**Exercise 1. Find annual grades of each students for each subject**

CREATE TABLE annual\_grade

(

student\_id integer,

subject\_id integer,

student\_name varchar(50),

subject varchar(20),

avg\_grade numeric,

CONSTRAINT id\_ PRIMARY KEY (student\_id ,subject\_id )

);

INSERT INTO annual\_grade (student\_id, subject\_id,student\_name,subject,avg\_grade)

SELECT students.id\_number, teachers.teacher\_id,

students.first\_name ||' ' ||students.sur\_name AS Student,

teachers.subject AS Subject, round(AVG(grade),2) AS avg\_grade

FROM subject\_grades JOIN students

ON subject\_grades.student\_id=students.id\_number

JOIN teachers

ON subject\_grades.subject=teachers.teacher\_id

GROUP BY students.id\_number, teachers.teacher\_id, Student, teachers.subject

ORDER BY teachers.subject;

SELECT\*FROM annual\_grade;

-----Let’s assume each student will receive extra point to annual grade if they did not get ‘1’ grade within the whole year.----

ALTER TABLE annual\_grade

ADD COLUMN get\_1 boolean;

---- Creating view that helps to assign students who got ‘1’ to table annual\_grade ----

CREATE VIEW get\_1 AS

SELECT student\_id,subject,grade

FROM subject\_grades

WHERE grade=1;

SELECT\*FROM get\_1;

UPDATE annual\_grade

SET annual\_grade.get\_1=TRUE

FROM get\_1

WHERE (annual\_grade.student\_id=get\_1.student\_id) AND (annual\_grade.subject\_id=get\_1.subject);

UPDATE annual\_grade

SET get\_1=FALSE

WHERE get\_1 IS NULL;

UPDATE annual\_grade

SET avg\_grade=avg\_grade+1

WHERE get\_1=TRUE;

ALTER TABLE annual\_grade

ADD COLUMN final\_grade int;

UPDATE annual\_grade

SET final\_grade=ROUND(avg\_grade,0)

SELECT\*FROM annual\_grade;

--- Reducing table size ---

SELECT pg\_size\_pretty( pg\_total\_relation\_size('annual\_grade'));

VACUUM FULL annual\_grade;

SELECT pg\_size\_pretty( pg\_total\_relation\_size('annual\_grade'));

**Exercise 2. The best student in each class gets a scholarship of 2000$ and a holiday ticket to Greece. Teacher, whose students received the highest annual grade, also gets a holiday ticket to Greece.**

CREATE TABLE scholarship

(

id\_ serial PRIMARY KEY,

name\_ varchar(25),

class\_ integer,

avg\_grade numeric

);

--- Insert students into table ---

CREATE VIEW avg\_annl\_grade AS

SELECT student\_name, ROUND(AVG (final\_grade),2) AS avg\_grade,

students.class\_nr AS class\_number

FROM annual\_grade JOIN students

ON annual\_grade.student\_id=students.id\_number

GROUP BY students.class\_nr,student\_name

ORDER BY avg\_grade DESC;

INSERT INTO scholarship (name\_,class\_,avg\_grade)

SELECT student\_name, class\_number, avg\_grade

FROM avg\_annl\_grade

WHERE (class\_number =3 AND avg\_grade=

(

SELECT MAX(avg\_grade)

FROM avg\_annl\_grade

HAVING class\_number=3

GROUP BY class\_number

))

OR

(class\_number =4 AND avg\_grade=

(

SELECT MAX(avg\_grade)

FROM avg\_annl\_grade

HAVING class\_number=4

GROUP BY class\_number

))

OR

(class\_number =5 AND avg\_grade=

(

SELECT MAX(avg\_grade)

FROM avg\_annl\_grade

HAVING class\_number=5

GROUP BY class\_number

))

GROUP BY student\_name, class\_number, avg\_grade;

SELECT\*FROM scholarship

--- Insert teacher into table ---

INSERT INTO scholarship (name\_)

SELECT name\_

FROM (

SELECT teachers.first\_name ||' '||teachers.sur\_name AS name\_,

ROUND(AVG(grade),2) AS avg\_grade

FROM subject\_grades JOIN teachers

ON subject\_grades.subject=teachers.teacher\_id

GROUP BY name\_

ORDER BY avg\_grade DESC

LIMIT 1

);

**Exercise 3. Create a table to track changes in table ‘subject\_grades’.**

CREATE TABLE grades\_history

(

student\_id integer NOT NULL,

subject\_id integer NOT NULL,

change\_time timestamp with time zone NOT NULL,

old\_grade integer NOT NULL,

new\_grade integer NOT NULL,

PRIMARY KEY (student\_id,subject\_id, change\_time)

);

--- We need to create function in PL/SQL ---

CREATE OR REPLACE FUNCTION track\_grade\_change()

RETURNS trigger AS

$$

BEGIN --- function of PL/pqSQL

IF NEW.grade != OLD.grade THEN

INSERT INTO grades\_history

(

student\_id,

subject\_id,

change\_time,

old\_grade,

new\_grade)

VALUES

(

OLD.student\_id,

OLD.subject,

now(),

OLD.grade,

NEW.grade

);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

--- creating trigger ---

CREATE TRIGGER grade\_update

AFTER UPDATE

ON subject\_grades

FOR EACH ROW

EXECUTE PROCEDURE track\_grade\_change();

--- trigger test ---

SELECT\*FROM subject\_grades

WHERE grade=1;

--- transaction will help avoid problems like: table 'subject\_grades' will be updated and 'grades\_history' will not. ---

START TRANSACTION;

UPDATE subject\_grades

SET grade=3

WHERE grade\_id=88;

SELECT\*FROM grades\_history;

--- trigger works properly---

COMMIT;