Assignment 1

PREPARATION OF THE DB MIGRATION

Ismayil Eyyublu

Production & Operations Management
INFT 3611

To begin with, for the migration and rollback processes, the initial step is to create tables. In this task, we have two tables: "STUDENTS" and "INTERESTS."

In this migration, we aim to modify the column name and adjust the existing length for this particular task.

> In this code, the first line defines the structure of the table. It specifies that "student_id" is a column of type integer, and "interest" is the second column with a string type and a maximum limit of 20 characters.

```
1  /* Creating Table "INTERESTS" */
2  create table INTERESTS(STUDENT_ID int, INTEREST varchar
     (20));
3  insert into INTERESTS(STUDENT_ID, INTEREST) values ('1',
     'Tennis'), ('1', 'Literature'), ('2', 'Math'), ('2',
     'Tennis'), ('3', 'Math'), ('3', 'Music'), ('2',
     'Football'), ('1', 'Chemistry'), ('3', 'Chess');
4
5  SELECT*FROM interests;
```

> On the second line, we use the "insert into" command to instruct the program to add values for the specified columns. We're saying that the first piece of information (value) we provide should go into the "student_id" column, and the second piece of information should go into the "interest" column.

```
 student_id |  interest
------------+------------
          1 | Tennis
          1 | Literature
          2 | Math
          2 | Tennis
          3 | Math
          3 | Music
          2 | Football
          1 | Chemistry
          3 | Chess
(9 rows)
```

In the process of altering a table, which involves making modifications, additions, or deletions, we are choosing to modify the "interests" table. Specifically, we are adding a new column called "interests" with the same data type we initially used when creating the table, which is varchar(20).

```
/* Creating new "INTERESTS" column */
alter table interests
add interests varchar(20);
```

After modifying the table by adding a new column named "interests," we use the "update" command to copy data from the existing "interest" column to the new "interests" column. This way, the new column gets the same data as the old one.

```
/* Making sure the information in the "interests" column
  is the same as in the "interest" column. */
update interests
set interests = interest;
```

Next mission is : “Change the name of the INTERESTS.INTEREST to INTERESTS and its type to array of strings. Migrate the data (table content) correspondingly (20%). Your INTERESTS table shall look like this after the migration: “

To create an array column, we first make a new table with a column called "array_agg." Then, we gather data from the old table and organize it into arrays based on the students' IDs.

```
/* Creating new "interests_new" table with array_agg -
  The PostgreSQL ARRAY_AGG() function is an aggregate
  function that accepts a set of values and returns an
  array in which each value in the set is assigned to an
  element of the array. */
CREATE TABLE interests_new AS select student_id, array_agg
  (interests) from interests group by(student_id) order
  by (student_id);
```

We're making a new table called "interests_new" by taking data from the old table. We keep the student IDs the same, but the "interests" column becomes an array, organized and ordered by student IDs. Once the new table is ready, we rename the old one so that the updated table can use the original name "INTERESTS."

And then switching the name of the new table as "INTERESTS" table:

```
/* Switching the name of the "interests" table */
ALTER TABLE interests RENAME TO interests_old;

/* Switching the name of the newly created
  "interests_new" table tointerests */
ALTER TABLE interests_new RENAME TO interests;
```

Then we drop the old table;

```
/* Dropping the old table */
DROP TABLE interests_old;
```

Then we are going to create STUDENTS Table;

```
1   /* CREATING new table STUDENTS */
2   create table STUDENTS(st_id int, st_name varchar(20),
      st_last varchar(20));
3   insert into STUDENTS(st_id, st_name, st_last)
4   values ('1', 'Konul', 'Gurbanova'), ('2', 'Shahnur',
      'Isgandarli'),('3', 'Natavan', 'Mammadova');
5
```

Following this command, we add a new column to the table

```
alter table students
add STUDENT_ID int;
```

Afterwards, making "student_id" to be equal to "st_id"

```
UPDATE students
set STUDENT_ID=ST_ID;
```

Next, we adjust the character limits for the "st_name" and "st_last" columns from 20 to 30

```
alter table students
alter column st_name type varchar(30);
alter table students
alter column st_last type varchar(30);
```

Following the creation and update of the column, the next step is to remove the old "st_id" column from the table

```
alter table students
drop column st_id;
```

WHY THE ROLLBACK IS USED?

In the context of migration, ROLLBACK in SQL is like a safety net. Imagine you're making changes to your database – adding tables, modifying data, etc. If something goes wrong during the migration, like an unexpected error or undesired changes, using ROLLBACK lets you undo everything and return the database to its previous state.

Remember, we adjusted the length to 30; now, we need to revert them to 20

```
alter table students
alter column st_name type varchar(20);
alter table students
alter column st_last type varchar(20);
```