| **Command** | **Description** |
| --- | --- |
| CREATE DATABASE DATABASE; | Create database |
| CREATE DATABASE NOT EXISTS database1; | IF NOT EXISTS let you to instruct MySQL server to check the existence of a database with a similar name prior to creating database. |
| CREATE DATABASE IF NOT EXISTS database1 CHARACTER SET latin1 COLLATE latin1\_swedish\_ci | the Latin1 character set uses the latin1\_swedish\_ci collation which is the Swedish case insensitive order. |
| SHOW DATABASES | You can see list of existing databases by running following SQL command. |
| CREATE TABLE [IF NOT EXISTS] TableName (fieldname dataType [optional parameters]) ENGINE = storage Engine; | Create table syntax |

**DATA TYPES**

**Numeric Data types**

| **Command** | **Description** |
| --- | --- |
| TINYINT( ) | -128 to 127 normal 0 to 255 UNSIGNED. |
| SMALLINT( ) | -32768 to 32767 normal 0 to 65535 UNSIGNED. |
| MEDIUMINT( ) | -8388608 to 8388607 normal 0 to 16777215 UNSIGNED. |
| INT( ) | -2147483648 to 2147483647 normal 0 to 4294967295 UNSIGNED. |
| BIGINT( ) | -9223372036854775808 to 9223372036854775807 normal 0 to 18446744073709551615 UNSIGNED. |
| FLOAT | A small approximate number with a floating decimal point. |
| DOUBLE( , ) | A large number with a floating decimal point. |
| DECIMAL( , ) | A DOUBLE stored as a string , allowing for a fixed decimal point. Choice for storing currency values. |

**Text Data Types**

| **Command** | **Description** |
| --- | --- |
| CHAR( ) | A fixed section from 0 to 255 characters long. |
| VARCHAR( ) | A variable section from 0 to 255 characters long. |
| TINYTEXT | A string with a maximum length of 255 characters. |
| TEXT | A string with a maximum length of 65535 characters. |
| BLOB | A string with a maximum length of 65535 characters. |
| MEDIUMTEXT | A string with a maximum length of 16777215 characters. |
| MEDIUMBLOB | A string with a maximum length of 16777215 characters. |
| LONGTEXT | A string with a maximum length of 4294967295 characters. |
| LONGBLOB | A string with a maximum length of 4294967295 characters. |

**Date / Time data types**

| **Command** | **Description** |
| --- | --- |
| DATE | YYYY-MM-DD |
| DATETIME | YYYY-MM-DD HH:MM:SS |
| TIMESTAMP | YYYYMMDDHHMMSS |
| TIME | HH:MM:SS |

**Other data types**

| **Command** | **Description** |
| --- | --- |
| ENUM | To store text value chosen from a list of predefined text values. |
| SET | This is also used for storing text values chosen from a list of predefined text values. It can have multiple values. |
| BOOL | Synonym for TINYINT(1), used to store Boolean values |
| BINARY | Similar to CHAR, difference is texts are stored in binary format. |
| VARBINARY | Similar to VARCHAR, difference is texts are stored in binary format. |

**MySQL SELECT statement command**

| **Command** | **Description** |
| --- | --- |
| SELECT [DISTINCT|ALL ] { \* | [fieldExpression [AS newName]} FROM tableName [alias] [WHERE condition][GROUP BY fieldName(s)] [HAVING condition] ORDER BY fieldName(s) | SQL SELECT statement syntax |
| SELECT \* FROM table1; | select the table |
| SELECT t1,t2,t3, t4 FROM table1; | we are only interested in getting only the t1, t2, t3 and t4 fields only. |
| SELECT Concat(t1, (, t3, )) , t4 FROM table2; | Getting table2 listing |
| SELECT column\_name|value|expression [AS] alias\_name; | Alias field names syntax |

**MySQL WHERE clause with AND,OR,IN,NOT IN commands**

| **Command** | **Description** |
| --- | --- |
| SELECT \* FROM tableName WHERE condition; | WHERE clause Syntax |
| SELECT \* FROM table1 WHERE t1 = 2 AND t2 = 2008; | WHERE clause combined with - AND LOGICAL Operator |
| SELECT \* FROM table1 WHERE t1 = 1 OR t1 = 2; | WHERE clause combined with - OR LOGICAL Operator |
| SELECT \* FROM table2 WHERE t1 IN (1,2,3); | WHERE clause combined with - IN Keyword |
| SELECT \* FROM table2 WHERE t1 NOT IN (1,2,3); | WHERE clause combined with - NOT IN Keyword |
| SELECT \* FROM table2 WHERE t3 = Female; | WHERE clause combined with Equal(=) to COMPARISON OPERATORS |
| SELECT \* FROM table3 WHERE t3 > 2000; | WHERE clause combined with greater than(>) to COMPARISON OPERATORS |
| SELECT \* FROM table1 WHERE t1<> 1; | WHERE clause combined with Not Equal to (<>)COMPARISON OPERATORS |

**MySQL Command INSERT INTO Table**

| **Command** | **Description** |
| --- | --- |
| INSERT INTO table\_name(column\_1,column\_2,...) VALUES (value\_1,value\_2,...); | basic syntax of the SQL INSERT command |
| INSERT INTO table1 (t1,t2,t3,t4) VALUES (X1,X2,X3,X4); | INSERT data into table |
| INSERT INTO table\_1 SELECT \* FROM table\_2; | Inserting into a Table from another Table |

**MySQL DELETE command**

| **Command** | **Description** |
| --- | --- |
| DELETE FROM table\_name [WHERE condition]; | Delete a row in MySQL |

**Example :-**  
DELETE FROM table1 WHERE table1\_id = 18;  
(delete entry of 18 number id form table1.)  
DELETE FROM table1 WHERE table1\_id IN (20,21);  
(delete entry of 20 and 21 number id form table1)

**MySQL Update Command**

| **Command** | **Description** |
| --- | --- |
| UPDATE table\_name SET column\_name = new\_value [WHERE condition]; | update command syntax |

**Example :-**  
SELECT \* FROM table1 WHERE t1 = 1;  
(retrieve the record for t1 =1)  
UPDATE table1 SET t4 = X1 WHERE t1 = 1;  
(update the t4 value in table)

**ORDER BY in MySQL: DESC & ASC command**

| **Command** | **Description** |
| --- | --- |
| SELECT statement... [WHERE condition | GROUP BY field\_name(s) HAVING condition] ORDER BY field\_name(s) [ASC | DESC]; | Order by clause basic syntax |
| SELECT {fieldName(s) | \*} FROM tableName(s) [WHERE condition] ORDER BY fieldname(s) ASC /DESC [LIMIT N] | DESC and ASC syntax |

**Example :-**  
For DESC (descending)  
SELECT \* FROM table1 ORDER BY t3 DESC;  
For ASC (ascending)  
SELECT \* FROM table1 ORDER BY t3 ASC;

**MySQL GROUP BY and HAVING Clause command**

**Group by**

| **Command** | **Description** |
| --- | --- |
| SELECT statements... GROUP BY column\_name1[,column\_name2,...] [HAVING condition]; | GROUP BY Syntax |

**Example for grouping a single column :-**  
SELECT t4 FROM table1 ;  
SELECT t4 FROM table1 GROUP BY t4;( suppose we want to get the unique values for t4.)

**Example for grouping a multiple columns :-**  
SELECT t1\_id,t4 FROM table2 ;  
SELECT t1\_id,t4 FROM table2 GROUP BY t1\_id,t4;(using group by method )

**Grouping and aggregate functions**

| **Command** | **Description** |
| --- | --- |
| SELECT t2,COUNT(t1) FROM table1 GROUP BY t2; | Suppose we want the total number of t2 column values in our database. |

**HAVING clause**

| **Command** | **Description** |
| --- | --- |
| SELECT \* FROM table2 GROUP BY t1\_id,t4 HAVING t1\_id = x1; | all the t4 for table2 t1 id x1. We would use the following script to achieve our results. |

**MySQL Wildcards commands for Like, NOT Like, Escape, ( % ), ( \_ )**

**% the percentage wildcards commmand in MySQL**

| **Command** | **Description** |
| --- | --- |
| SELECT statements... WHERE fieldname LIKE xxx%; | basic syntax for % percentage wildcard |

**Example :- we would use the percentage wildcard to perform a pattern match on both sides of the word "X1" as part t2 of table1**  
SELECT \* FROM table1 WHERE t2 LIKE %X1%;  
SELECT \* FROM table1 WHERE t2 LIKE %X1;  
(the percentage wildcard at the beginning of the search criteria only)  
SELECT \* FROM table1 WHERE t2 LIKE X1%;  
(the percentage wildcard to the end of the specified pattern to be matched.)

**\_ underscore wildcard command**

| **Command** | **Description** |
| --- | --- |
| SELECT \* FROM table1 WHERE t3 LIKE x2\_; | all the table1 that were t3 in the year "x2" |

**NOT Like wildcard command**

| **Command** | **Description** |
| --- | --- |
| SELECT \* FROM table1 WHERE t3 NOT LIKE X2\_; | Suppose we want to get table1 that were not t3 in the year X2\_ |

**Escape keyword wildcard command**

| **Command** | **Description** |
| --- | --- |
| LIKE 67#%% ESCAPE #; | we want to check for the string "67%" |

**MYSQL Regular Expressions (REGEXP)**

| **Command** | **Description** |
| --- | --- |
| SELECT statements... WHERE fieldname REGEXP pattern; | basic syntax of Regular Expression |

**Example :-** all the table1 t1 that have the word X1 in them. It does not matter whether the "X1" is at the beginning, middle or end of the title.  
SELECT \* FROM table1 WHERE t1 REGEXP X1;

**Regular expression Metacharacters**

| **Command** | **Description** |
| --- | --- |
| \* | The asterisk (\*) metacharacter is used to match zero (0) or more instances of the strings preceding it |
| + | The plus (+) metacharacter is used to match one or more instances of strings preceding it. |
| ? | The question(?) metacharacter is used to match zero (0) or one instances of the strings preceding it. |
| . | The dot (.) metacharacter is used to match any single character in exception of a new line. |
| [abc] | The charlist [abc] is used to match any of the enclosed characters. |
| [^abc] | The charlist [^abc] is used to match any characters excluding the ones enclosed. |
| [A-Z] | The [A-Z] is used to match any upper case letter |
| [a-z] | The [a-z] is used to match any lower case letter |
| [0-9] | The [0-9] is used to match any digit from 0 through to 9. |
| ^ | The caret (^) is used to start the match at beginning. |
| | | The vertical bar (|) is used to isolate alternatives. |
| [[:<:]] | The[[:<:]] matches the beginning of words. |
| [[:>:]] | The [[:>:]] matches the end of words. |
| [:class:] | The [:class:] matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters. |

**SQL Functions commands**

**String functions**

| **Command** | **Description** |
| --- | --- |
| SELECT t1\_id,t2, UCASE(t2) FROM table1; | the "UCASE" function to do that. It takes a string as a parameter and converts all the letters to upper case. |

**Numeric functions**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| DIV | Integer division | SELECT 23 DIV 6; |
| / | Division | SELECT 23 / 6 ; |
| - | Subtraction | SELECT 23 - 6 ; |
| + | Addition | SELECT 23 + 6 ; |
| \* | Multiplication | SELECT 23 \* 6 AS multiplication\_result; |
| % or MOD | Modulus | SELECT 23 % 6 ; or SELECT 23 MOD 6; |
| Floor | this function removes decimals places from a number and rounds it to the nearest lowest number. | SELECT FLOOR(23 / 6) AS floor\_result; |
| Round | this function rounds a number with decimal places to the nearest whole number. | SELECT ROUND(23 / 6) AS round\_result; |

**Stored functions**

| **Command** | **Description** |
| --- | --- |
| CREATE FUNCTION sf\_name ([parameter(s)]) RETURNS data type DETERMINISTIC STATEMENTS | basic syntax for creating a stored function |
| CREATE FUNCTION sf\_name ([parameter(s)]) | Mandatory and tells MySQL server to create a function named `sf\_name' with optional parameters defined in the parenthesis. |
| RETURNS data type | Mandatory and specifies the data type that the function should return. |
| DETERMINISTIC | The function will return the same values if the same arguments are supplied to it. |
| STATEMENTS | The procedural code that the function executes. |

**MySQL Aggregate function commands**

| **Command** | **Description** |
| --- | --- |
| SELECT COUNT(t1\_id) FROM table1 WHERE t1\_id = 2; | COUNT Function |
| SELECT MIN(t3) FROM table2; | MIN function |
| SELECT MAX(t3) FROM table2; | MAX function |
| SELECT SUM(t4) FROM table3; | SUM function |
| SELECT AVG(t4) FROM table3; | AVG function |

**MySQL IS NULL & IS NOT NULL commands**

| **Command** | **Description** |
| --- | --- |
| SELECT COUNT(t3) FROM table1; ( if t3 have null value present that not count) | Null as a Value |
| CREATE TABLE table2( t1\_number int NOT NULL, t2\_names varchar(255) , t3 varchar(6) ); | NOT NULL Values |
| comlumn\_name IS NULL comlumn\_name NOT NULL | NULL Keywords Basic syntax |
| SELECT \* FROM table1 WHERE t2\_number IS NULL; | Example of IS NULL |
| SELECT \* FROM table1 WHERE t2\_number IS NOT NULL; | Example of IS NOT NULL |

**MySQL AUTO\_INCREMENT commands**

| **Command** | **Description** |
| --- | --- |
| CREATE TABLE table1 ( t1\_id int(11) AUTO\_INCREMENT, t2\_name varchar(150) DEFAULT NULL, t3 varchar(500) DEFAULT NULL, PRIMARY KEY (t1\_id) ); | Auto increment syntax |

**MYSQL - ALTER, DROP, RENAME, MODIFY**

| **Command** | **Description** |
| --- | --- |
| ALTER TABLE table\_name ADD COLUMN column\_name data\_type; | Alter- syntax |
| DROP TABLE sample\_table; | DROP TABLE syntax |
| RENAME TABLE current\_table\_name TO new\_table\_name; | RENAME COMMAND syntax |
| ALTER TABLE table1 CHANGE COLUMN t1\_names t1name char(250) NOT NULL; | CHANGE KEYWORD |
| ALTER TABLE table1MODIFY t1name char(50) NOT NULL; | MODIFY KEYWORD |
| ALTER TABLE table1 ADD t4 date NULL AFTER t3; | AFTER KEYWORD |

**MySQL LIMIT & OFFSET**

| **Command** | **Description** |
| --- | --- |
| SELECT {fieldname(s) | \*} FROM tableName(s) [WHERE condition] LIMIT N; | LIMIT keyword syntax |
| SELECT \* FROM table1 LIMIT 1, 2; | OFF SET in the LIMIT query |

**MySQL SubQuery commands :**

| **Command** | **Description** |
| --- | --- |
| SELECT t1\_name FROM table1 WHERE category\_id =( SELECT MIN(t1\_id) from table2); | sub queries |

**MySQL JOINS commands**

| **Command** | **Description** |
| --- | --- |
| SELECT \* FROM table1 CROSS JOIN table2 | Cross JOIN |
| SELECT table1.t1 , table1.t2 , table2.t1 FROM table1 ,table2 WHERE table2.id = table1.table2\_id | INNER JOIN |
| SELECT A.t1 , B.t2 , B.t3 FROM table2 AS A LEFT JOIN table1 AS B ON B.table2\_id = A.id | LEFT JOIN |
| SELECT A.t1 , A.t2, B.t3 FROM table1 AS A RIGHT JOIN table2 AS B ON B.id = A.table2\_id | RIGHT JOIN |
| SELECT A.t1 , B.t2 , B.t3 FROM table2 AS A LEFT JOIN table1 AS B USING ( table2\_id ) | "ON" and "USING" clauses |

**MySQL UNION commands**

| **Command** | **Description** |
| --- | --- |
| SELECT column1, column2 FROM table1 | UNION syntax |
| SELECT column1,column2 FROM table2; | UNION DISTINCT |

**MySQL in Views commands**

| **Command** | **Description** |
| --- | --- |
| CREATE VIEW view\_name AS SELECT statement; | Views syntax |
| DROP VIEW general\_v\_movie\_rentals ; | Dropping views |

**MySQL Index commands**

| **Command** | **Description** |
| --- | --- |
| CREATE INDEX id\_index ON table\_name(column\_name); | Add index basic syntax |
| DROP INDEX index\_id ON table\_name; | Drop index basic syntax |