CREATE TABLE IF NOT EXISTS courses (

course\_id SERIAL PRIMARY KEY,

course\_name VARCHAR(100),

credit\_hours INTEGER

);

CREATE TABLE IF NOT EXISTS students(

student\_id serial primary key,

first\_name varchar,

last\_name varchar,

birthdate date,

envanment\_year int);

CREATE TABLE IF NOT EXISTS enrollments (

enrollment\_id SERIAL PRIMARY KEY,

grade INTEGER,

student\_id INT REFERENCES students(student\_id),

course\_id INT REFERENCES courses(course\_id)

);

INSERT INTO courses (course\_name, credit\_hours) VALUES

('Mathematics', 3),

('Physics', 4),

('Chemistry', 3),

('Biology', 3),

('Computer Science', 4),

('English Literature', 3),

('History', 3),

('Psychology', 3),

('Sociology', 3),

('Philosophy', 3);

INSERT INTO students (first\_name, last\_name, birthdate, envanment\_year) VALUES

('John', 'Doe', '2001-03-15', 2019),

('Jane', 'Smith', '2000-07-22', 2018),

('Michael', 'Johnson', '2001-02-10', 2019),

('Emily', 'Williams', '2002-04-01', 2020),

('Daniel', 'Brown', '2000-12-12', 2018),

('Olivia', 'Davis', '2001-05-14', 2019),

('James', 'Miller', '2001-08-23', 2019),

('Sophia', 'Wilson', '2002-01-11', 2020),

('Jackson', 'Moore', '2000-09-05', 2018),

('Ava', 'Taylor', '2001-06-19', 2019),

('Benjamin', 'Anderson', '2000-11-30', 2018),

('Isabella', 'Thomas', '2001-03-17', 2019),

('Lucas', 'Jackson', '2002-10-25', 2020),

('Mason', 'White', '2001-04-28', 2019),

('Liam', 'Harris', '2002-02-14', 2020),

('Mia', 'Martin', '2001-07-07', 2019),

('Ethan', 'Thompson', '2000-06-12', 2018),

('Charlotte', 'Garcia', '2002-09-21', 2020),

('Amelia', 'Martinez', '2001-11-03', 2019),

('Alexander', 'Rodriguez', '2002-05-16', 2020);

INSERT INTO enrollments (grade, student\_id, course\_id) VALUES

(85, 1, 1),

(90, 1, 2),

(80, 2, 3),

(75, 2, 4),

(88, 3, 5),

(92, 3, 6),

(70, 4, 7),

(95, 4, 8),

(78, 5, 9),

(82, 5, 10),

(85, 6, 1),

(76, 6, 2),

(90, 7, 3),

(84, 7, 4),

(92, 8, 5),

(88, 8, 6),

(73, 9, 7),

(79, 9, 8),

(81, 10, 9),

(83, 10, 10),

(87, 11, 1),

(93, 11, 2),

(80, 12, 3),

(86, 12, 4),

(89, 13, 5),

(94, 13, 6),

(77, 14, 7),

(78, 14, 8),

(90, 15, 9),

(85, 15, 10),

(84, 16, 1),

(82, 16, 2),

(91, 17, 3),

(79, 17, 4),

(92, 18, 5),

(88, 18, 6),

(80, 19, 7),

(86, 19, 8),

(93, 20, 9),

(81, 20, 10);

-- 3.1

-- 3.1.1

SELECT first\_name, last\_name, birthdate

FROM students;

-- 3.1.2

SELECT s.first\_name, s.last\_name, c.course\_name

FROM students s

JOIN enrollments e ON s.student\_id = e.student\_id

JOIN courses c ON e.course\_id = c.course\_id

WHERE c.course\_name = 'Mathematics';

-- 3.1.3

SELECT s.first\_name, s.last\_name, e.grade

FROM students s

JOIN enrollments e on s.student\_id = e.student\_id

where grade < 80;

-- 3.2

--3.2.1

SELECT s.first\_name, s.last\_name, c.course\_name

FROM enrollments e

JOIN courses c ON c.course\_id = e.course\_id

JOIN students s ON s.student\_id = e.student\_id;

-- 3.2.2

SELECT s.student\_id, s.first\_name, s.last\_name

FROM students s

LEFT JOIN enrollments e ON s.student\_id = e.student\_id

WHERE e.enrollment\_id IS NULL;

-- 3.3

-- 3.3.1

SELECT c.course\_name, COUNT(e.student\_id) AS student\_count

FROM courses c

LEFT JOIN enrollments e ON c.course\_id = e.course\_id

GROUP BY c.course\_name;

--3.3.2

SELECT c.course\_name, COUNT(e.student\_id) AS student\_count

FROM courses c

LEFT JOIN enrollments e ON c.course\_id = e.course\_id

GROUP BY c.course\_name

ORDER BY student\_count

LIMIT 1;

-- 3.4

--3.4.1

SELECT last\_name, first\_name

FROM students

ORDER BY last\_name ASC;

-- 3.4.2

SELECT s.first\_name, s.last\_name, s.birthdate, c.course\_name

FROM students s

JOIN enrollments e ON s.student\_id = e.student\_id

JOIN courses c ON e.course\_id = c.course\_id

WHERE c.course\_name = 'History'

AND s.birthdate > '2000-01-01';

-- 3.5

--3.5.1

SELECT s.first\_name, s.last\_name

FROM students s

WHERE (SELECT COUNT(e.course\_id)

FROM enrollments e

WHERE e.student\_id = s.student\_id) >

(SELECT AVG(course\_count)

FROM (SELECT COUNT(e.course\_id) AS course\_count

FROM enrollments e

GROUP BY e.student\_id) subquery);

--3.5.2

SELECT c.course\_name

FROM courses c

JOIN enrollments e ON c.course\_id = e.course\_id

JOIN students s ON e.student\_id = s.student\_id

WHERE s.student\_id IN (

SELECT e.student\_id

FROM enrollments e

JOIN students s ON e.student\_id = s.student\_id

GROUP BY e.student\_id

HAVING AVG(e.grade) = (

SELECT MIN(average\_grade)

FROM (

SELECT e.student\_id, AVG(e.grade) AS average\_grade

FROM enrollments e

GROUP BY e.student\_id

) AS subquery

)

);

-- 3.6

--3.6.1

UPDATE enrollments

SET grade = 70

WHERE grade > 70

--3.6.2

DELETE FROM enrollments

WHERE course\_id is null

--3.6.3

INSERT INTO enrollments (student\_id, course\_id, grade)

VALUES

((SELECT MAX(student\_id) from enrollments), 7 , 70),

((SELECT MAX(student\_id) from enrollments), 3 , 90)