AUTO\_INCREMENT = 10)
3. ALTER …. Modify existing table or database
4. UNION…….. combine the results from multiple SELECT queries into a consolidated result
5. UNION DISTINCT / UNION ALL
6. JOINS
7. RIGHT ….. To compare string from back side …

e.g. SELECT DISTINCT CITY FROM STATION WHERE RIGHT(CITY,1) IN (‘a’,’e’,’i’,’o’,’u’);

1. REPLACE E.g. - <https://www.hackerrank.com/challenges/the-blunder/forum>