DROP TABLE student; //brisemo tabelu koja se zove student

CREATE TABLE student( //pravimo tabelu koja se zove student I koja u sebi sadrzi sledece kolone

student\_id INT PRIMARY KEY, //ovu kolonu smo oznacili kao PK, tip je integer

student\_name VARCHAR (20), //tipa varchar I moze da sadrzi 20 karaktera

student\_major VARCHAR (20)

);

CREATE TABLE student(

student\_id INT PRIMARY KEY,

student\_name VARCHAR (20) NOT NULL, // vrednost ne moze da bude podesena na NULL

student\_major VARCHAR (20) UNIQUE //sve vrednosti u koloni moraju da budu razlicite

);

CREATE TABLE student(

student\_id INT AUTO\_INCREMENT, (MySQL AUTO\_INCREMENT) // da sam uvecava sledeci red za 1

student\_name VARCHAR (20),

student\_major VARCHAR (20) DEFAULT 'undecided',

PRIMARY KEY (student\_id) //PK mozemo I ovako da oznacimo

);

INSERT INTO student (student\_name, student\_major) VALUES('Jack', 'Biology'); //ubacujemo podatke u tabelu

INSERT INTO student (student\_name, student\_major) VALUES('Kate', 'Sociology');//redom kako smo definisali

ALTER TABLE student ADD gpa DECIMAL(3, 2);//promeni tabelu student tako sto ces dodati kolonu gpa koja ima

//decimalnu vrednost gde ispred tacke imamo 3 cifre a iza 2

SELECT \* FROM student; //izaberi sve kolone iz tabele student

ALTER TABLE student DROP COLUMN gpa; //promeni tabelu tako sto ces izbrisati kolonu gpa

INSERT INTO student VALUES(1, 'Jack', 'Biology'); // ubaci u tabelu student red sa vrednostima

//student\_id=1; student\_name=’Jack’; student\_major = ‘Biology’

INSERT INTO student VALUES(2, 'Kate', 'Sociology');

INSERT INTO student(student\_id, student\_name) VALUES(3, 'Claire'); //ubaci u tabelu student red gde ces u samo ove dve

//kolone ubaciti ove vrednosti

INSERT INTO student(student\_id, student\_name) VALUES(4, 'Claire');

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

UPDATE student //promeni tabelu student

SET student\_major = 'Biology' //tako sto ces podesiti kolonu student\_major da ima vrednost ‘Biology’

WHERE student\_id = 4; //svuda gde je vrednost kolone student\_id = 4

UPDATE student

SET student\_major = 'Bio'

WHERE student\_major = 'Biology';

UPDATE student //promeni tabelu student

SET student\_major = 'Biochemistry' //tako sto ces podesiti kolonu student\_major da ima vrednost ‘Bichemistry’

WHERE student\_major = 'Bio' OR student\_major = 'Chemistry'; // svuda gde je vrednost kolone student\_major ‘Bio’ ILI ‘Chemistry’

UPDATE student

SET student\_name = 'Tom', student\_major = 'undecided' // podesi student\_name na ‘Tom’ I student\_major na ‘undecided’

WHERE student\_id = 1; //svuda gde je student\_id = 1

DELETE FROM student // izbrisi iz tabele student

WHERE student\_id = 5; //sve redove gde je student\_id = 5

QUERIES

SELECT student\_name //izaberi kolonu student\_name

FROM student; //iz tabele student

SELECT student\_name, student\_major //izaberi kolonu student\_name I kolonu student\_major

FROM student; //iz tabele student

SELECT student.student\_name, student.student\_major //moze I spred kolone da se stavi ime tabelei tacka – tako je preciznije

FROM student;

SELECT student.student\_name, student.student\_major

FROM student

ORDER BY student\_name; //I poredjaj po rastucem redu kolone student\_name (od A do Z)

SELECT student.student\_name, student.student\_major

FROM student

ORDER BY student\_name DESC; //poredjaj po opadajucem redu (Od Z do A)

SELECT student.student\_name, student.student\_major

FROM student

ORDER BY student\_id DESC;//mozemo da poredjamo I po kriterijumu koji nismo stavili u SELECT

SELECT \*

FROM student

ORDER BY student\_major, student\_id; //moze da poredja I po vise kriterijuma, prvo po major pa onda id

SELECT \*

FROM student

LIMIT 2; //ispisuje samo prva 2 reda

SELECT \*

FROM student

ORDER BY student\_id DESC //prvo ce poredjati od najvecg do najmanjeg id

LIMIT 2; // I onda ispisati prva dva … u ovom slucaju redove sa student\_ide 5 I 4

SELECT \* //izaberi sve kolone

FROM student //iz tabele student

WHERE student\_major = 'Comp Sci'; //gde je kolona student\_major “Comp Sci”

SELECT student\_name, student\_major //izaberi kolone student\_name I student\_major

FROM student //iz tabele student

WHERE student\_major = 'Biochemistry'; //gde je kolona student\_major “Biochemistry”

SELECT student\_name, student\_major

FROM student

WHERE student\_major = 'Biochemistry' OR student\_major = 'Comp Sci'; // gde je kolona student\_major “Biochemistry” ili “Comp Sci”

SELECT student\_name, student\_major

FROM student

WHERE student\_major = 'Biochemistry' OR student\_name = 'Kate'; // gde je kolona student\_major “Biochemistry” ili ime “Kate”

-- //komentar su dve crte

<, >, <=, >=, =, <>, AND, OR //operatori poredjenja…<> znaci not equal to

SELECT student\_name, student\_major

FROM student

WHERE student\_major <> 'Biochemistry'; //gde student\_major nije “Biochemistry”

SELECT \*

FROM student

WHERE student\_id <3;

SELECT \*

FROM student

WHERE student\_id <3 AND student\_name <> 'Kate';

SELECT \*

FROM student

WHERE student\_name IN ('Tom', 'Kate', 'Jack'); // gde su u koloni student\_name vrednosti Tom ili Kate ili Jack

SELECT \*

FROM student

WHERE student\_name IN ('Tom', 'Kate', 'Jack') AND student\_id >2; // gde su u koloni student\_name vrednosti Tom ili Kate ili Jack I gde je student\_id veci od 2

COMPANY DATABASE

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, //kad napravimo tabelu gde je super\_id PK ovde cemo moci da naznacimo da je ovo FOREIGN KEY

branch\_id INT //kad napravimo tabelu gde je branch\_id PK ovde cemo moci da naznacimo da je ovo FOREIGN KEY);

CREATE TABLE branch(

branch\_id INT PRIMARY KEY,

branch\_name VARCHAR(40),

mgr\_id INT,

mgr\_start\_date DATE,

FOREIGN KEY(mgr\_id) REFERENCES employee(emp\_id) ON DELETE SET NULL

);

// FOREIGN KEY(mgr\_id) REFERENCES employee(emp\_id) ON DELETE SET NULL

//postavi kolonu mgr\_id iz ove tabele koju kreiras (TABLE BRANCH) da bude FOREIGN KEY za kolonu emp\_id tabele employee

//ON DELETE SET NULL se uvek pise za FK…kasnije ce biti objasnjeno

//sada mozemo da dodamo FOREIGN KEY u tabelu employee

ALTER TABLE employee //napravi izmenu u tabeli employee

ADD FOREIGN KEY(branch\_id) //tako sto ces dodati (ADD) strani kljuc(FOREIGN KEY) da bude kolona branch\_id

REFERENCES branch(branch\_id) //tako da u tabeli branch kolona branch\_id je veza (ima iste vrednosti)

ON DELETE SET NULL;

ALTER TABLE employee //napravi izmenu u tabeli employee

ADD FOREIGN KEY(super\_id) //tako sto ces postaviti kolonu super\_id da bude strani kljuc

REFERENCES employee(emp\_id) // koji je isto sto I emp\_id u tabeli employee …u ovom slucaju znaci super\_id je employee id supervizora

ON DELETE SET NULL;

CREATE TABLE client (

client\_id INT PRIMARY KEY,

client\_name VARCHAR(40),

branch\_id INT,

FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id) ON DELETE SET NULL

);

CREATE TABLE works\_with (

emp\_id INT,

client\_id INT,

total\_sales INT,

PRIMARY KEY(emp\_id, client\_id), // kompozitni PK

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), //kompozitni PK

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);

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

UPDATE employee

SET branch\_id = 1

WHERE emp\_id = 100;

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', 'F', 63000, 102, 2);

INSERT INTO employee VALUES(104, 'Kelly', 'Kapoor', '1980-02-05', 'F', 55000, 102, 2);

INSERT INTO employee VALUES(105, 'Stanley', 'Hudson', '1958-02-19', 'M', 69000, 102, 2);

-- Stamford

INSERT INTO employee VALUES(106, 'Josh', 'Porter', '1969-09-05', 'M', 78000, 100, NULL);

INSERT INTO branch 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', 'Writing Utensils');

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 Whitepages', 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, 5000);

INSERT INTO works\_with VALUES(108, 403, 12000);

INSERT INTO works\_with VALUES(105, 404, 33000);

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 employees

SELECT \*

FROM employee;

-- Find all clients

SELECT \*

FROM clients;

-- Find all employees ordered by salary

SELECT \*

from employee

ORDER BY salary ASC/DESC;

-- Find all employees ordered by sex then name

SELECT \*

from employee

ORDER BY sex, name;

-- Find the first 5 employees in the table

SELECT \*

from employee

LIMIT 5;

-- Find the first and last names of all employees

SELECT first\_name, employee.last\_name

FROM employee;

-- Find the forename and surnames names of all employees

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

FROM employee;

-- Find out all the different genders

SELECT DISCINCT sex

FROM employee;

-- Find all male employees

SELECT \*

FROM employee

WHERE sex = 'M';

-- Find all employees at branch 2

SELECT \*

FROM employee

WHERE branch\_id = 2;

-- Find all employee's id's and names who were born after 1969

SELECT emp\_id, first\_name, last\_name

FROM employee

WHERE birth\_day >= 1970-01-01;

-- Find all female employees at branch 2

SELECT \*

FROM employee

WHERE branch\_id = 2 AND sex = 'F';

-- Find all employees who are female & born after 1969 or who make over 80000

SELECT \*

FROM employee

WHERE (birth\_day >= '1970-01-01' AND sex = 'F') OR salary > 80000;

-- Find all employees born between 1970 and 1975

SELECT \*

FROM employee

WHERE birth\_day BETWEEN '1970-01-01' AND '1975-01-01';

-- Find all employees named Jim, Michael, Johnny or David

SELECT \*

FROM employee

WHERE first\_name IN ('Jim', 'Michael', 'Johnny', 'David');

//-----------------------------------------------------------------------

-- Find the number of employees

SELECT COUNT(super\_id)

FROM employee;

-- Find the average of all employee's salaries

SELECT AVG(salary)

FROM employee;

-- Find the sum of all employee's salaries

SELECT SUM(salary)

FROM employee;

-- Find out how many males and females there are

SELECT COUNT(sex), sex

FROM employee

GROUP BY sex

-- Find the total sales of each salesman

SELECT SUM(total\_sales), emp\_id

FROM works\_with

GROUP BY client\_id;

-- Find the total amount of money spent by each client

SELECT SUM(total\_sales), client\_id

FROM works\_with

GROUP BY client\_id;

//------------------------------------------------------

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

-- Find any client's 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 on the 10th day of the month

SELECT \*

FROM employee

WHERE birth\_day LIKE '\_\_\_\_\_10%';

-- Find any clients who are schools

SELECT \*

FROM client

WHERE client\_name LIKE '%Highschool%';

//--------------------------------------------------------------------

-- Find a list of employee and branch names

SELECT employee.first\_name AS Employee\_Branch\_Names

FROM employee

UNION

SELECT branch.branch\_name

FROM branch;

-- Find a list of all clients & branch suppliers' names

SELECT client.client\_name AS Non-Employee\_Entities, client.branch\_id AS Branch\_ID

FROM client

UNION

SELECT branch\_supplier.supplier\_name, branch\_supplier.branch\_id

FROM branch\_supplier;

//-------------------------------------------

-- Add the extra branch

INSERT INTO branch VALUES(4, "Buffalo", NULL, NULL);

SELECT employee.emp\_id, employee.first\_name, branch.branch\_name

FROM employee

JOIN branch -- LEFT JOIN, RIGHT JOIN

ON employee.emp\_id = branch.mgr\_id;

//-----------------------------------------------------

-- Find names of all employees who have sold over 50,000

SELECT employee.first\_name, employee.last\_name

FROM employee

WHERE employee.emp\_id IN (SELECT works\_with.emp\_id

FROM works\_with

WHERE works\_with.total\_sales > 50000);

-- Find all clients who are handles by the branch that Michael Scott manages

-- Assume you know Michael's ID

SELECT client.client\_id, client.client\_name

FROM client

WHERE client.branch\_id = (SELECT branch.branch\_id

FROM branch

WHERE branch.mgr\_id = 102);

-- Find all clients who are handles by the branch that Michael Scott manages

-- Assume you DONT'T know Michael's ID

SELECT client.client\_id, client.client\_name

FROM client

WHERE client.branch\_id = (SELECT branch.branch\_id

FROM branch

WHERE branch.mgr\_id = (SELECT employee.emp\_id

FROM employee

WHERE employee.first\_name = 'Michael' AND employee.last\_name ='Scott'

LIMIT 1));

-- Find the names of employees who work with clients handled by the scranton branch

SELECT employee.first\_name, employee.last\_name

FROM employee

WHERE employee.emp\_id IN (

SELECT works\_with.emp\_id

FROM works\_with

)

AND employee.branch\_id = 2;

-- Find the names of all clients who have spent more than 100,000 dollars

SELECT client.client\_name

FROM client

WHERE client.client\_id IN (

SELECT client\_id

FROM (

SELECT SUM(works\_with.total\_sales) AS totals, client\_id

FROM works\_with

GROUP BY client\_id) AS total\_client\_sales

WHERE totals > 100000

);

//----------------------------------------------------------------

-- CREATE

-- TRIGGER `event\_name` BEFORE/AFTER INSERT/UPDATE/DELETE

-- ON `database`.`table`

-- FOR EACH ROW BEGIN

-- -- trigger body

-- -- this code is applied to every

-- -- inserted/updated/deleted row

-- END;

CREATE TABLE trigger\_test (

message VARCHAR(100)

);

DELIMITER $$

CREATE

TRIGGER my\_trigger BEFORE INSERT

ON employee

FOR EACH ROW BEGIN

INSERT INTO trigger\_test VALUES('added new employee');

END$$

DELIMITER ;

INSERT INTO employee

VALUES(109, 'Oscar', 'Martinez', '1968-02-19', 'M', 69000, 106, 3);

DELIMITER $$

CREATE

TRIGGER my\_trigger BEFORE INSERT

ON employee

FOR EACH ROW BEGIN

INSERT INTO trigger\_test VALUES(NEW.first\_name);

END$$

DELIMITER ;

INSERT INTO employee

VALUES(110, 'Kevin', 'Malone', '1978-02-19', 'M', 69000, 106, 3);

DELIMITER $$

CREATE

TRIGGER my\_trigger BEFORE INSERT

ON employee

FOR EACH ROW BEGIN

IF NEW.sex = 'M' THEN

INSERT INTO trigger\_test VALUES('added male employee');

ELSEIF NEW.sex = 'F' THEN

INSERT INTO trigger\_test VALUES('added female');

ELSE

INSERT INTO trigger\_test VALUES('added other employee');

END IF;

END$$

DELIMITER ;

INSERT INTO employee

VALUES(111, 'Pam', 'Beesly', '1988-02-19', 'F', 69000, 106, 3);

DROP TRIGGER my\_trigger;