CREATE TABLE student (

student\_id INT PRIMARY KEY,

St\_name VARCHAR(20),

major VARCHAR(20)

-- PRIMARY KEY(student\_id)

);

DESCRIBE student;

DROP TABLE student;

ALTER TABLE student ADD gpa DECIMAL(3,2);

ALTER TABLE student DROP COLUMN gpa;

INSERT INTO student VALUES(1, ‘Jack’, ‘Biology’);

SELECT \* FROM student;

INSERT INTO student VALUES(2, ‘Kate’, ‘Sociology’);

INSERT INTO student(student\_id, St\_name) VALUES(3, ‘Claire’);

INSERT INTO student VALUES(4, ‘Jack’, ‘Biology’);

INSERT INTO student VALUES(5, ‘Mike’, ‘Computer Science’);

DROP TABLE student;

CREATE TABLE student (

student\_id INT,

St\_name VARCHAR(20) NOT NULL,

major VARCHAR(20) UNIQUE -- major VARCHAR(20) DEFAULT ‘undecided’,

PRIMARY KEY(student\_id)

);

INSERT INTO student VALUES(1, ‘Jack’, ‘Biology’);

INSERT INTO student VALUES(2, ‘Kate’, ‘Sociology’);

INSERT INTO student VALUES(3, NULL, “Chemistry”); -- Error because of NULL

INSERT INTO student VALUES(4, ‘Jack’, ‘Biology’); -- Error because of Duplicate entry

INSERT INTO student VALUES(5, ‘Mike’, ‘Computer Science’);

INSERT INTO student(student\_id, St\_name) VALUES(6, ‘Jack’); -- Check when using DEFAULT ‘undecided’

DROP TABLE student;

CREATE TABLE student (

student\_id INT AUTO\_INCREMENT,

St\_name VARCHAR(20),

major VARCHAR(20),

PRIMARY KEY(student\_id)

);

INSERT INTO student(St\_name, major) VALUES(‘Jack’, ’Biology’);

INSERT INTO student(St\_name, major) VALUES(‘Kate’, ’Sociology’);

INSERT INTO student(St\_name, major) VALUES(‘Craile’, ’Biology’);

DROP TABLE student;

CREATE TABLE student (

student\_id INT PRIMARY KEY,

St\_name VARCHAR(20),

major VARCHAR(20)

);

INSERT INTO student VALUES(1, ‘Jack’, ‘Biology’);

INSERT INTO student VALUES(2, ‘Kate’, ‘Sociology’);

INSERT INTO student VALUES (3, ‘Claire’, ‘Chemistry’);

INSERT INTO student VALUES(4, ‘Jack’, ‘Biology’);

INSERT INTO student VALUES(5, ‘Mike’, ‘Computer Science’);

SELECT \* FROM student;

UPDATE student

SET major = ‘Bio’

WHERE major = ‘Biology’;

-- Other comparison operations :

= : equals

<> : not equals

> : greater than

< : less than

>= : greater that or equal

<= : less than or equal –

UPDATE student

SET major = ‘Comp Sci’

WHERE student\_id = 4;

UPDATE student

SET major = ‘Biochemistry’

WHERE major = ‘Bio’ OR major = ‘Chemistry’;

UPDATE student

SET St\_name =’Tom’, major = ‘Undecided’

WHERE student\_id=1;

UPDATE student

SET major = ‘Undecided’;

DELETE FROM student;

-- It’ll delete all the rows inside the table

DELETE FROM student

WHERE student\_id = 5;

-- Delete the rows where student\_id = 5

DELETE FROM student

WHERE name = ‘Tom’ AND major = ‘undecided’;

-- Getting information from the table. More specifically we’ll be looking at SELECT keyword.

/\* Queries are essentially just block of SQL that’s designed to ask the database management system for a particular piece of information. \*/

SELECT \* FROM student;

SELECT St\_name FROM student;

SELECT St\_name, major FROM student;

SELECT student.St\_name, student.major FROM student; -- It’s same as the previous line of code

SELECT student.St\_name, student.major FROM student

ORDER BY St\_name; -- By default it’s going to be ascending alphabetic order

SELECT student.St\_name, student.major FROM student

ORDER BY St\_name DESC; -- For getting in descending order

SELECT student.St\_name, student.major FROM student

ORDER BY student\_id DESC; -- We can order by anything whether we’re returning that column or not like here.

SELECT \* FROM student

ORDER BY student\_id ASC;

SELECT \* FROM student

ORDER BY major, student\_id; -- It’s getting ordered by major first and then student\_id.

SELECT \* FROM student

ORDER BY major, student\_id DESC;

SELECT \* FROM student

LIMIT 2; -- Limiting amount of results that we might be getting

SELECT \* FROM student

ORDER BY student\_id DESC

LIMIT 2;

SELECT \* FROM student

WHERE major = ‘Biology’;

SELECT St\_name, major FROM student

WHERE major = ‘Chemistry’ OR major = ‘Biology’;

SELECT St\_name, major FROM student

WHERE major = ‘Chemistry’ OR name = ‘Kate’;

-- <, >, <=, >=, =, <>, AND, OR

SELECT St\_name, major FROM student

WHERE major <> ‘Chemistry’;

SELECT \* FROM student

WHERE student\_id <= 3 AND name <> ‘Jack’;

SELECT \* FROM student

WHERE St\_name IN (‘Claire’, ‘Kate’, ‘Mike’);

SELECT \* FROM student

WHERE major IN (‘Biology’, ‘Chemistry’) AND student\_id > 2 ;

----------------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE TABLE employee (

Emp\_id INT PRIMARY KEY,

First\_name VARCHAR(40),

Last\_name VARCHAR(40),

Birth\_day DATE,

Sex VARCHAR(1),

Salary INT,

Super\_id INT,

Branch\_id INT

/\* Although Super\_id and Branch\_id are foreign keys but we can’t technically define them as foreign key now because neither employee table nor branch table exist yet so we’re defining them as normal columns here. \*/

);

CREATE TABLE branch (

Branch\_id INT PRIMARY KEY,

Branch\_name VARCHAR(40),

Mgr\_id INT,

Mgr\_start\_date DATE,

FOREIGN KEY(mgr\_id) REFERNCES employee(Emp\_id) ON DELETE SET NULL

/\* We can also use ON DELETE CASCADE in place of SET NULL. Whenever we’re creating a Foreign Key, we’re gonna put either of the two. This makes a lot easier for us to manage the foreign key. \*/

);

-- Now setting Super\_id and Branch\_id of employee table as foreign keys.

ALTER TABLE employee

ADD FOREIGN KEY(branch\_id)

REFERENCES branch(branch\_id)

ON DELETE SET NULL;

ALTER TABLE employee

ADD FOREIGN KEY(Super\_id)

REFERENCES employee(Emp\_id)

ON DELETE SET NULL;

CREATE TABLE client(

Client\_id INT PRIMARY KEY,

Client\_name VARCHAR(40),

Branch\_id INT,

FOREIGN KEY(Branch\_id) REFERNCES branch(branch\_id) ON DELETE SET NULL

);

-- This Table works\_with is now having composite primary key and that too both are Foreign Keys.

CREATE TABLE works\_with(

Emp\_id INT,

Client\_id INT,

Total\_sales INT,

PRIMARY KEY (Emp\_id, Client\_id),

FOREIGN KEY (Emp\_id) REFERENCES employee(Emp\_id) ON DELETE CASCADE,

FOREIGN KEY (Client\_id) REFERENCES client(Client\_id) ON DELETE CASCADE

);

CREATE TABLE branch\_supplier (

Branch\_id INT,

supplier\_name VARCHAR(40),

supply\_type VARCHAR(40),

PRIMARY KEY (Branch\_id, supplier\_name),

FOREIGN KEY (Branch\_id) REFERENCES branch(Branch\_id) ON DELETE CASCADE

);

-- Corporate

INSERT INTO employee VALUES(100, ‘David’, ‘Wallace’, ‘1967-11-17’, ‘M’, 250000, NULL, NULL);

-- It was NULL there because that Branch\_id has not been created yet.

INSERT INTO branch VALUES (1, ‘Corporate’, 100, ‘2006-02-09’);

UPDATE employee

SET Branch\_id =1

WHERE Emp\_id =101;

INSERT INTO employee VALUES(101, ‘Jan’, ‘Levinson’, ‘1961-05-11’, ‘F’, 110000, 100, 1);

-- Scranton

INSERT INTO employee VALUES(102, ‘Michael’, ‘Scott’, 1964-03-15’, ‘M’, 75000, 100, NULL);

INSERT INTO branch VALUES (2, ‘Scranton’, 102, ‘1992-04-06’);

UPDATE employee

SET Branch\_id =2

WHERE Emp\_id =102;

INSERT INTO employee VALUES(103, ‘Angela’, ‘Martin’, 1971-06-25’, ‘M’, 75000, 102, 2);

INSERT INTO employee VALUES(104, ‘Kelly’, ‘Kapoor’, 1980-02-05’, ‘M’, 75000, 102, 2);

INSERT INTO employee VALUES(105, ‘Stanley’, ‘Hudson’, 1958-02-19’, ‘M’, 75000, 102, 2);

-- Stamford

INSERT INTO employee VALUES(106, ‘Josh’, ‘Partner’, ‘1969-09-05’, ‘M’, 78000, 100, NULL);

INSERT INTO employee VALUES (3, ‘Stamford’, 106, ‘1998-02-13’);

UPDATE employee

SET Branch\_id =3

WHERE Emp\_id =106;

INSERT INTO employee VALUES(107, ‘Andy’, Bernard’, 1973-07-22’, ‘M’, 65000, 106, 3);

INSERT INTO employee VALUES(108, ‘Jim’, ‘Halpert’, 1978-10-01’, ‘M’, 71000, 106, 3);

-- Branch Supplier

INSERT INTO branch\_supplier VALUES(2, ‘Hammer Mill’, ‘Paper’);

INSERT INTO branch\_supplier VALUES(2, ‘Uni-ball’, ‘Paper’);

INSERT INTO branch\_supplier VALUES(3, ‘Patriot Paper’, ‘Paper’);

INSERT INTO branch\_supplier VALUES(2, ‘J.T. Forms & Labels’, ‘Custom Forms’);

INSERT INTO branch\_supplier VALUES(3, ‘Uni-ball’, ‘Writing Utensils’);

INSERT INTO branch\_supplier VALUES(3, ‘Hammer Mill’, ‘Paper’);

INSERT INTO branch\_supplier VALUES(3, ‘Stamford Lables’, ‘Custom Forms’);

-- CLIENT

INSERT INTO client VALUES(400, ‘Dunmore Highschool’, 2);

INSERT INTO client VALUES(401, ‘Lackawana Country’, 2);

INSERT INTO client VALUES(402, ‘FedEx’, 3);

INSERT INTO client VALUES(403, ‘John Daly Law, LLC’, 3);

INSERT INTO client VALUES(404, ‘Scranton Whiepages’, 2);

INSERT INTO client VALUES(405, ‘Times Newspaper’, 3);

INSERT INTO client VALUES(406, ‘FedEx’, 2);

-- WORKS\_WITH

INSERT INTO works\_with VALUES(105, 400, 55000);

INSERT INTO works\_with VALUES(102, 401, 267000);

INSERT INTO works\_with VALUES(108, 402, 22500);

INSERT INTO works\_with VALUES(107, 403, 12000);

INSERT INTO works\_with VALUES(108, 403, 33000);

INSERT INTO works\_with VALUES(105, 404, 5000);

INSERT INTO works\_with VALUES(107, 405, 26000);

INSERT INTO works\_with VALUES(102, 406, 15000);

INSERT INTO works\_with VALUES(105, 406, 130000);

-- Find all empolyees

SELECT \* FROM employee;

-- Find all clients

SELECT \* FROM client;

-- Find all employee ordered by salary

SELECT \* FROM employee

ORDER BY salary DESC;

-- Find all employees ordered by sex then name

SELECT \* FROM employee

ORDER BY sex, first\_name, last\_name;

-- Find the first 5 employees in the table

SELECT \* FROM employee

LIMIT 5;

-- Find the first and last name of all employees

SELECT first\_name, last\_name FROM employee;

-- Find the forename and surname of all employees

SELECT first\_name AS forename, last\_name AS surname

FROM employee;

-- Find out all the different genders

SELECT DISTINCT sex

FROM employee;

SELECT DISTINCT Branch\_id

FROM employee;

------------------------------------------------------- Functions in SQL-------------------------------------------------------------------------------

-- Find the number of employees

SELECT COUNT (Emp\_id)

FROM employee;

SELECT COUNT (Super\_id)

FROM employee;

-- Here it won’t count the NULL value

-- Find the number of female employees born after 1970

SELECT COUNT (Emp\_id)

FROM employee;

WHERE sex = ‘F’ AND Birth\_date >= ‘1971-01-01’;

-- Find the average of all employee’s salaries

SELECT AVG(salary)

FROM employee;

SELECT AVG(salary)

FROM employee

WHERE sex = ‘M’;

-- Find the sum of all employee’s salaries

SELECT SUM(salary)

FROM employee;

-- Find out how many males and females are there

SELECT COUNT (Sex), Sex

FROM employee;

SELECT COUNT (Sex), Sex

FROM employee

GROUP BY sex;

-- Find out total sales of each salesman

SELECT SUM (total\_sales), Emp\_id

FROM works\_with

GROUP BY Emp\_id;

-- Find out total sales by each client

SELECT SUM (total\_sales), Client\_id

FROM works\_with

GROUP BY Clent\_id;

-- WildCards

-- % = any # characters, \_= one character

-- Find any client who are an LLC

SELECT \*

FROM client

WHERE Client\_name LIKE ‘%LLC’;

-- Find any branch suppliers who are in the label business

SELECT \*

FROM branch\_supplier

WHERE supplier\_name LIKE ‘% Label%’;

-- Find any employee born in February

SELECT \*

FROM employee

WHERE birth\_date LIKE ‘\_\_\_\_-02%’;

-- Find any clients who are schools

SELECT \*

FROM client

WHERE Client\_name LIKE ‘%school%’;