LAP 3

3. Insert new student and his score in exam in different subjects as transaction and save it

```
postgres=# create view student_tracks_view as select students.first_name,
        tracks.track_name from students join tracks on students.track_id = tracks.track_id;

CREATE VIEW
```

4. Create a view for Tracks names and the subjects which is belong/study to it.

```
postgres=# create view tracks_cources_view as select tracks.track_name, cources.course_name
from tracks join cources on tracks.track_id = cources.course_id;

CREATE VIEW
```

5. Create a view for student names with their subject's names which will study.

```
postgres=# CREATE VIEW student_course_view AS SELECT students.first_name,
cources.course_name FROM students JOIN student_courses ON students.student_id =
student_courses.student_id JOIN cources ON student_courses.course_id = cources.course_id;

CREATE VIEW
```

6. Create a view for all students name (Full Name) with their score in each subject and its date.

```
postgres=# create view student_exam_result_view as select students.first_name,
students.last_name, exam_result.score_student, exam_result.exam_date from students join
exam_result on students.student_id = exam_result.student_id;

CREATE VIEW
```

7. Create a temporary view for all subjects with their max_score.

```
postgres=# create view cources_view as select cources.max_score, cources.course_name from
cources ;

CREATE VIEW
```

10.(from Q.6) Display the date of exam as the following: day 'month name' year.

postgres=# select TO_CHAR(student_exam_result_view.exam_date, 'DD "Month" YYYY') from student_exam_result_view;

to_char

---

23 Month 2023

25 Month 2023

20 Month 2023

19 Month 2023

15 Month 2023

10 Month 2023

05 Month 2023

(7 rows)

11.Display name and age of each students

postgres=# select students.birth_date, student_exam_result_view.first_name from students join student_exam_result_view on students.first_name = student_exam_result_view.first_name;

birth_date | first_name

---+---

(0 rows)

12.Display the name of students with their Rounded score in each subject

postgres=# create view student_scors_view as select students.first_name, cources.course_name, ROUND(exam_result.score_student) from students join exam_result on students.student_id = exam_result.student_id join cources on exam_result.course_id = cources.course_id;

CREATE VIEW

13.Display the name of students with the year of Birthdate

postgres=# create view student_birth_year_view as select students.first_name, EXTRACT(YEAR FROM birth_date) from students;

CREATE VIEW

14.Add new exam result, in date column use NOW() function;

postgres=# insert into exam_result (result_id,student_id,course_id,score_student,exam_date) values (8,5,1,30,NOW());

INSERT 0 1

15.Create database called ITI, and create different schema and Tables inside this schema

postgres=# insert into exam_result (result_id,student_id,course_id,score_student,exam_date) values (8,5,1,30,NOW());

INSERT 0 1

postgres=# create schema education;

CREATE SCHEMA

postgres=# create table education.students (student_id SERIAL primary key, first_name varchar(50), last_name varchar(50), birthdate DATE);

CREATE TABLE

postgres=# CREATE TABLE education.subjects (subject_id SERIAL PRIMARY KEY, name VARCHAR(100), description TEXT);

CREATE TABLE

postgres=# CREATE TABLE education.exams (exam_id SERIAL PRIMARY KEY, student_id INT REFERENCES education.students(student_id), subject_id INT REFERENCES education.subjects(subject_id), score NUMERIC(5,2), exam_date TIMESTAMP DEFAULT NOW());

CREATE TABLE

postgres=# INSERT INTO education.students (first_name, last_name, birthdate) VALUES ('John', 'Doe', '2000-05-15');

INSERT 0 1

LAP 4

1. Create multiply function which take two number and return the multiply of them

```
postgres=# CREATE OR REPLACE FUNCTION multiply(a NUMERIC, b NUMERIC)RETURNS
NUMERIC AS $$BEGIN    RETURN a * b;

postgres$# END

postgres$# ;

postgres$# $$ LANGUAGE plpgsql;

CREATE FUNCTION
```

2. Create Hello world function which take username and return welcome message to user using his name

```
postgres=# CREATE OR REPLACE FUNCTION hello_world(username VARCHAR)RETURNS
VARCHAR AS $$BEGIN    RETURN 'Welcome, ' || username || '!';

postgres$# End;

postgres$# $$ LANGUAGE plpgsql;

CREATE FUNCTION
```

3. Create function which takes number and return if this number is odd or even.

```
postgres=# CREATE OR REPLACE FUNCTION check_odd_even(num INT)RETURNS
VARCHAR AS $$BEGIN    IF num % 2 = 0 THEN       RETURN 'Even';

postgres$#  ELSE       RETURN 'Odd';

postgres$# End if;

postgres$# end;

postgres$# $$ LANGUAGE plpgsql;

CREATE FUNCTION
```

4. Create AddNewStudent function which take Student firstName and lastname and birthdate and TrackName and add this new student info at database

```
postgres=# CREATE OR REPLACE FUNCTION AddNewStudent(firstName VARCHAR, lastName
VARCHAR, birthdate DATE, trackName VARCHAR)RETURNS VOID AS $$DECLARE    track_id INT;

postgres$# BEGIN
```

```
postgres$# SELECT t.track_id INTO track_id    FROM education.tracks t    WHERE t.track_name =
trackName;

postgres$# IF track_id IS NOT NULL THEN       INSERT INTO education.students (first_name,
last_name, birthdate, track_id)       VALUES (firstName, lastName, birthdate, track_id);

postgres$# ELSE       RAISE EXCEPTION 'Track not found: %', trackName;

postgres$# End if;

postgres$# end;

postgres$# $$ LANGUAGE plpgsql;

CREATE FUNCTION
```

5. Create function which takes StudentId and return the string/text that describe the use info(firstname, last name, age, TrackName).

```
postgres=# CREATE OR REPLACE FUNCTION GetStudentInfo(studentId INT)RETURNS TEXT AS
$$DECLARE    first_name VARCHAR;    last_name VARCHAR;    birthdate DATE;    track_name
VARCHAR;    age INT;BEGIN

postgres$# SELECT s.first_name, s.last_name, s.birthdate, t.track_name    INTO first_name,
last_name, birthdate, track_name    FROM education.students s    JOIN education.tracks t ON
s.track_id = t.track_id    WHERE s.student_id = studentId;

postgres$# age := DATE_PART('year', AGE(birthdate));

postgres$# RETURN 'Student: ' || first_name || ' ' || last_name || ', Age: ' || age || ', Track: ' ||
track_name;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION
```

6. Create function which takes TrackName and return the students names in this track.

```
postgres=# CREATE TABLE education.tracks ( track_id SERIAL , track_name varchar(50) );

CREATE TABLE

postgres=# ALTER TABLE education.students ADD COLUMN track_id SERIAL;

ALTER TABLE

postgres=# ALTER TABLE education.tracks ADD CONSTRAINT pk_track_id PRIMARY KEY (track_id);

ALTER TABLE
```

```
postgres=# CREATE OR REPLACE FUNCTION GetStudentsByTrack(trackName
VARCHAR)RETURNS TABLE(student_name TEXT) AS $$BEGIN    RETURN QUERY

postgres$# SELECT s.first_name || ' ' || s.last_name AS student_name    FROM public.students s

postgres$# JOIN public.tracks t ON s.track_id = t.track_id    WHERE t.track_name = trackName;

postgres$# END;$$ LANGUAGE plpgsql;

CREATE FUNCTION

postgres=# SELECT * FROM GetStudentsByTrack('Java');

student_name

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

(0 rows)
```

7. Create function which takes student id and subject id and return score the student in subject.

```
postgres=# CREATE OR REPLACE FUNCTION GetStudentScore(studentId INT, subjectId INT)
RETURNS NUMERIC AS $$ DECLARE score NUMERIC; BEGIN select education.exams.score into
score from education.exam.score ss where ss.student_id = studentId and ss.subject_id =
subjectId;

postgres$# RETURN score;

postgres$# End;

postgres$# $$ LANGUAGE plpgsql;

CREATE FUNCTION
```

8. Create function which takes subject id and return the number of students who failed in a subject (Score less than 50).

```
postgres=# CREATE OR REPLACE FUNCTION GetNumberOfStudentsFailed (subjectId INT)
RETURNS INT AS $$DECLARE num_failed INT; BEGIN

postgres$# SELECT COUNT(*)    INTO num_failed    FROM education.exam_scores    WHERE
subject_id = subjectId AND score < 50;

postgres$# RETURN num_failed;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION
```

9. Create function which take subject name and return the average grades for subject

```
postgres=# CREATE OR REPLACE FUNCTION GetNumberOfStudentsFailed (subjectId INT)
RETURNS INT AS $$DECLARE num_failed INT; BEGIN

postgres$# SELECT COUNT(*)  INTO num_failed  FROM education.exam_scores  WHERE
subject_id = subjectId AND score < 50;

postgres$# RETURN num_failed;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION

postgres=# CREATE OR REPLACE FUNCTION GetAverageGrade(subjectName VARCHAR)RETURNS
NUMERIC AS $$DECLARE  avg_grade NUMERIC;BEGIN

postgres$# SELECT AVG(es.score)  INTO avg_grade  FROM education.exam_scores es  JOIN
education.subjects s ON es.subject_id = s.subject_id  WHERE s.subject_name = subjectName;

postgres$# RETURN avg_grade;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION
```

LAST LAP

1. Create trigger to prevent insert new Course with name length greater than 20 chars;

```
postgres=# CREATE OR REPLACE FUNCTION check_course_name_length()RETURNS
TRIGGER AS $$BEGIN

postgres$# IF LENGTH(NEW.course_name) > 20 THEN      RAISE EXCEPTION 'Course name
cannot be longer than 20 characters.';  END IF;

postgres$# RETURN NEW;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION
```

2. Create trigger to prevent user to insert or update Exam with Score greater than 100 or less than zero

```
postgres=# CREATE OR REPLACE FUNCTION check_exam_score()RETURNS TRIGGER AS
$$BEGIN

postgres$# IF NEW.score < 0 OR NEW.score > 100 THEN      RAISE EXCEPTION 'Score must
be between 0 and 100.';  END IF;
```

postgres$# RETURN NEW;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION

3. (bonus) Create trigger to prevent any user to update/insert/delete to all tables (Students, Exams, Tracks,..) after 7:00 PM

postgres=# CREATE OR REPLACE FUNCTION check_time_restriction()RETURNS TRIGGER AS $$BEGIN

postgres$# IF EXTRACT(HOUR FROM CURRENT_TIME) >= 19 THEN RAISE EXCEPTION 'Updates, inserts, and deletes are not allowed after 7:00 PM.'; END IF;

postgres$# RETURN NEW;END;$$ LANGUAGE plpgsql;

CREATE FUNCTION

4. Backup your Database to external file

C:\Program Files\PostgreSQL\16\bin>#pg_dump ITI > outfile;

Access is denied.

5. Backup your Student table to external file

C:\Program Files\PostgreSQL\16\bin>pg_dump -U postgres -d ITI -t education.students -F p -f "C:\Users\YourUsername\Documents\students_backup.sql"

Password: