mysql -u root -p

Command are in sql

06 NOVEMBER WEDNESDAY
310-055 • WK 45

DDL :- DATA DEFINITION LANGUAGE

details = variablename

☆ start - mysql -u root -p
☆ show all database record : show databases ;
☆ for create databases : create databases ___ ;
☆ if show many databases :- use ___ ;
☆ for create table : create table details(id int primary key, name varchar(20), salary float);
☆ for see create table : desc details ;

field	Type	Null	Key	default	extra
id	int(11)	No	PRI	Null	
name	varchar(20)	Yes		Null	
salary	float	Yes		Null	
age	int(11)	Yes		"	

☆ for add new raw
= alter table details add age int;

☆ for add new raw on first : alter table details add Email varchar(20) first;

field					
Email					
id					
name					
Salary					
Age					

☆ for add random place : alter table details add contactno bigint after name ;

field	Type	null	key	def	ext.
Email					
id					
name					
contactno					
salary					
Age					

2013

✯ for change key name :- alter table details change id Golu int ;

field	Type				
email	v(20)				
Golu	int(11)				
name	varchar(30)				
contactno	bigint				
Salary	float				
Age	int(11)				

This raw delete.

✯ for delete particular key :- alter table details drop salary

✯ for change datatype of key :- alter table modify name char(30)

change :

name	char(30)				

✯ for check change details : alter table rename details

✯ for Delete table : drop table details ;

✯ for check table : desc details

DML :- DATA MANIPULATION LANGUAGE

✯ Insert ~~into~~ value in table : insert into details values("golu@123", 10, "golu", 9135445880, 20,000, 19) ;

Email	golu (change id)	name	contactno	salary	Age
golu@123	10	Golu	9135445880	20,000	19

✯ for see details : select * from details ;

* for insert a single data: insert into details(name) values("Rohan");

cd - select * from details;

Email	~~Golu~~ id	name	contactno	salary	Age
null	0	Rohan	Null	Null	Null
golu@123	10	Golu	9135448850	20,000	19

change 20

* change id no: update details set id = 20 where id=0;

* updata details taking refrence id: - update details set Email= 'abhi@123', contactno = 85210, salary= 10,000, Age = 20 where id= 20;

cd- select * from details;

Email	~~Golu~~ Id	name	contactno	salary	Age
abhi@123	20	Rohan	85210	10,000	20
golu@123	10	Golu	9135448850	20,000	19

Note: Taking refrence Id we can change any value.
like:- update details set name="Abhijeet" where id=10;

* for Delete row: delete from details where id = ----;

* for Remove all Details: truncate details;

* for see table after truncate command:- desc details;

Field	type	null
Email	v(20)	N
Golu	int(11)	Y
name	v(30)	Y
contactno	bigint	Y
Salary	Float	Y

OCTOBER 2013

M	T	W	T	F	S	S	M	T	W	T	F	S	S
	1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31										

Table :- insert into details values("sk@123", 30, "John", 7913, 50,000, 35);
("Tk@123, 40, "Ranjeet", 8723, 5000, 24);

Email	Holly (id)	name	contactno	salary	Age
gk@123	Holly 10	Holly	9133445880	20000	19
abhinav@123	Rohan 20	Rohan	852110	1000	20
sk@123	John 30	John	7913	50000	35
Tk@123	Ranjeet 40	Ranjeet	8723	5000	24

Ⓐ Select few value :- select name, Age from details

Holly	19
Rohan	20
John	35
Ranjeet	24

conditions

Ⓐ cond :- select name from details where age > 24;

name
John

Select Age from details where age < 35 && age > 20;

Age
24

Select age from details order by ~~[illegible]~~ Age;

Age
19
20
24
35

select name, age from details order by name, age

name	age
Golu	19
John	35
Rohan	20
Ronjeet	24

- Select id from details where name="holu";

id
10

- select sum(age) from details;

sum(Age)
98

→ Select max(age) from details;

max(age)
35

Students **Joining** . courses

name	course-id
Golu	101
Rohan	102
Mukesh	null.
Ravi	103

id	course name
101	CS
102	math
103	Phy

(i) inner join

```
Select students.name, courses.coursename
from students INNER JOIN courses on
students.course_id = courses.id;
```

(ii) left join

```
Select students.name, courses.course_name from students
LEFT JOIN courses
ON students.course_id = courses.id;
```

(iii)

use union/Intersection

```
create database sales;
 use sales;

create table 2015sales( pid int, pname varchar(20),
                              cost float);

create table 2016sales( pid int, pname varchar(20, cost flo

insert into 2015sales values (1, "shampoo", 346.8),
                   (6, "soap", 234.6), (7, "biscuit", 441.3)
                   ( 8, "lolipop", 783.3);
          select * from 2015sales;
```

pid	pname	cost
1	shampoo	346.8
6	soap	234.6
7	biscuit	441.3
8	lolipop	783.3

```
insert into 2016sales values (1, "shampoo", 346.8),
      (6, "soap", 234.6), (7, "biscuit", 446.0),
                (8, "lolipop", 7898.3), (9, "toothpaste", 517.
       , (10, "choclate", 1234.5), (10, "choclate", 123.5);
```

pid	pname	cost
1	shampoo	346.8
6	soap	234.6
7	biscuit	446.0
8	lollipop	7898.3
9	toothpaste	517.2
10	choclate	1234.5

→ select * from 2015sales union all select * from 2016sales;

pid	pname	cost
1	shampoo	346.8
6	soap	234.6
7	biscuit	446.3
8	lolipop	783.3
1	shampoo	341.8
6	soap	234.6
7	biscuit	446.0
8	lolipop	4798.3
9	toothpaste	567.2
10	choclate	1234.5

union all
it give o/p without removing duplicate

→ select * from 2015sales union select * from 2016sales;

pid	pname	cost
1	shampoo	346.8
6	soap	234.6
7	biscuit	446
8	lolipop	7803.3
9	toothpaste	567.2
10	choclate	1234.9

use foregin key

foregin key is a field in a database table that creates a relationship between 2 tables. it refrer to primary key in another table, ensuring that the data stored in one table corrosponds to valid entire in anothers.

__Referential Integrity__ :- when inserting a new record into the table containning the foregin key, the value for the foregin key must exit in the refrenced table.

DDL

```
ex— create table Customers (customer_id int Primary key,
                 name varchar(50), email varchar(50) );

create table Orders( order_id int primary key ,
          orderdate Date , amount Decimal(10,2),
      customer_id int,

    foregin key (customer_id) refrences Customers(custm
);
```

DML

```
Insert into Customers values(1, 'Golu', 'Golu@example.in'),
      (2, 'kumar', 'kumar@example.in'), (3, 'Abhijeet', 'abhijeet
                                                @example.in');

insert into Orders values(101, '2024-11-07', 250.00, 1),
                         (102, "  08', 150.00, 2),
                         (103, '  "  09', 325.00, 1),
                         (104, '  "  10', 475.00, 3);
```

referenced table

Same col-name

referencial To.

customer-id	name	email
1	Golu	Golu@example.in
2	kumar	kumar@example.in
3	Abhijeet	Abhijeet@example.in

customer table

order-id	order-date	amount	customerid
101	2024-11-07	250.00	1
102	" -08	150.00	2
103	" -09	325.00	1
104	" -10	475.00	3

Ordertable

→ customerid(1,2,3) ——— same ——— customerid(1,2,1,3)

when we will add in order table

Insert into Orders values (105, '2024-11-11', 400.00, 4);
→ it give an error:→ customerid 4 doesnot exist in Customers (referenced table) then we can not insert

resolve error: 1. first we have to insert customerid 4 in table Customer (referenced table).
2. After inser orders table.

1. insert into Customers values (4, 'New customer', 'newcustomer@example.in');

2. insert into orders

Group by, order by, Having

Group by: it is used to Group raws by one or
columns and allow aggregate functio

Orader by: sort the result set by one or
(ASC) for adending dec for decending.

Having: filter the result set after aggregat
allowind you to set condition on

ex - create table sales(product-id, quantity, sales
values(1, 50, '2024-10-01'),
(1, 60, ' " 02'),
(2, 40, ' " 01'),
(2, 70, ' " 03'),
(3, 150, ' " 02'),
(3, 20, ' " 04');

Product-id	Quantity	salesdate
1	50	2024-10-01
1	60	-02
2	40	-01
2	70	-03
3	150	-02
3	20	-04

result-table

product-id
3
1
2

Apply →

```
→ Select product id, sum(quantity) as total sales
  from sales Group by product id
  Having sum(quantity) >100
  order by total_sales DESC;
```

Page

Trigger

Triger is a special type of stored procedure that automatically executes or "fire" when a specified event occurs in the data base:

This event typically involves change to the data. such as insert, update, or delete operations.

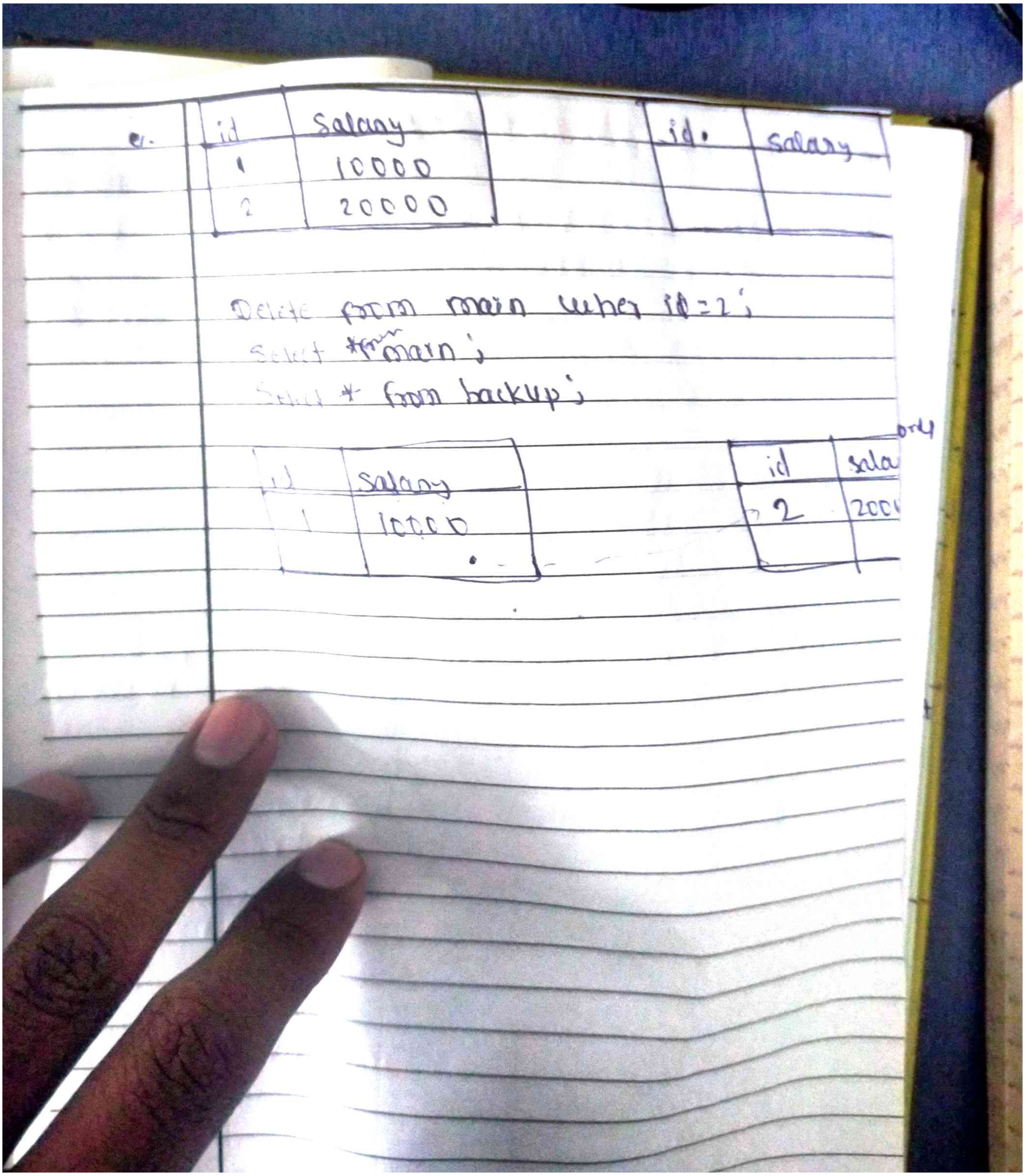
Type → (i) Before trigger
(ii) After trigger
(iii) instead of trigger.

ex — we will create a trigger when any row delete in main table then that row will be stored in backup-table ?

```
Create table main( id int, salary int);
insert into main values( 1,10000), (2, 20000);

Create table backup ( id int, salary int);

Delimiter $$
    Create Trigger before_delete_main After delete on
    main
     For each row
      begin
        insert into backup (id , salary)
        values( old.id, OLD.salary);
       End $$
  Delimiter ;
```



a.

id	salary
1	10000
2	20000

id.	salary

Delete from main wher id=2;
Select * from main;
Select * from backup;

id	salary
1	10000

id	sala
2	200