MySQL - How to

ALL HOW TOs

Enter command line mode for MySQL

```
    # enter without user specified
mysql

# enter for specific user and prompt for password
mysql -u _USERNAME_ -p
```

NOTE: Substitute the user's MySQL login name for _USERNAME_.

Database level commands

Create database

```
    CREATE DATABASE _DATABASE_NAME_ DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;
```

NOTE: Substitute the name of the database for _DATABASE_NAME_.

Delete database

```
DROP DATABASE _DATABASE_NAME_;
```

NOTE: Substitute the name of the database for _DATABASE_NAME_.

Switch to a database

```
use _DATABASE_NAME_;
```

User level commands

List of users

```
SELECT User FROM mysql.user;
```

NOTE: No substitutions.

Add user

```
CREATE USER '_USERNAME_'@'localhost' IDENTIFIED BY '_USERPASSWORD_';
```

NOTE: Substitute the user's MySQL login name for _USERNAME_ *and the user's MySQL password for* _USERPASSWORD_.

Give user permissions for a specific database

```
    use _DATABASE_NAME_;
GRANT ALL ON _DATABASE_NAME_.* TO _USERNAME_@localhost IDENTIFIED BY '_USERPASSWORD_';
```

NOTE: Substitute the user's MySQL login name for _USERNAME_*, the user's MySQL password for* _USERPASSWORD_*, and the name of the database for _DATABASE_NAME_.*

List databases for current user

```
show databases;
```

NOTE: No substitutions.

Table level commands

Queries for table structure

List tables in a database

```
    show tables from _DATABASE_NAME_;

OR

use _DATABASE_NAME_;
show tables;
```

Show row counts for each table

```
    select table_name, table_rows from information_schema.tables where table_schema='_DATABASE_NAME_';
```

List tables with specified column name

```
    select distinct table_name from information_schema.columns where column_name in ('_COLUMN_NAME_') and
table_schema='_DATABASE_NAME_';
```

List of tables with foreign key IDs -- LIKE ('%_id')

```
    select distinct table_name from information_schema.columns where column_name LIKE ('%_id') and
table_schema='exhibits_staging';
```

List columns in a table

```
    show columns from _TABLE_NAME_ in _DATABASE_NAME_;

OR

use _DATABASE_NAME_;
show columns from _TABLE_NAME_;
```

Queries for table values

List values for a specific column

```
    select _COLUMN_NAME_ from _DATABASE_NAME_._TABLE_NAME_;

OR

use _DATABASE_NAME_;
select _COLUMN_NAME_ from _TABLE_NAME_;
```

List values for a range of rows

```
    select _COLUMN_NAMES_ from _DATABASE_NAME_._TABLE_NAME_ limit _START_, _NUM_ROWS_;

OR

use _DATABASE_NAME_;
select _COLUMN_NAMES_ from _TABLE_NAME_ limit _START_, _NUM_ROWS_;

OR

# sort by id for subset (NOTE: where clause has to come before sorting)
select _COLUMN_NAMES_ from _TABLE_NAME_ where _WHERE_CLAUSE_ order by id;

# sort by id and list last N rows (NOTE: limit has to come after sorting)
select _COLUMN_NAMES_ from _TABLE_NAME_ order by id desc limit _NUM_ROWS_;
```

NOTE: It starts one row after the _START_ value. If you have 0, it will list rows 21-25.

Where with wildcard using LIKE

```
    # % is wildcard.  To force results to start with the query string, eliminate the first %
select _COLUMN_NAMES_ from _TABLE_NAME_ where _COLUMN_NAME_ like '%_QUERY_STRING_%';
```

Get all that don't start with X using <> operator

```
    select id, COL2 from users where substr(COL2,1,5) <> 'guest';
```

Get all that have one of a set of values using IN

```
    select id, COL2 from users where COL2 IN ('vulue1','value2','value3');
```

Count and Count Distinct

```
    select count(id) from TABLE_NAME;
select count(distinct COLUMN_NAME) from TABLE_NAME;
```

Modifying values in tables

Add a record

```
    insert into TABLE_NAME (id, COL2, COL3) values(214,2,'three');
```

Edit a record

```
    replace into TABLE_NAME values(54,2,'new three');            # replaces entire record
update TABLE_NAME set COL_NAME = 'new three' where id = 54;  # where id = 54 limits update specific record
```

Search and replace part of a value in a record's column

```
    update TABLE_NAME set COLUMN_NAME = replace(COLUMN_NAME,'OLD_SUBSTRING', 'NEW_SUBSTRING') where id=35;
   # where id=35 limits update to specific record
```

Delete a record

```
    delete from TABLE_NAME where id=897;
```

Joins

blocked URL

INNER

only when there is an A and a B such that A.Key = B.Key; Ignores all entries in A and B where there isn't a match

```
    SELECT <select_list> FROM TableA A
  INNER JOIN TableB B
  ON A.Key = B.Key
```

LEFT

everything from A + extra stuff from B when A.Key = B.Key

```
    SELECT <select_list> FROM TableA A
  LEFT JOIN TableB B
  ON A.Key = B.Key
```

RIGHT

everything from B + extra stuff from A when A.Key = B.Key

```
    SELECT <select_list> FROM TableA A
  RIGHT JOIN TableB B
  ON A.Key = B.Key
```

OUTER

everything in A and everything in B; lots of sparse data when there is no match A.Key = B.Key

```
  SELECT <select_list> FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```