CRUD - I

Hey everyone!

We start at 7:05 AM

Agenda

↳ CRUD operation.

- → Create ✓
- → Read ✓
  - → WHERE ✓
  - → AND, NOT & OR
  - → IN, BETWEEN
  - → LIKE
  - → IS NULL
  - → ORDER BY
  - → LIMIT
- → Update
- → Delete

What is CRUD

students.

id	name	psp	batch
1	Ujjwal	100	1
2	Ajaay	25	1
3	Gaytri	75	2

What all type of operation can you do on data of this/any table

→ Update (U)
→ Delete (D)
→ create (C)
→ Read (R)

} CRUD

↓

sakila

Sakila database overview

↳ Digital video rental store

C : Create

→ put new data in a table

Statement → INSERT.

Q create a new record in film table.

→ create data → table name

INSERT into {table_name} (
  {column_names} → column
) VALUES
 (______ .),
 ( ),
 ( );

R: Read

Statement → select.

select * → fetch me all the columns.
  from {table_name};

select * from film;
select title, release-year from film.

⇒ Let see how these queries might work internally select * from table_name

```
                        [ - - - - - ]
table_name = [{ }, { } , - - - - - - - { } ]
ans = [ ]

for row in table_name:
        ans.add(row)

print(ans)
```

select title, film_id from film

```
table_name = [ { }, { } - - - - - { } ]
ans = [ ]

for row in table_name :
        ans.add(row)

for row in ans :
    print(row[title] , row[film_id])
```

[BREAK till 08:17 AM]

Composite keys.

Students.

id name email phone_no batch-id

CK

(id) ← PK

~~(email)~~ ×

~~(phone_no)~~ ×

(email, phone no,) ⟹ composite key

(id, name, batch-id)

(name, batch-id) =

DISTINCT.

1. distinct should be very first thing after select
2. distinct is applied to pair/set of values.

```
select distinct rating, release_year from film.
```

```
table_name = [ { }, { }, - - - - { } ]
    ans    = [   ]

for each row in table_name :
        ans.add( row)

filtered_ans = [ ]
  for each row in ans :
      filtered_ans.add(row[rating], row[release
                                          year])

print( set (filtered_ans))
```

↓

distinct → removes duplicates

=> it gets applied after the entire data set is formed.

```
select * from film;
select title from film;
```

```
select 1;
select "Hello world";
```

Q Print title of every movie and alongside print (Hello) ?

Jab tak hai Jaan	hello
Tees Mar khan	hello
K3G	hello
pushpa	hello

select title, 'Hello' from film.

value in table is in min

Q Print name, length (in hours)

```
select title, length/60
        from film;
```

Table → film =

Duplicate table → film_copy ;)

```
insert into film_copy
        ( colm names )
values
 (                    ),
```

( ),
( );

non null

id name rating

insert into film_copy (
coln names) 2
= id , name
VALUES
select coln_names (id , name)
from film

insert into film_copy
values
select * from film; =

insert into film_copy (title)
values
description
select ~~date~~ from film

WHERE CLAUSE

select * from film;

Where statement.

title , film_id

Q select ~~*~~ from film where rating = 'PG-13' ;

```
table-name - [ { }, { } - - - - - { } ]
ans [ ] → intermediary table

for row in table_name :
    ① ans. add (row)  - - - - → if rating == 'PG-13'
                                removing redundant
                                data ASAP.

for row in ans:
    ② print ( row[title], row[film_id] )
```

WHERE

⇒ allows to filter row based on condition.

```
table = [                    ]
ans = [                      ]

for each row in table :           // from
        if row matches condition:   // where
                ans.add(row)

for each row in ans :          // print
        print(row)
```