

---

*Lab 09*

---

---

*Q1*

---

In the MY\_STAFF table that you created in the last lab, Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

SQL QUERY:

```
ALTER TABLE MY_STAFF
ADD Commission NUMBER (2,2);
ALTER TABLE MY_STAFF
ADD CONSTRAINT CHK_Commission CHECK( Comission>0 );
```

---

*Q2*

---

Create a table STAFF\_UPDATES with the same structure as MY\_STAFF. Keep it unconstrained i.e. do not define the primary key and foreign key constraints in this table. Populate it with the following data:

SQL QUERY:

```
insert INTO STAFF_UPDATES (employee_id,first_name,last_name,user_id, salary, commission,  
depart_id) values (1, 'Ralph', 'Patel','rpatel' ,920,0.05 ,101);
```

```
insert INTO STAFF_UPDATES (employee_id,first_name,last_name,user_id, salary, commission,  
depart_id) values (6,' Zara', 'Lee', 'zlee', 110, 0.01,103);
```

The screenshot shows an SQL Worksheet with two insert statements. The first statement is on line 3, and the second is on line 5. The interface includes a toolbar with icons for running queries, saving, and other functions. Below the code editor, there are tabs for 'Query result', 'Script output', 'DBMS output', 'Explain Plan', and 'SQL history'. The 'Script output' tab is selected, showing the execution of the second insert statement. The output indicates that 1 row was inserted successfully, and the elapsed time was 00:00:00.003.

```
1  
2  
3 insert INTO STAFF_UPDATES (employee_id,first_name,last_name,user_id, salary, commission,  
4 depart_id) values (1, 'Ralph', 'Patel','rpatel' ,920,0.05 ,101);  
5 insert INTO STAFF_UPDATES (employee_id,first_name,last_name,user_id, salary, commission,  
6 depart_id) values (6,' Zara', 'Lee', 'zlee', 110, 0.01,103);  
7
```

Query result   **Script output**   DBMS output   Explain Plan   SQL history

Error at Line: 4 Column: 0

```
SQL>  
insert INTO STAFF_UPDATES (employee_id,first_name,last_name,user_id, salary, commission,  
depart_id) values (6,' Zara', 'Lee', 'zlee', 110, 0.01,103);
```

1 row inserted.

Elapsed: 00:00:00.003

---

Q3

---

Give logical reason why we made MY\_STAFF table unconstrained.

We deliberately made STAFF\_UPDATES unconstrained to allow flexible data manipulation without restrictions. For Testing or Temporary Data, bypassing validations and data cleaning and correction

---

#### Q 4

---

Insert a few departments in MY\_DEPART (that you created in the last lab session),  
(e.g., 101 – HR, 102 – Sales, 103 – IT) and update the existing staff to be part of one of these departments.

The screenshot shows a SQL worksheet interface with a toolbar at the top containing icons for running queries, saving, and other functions. The main area displays the following SQL code:

```
1  insert into MY_DEPART (DEPART_ID, DEPART_NAME) values (101, 'HR');
2  insert into MY_DEPART (DEPART_ID, DEPART_NAME) values (102, 'Sales');
3  insert into MY_DEPART (DEPART_ID, DEPART_NAME) values (103, 'IT');
4
5  update staff_updates
6  set depart_id = 101
7  where employee_id=1;
8
9
```

Below the code editor, there are tabs for "Query result", "Script output", "DBMS output", "Explain Plan", and "SQL history". The "Script output" tab is currently selected. It shows the execution of the update query:

```
SQL> update staff_updates
      set depart_id = 101
      where employee_id=1
```

Below the query execution, it states "1 row updated." and "Elapsed: 00:00:00.005".

---

q5

---

Use a MERGE statement to update existing records in MY\_STAFF if employee\_id matches, and insert new rows if they don't exist. Verify the changes in MY\_STAFF.

Sql query:

Merge into staff\_updates s using my\_staff m

On (s.employee\_id=m.employee\_id)

When matched then

Update set s.first\_name=m.first\_name, s.last\_name=m.last\_name, s.user\_id=m.userid,  
s.salary=m.salary

When not matched then

Insert (employee\_id ,first\_name,last\_name, user\_id,salary ) values  
(m.employee\_id,m.first\_name,m.last\_name,m.userid,m.salary);

---

Q6

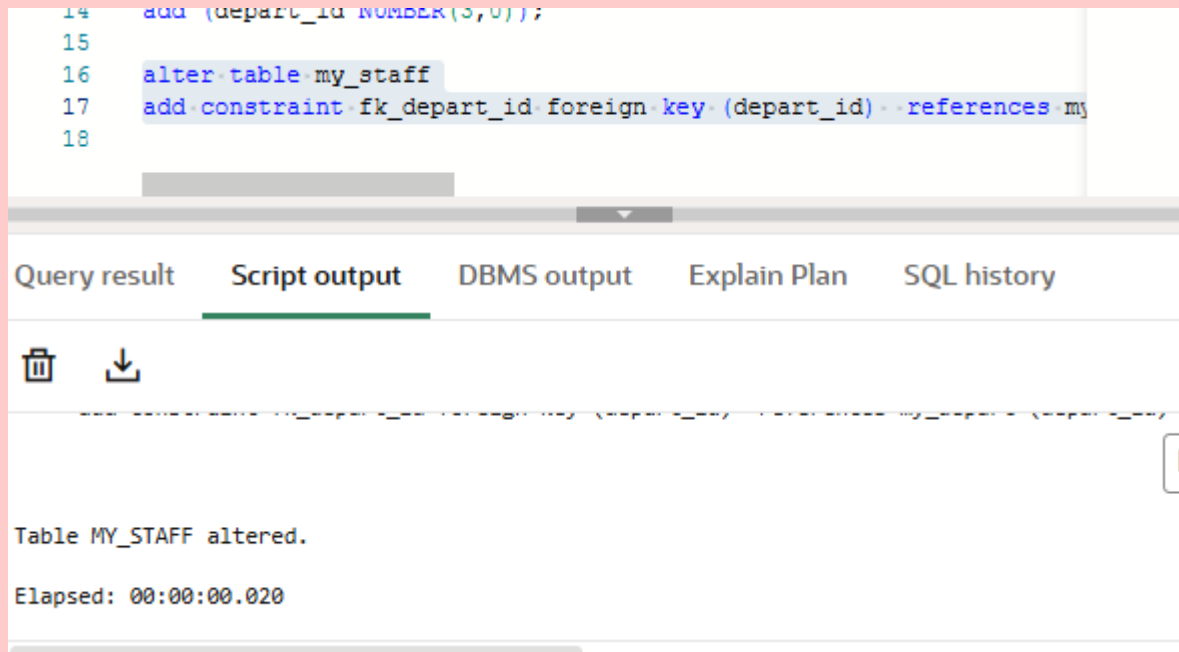
---

Alter the MY\_STAFF table to add a foreign key constraint on depart\_id that references the MY\_DEPART table with ON DELETE CASCADE.

SQL Query:

```
alter table my_staff
```

```
add constraint fk_depart_id foreign key (DEPART_ID) REFERENCES MY_DEPART (DEPART_ID) on  
delete CASCADE;
```



```
14 add (depart_id NUMBER(3,0));  
15  
16 alter table my_staff  
17 add constraint fk_depart_id foreign key (depart_id) references my  
18
```

Query result   **Script output**   DBMS output   Explain Plan   SQL history

Table MY\_STAFF altered.  
Elapsed: 00:00:00.020

---

*Q7*

---



Delete department 103 and verify that all staff associated with it are automatically removed from MY\_STAFF.

Delete from my\_depart where depart\_id =103;

Select employee\_id from my\_staff where depart\_id=103;

```
18
19 delete from MY_DEPART where depart_id=103;
20
21 Select employee_id FROM my_staff where depart_id=103;
22
```

Query result   Script output   DBMS output   Explain Plan   SQL history

  Download   Execution time: 0.063 seconds

EMPLOYEE\_ID

0 items to display.

*Q8*

Drop the foreign key constraint on depart\_id, and recreate it using ON DELETE SET NULL.

```

23 alter TABLE my_staff
24 drop constraint fk_depart_id;
25 |

```

Query result   **Script output**   DBMS output   Explain Plan   SQL history



Table MY\_STAFF altered.

Elapsed: 00:00:00.017

```

25
26 alter table my_staff
27 add constraint fk_depart_id FOREIGN key (depart_id) references my

```

Query result   **Script output**   DBMS output   Explain Plan   SQL history



Table MY\_STAFF altered.

Elapsed: 00:00:00.016

9.

Insert a department 104 – "Finance" and assign it to a new staff member.

```
insert into MY_DEPART (depart_id,depart_name) values (104,'Finance');
```

```
update MY_STAFF
```

```
set depart_id=104
```

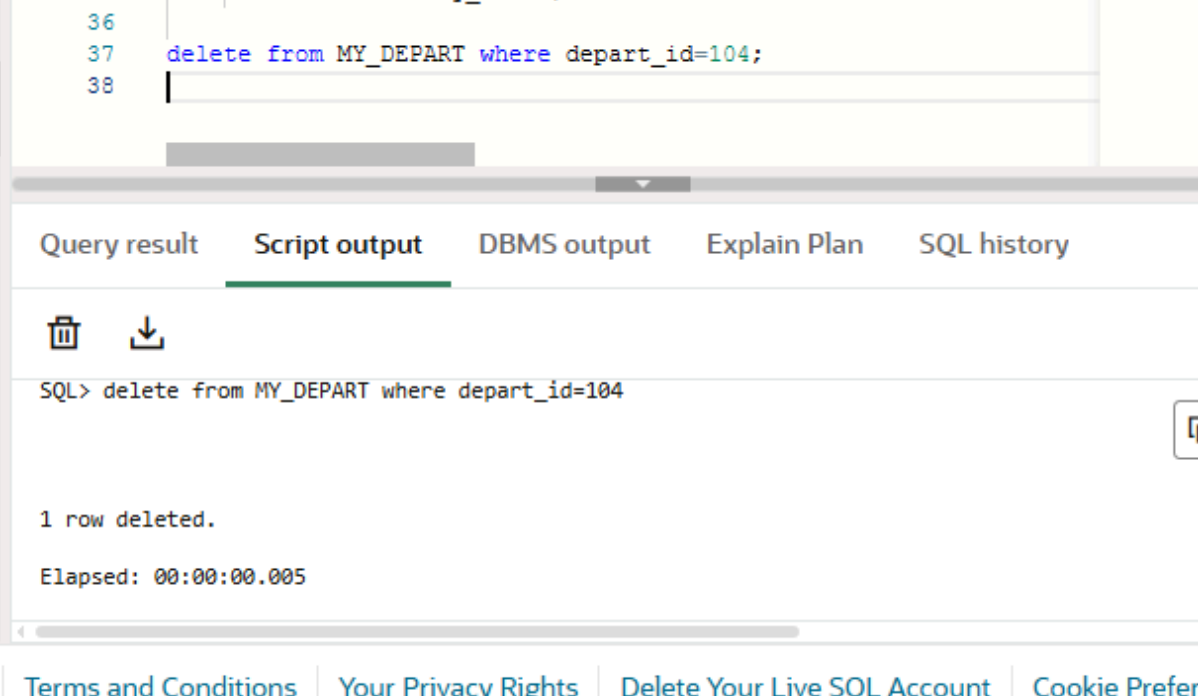
```
where employee_id=2;
```

---

10.

---

Delete department 104 and confirm that the depart\_id of affected staff is now set to NULL.



The screenshot shows a SQL IDE interface. The top section contains a code editor with the following SQL query:

```
36  
37 delete from MY_DEPART where depart_id=104;  
38
```

Below the code editor, there are tabs for "Query result", "Script output", "DBMS output", "Explain Plan", and "SQL history". The "Script output" tab is currently selected and highlighted with a green underline. Below the tabs, there are icons for a trash can and a download arrow. The main output area displays the following text:

```
SQL> delete from MY_DEPART where depart_id=104
```

1 row deleted.

Elapsed: 00:00:00.005

At the bottom of the IDE, there are links for "Terms and Conditions", "Your Privacy Rights", "Delete Your Live SQL Account", and "Cookie Preferences".

```
Select * from my_staff where depart_id=104;
```



---

11.

---

Create an audit table named STAFF\_AUDIT with the following columns:  
(employee\_id NUMBER, action\_type VARCHAR2(10), action\_date DATE).

```
39 select * from my_staff where depart_id=104,  
40  
41 create table staff_audit (employee_id NUMBER, action_type VARCHAR2(10), action_date DATE);
```

---

Query result   **Script output**   DBMS output   Explain Plan   SQL history




 

Table STAFF\_AUDIT created.

Elapsed: 00:00:00.010



---

12

---

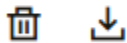
Create a trigger on MY\_STAFF that fires AFTER INSERT OR DELETE and records the employee\_id, action type ('INSERT' or 'DELETE'), and current date into STAFF\_AUDIT.

```

1
2 CREATE OR REPLACE TRIGGER trg_staff_audit
3 AFTER INSERT OR DELETE ON MY_STAFF
4 FOR EACH ROW
5 BEGIN
6     IF inserting THEN
7         INSERT INTO STAFF_AUDIT (employee_id, action_type, action_
8             VALUES (:NEW.employee_id, 'INSERT', SYSDATE);
9     ELSIF deleting THEN
10        INSERT INTO STAFF_AUDIT (employee_id, action_type, action_
11            VALUES (:OLD.employee_id, 'DELETE', SYSDATE);
12    END IF;
13 END;
14

```

Query result   **Script output**   DBMS output   Explain Plan   SQL history



BEGIN...  
Show more...

Trigger TRG\_STAFF\_AUDIT compiled

Elapsed: 00:00:00.021

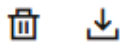
### Q 13.

Test the trigger by inserting and deleting a record in MY\_STAFF. Query STAFF\_AUDIT to verify entries

insert into MY\_STAFF (employee\_id,first\_name,last\_name,user\_id,salary,depart\_id) values (114,'Sam','Luke',1454,15000,101);

```
14
15  select * from MY_DEPART;
16
17  insert into MY_STAFF (employee_id,first_name,last_name,user_id,salary,depart_id) values (101,'John','Doe',1001,12000,101);
18
19
```

Query result   **Script output**   DBMS output   Explain Plan   SQL history



SQL>  
insert into MY\_STAFF (employee\_id,first\_name,last\_name,user\_id,salary,depart\_id) values (101,'John','Doe',1001,12000,101);  
  
1 row inserted.

delete from MY\_STAFF where depart\_id=101;

```
3 delete from MY_STAFF where depart_id=101;
4
5 select *.FROM staff_audit;
6
7
```

Query result   Script output   DBMS output   Explain Plan   SQL history

Download   Execution time: 0.009 seconds

	EMPLOYEE_ID	ACTION_TYPE	ACTION_DATE
	114	INSERT	5/7/2025, 7:26:44 P
	115	INSERT	5/7/2025, 7:31:45 P
	114	DELETE	5/7/2025, 7:34:06 P
	115	DELETE	5/7/2025, 7:34:06 P