CPS 610
Assignment 4

Group 12

Mohamed Shrief 500981017
Chris Fontein 500189282
Ekrem Yilmaz 501108034

Task 1: Creating Professors Tables

Task1-Part1: Professor1 Table with Address Object

In this task we defined a user-defined type for addresses and then created an object type for professors (Professor1) that incorporates this address type. This design encapsulates the professor’s address information within the object, ensuring that related attributes are kept together and data integrity is maintained.

Below is the exact code from our SQL file that defines an object type for addresses, extends the professor object to include an address attribute, and creates the Professor1 table:

```sql
-- Creating the address object type
CREATE TYPE address_typ AS OBJECT (
   StreetNo NUMBER(10),
   StreetName VARCHAR2(100),
   AptNo NUMBER(5),
   City VARCHAR2(100),
   State VARCHAR2(100),
   ZipCode NUMBER(9),
   Country VARCHAR2(100)
);
/

-- Creating the professor1 object type with an address attribute
CREATE TYPE professor1_typ AS OBJECT (
   professor_id NUMBER,
   professor_name VARCHAR2(100),
   department VARCHAR2(100),
   num_courses NUMBER,
   address address_typ
);
/

-- Creating the Professor1 table using the professor1_typ object
CREATE TABLE Professor1 OF professor1_typ;
/
```

Tak1-Part2: Professor2 Table with Circular Object Type
This code defines the professor2 object type to include a REF attribute named mentor. The REF serves as a pointer to another professor2 object, enabling us to establish a circular relationship (for example, a mentor-mentee association) without duplicating data. This approach ensures efficient relationship management and data integrity.

Below is the exact code from our SQL file that creates an object type for professors with a circular reference (using REF) and then creates the Professor2 table:

```
-- Creating the professor2 object type with a REF for circular reference (mentor)
CREATE TYPE professor2_typ AS OBJECT (
   professor_id NUMBER,
   professor_name VARCHAR2(100),
   department VARCHAR2(100),
   num_courses NUMBER,
   mentor REF professor2_typ
);
/

-- Creating the Professor2 table using the professor2_typ object
CREATE TABLE Professor2 OF professor2_typ;
/
```

Task 2: Explanation of REF

What is REF?
Based on our observations and what we implemented, a REF in Oracle is essentially a pointer or reference to an object stored in an object table. Instead of duplicating the entire object, a REF stores the memory address or location of that object. This provides several advantages:

Data Integrity & Efficiency:
Changes made to the original object are automatically reflected wherever the REF is used, avoiding redundancy.
Relationship Management:
By using REF, we can create clear relationships between objects. In our Professor2 table, the mentor attribute uses a REF to point to another professor2 object. This allows for circular references (e.g., mutual mentor relationships) without storing duplicate data.
In summary, using REF lets us maintain efficient, non-redundant links between objects, which is particularly useful for modeling complex relationships.

Task 3: PL/SQL Procedure for Increasing Course Count

In our implementation, the professor1 object type includes the num_courses attribute to represent the number of courses taught by a professor. The increase_courses procedure takes a professor's ID and an increment value as parameters. It then updates the corresponding professor's course count by adding the given increment. This procedure demonstrates the integration of procedural logic with our object-oriented database design, ensuring that updates to the professor's teaching load are handled dynamically.

Below is the exact PL/SQL code from our SQL file that adds an attribute for the number of courses a professor is teaching and implements a procedure to increase this count:

```
CREATE OR REPLACE PROCEDURE increase_courses (
   p_professor_id IN NUMBER,
   p_increment IN NUMBER
) IS
BEGIN
   UPDATE Professor1
   SET num_courses = num_courses + p_increment
   WHERE professor_id = p_professor_id;
   COMMIT;
END;
/
```