Islamic University of Technology
Lab Task 04
CSE 4308 - DBMS Lab

Submitted To :

MD Bakhtiar Hasan
Ast. Professor, CSE Department
Islamic University of Technology

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Sian Ashsad
ID : 200042151
Prog. : SWE
Dept. : CSE

Task 1 :

Working code :

```
--> 1 <--

create or replace view Advisor_Selection as
select ID, name, dept_name
from instructor;
```

Report :

Analysis : In this task, we need to create a view named Advisor_Selection that shows the ID, name and department name of instructors.

Explanation of solution : We use commands create or replace to either create or replace the required view. Then we select ID, name and dept_name from the provided instructor table.

Task 2 :

Working code :

```
--> 2 <--

create or replace view Student_Count AS
select name, count(s_id) as count_student
from Advisor_Selection natural join advisor
GROUP BY name;

select * from Student_Count;
```

Report :

Analysis : In this task, we need to create another view named Student_Count using Advisor_Selection and advisor to show the name of the instructor and the number students assigned under them.

Explanation of solution : Here we initiate the Student_Count view with create or replace commands and select name from Advisor_Selection view and select an aggregate function count over s_id from advisor table. Advisor_Selection and advisor table will have a natural join. Afterwards we will use the group by function on the name attribute. Finally we will show all information of Studnet_Count view.

Findings : A view can be used like a table in queries provided that the table connected with the view is present in that database.

Problems : It took some time to realize that views can be used like tables in queries. Other than that, there were no further issues.

Task 3 :

Working code :

```
--> 3 <--

--a
create role Students;

GRANT SELECT ON s200042151.advisor to Students;
GRANT SELECT ON s200042151.course to Students;

--b
create role course_teacher;

GRANT SELECT ON s200042151.student to course_teacher;
GRANT SELECT ON s200042151.course to course_teacher;

--c
create role dept_head;

GRANT course_teacher to dept_head;
GRANT INSERT ON s200042151.instructor to dept_head;

--d
create role Administrator;

GRANT SELECT ON s200042151.department to Administrator;
GRANT SELECT ON s200042151.instructor to Administrator;
GRANT UPDATE (budget) ON s200042151.department to Administrator;
```

Report :

Analysis : In this task we have to create certain roles and grant them appropriate privileges.

Explanation of solution :

a) Students role will be able to perform only the SELECT command in advisor and course tables.
b) course_teacher will be able to perform only the SELECT command in student and course tables.
c) dept_head will be able to perform only the SELECT command in student and course tables. This is because we have granted the course_teacher role to dept_head. Furthermore, dept_head will be able to perform INSERT command in the instructor table.
d) Administrator will be able to perform SELECT command in department and instructor tables. Additionally, it can UPDATE the budget attribute for the department table.

Findings : Not adding s200042151.(table name) leads to error. This is because the query cannot find which database the table should perform operations on.

Task 4 :

Working code :

```
--> 4 <--
CREATE USER Adon IDENTIFIED BY s1234;
GRANT Students TO Adon;

DROP USER Dihan;
CREATE USER Dihan IDENTIFIED BY d1234;
GRANT course_teacher TO Dihan;

DROP USER ARMK;
CREATE USER ARMK IDENTIFIED BY a1234;
GRANT dept_head TO ARMK;

DROP USER Mirza;
CREATE USER Mirza IDENTIFIED BY m1234;
GRANT Administrator TO Mirza;

GRANT CREATE SESSION TO Adon;
GRANT CREATE SESSION TO Dihan;
GRANT CREATE SESSION TO ARMK;
GRANT CREATE SESSION TO Mirza;

conn Adon/s1234
SELECT * FROM s200042151.advisor;
SELECT * FROM s200042151.course;
--will not show--
SELECT * FROM s200042151.instructor;
--will not show--
SELECT * FROM s200042151.teaches;
--will not work--
INSERT INTO s200042151.advisor values ('10102', 'ARMK', 'Comp. Sci.', '125000');


conn Dihan/d1234
SELECT * FROM s200042151.student;
```

```
SELECT * FROM s200042151.course;
--will not show--
SELECT * FROM s200042151.instructor;
--will not show--
SELECT * FROM s200042151.teaches;
--will not show--
SELECT * FROM s200042151.instructor;
--will not show--
SELECT * FROM s200042151.teaches;
--will not work--
INSERT INTO s200042151.student values ('10102', 'Sian', 'Comp. Sci.', '45000');

conn ARMK/a1234
SELECT * FROM s200042151.student;
SELECT * FROM s200042151.course;
SELECT * FROM s200042151.instructor;
--will not show--
SELECT * FROM s200042151.teaches;
--will not work--
INSERT INTO s200042151.student values ('10102', 'Sian', 'Comp. Sci.', '45000');

INSERT INTO s200042151.instructor values ('10102', 'Sian', 'Comp. Sci.',
'45000');
--add new instructor--
SELECT * FROM s200042151.instructor;

conn Mirza/m1234
SELECT * FROM s200042151.department;
SELECT * FROM s200042151.instructor;
--doesn't show--
SELECT * FROM s200042151.teaches;
UPDATE s200042151.department
SET budget = 12000
WHERE dept_name = 'History';

SELECT *
FROM s200042151.department;
```

Report :

Analysis : In this task, we need to create users under previously created roles and write relevant SQL queries to demonstrate that the imposed access control works.

Explanation of solution : First, we will create 4 roles with the 4 roles assigned to them individually.
Adon : has student role.
Dihan : has course_teacher role.
ARMK : has dept_head role.
Mirza : has Administrator role.
All of these roles will have CREATE SESSION access so that we can connect to them and perform queries.
In the given code, we can see Adon can SELECT any entry from both advisor and course tables but cannot do the same for any other table. Doing so will show an error message. Adon cannot perform any other command on advisor and course tables other than SELECT.

Similarly, Dihan can only SELECT from student and course tables. We can not perform any query on any other table.

ARMK user can do the operations as Dihan. Additionally, ARMK can INSERT INTO values in the instructor table only. We can not perform any other queries other than these.

Mirza user can only SELECT from student and instructor tables. Furthermore, Mirza can update the budget attribute in the instructor table. We can not perform any other queries other than these through Mirza.

Findings : The errors shown by performing queries which are not granted don't show the reason for the error. Which provides additional safety of access.

Problems : Connecting to the users was a challenge as I had to grant create session privileges to each user. This is something I had to figure out. Disconnecting from a user resulted in disconnection from the main database. Which proved to be a hassle. After performing the right queries these problems were resolved.