Commands

Access monitor: `mysql -u [username] -p;` (will prompt for password)

Show all databases: `show databases;`

Access database: `mysql -u [username] -p [database]` (will prompt for password)

Create new database: `create database [database];`

Select database: `use [database];`

Determine what database is in use: `select database();`

Show all tables: `show tables;`

Show table structure: `describe [table];`

List all indexes on a table: `show index from [table];`

Create new table with columns: `CREATE TABLE [table] ([column] VARCHAR(120), [another-column] DATETIME);`

Adding a column: `ALTER TABLE [table] ADD COLUMN [column] VARCHAR(120);`

Adding a column with an unique, auto-incrementing ID: `ALTER TABLE [table] ADD COLUMN [column] int NOT NULL AUTO_INCREMENT PRIMARY KEY;`

Inserting a record: `INSERT INTO [table] ([column], [column]) VALUES ('[value]', '[value]');`

MySQL function for datetime input: `NOW()`

Selecting records: `SELECT * FROM [table];`

Explain records: `EXPLAIN SELECT * FROM [table];`

Selecting parts of records: `SELECT [column], [another-column] FROM [table];`

Counting records: `SELECT COUNT([column]) FROM [table];`

Counting and selecting grouped records: `SELECT *, (SELECT COUNT([column]) FROM [table]) AS count FROM [table] GROUP BY [column];`

Selecting specific records: `SELECT * FROM [table] WHERE [column] = [value];` (Selectors: `<`, `>`, `!=`; combine multiple selectors with `AND`, `OR`)

Select records containing `[value]`: `SELECT * FROM [table] WHERE [column] LIKE '%[value]%';`

Select records starting with `[value]`: `SELECT * FROM [table] WHERE [column] LIKE '[value]%';`

Select records starting with `val` and ending with `ue`: `SELECT * FROM [table] WHERE [column] LIKE '[val_ue]';`

Select a range: `SELECT * FROM [table] WHERE [column] BETWEEN [value1] and [value2];`

Select with custom order and only limit: `SELECT * FROM [table] WHERE [column] ORDER BY [column] ASC LIMIT [value];` (Order: `DESC`, `ASC`)

Updating records: `UPDATE [table] SET [column] = '[updated-value]' WHERE [column] = [value];`

Deleting records: `DELETE FROM [table] WHERE [column] = [value];`

Delete *all records* from a table (without dropping the table itself): `DELETE FROM [table];` (This also resets the incrementing counter for auto generated columns like an id column.)

Delete all records in a table: `truncate table [table];`

Removing table columns: `ALTER TABLE [table] DROP COLUMN [column];`

Deleting tables: `DROP TABLE [table];`

Deleting databases: `DROP DATABASE [database];`

Custom column output names: `SELECT [column] AS [custom-column] FROM [table];`

Export a database dump (more info here): `mysqldump -u [username] -p [database] > db_backup.sql`

Use `--lock-tables=false` option for locked tables (more info here).

Import a database dump (more info here): `mysql -u [username] -p -h localhost [database] < db_backup.sql`

Logout: `exit;`

Aggregate functions

Select but without duplicates: `SELECT distinct name, email, acception FROM owners WHERE acception = 1 AND date >= 2015-01-01 00:00:00`

Calculate total number of records: `SELECT SUM([column]) FROM [table];`

Count total number of `[column]` and group by `[category-column]`: `SELECT [category-column], SUM([column]) FROM [table] GROUP BY [category-column];`

Get largest value in `[column]`: `SELECT MAX([column]) FROM [table];`

Get smallest value: `SELECT MIN([column]) FROM [table];`

Get average value: `SELECT AVG([column]) FROM [table];`

Get rounded average value and group by `[category-column]`: `SELECT [category-column], ROUND(AVG([column]), 2) FROM [table] GROUP BY [category-column];`

Multiple tables

Select from multiple tables: `SELECT [table1].[column], [table1].[another-column], [table2].[column] FROM [table1], [table2];`

Combine rows from different tables: `SELECT * FROM [table1] INNER JOIN [table2] ON [table1].[column] = [table2].[column];`

Combine rows from different tables but do not require the join condition: `SELECT * FROM [table1] LEFT OUTER JOIN [table2] ON [table1].[column] = [table2].[column];` (The left table is the first table that appears in the statement.)

Rename column or table using an *alias*: `SELECT [table1].[column] AS '[value]', [table2].[column] AS '[value]' FROM [table1], [table2];`

Users functions

List all users: `SELECT User,Host FROM mysql.user;`

Create new user: `CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';`

Grant `ALL` access to user for `*` tables: `GRANT ALL ON database.* TO 'user'@'localhost';`

Find out the IP Address of the Mysql Host

`SHOW VARIABLES WHERE Variable_name = 'hostname';` (source)

MySQL Database

To be able to experiment with the code examples, you should have MySQL installed on your computer.

You can download a free MySQL database at https://www.mysql.com/downloads/.

Install MySQL Driver

Once you have MySQL up and running on your computer, you can access it by using Node.js.

To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.

To download and install the "mysql" module, open the Command Terminal and execute the following:

```
C:\Users\Your Name>npm install mysql
```

Now you have downloaded and installed a mysql database driver.

Node.js can use this module to manipulate the MySQL database:

```
var mysql = require('mysql');
```

Create Connection

Start by creating a connection to the database.

Use the username and password from your MySQL database.

demo_db_connection.js

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
```

```
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Save the code above in a file called "demo_db_connection.js" and run the file:

```
Run "demo_db_connection.js"

C:\Users\Your Name>node demo_db_connection.js
```

Which will give you this result:

```
Connected!
```

Now you can start querying the database using SQL statements.

Query a Database

Use SQL statements to read from (or write to) a MySQL database. This is also called "to query" the database.

The connection object created in the example above, has a method for querying the database:

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

The query method takes an sql statements as a parameter and returns the result.

```
// npm install mysql

//import mysql module
const mysql = require('mysql');

//create connection object
const con = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'mydb'
});

//use connect method to connect with db
con.connect((error) => {
  if (error) throw error;
  console.log('Connected to MySQL database!');
});

//query the db
con.query("CREATE DATABASE IF NOT EXISTS mydb ", function (err, result) {
    if (err) throw err;
    console.log("Database created");
  });

  //create table
var sql = "CREATE TABLE IF NOT EXISTS customers (name VARCHAR(255), address
VARCHAR(255))";
con.query(sql, function (err, result) {
  if (err) throw err;
  console.log("Table created");
});

//alter table
var sql = "ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY
KEY";
con.query(sql, function (err, result) {
  if (err) throw err;
  console.log("Table altered");
});

//insert records
var sql = "INSERT INTO customers (name, address) VALUES ('Company Inc',
'Highway 37')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
```

```
  var sql = "INSERT INTO customers (name, address) VALUES ?";
  var values = [
    ['John', 'Highway 71'],
    ['Peter', 'Lowstreet 4'],
    ['Amy', 'Apple st 652'],
    ['Hannah', 'Mountain 21'],
    ['Michael', 'Valley 345'],
    ['Sandy', 'Ocean blvd 2'],
    ['Betty', 'Green Grass 1'],
    ['Richard', 'Sky st 331'],
    ['Susan', 'One way 98'],
    ['Vicky', 'Yellow Garden 2'],
    ['Ben', 'Park Lane 38'],
    ['William', 'Central st 954'],
    ['Chuck', 'Main Road 989'],
    ['Viola', 'Sideway 1633']
  ];
  con.query(sql, [values], function (err, result) {
    if (err) throw err;
    console.log("Number of records inserted: " + result.affectedRows);

    console.log(result);
  });


  var sql = "INSERT INTO customers (name, address) VALUES ('Michelle', 'Blue
Village 1')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted, ID: " + result.insertId);
  });

const user = { name: 'hiteshri', address: 'ahmedabad' };
con.query('INSERT INTO customers SET ?', user, (error, results) => {
  if (error) throw error;
  console.log('User inserted successfully!');
});


//select query
  con.query("SELECT * FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });

  con.query("SELECT name, address FROM customers", function (err, result,
fields) {
    if (err) throw err;
```

```
    console.log(result);
  });

  con.query("SELECT name, address FROM customers", function (err, result,
fields) {
    if (err) throw err;
    //console.log(fields);
    console.log(fields[1].name);
  });

  con.query("SELECT * FROM customers WHERE address = 'Park Lane 38'",
            function (err, result) {
    if (err) throw err;
    console.log(result);
  });

  con.query("SELECT * FROM customers WHERE address LIKE 'S%'", function (err,
result) {
    if (err) throw err;
    console.log(result);
  });

  var adr = 'Mountain 21';
  //Escape the address value:
  var sql = 'SELECT * FROM customers WHERE address = ?';
  //Send an array with value(s) to replace the escaped values:
  con.query(sql, [adr], function (err, result) {
    if (err) throw err;
    console.log(result);
  });

  var name = 'Amy';
var adr = 'Mountain 21';
var sql = 'SELECT * FROM customers WHERE name = ? OR address = ?';
con.query(sql, [name, adr], function (err, result) {
  if (err) throw err;
  console.log(result);
});


con.query("SELECT * FROM customers ORDER BY name", function (err, result) {
  if (err) throw err;
  console.log(result);
});

con.query("SELECT * FROM customers ORDER BY name DESC", function (err, result)
{
  if (err) throw err;
  console.log(result);
```

```
});

var sql = "DELETE FROM customers WHERE address = 'Mountain 21'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Number of records deleted: " + result.affectedRows);
  });


var sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley
345'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result.affectedRows + " record(s) updated");
  });

  var sql = "SELECT * FROM customers LIMIT 5";
  //var sql = "SELECT * FROM customers LIMIT 5 OFFSET 2";
  //var sql = "SELECT * FROM customers LIMIT 2, 5";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });

  var sql = "DROP TABLE IF EXISTS customers";
con.query(sql, function (err, result) {
  if (err) throw err;
  console.log("Table dropped");
});



con.end();
```