Lap1

1. Add gender column for the student table[Enum]. It holds two value (male or female).

```
postgres=# CREATE TYPE gender_enum AS ENUM ('male', 'female');

        postgres=# ALTER TABLE students ADD COLUMN gender gender_enum;
```

2. Add birth date column for the student table.

```
ALTER TABLE students ADD COLUMN birth_date DATE;
```

3. Delete the name column and replace it with two columns first name and last name.

```
        postgres=# ALTER TABLE students DROP COLUMN student_name;

        ALTER TABLE

        postgres=# ALTER TABLE students ADD COLUMN first_name VARCHAR(255), ADD
        COLUMN last_name VARCHAR(255);
```

4. Delete the address and email column and replace it with contact info (Address, email) as object/Composite Data type.

```
        postgres=# CREATE TYPE contact_info AS (    address TEXT,    email TEXT);

        CREATE TYPE

        postgres=# ALTER TABLE students DROP COLUMN address,DROP COLUMN email;

        ALTER TABLE

        postgres=# ALTER TABLE students ADD COLUMN contact_info contact_info;

        ALTER TABLE
```

5. Change any Serial Datatype at your tables to smallInt

```
            postgres=# ALTER TABLE students ALTER COLUMN student_id TYPE SMALLINT;

ALTER TABLE
```

6. Add/Alter foreign key constrains in Your Tables.

```
postgres=# ALTER TABLE students DROP CONSTRAINT IF EXISTS track_id;

NOTICE:  constraint "track_id" of relation "students" does not exist, skipping

ALTER TABLE

postgres=# ALTER TABLE students ADD COLUMN new_track_id INTEGER;

ALTER TABLE

postgres=# ALTER TABLE students ADD CONSTRAINT new_track_id FOREIGN KEY
(new_track_id) REFERENCES tracks(track_id);

ALTER TABLE
```

7. Insert new data in all Tables.

```
postgres=# INSERT INTO students (student_id, track_id, gender, birth_date, first_name,
last_name, contact_info, new_track_id) VALUES (5,1, 'female', '2023-07-07', 'amira',
'hassan', ROW('123 Main St', 'ah@gmail.com'), 1);

INSERT 0 1
```

9. Display male students only.

```
postgres=# INSERT INTO students (student_id, track_id, gender, birth_date, first_name,
last_name, contact_info, new_track_id) VALUES (6,1, 'male', '2023-07-07', 'hassan', 'ali',
ROW('123 Main St', 'hm@gmail.com'), 1);

INSERT 0 1

postgres=# SELECT * FROM students WHERE gender = 'male';

student_id | track_id | gender | birth_date | first_name | last_name |       contact_info       |
new_track_id
```

```
------------+----------+--------+------------+------------+-----------+-----------------------------+---------
-----
          6 |        1 | male   | 2023-07-07 | hassan     | ali       | ("123 Main St",hm@gmail.com) |            1
(1 row)
```

10.Display the number of female students.

```
postgres=# SELECT COUNT(*) AS number_of_female_students FROM students WHERE
gender = 'female';
number_of_female_students
---------------------------
                         1
(1 row)
```

11.Display the students who are born before 1992-10-01.

```
postgres=# INSERT INTO students (student_id, track_id, gender, birth_date, first_name,
last_name, contact_info, new_track_id) VALUES (7,1, 'male', '1930-07-07', 'hassan', 'ali',
ROW('123 Main St', 'hm@gmail.com'), 1);
INSERT 0 1
postgres=# INSERT INTO students (student_id, track_id, gender, birth_date, first_name,
last_name, contact_info, new_track_id) VALUES (8,1, 'male', '1940-07-07', 'ali', 'ali',
ROW('123 Main St', 'hm@gmail.com'), 1);
INSERT 0 1
postgres=# SELECT * FROM students WHERE birth_date < '1992-10-01';
student_id | track_id | gender | birth_date | first_name | last_name |        contact_info         |
new_track_id
------------+----------+--------+------------+------------+-----------+-----------------------------+---------
-----
          7 |        1 | male   | 1930-07-07 | hassan     | ali       | ("123 Main St",hm@gmail.com) |            1
          8 |        1 | male   | 1940-07-07 | ali        | ali       | ("123 Main St",hm@gmail.com) |            1
(2 rows)
```

12.Display male students who are born before 1991-10-01.

```
postgres=# SELECT *FROM students WHERE gender = 'male'  AND birth_date < '1991-10-01';
student_id | track_id | gender | birth_date | first_name | last_name |       contact_info        | new_track_id
------------+----------+--------+------------+------------+-----------+-----------------------------+--------------
          7 |        1 | male   | 1930-07-07 | hassan     | ali       | ("123 Main St",hm@gmail.com) |            1
          8 |        1 | male   | 1940-07-07 | ali        | ali       | ("123 Main St",hm@gmail.com) |            1
(2 rows)
```

13.Display subjects and their max score sorted by max score.

```
postgres=# SELECT course_name, max_score FROM cources ORDER BY max_score DESC;
course_name | max_score
-------------+-----------
HTML        |        90
CPP         |        80
C           |        50
(3 rows)
```

14.Display the subject with highest max score

```
postgres=# SELECT course_name, max_score FROM cources ORDER BY max_score DESC LIMIT 1;
course_name | max_score
-------------+-----------
HTML        |        90
```

```
(1 row)
```

15.Display students' names that begin with A.

```
postgres=# SELECT first_name, last_name FROM students WHERE first_name LIKE 'a%';

first_name | last_name

------------+-----------

amira      | hassan

ali        | ali

(2 rows)
```

16.Display the number of students' their name is "Mohammed"

```
postgres=# SELECT COUNT(*) AS number_of_students FROM students WHERE first_name
= 'Mohammed';

number_of_students

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

                  0

(1 row)
```

17.Display the number of males and females.

```
postgres=# SELECT gender, COUNT(*) AS number_of_students FROM students GROUP BY
gender;

gender | number_of_students

--------+--------------------

male   |                  3

       |                  4

female |                  1

(3 rows)
```

18.Display the repeated first names and their counts if higher than

```
postgres=# SELECT first_name, COUNT(*) AS name_count FROM students GROUP BY
first_name HAVING COUNT(*) > 2;

first_name | name_count

------------+------------

           |     4

(1 row)
```

19.Display the all Students and track name that belong to it

```
postgres=# SELECT s.student_id, s.first_name, s.last_name, t.track_name FROM students
s JOIN tracks t ON s.track_id = t.track_id;

student_id | first_name | last_name | track_name

------------+------------+-----------+------------

          8 | ali        | ali       | Telecom

          7 | hassan     | ali       | Telecom

          6 | hassan     | ali       | Telecom

          5 | amira      | hassan    | Telecom

          3 |            |           | Telecom

          1 |            |           | Telecom

          4 |            |           | OpenSource

          2 |            |           | OpenSource

(8 rows)
```

____

Lap 3

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

```
postgres=!# BEGIN; WITH new_student AS ( INSERT INTO students (first_name, last_name)
VALUES ('Ahmed', 'Mora')  RETURNING student_id) INSERT INTO cources (student_id,
course_id, max_score) SELECT student_id, 1, 85 FROM new_student UNION ALL SELECT
student_id, 2, 90 FROM new_student UNION ALL SELECT student_id, 3, 75 FROM
new_student;

ERROR:  current transaction is aborted, commands ignored until end of transaction block

ERROR:  current transaction is aborted, commands ignored until end of transaction block

postgres=!# commit;

ROLLBACK
```

2. Insert new students and his score in exam in different subjects as transaction and undo it.

```
postgres=# BEGIN;

BEGIN

postgres=*# INSERT INTO cources (student_id, course_id, max_score) SELECT student_id,
1, 85 FROM students WHERE first_name = 'hassan' AND last_name = 'ali' UNION ALL
SELECT student_id, 2, 90 FROM students WHERE first_name = 'ali' AND last_name = 'ali';

ERROR:  column "student_id" of relation "cources" does not exist

LINE 1: INSERT INTO cources (student_id, course_id, max_score) SELEC...

postgres=!# ROLLBACK;

ROLLBACK
```

8. Create user and give him all privileges.

```
postgres=# CREATE USER amira WITH PASSWORD 'amira123';
```

```
CREATE ROLE

postgres=# GRANT ALL PRIVILEGES ON DATABASE gradess TO amira;

GRANT
```

9. Create another new user and make the authentication method is “trust” and give him all privileges if he login from his “local” server.

```
postgres=# CREATE USER nada WITH PASSWORD 'nada123';

CREATE ROLE

postgres=# GRANT ALL PRIVILEGES ON DATABASE gradess TO nada;

GRANT
```