1. Windows+r 🡪 Cmd
2. cd C:\Program Files\MySQL\MySQL Server 5.6\bin
3. mysql –u root –p
4. iamrahul

For importing a table

1. use database\_name;
2. source file\_location/filename.sql;

For maximizing the size of the command prompter

1. wmiz
2. maximize it from top right.
3. Exit

Few

SHOW DATABASES;  
CREATE DATABASE company;  
USE new;  
DROP TABLE table1;  
CREATE TABLE table1 (  
id INT PRIMARY KEY,  
name VARCHAR(15),  
address VARCHAR(30) NOT NULL,  
age INT(3)  
);  
  
DESCRIBE table1;  
  
INSERT INTO table1 VALUES (“rahul”, 20);  
  
INSERT INTO table1 VALUES   
(“rahul”, 20),  
(“nakul”,25);  
  
INSERT INTO table1 (age,name) VALUES (20,”rahul”);  
  
INSERT INTO table1 (name, age, addr) VALUES   
SELECT e\_name, e\_age, e\_address FROM table2;  
  
  
 SELECT \* FROM customers;  
  
UPDATE items SET name=”new value”, bids=54 WHERE id=106;  
DELETE FROM items WHERE id=105;  
  
ALTER TABLE table1 ADD new\_column VARCHAR(15) NOT NULL; //to add new column  
  
ALTER TABLE table1 DROP COLUMN new\_column;

RENAME TABLE table1 TO table2;

SELECT col1, col2 FROM customers;  
SELECT DISTINCT city FROM customers;  
SELECT \* FROM customers LIMIT 5;  
SELECT \* FROM customers LIMIT 2,7; // 7 elements selected starting from element 2  
SELECT \* FROM customers ORDER BY id;  
SELECT \*FROM customers ORDER BY name;  
SELECT \*FROM customers ORDER BY state, name;  
SELECT \* FROM customers ORDER BY id DESC;  
SELECT \* FROM customers ORDER BY id DESC LIMIT 1;  
SELECT \* FROM customers WHERE id=5/id!=5/id>5;  
SELECT \* FROM customers WHERE id BETWEEN 5 AND 72; //can use AND statement for the same  
SELECT \* FROM customers WHERE state =”NY”;  
SELECT \* FROM customers WHERE state =”NY” AND/OR id>46;  
SELECT \* FROM customers WHERE (state =”NY” AND city=”Hollywood”) OR (id=1 And zip=342);  
SELECT \* FROM customers WHERE state IN (“NY”,”NC”,”CA”) ORDER BY state;// instead of using multiple OR statements.  
SELECT \* FROM customers WHERE state NOT IN (“NY”,”NC”) ORDER BY state;   
SELECT \* FROM items WHERE name LIKE “%computer%” / “computer%” / “c%m%r”;   
  
//character ‘%’ is used in place of string (including spaces), for single character we use ’\_’

SELECT CONCAT(city,” IN STATE “,state) AS new\_address FROM customers;  
SELECT \*, (cost\*0.8) AS sale\_80\_percentage\_price FROM items;  
SELECT \*, UPPER(name) AS upper\_name FROM customers;  
  
SELECT SQRT(cost), AVG(cost), SUM(cost) FROM customers;  
  
SELECT COUNT(name/id/…) FROM customers WHERE id>87;

SELECT   
COUNT(\*) AS max\_count,  
MAX(cost) AS max\_cost,  
AVG(cost) AS avg\_cost  
FROM items WHERE seller\_id<12;  
  
OBSERVE the functionality of COUNT  
1. SELECT id, name, COUNT(\*) FROM customers;  
2. SELECT id, name, COUNT(\*) FROM customers WHERE id=7;  
3. SELECT id, name, COUNT(\*) FROM customers WHERE id>7;

SELECT seller\_id, name, COUNT(\*) FROM items;  
SELECT seller\_id, name, COUNT(\*) FROM items GROUP BY seller\_id;  
SELECT seller\_id, name, COUNT(\*) FROM items GROUP BY seller\_id HAVING COUNT(\*)>3;

SELECT seller\_id, name, COUNT(\*) AS count\_max  
FROM items //each seller\_id corresponds to different name, thus only the first name comes.  
GROUP BY seller\_id  
HAVING COUNT(\*) >3  
ORDER BY count\_max ASC;  
  
SELECT seller\_id, MIN(cost), MAX(cost), AVG(cost), COUNT(\*)  
FROM items // min, max and avg cost at which one particular sells his product   
GROUP BY seller\_id  
HAVING COUNT(\*)>1

SELECT bids, name, COUNT(\*) AS count\_bid  
FROM items //each bids corresponds to different name, thus only the first name comes.  
GROUP BY bids  
HAVING COUNT(\*) >1;

SELECT \*   
FROM items   
WHERE cost > (SELECT AVG(cost) FROM items); //subquery, nesting

SELECT \*   
FROM items   
WHERE cost > AVG(cost); //will give wrong output  
  
SELECT id, cost, (cost –(SELECT AVG(cost) FROM items)) AS deviation,   
(SELECT AVG(cost) FROM items)AS average  
FROM items  
GROUP BY id;

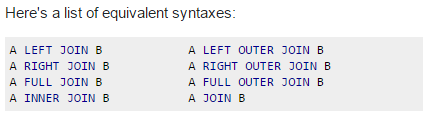
SELECT id, cost,(cost-AVG(cost)) AS average FROM items; //won’t give correct output  
  
SELECT name, MIN(cost) FROM items WHERE name Like “%frog%”;   
  
SELECT \* FROM items WHERE cost=( SELECT MAX(cost) FROM items WHERE name LIKE ”%frog%”);

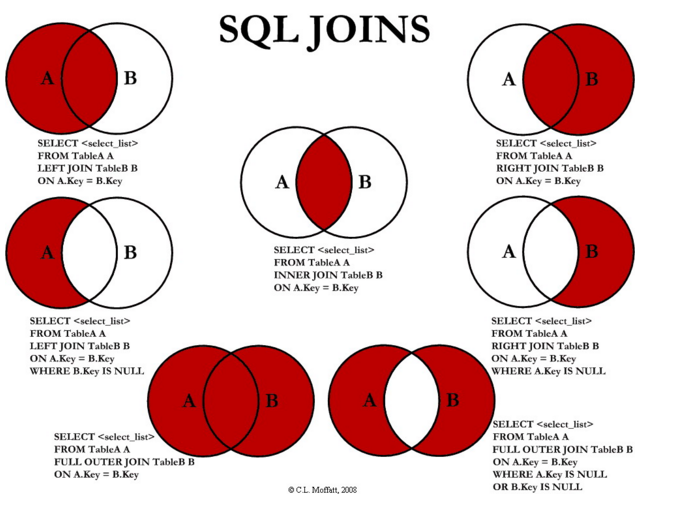
JOINING TABLE  
//ASSUMING THAT CUSTOMERS ARE BASICALLY THE VENDORS/SELLERS

SELECT customers.id, customers.name, items.name, items.cost  
FROM customers, items  
WHERE customers.id=items.seller\_id   
ORDER BY customers.id  
LIMIT 10;  
// items.seller\_id OR seller\_id  
// few customers.id are repeated also as one customer is selling more than one items and few of them are missing too!  
  
SELECT c.id, c.name, i.name, i.cost  
FROM customers AS c, items AS I //defining name for tables as we did for columns   
WHERE c.id=i.seller\_id   
ORDER BY c.id  
LIMIT 10;  
  
  
SELECT name, cost, bids FROM items WHERE bids>190 OR cost>1000;

OR

SELECT name, cost, bids FROM items WHERE bids>190  
UNION  
SELECT name, cost, bids FROM items WHERE cost>1000;  
  
SELECT name, cost, bids FROM items WHERE bids>190  
UNION ALL   
SELECT name, cost, bids FROM items WHERE cost>1000;  
//if an entry satisfies both conditions then it will appear twice  
  
FULL TEXT SEARCHING WAY BETTER THAN USING LIKE=”%xyz%”  
  
Firstly alter the attributes in which you want to activate full text searching OR initially define it like that.  
ALTER TABLE items ADD FULLTEXT(name);   
  
  
SELECT \* FROM items WHERE name LIKE “%baby%”;  
OR  
SELECT \* FROM ITEMS WHERE MATCH(name) AGAINST(“baby”);  
  
  
  
SELECT \* FROM items WHERE name LIKE “%baby%” AND name NOT LIKE “%coat%” ;  
OR  
SELECT \* FROM ITEMS WHERE MATCH(name) AGAINST(“+baby -coat” IN BOOLEAN MODE);

**JOIN**Note: outer keyword is OPTIONAL  
  




EXAMPLE of left join  
Suppose a company wants to see the sales by the already existing customers, new customers. Consider two tables  
Table1(A) – contains details of all the customers till july, it contains “cid” from 1-100  
Table2(B)- contains sales data of august which has “cid”=[1,7,24,102,107] where 102, 107 are new customers.  
  
  
  
 A B

A=[1….100]  
B=[1,7,24,102,107]  
C=A intersection B=[1,7,24]  
  
Query 1 //will give A  
SELECT \*  
FROM A  
LEFT JOIN B  
ON A.cid=B.cid  
  
Query 2 //will give A-C  
SELECT \*  
FROM A  
LEFT JOIN B  
ON A.cid=B.cid  
WHERE B.cid IS NULL  
  
Query 3 // will give B-C  
SELECT \*  
FROM A  
RIGHT JOIN B  
ON A.cid=B.cid  
WHERE A.cid IS NULL

COMPANY DATABASE

CREATE TABLE department (

dname VARCHAR(15) NOT NULL,  
dnumber INT NOT NULL,  
mgrssn CHAR(9) NOT NULL,  
mgrstartdate DATE,  
PRIMARY KEY (dnumber)  
  
);

CREATE TABLE employee(  
  
fname VARCHAR(15) NOT NULL,   
minit CHAR,  
lname VARCHAR(15) NOT NULL,  
ssn CHAR(9),  
bdate DATE,  
address VARCHAR(30),  
sex CHAR,  
salary DECIMAL(10,2),  
superssn CHAR(9),  
dno INT,  
PRIMARY KEY(ssn),  
FOREIGN KEY(superssn) REFERENCES employee(ssn),  
FOREIGN KEY (dno) REFERENCES department(dnumber)  
);

ALTER TABLE department ADD FOREIGN KEY (mgrssn) REFERENCES employee (ssn)

CREATE TABLE dept\_locations(

dnumber INT NOT NULL,  
dlocation VARCHAR(15) NOT NULL,  
PRIMARY KEY(dnumber, dlocation),  
FOREIGN KEY (dnumber) REFERENCES department (dnumber)

);

CREATE TABLE project (

pname VARCHAR(15) NOT NULL,  
pnumber INT NOT NULL,  
plocation VARCHAR(15),  
dnum INT NOT NULL,  
PRIMARY KEY(pnumber),  
FOREIGN KEY(dnum) REFERENCES dept\_locations(dnumber)

);

CREATE TABLE works\_on(

essn CHAR(9) NOT NULL,  
pno INT NOT NULL,  
hours DECIMAL(3,1) NOT NULL,  
FOREIGN KEY(essn) REFERENCES employee(ssn),  
FOREIGN KEY(pno) REFERENCES project(pnumber)  
);  
  
  
CREATE TABLE dependent(

essn CHAR(9) NOT NULL,  
dependent\_name VARCHAR(15) NOT NULL,  
sex CHAR,  
bdate DATE,  
relationship VARCHAR(8),  
PRIMARY KEY(essn, dependent\_name),  
FOREIGN KEY(essn) REFERENCES employee(ssn)

);  
  
//Insert into employee by keeping dno as NULL and superssn same as ssn

**At the time of joining you must mention 1 condition about which two tables are joined, for three tables, 2 conditions…**SELECT e.fname, d.dname  
FROM employee AS e, department AS d;  
**This will give you all combinations**.  
Thus correct query is  
SELECT e.fname, d.dname  
FROM employee AS e, department AS d  
WHERE e.dno=d.dnumber;  
  
OR  
SELECT e.ssn, w.pno, p.pname  
FROM employee AS e, works\_on AS w, project AS p  
WHERE e.ssn=w.essn AND w.pno=p.pnumber;