DATABASE MANAGEMENT SYSTEM

Wolaita Sodo University

School of Informatics

Department of Computer Science

Section 2 GROUP 3

ADVANCED DATABASE

Name	ID
HANA NIGUS	UGR/94290/16
BIFTU NADROS	UGR/91538/16
FISEHA ALEMU	UGR/91770/16
MESAY BIRHANU	UGR/92009/16
MEGFIRA ABDELA	UGR/91970/16

WULETAW IYASU

2017 E.C

Contents

Introduction

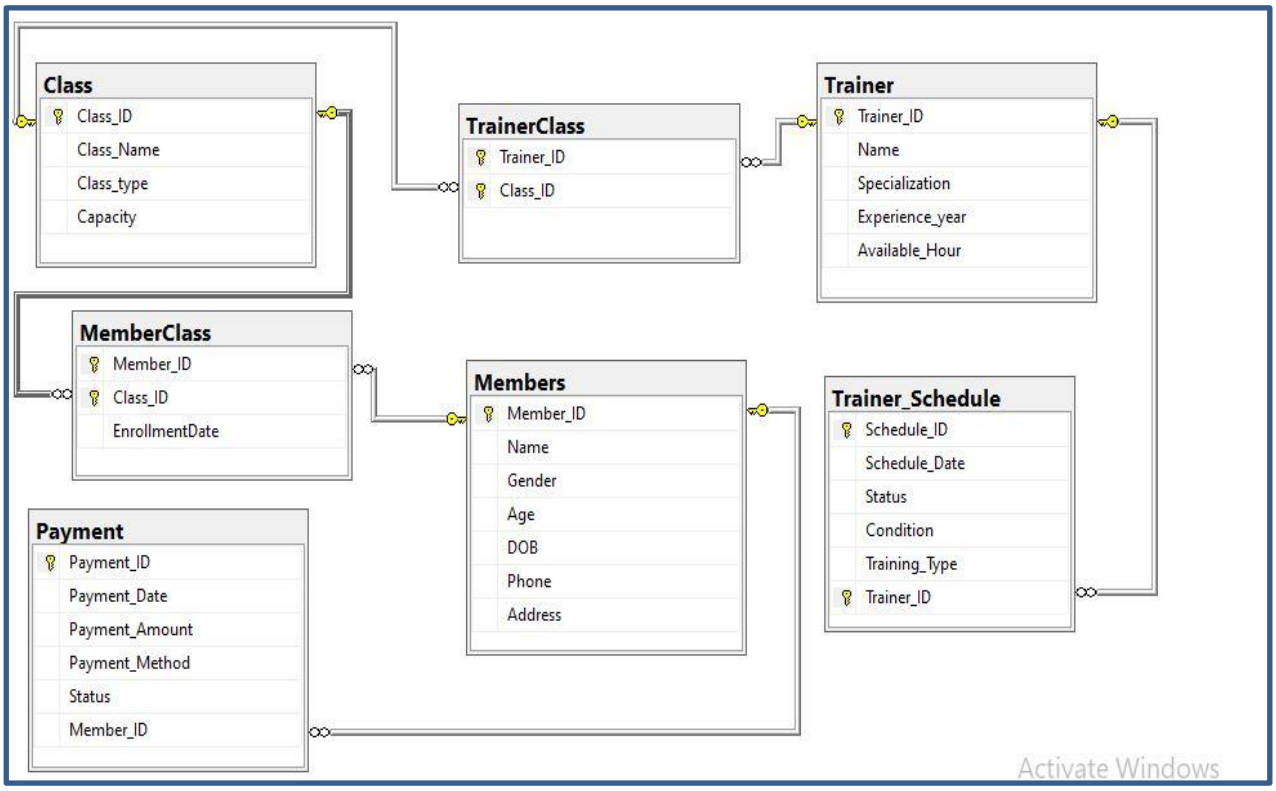
This project is about creating a database system for a fitness centre to help manage things like member sign-ups, trainer schedules, workout plans, and payments. The goal is to make daily tasks easier, reduce mistakes, and allow staff to quickly access important information. The system will also include tools to handle important operations safely, such as updating member details or processing payments, so that data stays correct and up to date.

As part of the project, we will use features like commit, rollback, and save points to manage changes in the database. We will also test how the system handles more than one person using it at the same time and make sure it works well in those situations. In addition, we will set up controls so that only certain users can see or change certain data, helping to keep the information safe. Lastly, we will show how to protect summary data so that private details cannot be guessed from it. This will help make the system secure, useful, and easy to manage.

#Tasks

Task_1

1. **Design a relational database application based on your chosen project title.**



Task_2

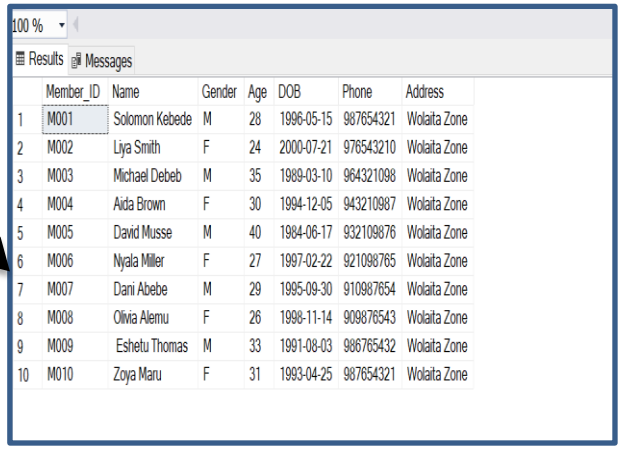
2. **Implement transaction management for critical operations. For example, in a bank management system, this would include fund transfers.**

```
--Transaction Implement
Begin Transaction
select*from Payment
Save Transaction first
commit
rollback
```

Task_3

3. **Demonstrate the use of commit, rollback and savepoints.**
   - Use of commit: used for save transaction permanently after commit no rollback!

Before Transaction



	Member_ID	Name	Gender	Age	DOB	Phone	Address
1	M001	Solomon Kebede	M	28	1996-05-15	987654321	Wolaita Zone
2	M002	Liya Smith	F	24	2000-07-21	976543210	Wolaita Zone
3	M003	Michael Debeb	M	35	1989-03-10	964321098	Wolaita Zone
4	M004	Aida Brown	F	30	1994-12-05	943210987	Wolaita Zone
5	M005	David Musse	M	40	1984-06-17	932109876	Wolaita Zone
6	M006	Nyala Miller	F	27	1997-02-22	921098765	Wolaita Zone
7	M007	Dani Abebe	M	29	1995-09-30	910987654	Wolaita Zone
8	M008	Olivia Alemu	F	26	1998-11-14	909876543	Wolaita Zone
9	M009	Eshetu Thomas	M	33	1991-08-03	986765432	Wolaita Zone
10	M010	Zoya Maru	F	31	1993-04-25	987654321	Wolaita Zone

```
select*from Members
--Transaction 1 Using insert opration on Members Table
begin transaction
INSERT INTO Members (Member_ID, Name, Gender, Age, DOB, Phone, Address) VALUES
('M0011', 'Selamawit Kebede', 'F', 28, '1996-05-15', 0987654344, 'Wolaita Zone');
Save transaction Memeber1;--save temporary
Commit transaction--save permantly
rollback
```

After Transaction and commit

	Member_ID	Name	Gender	Age	DOB	Phone	Address
1	M001	Solomon Kebede	M	28	1996-05-15	987654321	Wolaita Zone
2	M0011	Selamawit Kebede	F	28	1996-05-15	987654344	Wolaita Zone
3	M002	Liya Smith	F	24	2000-07-21	976543210	Wolaita Zone
4	M003	Michael Debeb	M	35	1989-03-10	964321098	Wolaita Zone
5	M004	Aida Brown	F	30	1994-12-05	943210987	Wolaita Zone
6	M005	David Musse	M	40	1984-06-17	932109876	Wolaita Zone
7	M006	Nyala Miller	F	27	1997-02-22	921098765	Wolaita Zone
8	M007	Dani Abebe	M	29	1995-09-30	910987654	Wolaita Zone
9	M008	Olivia Alemu	F	26	1998-11-14	909876543	Wolaita Zone
10	M009	Eshetu Thomas	M	33	1991-08-03	986765432	Wolaita Zone
11	M010	Zoya Maru	F	31	1993-04-25	987654321	Wolaita Zone

   - **Use of save point(Save Transaction):save transaction temporary allow rollback**

After Save point: saved temporary

```
select*from Payment
--Transaction 2 Using update opration on Members Table
begin transaction
update Payment set Payment_Amount=1400 where Payment_ID='P001'
Save transaction Payment1;--save temporary
Commit transaction--save permantly
rollback
```

	Payment_ID	Payment_Date	Payment_Amount	Payment_Method	Status	Member_ID
1	P001	2024-03-01	1400	Cash	Successful	M001
2	P002	2024-03-02	700	Credit Card	Successful	M002
3	P003	2024-03-03	600	Bank Transfer	Pending	M003
4	P004	2024-03-04	500	Cash	Successful	M004
5	P005	2024-03-05	800	Credit Card	Failed	M005
6	P006	2024-03-06	750	Bank Transfer	Successful	M006
7	P007	2024-03-07	550	Cash	Successful	M007
8	P008	2024-03-08	900	Credit Card	Pending	M008
9	P009	2024-03-09	650	Bank Transfer	Successful	M009
10	P010	2024-03-10	500	Cash	Successful	M010

- **Use of Rollback :Cancels all change made within transaction except commit transaction**
  - Above transaction save point state so used rollback cancel change

After rollback for above transaction

```
--Transaction 3 Rollback
begin transaction
rollback ;
```

100 %

Results | Messages

	Payment_ID	Payment_Date	Payment_Amount	Payment_Method	Status	Member_ID
1	P001	2024-03-01	500	Cash	Successful	M001
2	P002	2024-03-02	700	Credit Card	Successful	M002
3	P003	2024-03-03	600	Bank Transfer	Pending	M003
4	P004	2024-03-04	500	Cash	Successful	M004
5	P005	2024-03-05	800	Credit Card	Failed	M005
6	P006	2024-03-06	750	Bank Transfer	Successful	M006
7	P007	2024-03-07	550	Cash	Successful	M007
8	P008	2024-03-08	900	Credit Card	Pending	M008
9	P009	2024-03-09	650	Bank Transfer	Successful	M009
10	P010	2024-03-10	500	Cash	Successful	M010

Task_4

4. **<u>Simulate concurrent transactions and analyse the impact of concurrency control techniques</u>**
   - Use shard lock and exclusive lock

Use this cod for Check condition for time, Session ID and lock condition

chek.sql - BIFANAD...ANAD\Bifanad (56)) | Object Explorer | Fitness1.sql - BIFA...FANAD\Bifanad (52))

```
select
request_session_id ,
request_mode,
request_type ,
resource_type,
resource_description,
request_status
from sys.dm_tran_locks;
select
session_id,
wait_duration_ms,
wait_type ,
blocking_session_id,
resource_description
from sys.dm_os_waiting_tasks
select
session_id,
status ,
wait_type ,
wait_time ,
wait_resource,
command
from sys.dm_exec_requests
```

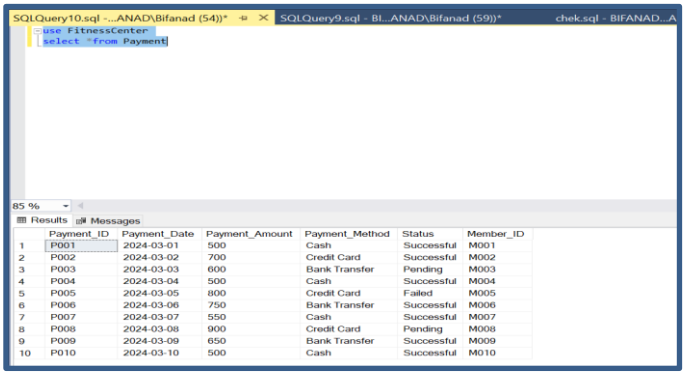
85 %

Results | Messages

	request_session_id	request_mode	request_type	resource_type	resource_description	request_status
1	56	S	LOCK	DATABASE		GRANT
2	52	S	LOCK	DATABASE		GRANT
3	55	S	LOCK	DATABASE		GRANT

- **Shared look**: in this case one or more Quires used the same table at the same time because of garneted for only read operation.

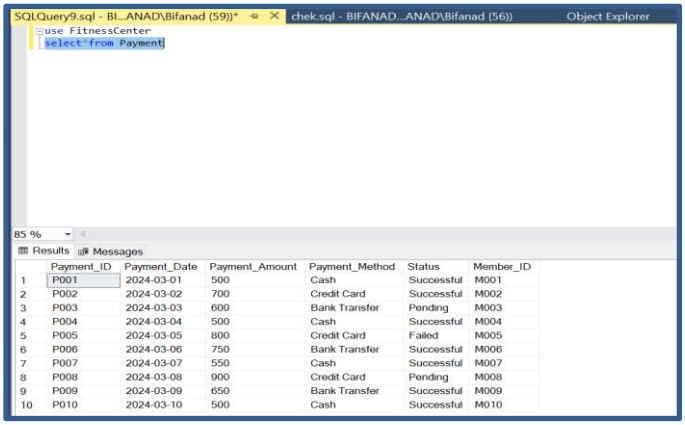
Session ID 54: select operation



```
use FitnessCenter
select *from Payment
```

	Payment_ID	Payment_Date	Payment_Amount	Payment_Method	Status	Member_ID
1	P001	2024-03-01	500	Cash	Successful	M001
2	P002	2024-03-02	700	Credit Card	Successful	M002
3	P003	2024-03-03	600	Bank Transfer	Pending	M003
4	P004	2024-03-04	500	Cash	Successful	M004
5	P005	2024-03-05	800	Credit Card	Failed	M005
6	P006	2024-03-06	750	Bank Transfer	Successful	M006
7	P007	2024-03-07	550	Cash	Successful	M007
8	P008	2024-03-08	900	Credit Card	Pending	M008
9	P009	2024-03-09	650	Bank Transfer	Successful	M009
10	P010	2024-03-10	500	Cash	Successful	M010

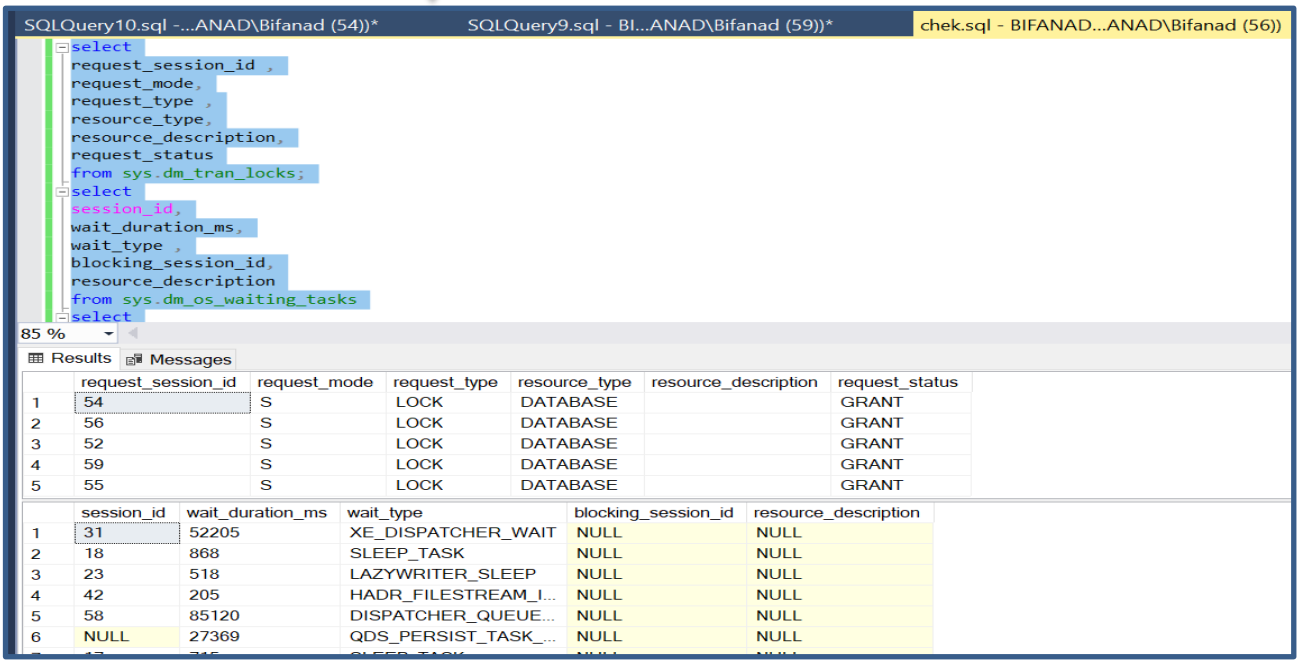
Session ID 57: select operation



```
use FitnessCenter
select*from Payment
```

	Payment_ID	Payment_Date	Payment_Amount	Payment_Method	Status	Member_ID
1	P001	2024-03-01	500	Cash	Successful	M001
2	P002	2024-03-02	700	Credit Card	Successful	M002
3	P003	2024-03-03	600	Bank Transfer	Pending	M003
4	P004	2024-03-04	500	Cash	Successful	M004
5	P005	2024-03-05	800	Credit Card	Failed	M005
6	P006	2024-03-06	750	Bank Transfer	Successful	M006
7	P007	2024-03-07	550	Cash	Successful	M007
8	P008	2024-03-08	900	Credit Card	Pending	M008
9	P009	2024-03-09	650	Bank Transfer	Successful	M009
10	P010	2024-03-10	500	Cash	Successful	M010

Using check code check above to session ID give the below out put this means the above two codes are sheared lock.



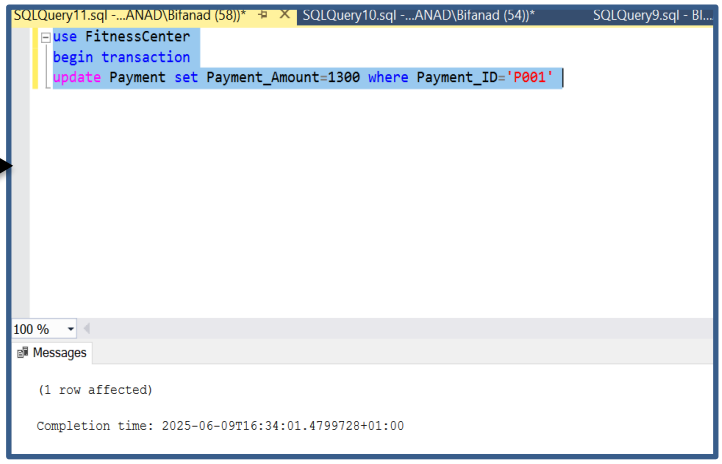
```
select
request_session_id ,
request_mode,
request_type ,
resource_type,
resource_description,
request_status
from sys.dm_tran_locks;
select
session_id,
wait_duration_ms,
wait_type ,
blocking_session_id,
resource_description
from sys.dm_os_waiting_tasks
select
```

	request_session_id	request_mode	request_type	resource_type	resource_description	request_status
1	54	S	LOCK	DATABASE		GRANT
2	56	S	LOCK	DATABASE		GRANT
3	52	S	LOCK	DATABASE		GRANT
4	59	S	LOCK	DATABASE		GRANT
5	55	S	LOCK	DATABASE		GRANT

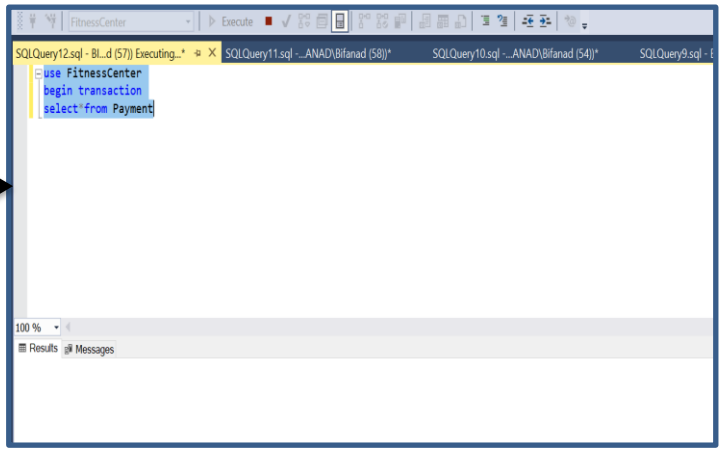
	session_id	wait_duration_ms	wait_type	blocking_session_id	resource_description
1	31	52205	XE_DISPATCHER_WAIT	NULL	NULL
2	18	868	SLEEP_TASK	NULL	NULL
3	23	518	LAZYWRITER_SLEEP	NULL	NULL
4	42	205	HADR_FILESTREAM_I...	NULL	NULL
5	58	85120	DISPATCHER_QUEUE...	NULL	NULL
6	NULL	27369	QDS_PERSIST_TASK_...	NULL	NULL

- **Exclusive Lock**: only on transaction are apply because of garneted for read and write

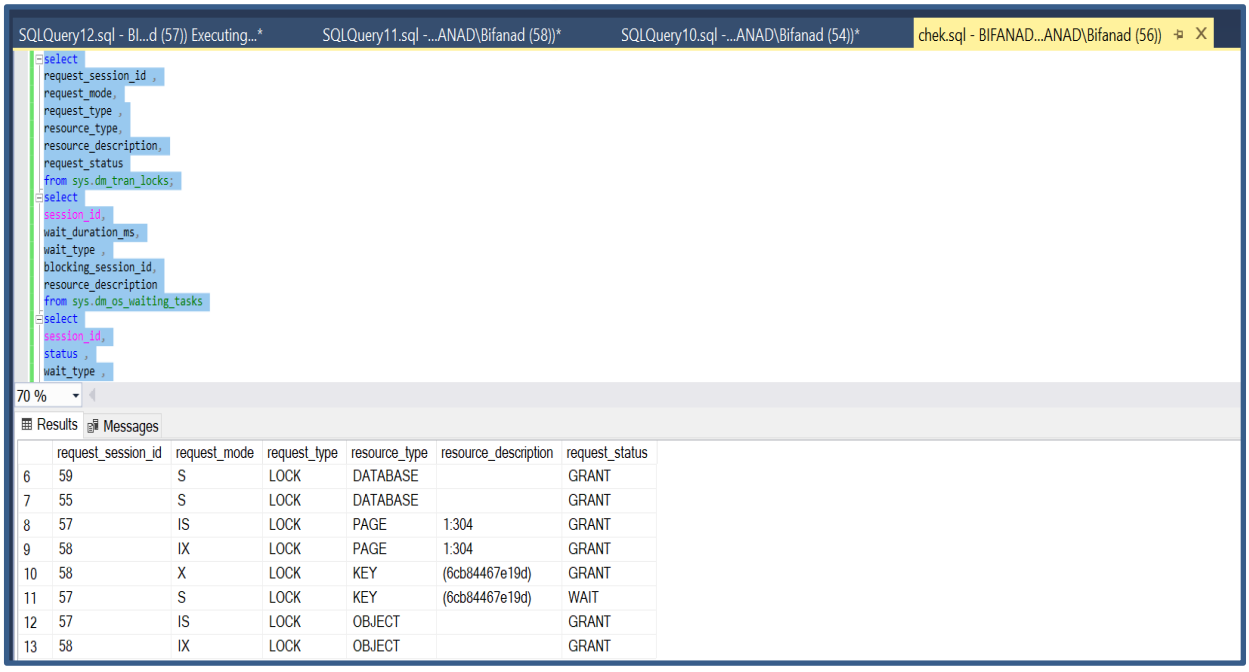
Session ID 58 Update operation



Session ID 57 select operation



Using check code check above to session ID give the below out put this means the above two codes are Exclusive lock session ID 57 wait Commit or rollback of session 58 so in this time Dead lock appear.



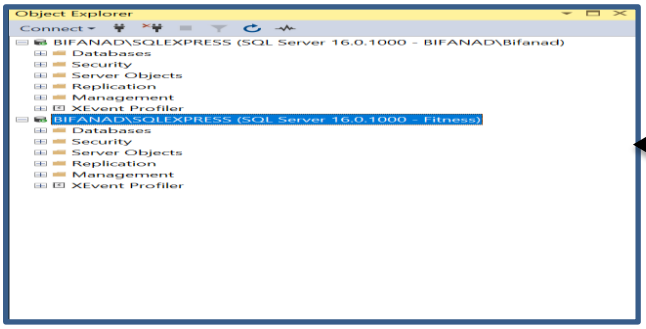
	request_session_id	request_mode	request_type	resource_type	resource_description	request_status
6	59	S	LOCK	DATABASE		GRANT
7	55	S	LOCK	DATABASE		GRANT
8	57	IS	LOCK	PAGE	1:304	GRANT
9	58	IX	LOCK	PAGE	1:304	GRANT
10	58	X	LOCK	KEY	(6cb84467e19d)	GRANT
11	57	S	LOCK	KEY	(6cb84467e19d)	WAIT
12	57	IS	LOCK	OBJECT		GRANT
13	58	IX	LOCK	OBJECT		GRANT

Task_5

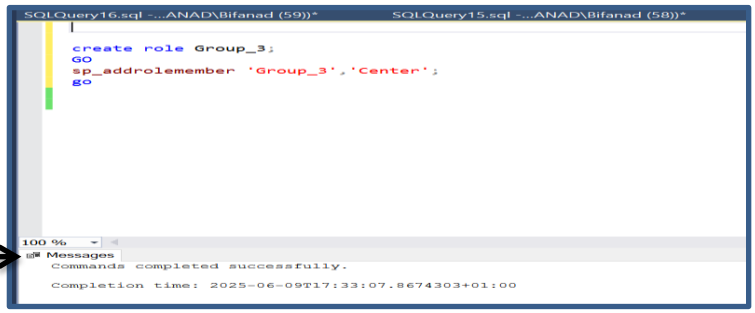
5. **<u>This are an additional task for your project "Implement discretionary access control by granting and revoking privileges to users. Demonstrate how to handle statistical database security.</u>''**

Create Login and user

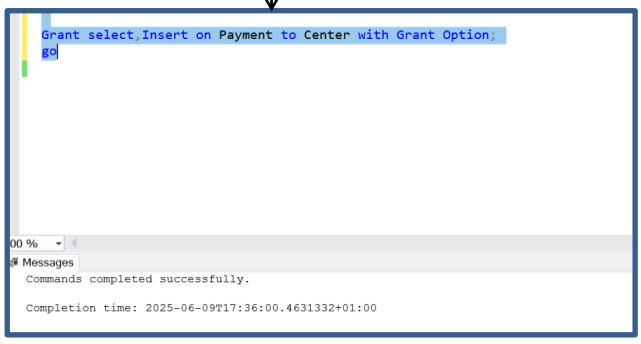
Login Fitness; user Center



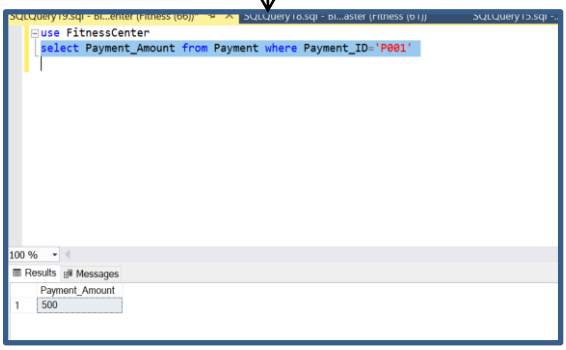
Create Role: Group_3



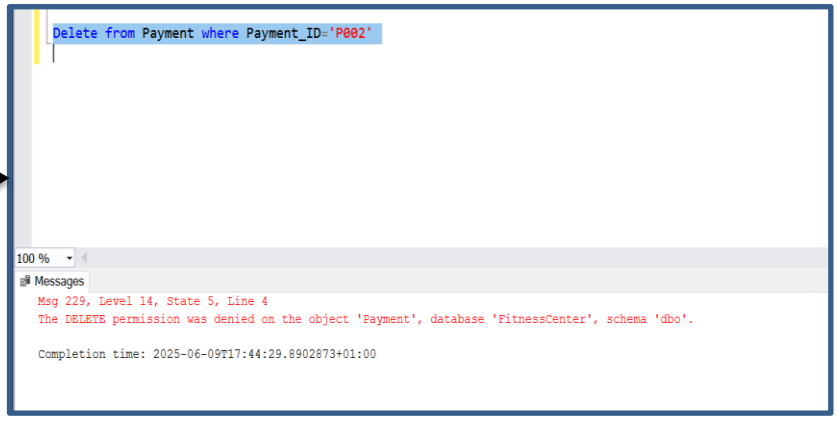
Give Grant option for Fitness from Bifanad on table

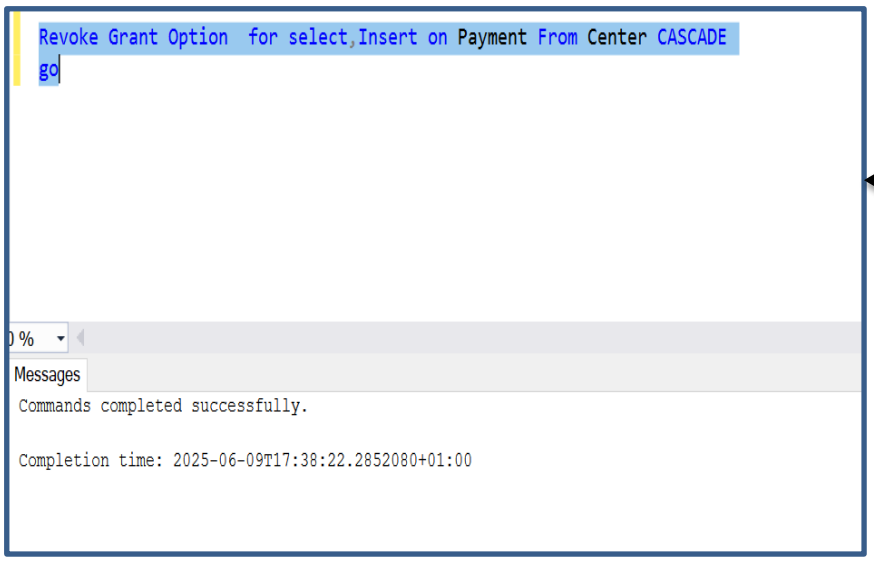


Check Select operation



Check Delete operation (Not Grant)





Remove all Grant Option

CONCLUSION

In conclusion, this project will help build a useful and secure database system for a fitness center that makes daily work easier and keeps information safe. By adding features like transaction controls, user access settings, and data protection methods, the system will be able to handle important tasks without losing or mixing up data. It will also work well when many people use it at the same time. Overall, the project will create a strong and easy-to-use system that supports the smooth running of the fitness center.