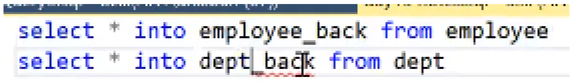
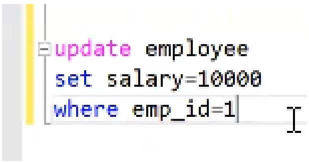
Advanced update-insert

1. Before learning Advanced_insert & update..we will take backup of our table



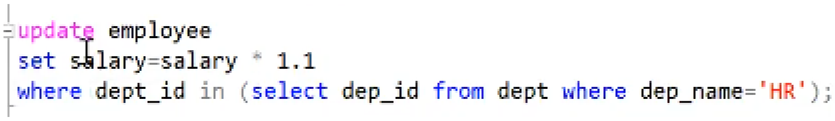
```
select * into employee_back from employee
select * into dept_back from dept
```

2. Update
3. The basic update which we have learned is looks like this….and this is within a table..



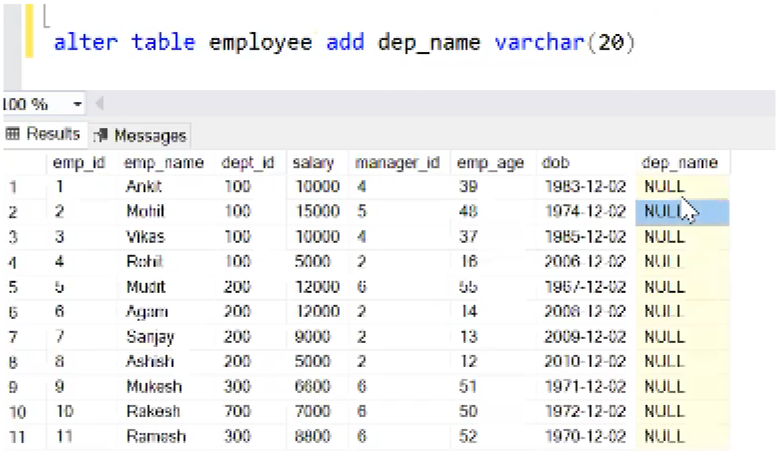
```
update employee
set salary=10000
where emp_id=1
```

4. Here we are updating the ..from the result of a query..see pic and understand
5.



```
update employee
set salary=salary * 1.1
where dept_id in (select dep_id from dept where dep_name='HR');
```

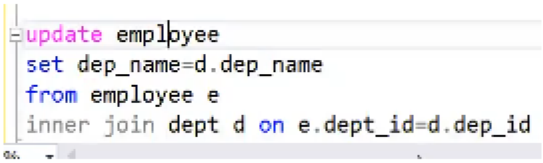
6. Now we have this dept name ..see pic below



```
alter table employee add dep_name varchar(20)
```

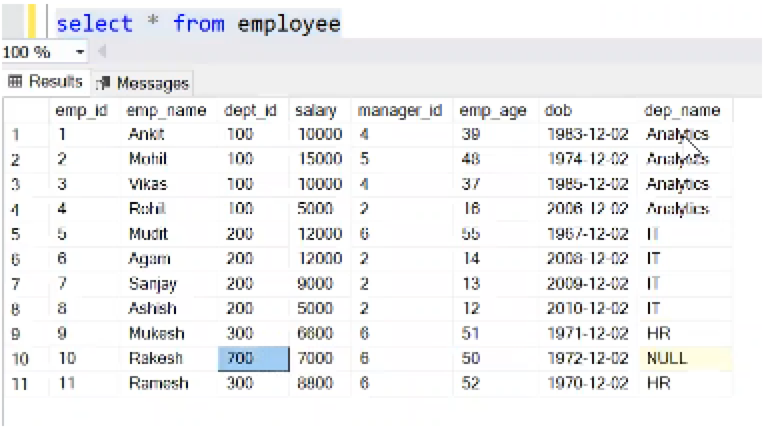
| | emp_id | emp_name | dept_id | salary | manager_id | emp_age | dob | dep_name |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Ankit | 100 | 10000 | 4 | 39 | 1983-12-02 | NULL |
| 2 | 2 | Mohil | 100 | 15000 | 5 | 48 | 1974-12-02 | NULL |
| 3 | 3 | Vikas | 100 | 10000 | 4 | 37 | 1985-12-02 | NULL |
| 4 | 4 | Rohit | 100 | 5000 | 2 | 16 | 2006-12-02 | NULL |
| 5 | 5 | Mudit | 200 | 12000 | 6 | 55 | 1967-12-02 | NULL |
| 6 | 6 | Agam | 200 | 12000 | 2 | 14 | 2008-12-02 | NULL |
| 7 | 7 | Sanjay | 200 | 9000 | 2 | 13 | 2009-12-02 | NULL |
| 8 | 8 | Ashish | 200 | 5000 | 2 | 12 | 2010-12-02 | NULL |
| 9 | 9 | Mukesh | 300 | 6800 | 6 | 51 | 1971-12-02 | NULL |
| 10 | 10 | Rakesh | 700 | 7000 | 6 | 50 | 1972-12-02 | NULL |
| 11 | 11 | Ramesh | 300 | 8800 | 6 | 52 | 1970-12-02 | NULL |

7. 
8. And we want to populate names for this dep_nam..we used inner join..and retrieved dep_names from dept table…see pic and understand
9.



```
update employee
set dep_name=d.dep_name
from employee e
inner join dept d on e.dept_id=d.dep_id
```

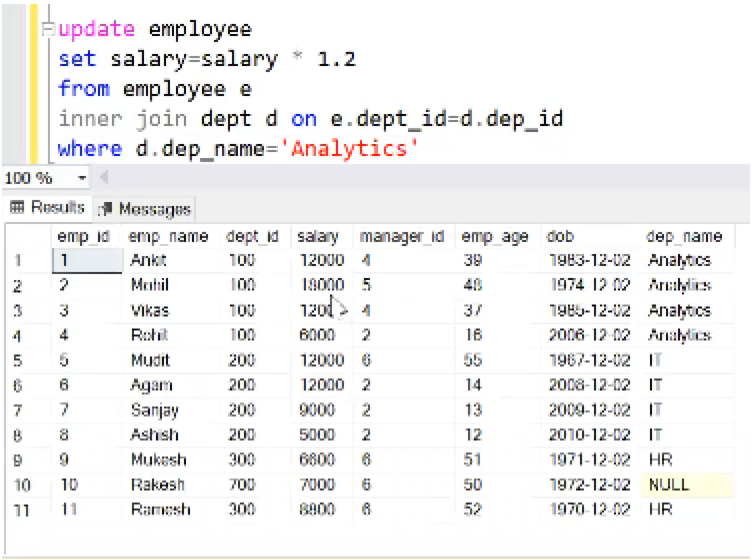
10.



```sql
select * from employee
```

| | emp_id | emp_name | dept_id | salary | manager_id | emp_age | dob | dep_name |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Ankit | 100 | 10000 | 4 | 39 | 1983-12-02 | Analytics |
| 2 | 2 | Mohit | 100 | 15000 | 5 | 48 | 1974-12-02 | Analytics |
| 3 | 3 | Vikas | 100 | 10000 | 4 | 37 | 1985-12-02 | Analytics |
| 4 | 4 | Rohit | 100 | 5000 | 2 | 16 | 2006-12-02 | Analytics |
| 5 | 5 | Mudit | 200 | 12000 | 6 | 55 | 1967-12-02 | IT |
| 6 | 6 | Agam | 200 | 12000 | 2 | 14 | 2008-12-02 | IT |
| 7 | 7 | Sanjay | 200 | 9000 | 2 | 13 | 2009-12-02 | IT |
| 8 | 8 | Ashish | 200 | 5000 | 2 | 12 | 2010-12-02 | IT |
| 9 | 9 | Mukesh | 300 | 6800 | 6 | 51 | 1971-12-02 | HR |
| 10 | 10 | Rakesh | 700 | 7000 | 6 | 50 | 1972-12-02 | NULL |
| 11 | 11 | Ramesh | 300 | 8800 | 6 | 52 | 1970-12-02 | HR |

11. Next…if you want to update the salary of a particular dept ..we use



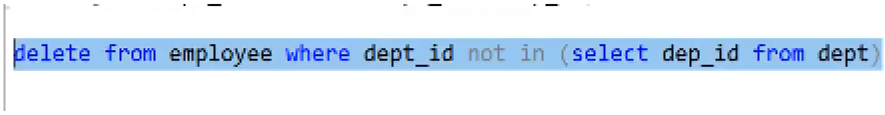
```sql
update employee
set salary=salary * 1.2
from employee e
inner join dept d on e.dept_id=d.dep_id
where d.dep_name='Analytics'
```

| | emp_id | emp_name | dept_id | salary | manager_id | emp_age | dob | dep_name |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Ankit | 100 | 12000 | 4 | 39 | 1983-12-02 | Analytics |
| 2 | 2 | Mohit | 100 | 18000 | 5 | 48 | 1974-12-02 | Analytics |
| 3 | 3 | Vikas | 100 | 1200 | 4 | 37 | 1985-12-02 | Analytics |
| 4 | 4 | Rohit | 100 | 6000 | 2 | 16 | 2006-12-02 | Analytics |
| 5 | 5 | Mudit | 200 | 12000 | 6 | 55 | 1967-12-02 | IT |
| 6 | 6 | Agam | 200 | 12000 | 2 | 14 | 2008-12-02 | IT |
| 7 | 7 | Sanjay | 200 | 9000 | 2 | 13 | 2009-12-02 | IT |
| 8 | 8 | Ashish | 200 | 5000 | 2 | 12 | 2010-12-02 | IT |
| 9 | 9 | Mukesh | 300 | 6800 | 6 | 51 | 1971-12-02 | HR |
| 10 | 10 | Rakesh | 700 | 7000 | 6 | 50 | 1972-12-02 | NULL |
| 11 | 11 | Ramesh | 300 | 8800 | 6 | 52 | 1970-12-02 | HR |

12.

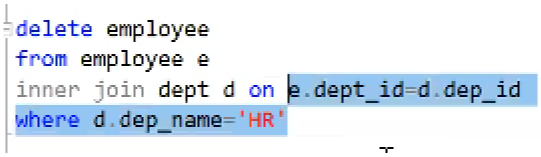
Delete

1. Here we are deleting the id…which were retrieved from the sub query…see pic



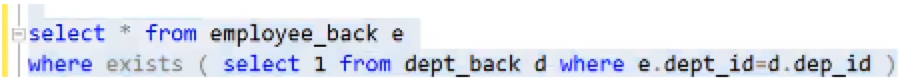
```sql
delete from employee where dept_id not in (select dep_id from dept)
```

2.
3. And to delete employee from particular dept..we use



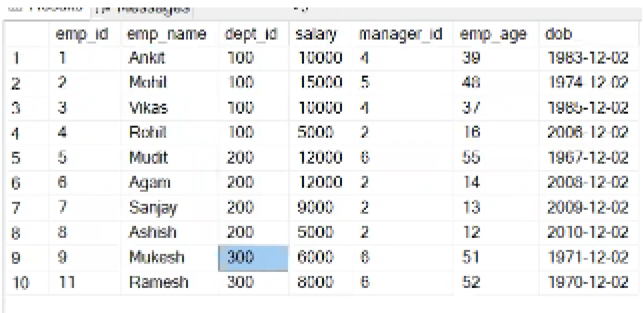
```sql
delete employee
from employee e
inner join dept d on e.dept_id=d.dep_id
where d.dep_name='HR'
```

4. The above result can also be achieved thru sub queries
5. Try the above query without any filters like where(task for myself)
6. Way of using Exists

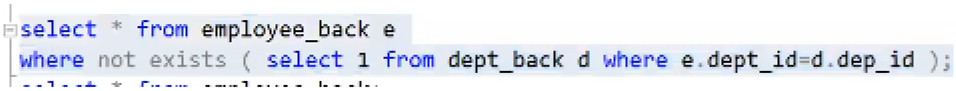


```sql
select * from employee_back e
where exists ( select 1 from dept_back d where e.dept_id=d.dep_id )
```

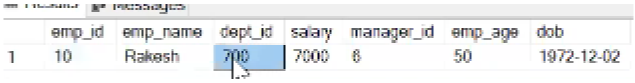
7.

8. 

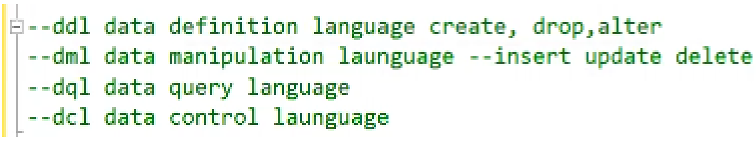
|  | emp_id | emp_name | dept_id | salary | manager_id | emp_age | dob |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Ankit | 100 | 10000 | 4 | 39 | 1983-12-02 |
| 2 | 2 | Mohil | 100 | 15000 | 5 | 48 | 1974-12-02 |
| 3 | 3 | Vikas | 100 | 10000 | 4 | 37 | 1985-12-02 |
| 4 | 4 | Rohil | 100 | 5000 | 2 | 16 | 2006-12-02 |
| 5 | 5 | Mudit | 200 | 12000 | 6 | 55 | 1967-12-02 |
| 6 | 6 | Agam | 200 | 12000 | 2 | 14 | 2008-12-02 |
| 7 | 7 | Sanjay | 200 | 9000 | 2 | 13 | 2009-12-02 |
| 8 | 8 | Ashish | 200 | 5000 | 2 | 12 | 2010-12-02 |
| 9 | 9 | Mukesh | 300 | 6000 | 6 | 51 | 1971-12-02 |
| 10 | 11 | Ramesh | 300 | 8000 | 6 | 52 | 1970-12-02 |

9. Here in the above query we retrieved all the employees whose dept_id are present in dept table
10. Basically what it is doing is….for each query(row) in employee table…it checks with the output of inner query….whether the dept_id is present in employee table or not(think this way)...if our inner query returns at least one row..then **exists** treat that as **true.**
11. And whenever there are no rows…then the inner query returns **0 and exists treats that as False.** (in our case for dept_id 700 from employees table…gives us FALSE)
12. Not Exists
13. 

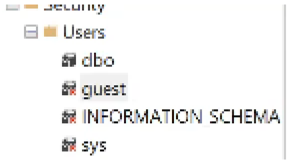
```
select * from employee_back e
where not exists ( select 1 from dept_back d where e.dept_id=d.dep_id );
```

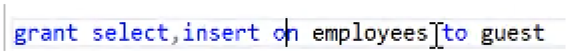
14. 

|  | emp_id | emp_name | dept_id | salary | manager_id | emp_age | dob |
|---|---|---|---|---|---|---|---|
| 1 | 10 | Rakesh | 700 | 7000 | 6 | 50 | 1972-12-02 |

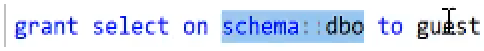
15. Refer above points…Not exists is opposite of exists
16. Generally we don't use exists..because we can perform this operation using the other means too
17. 

```
--ddl data definition language create, drop,alter
--dml data manipulation launguage --insert update delete
--dql data query language
--dcl data control launguage
```

18. Data control language
19. In DCL we have 2 main things..which are **grant , revoke**
20. Here we have a guest user
21. 

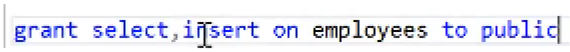
22. In the below line…we are giving permission to use select and insert commands for guest user
23. 

```
grant select,insert on employees to guest
```

24. If we want to give permission to all tables in the current dbo(for me Learning sql db)..then we will use **schema::dbo**
25. 

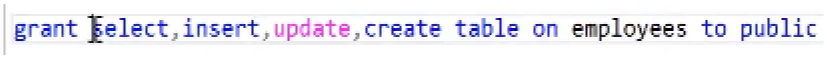
```
grant select on schema::dbo to guest
```

26. If we want to give permission to all the users at a time..then we will use **public**

27. 

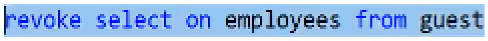
```
grant select,insert on employees to public
```

28. We can give all type of sql keyword commands like



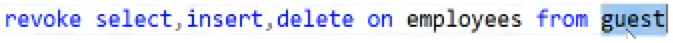
```
grant select,insert,update,create table on employees to public
```

29. And if we want to revoke..we use **revoke**
30. Here we have removed **select** keyword access on employees table for guest user



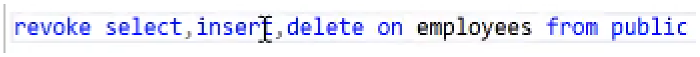
```
revoke select on employees from guest
```

31. We can give multiple keyword commands at once like this



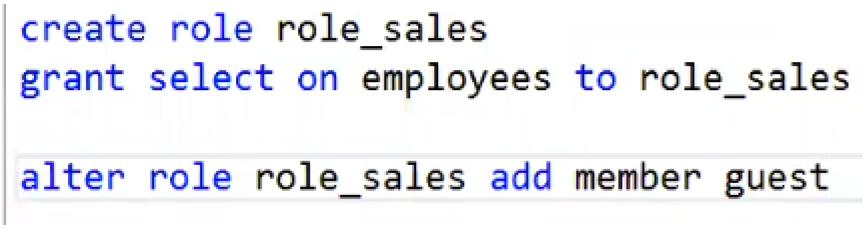
```
revoke select,insert,delete on employees from guest
```

32. If we want to revoke for all the users at once..we use public



```
revoke select,insert,delete on employees from public
```

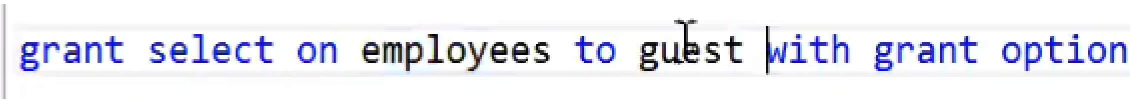
33. Next we have roles



```
create role role_sales
grant select on employees to role_sales

alter role role_sales add member guest
```

34. 
35. In roles…for example in admin role..there will be specific users for admin roles..All this users will be present in the admin role..
36. Now if we give grant access on the admin role…it applies to all the users in admin role..refer chatgpt if doubt
37. Generally guest users can use grant command..
38. If u want to give guest user the access of using grant command..we have to use



```
grant select on employees to guest with grant option
```

39. Now guest user can grant other users too
40. Next learn more about **TCL-Transaction control language(roll back and commit)** from online