Class starts at 9:05 PM.

Agenda:

Read:

- SELECT
- WHERE
- AND / OR / NOT
- IN
- BETWEEN
- LIKE
- IS NULL

Read:

SELECT Statement

Select * from {tableName}

→ Give me complete data of the table

→ For every row, give me all the columns.

How does MySQL work internally?

It is internally implemented in C++.

```
Select * from film
```

table_name = [{} {}, {}, {}, {} ..... {}]

row

ans = [ ]

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

print(ans)
```

Q. Select title, description from film

table_name = [{} {} {} {} {} ..... {}]

row

ans = [ ]

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

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

→ Distinct should be the very first thing after SELECT!

→ Distinct works on the combination of columns.
Unique values for the comb^n of columns.

Working of Distinct internally.

table_name = [{} {}, {} {}, {} {} .... {}]

row

ans = [ ]

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

final_ans = [ ]

```
for row in ans :
    final_ans . add (row [title],
                     row [desc])

print (set(final_ans))
```

distinct.

→ Select can be used as a print statement.

```
select 1;

select 'Hello World';
```

Q Print the title of every film, and along with that print hello.

title	
Jab Tak Hai Jaan	hello
K K H H	hello
X Y Z	hello

```
select title, 'hello'
from film;
```

Q what if I want to add, subtract do something with the columns.

we have a column called length,
→ length of the movie in minutes.

180 → 3
140 → 2. ...

Select title, length/60 from film.

Q given a table → Film

I want to copy all the contents into a new table film_copy

Insert Into film_copy (col^m names)
Select col^m names from film

same syntax

Insert into film_copy (title)
select title from film.

Insert into film_copy (title)
select date from film.

→ Datatype for the columns should be same!

Insert into film_copy (title)
select desc from film.

Break till 10:26 PM

where

allows you to filter rows based on a condition.

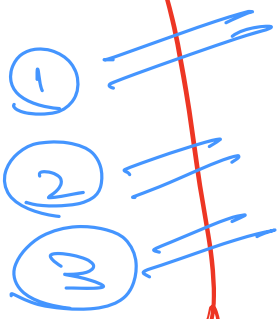
```
1 table_name = [ {} {} {} {} {} ... {} ]
                 row

2 ans = [ ]

3 for row in table_name :
4         if row satisfies condition :
              ans.add [row]
5 for row in ans :
6     print ( row[title], row[desc] );
```

handle condition/
filter rows as soon as possible.

```
Select  title.
from    film
where   rating = = 'PG-13'.
```



table_name = [{ }, { }, { }, { }, { } .... { }]

row

ans = [ ]

```
for row in table_name :          // FROM
    if row satisfies condition:
        ans.add(row)             // WHERE

for row in ans :
    print(row[title], row[desc]);
                                 // SELECT
```

Order of operation is
FROM, WHERE ..., Distinct,
SELECT.

AND / OR / NOT

where clause is analogous to the If condition.

&&	AND
||	OR
!	NOT

```
select * from film
where rating = 'PG-13'
  AND year = 2006
```

```
Select * from film
where rating = 'PG-13'
   OR year = 2006
```

Q Give me title of film ~~that have~~ are neither 'PG-13' Nor released in 2006.


```
Select * from film
where ratings != 'PG-13'
  AND year != 2006
```

```
Select * from film
where Not rating = 'PG-13'
  AND Not year = 2006
```

$1 + 2 - 3 / 6$ → BODMAS

$(1+2) - (3/6)$

$3 - 0.5 = 2.5$

Order of precedence:

( ), NOT, AND, OR

---

1. Select * from film
   where Not (rating == PG-13 AND
   year == 2006)

→ Select * from film
where rating != PG-13 OR
year != 2006

Select * from film
where rating != PG-13 AND
year != 2006

(2)

IN Operator.

Q get all the students of batch_id in 1, 3, 5, 7.

```
Select * from students
where batch_id = 1
   OR batch_id = 3
   OR batch_id = 5
   :
```

Use In operator, when we want a check on the

multiple values of the same col^y,

```
Select * from Student where
batch_id IN(1, 3, 5, 7);
```

Between

```
select * from Students
where (psp >= 50 and psp <= 70)
OR (psp >= 80 and psp <= 85)
```

(>= x AND <= y)

Between x and y

Both x and y are inclusive

```
select *
from Students
where (p&p between 50 AND 70)
   OR (p&p between 80 AND 85)
```

x = 50 ✓

x = 49 ✗

y = 70 ✓

y = 71 ✗