

CRUD - 1

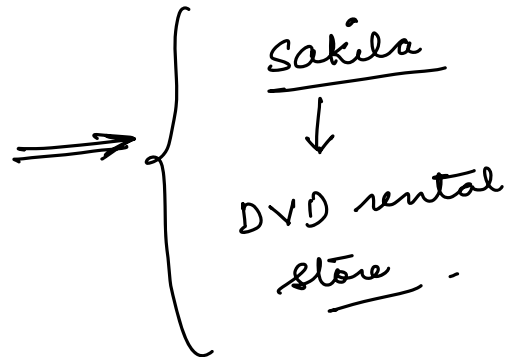
TABLE OF CONTENTS

1. CRUD Operations
2. Create
3. Read
 - 3.1 Distinct
 - 3.2 Where
4. Order By
5. AND, OR, NOT
6. IN Operator



Notes

C : create]
R : read
U : update
D : delete .





Create

- Create database ✓
- Create table ✓
- Adding new entries

< / > Syntax

Insert into table_name (col1, col2)

values (values_1, values_2);

*exact
same order -*



- Column names are optional. Let's see this scenario as well :

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_update

↑ ↑ ↑ ↑ →

INSERT INTO film
VALUES (default, 'The Dark Knight', 'Batman fights the Joker', 2008, 1, NULL, 3, 4.99, 152, 19.99, 'PG-13', 'Trailers', default);

↓ ↑ ↑ ↑ . —

create table — (

{

—

—

—

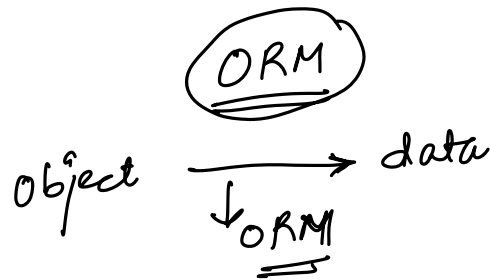
—

);



Drawbacks


1. This is not a good practice, as it makes the query prone to errors. So always specify the column names.
2. This makes writing queries tedious, as while writing query you have to keep a track of what column was where. And even a small miss can lead to a big error.
3. If you don't specify column names, then you have to specify values for all the columns, including film_id, original_language_id and last_update, which we may want to keep NULL.





Read ✓



- Print 
- You may print constant data or data from other tables.
- Most used query


< / > Syntax

select 50;

- Printing constant value :

SELECT constant_value;

- Printing data of whole table :

SELECT  *everything*
FROM table;

Students

id	first_name	last_name	psp
1	Virat	Kohli	80
2	Rahul	KL	75
3	Rohit	Sharma	95
4	Rahul	KL	80

*select * from students;*



- Printing some columns from a table :

```
SELECT col1, col4 as temp  
FROM table;
```

alias

select id, psp as problem from students;

Students

id	first_name	last_name	psp
1	Virat	Kohli	80
2	Rahul	KL	75
3	Rohit	Sharma	95
4	Rahul	KL	80

</> Pseudo-code

C++,
java

```
table_name : [ [], [], [], [] ]
```

```
for row in table_name :
```

```
ans.add( row )
```

ans.add(row[id],
row[psp])

```
for row in ans :
```

```
print( row )
```



Distinct (Gives output of all unique values)

- Distinct pair of names :

Students

id	psp	name
1	80	Virat
2	75	Rahul
3	95	Rohit
4	80	Rahul

Distinct Names

Virat
Rahul
Rohit

- Distinct pair of first_name and last_name :

Students

id	first_name	last_name
1	Virat	Kohli
2	Rahul	KL
3	Rohit	Sharma
4	Rahul	KL

Distinct Names

Virat Kohli
Rahul KL
Rohit Sharma



- It can be applied on pair as well.

or more no of columns as well

- Filters out duplicates.

```
SELECT distinct release_year, rating FROM film;
```

```
film : [ [], [], [], [] ]
```

```
for row in film :
```

```
ans.add( row )
```

```
for row in ans :
```

```
filtered_ans.add( row[release_year], row[rating] )
```

```
unique_ans = set(filtered_ans)
```

```
print (unique_ans)
```

distinct



Inserting data from other table using SELECT

- We want to create a copy of students table where the data includes their id, name and last_name.

Students					Students Copy		
id	name	last_name	psp	attendance	id	name	last_name
1	Rohit	Sharma	80	85			
2	Virat	Kohli	75	85			
3	Shubhman	Gill	95	95			
4	Rahul	KL	92	85			
5	Rishabh	Pant	80	88			

'Should I add all the data one by one?'



'No ! We have a solution for that'



< / > Syntax

Code to insert data from existing table :

```
insert into students_copy(first_name, last_name)
```

→ Where to insert

```
SELECT first_name, last_name
```

```
FROM students;
```



→ What to insert



Where (Similar to if condition)

Question : Get all the movies with ' PG-13 ' ratings.

Note : Assume if it is an array, how will you filter the data ? using If condition.

- We have where condition in SQL.

Films

film_id	title	release_year	language	rating
1	KGF	2018	Kannada	PG
2	Kung Fu Panda	2006	English	G
3	Janghu 007	1947	Bhojpuri	NC-17
4	Kantara	2022	Kannada	PG-13

select * from film
where rating = 'PG-13'

< / > Syntax

SELECT *

FROM film where rating = 'PG-13' ;



< / > Pseudo-code

table_name : [[], [], [], []]

for row in table_name :

if row.matches(condition in where clause)

ans.add(row)

for row in ans :

print(row)



AND, OR, NOT

- These are same as logical operators we have seen so far.
- AND = **AND**
- OR = **OR**
- NOT = **<>** , **!=** , **NOT**



Order By



- Order by clause allows to return value in a sorted order.
- By default the data is ordered in ascending order.

Question : Order the data in descending order according to rental_duration.

< / > *Syntax*

```
SELECT * FROM film ORDER BY rental_duration DESC;
```

- In case of tie, PK is always a tie-breaker.

Primary Key

film_id	title	rental_duration
1	KGF	1.5 hrs
2	Kung Fu Panda	2.2 hrs
3	Janghu 007	3.5 hrs
4	Kantara	2.2 hrs

Ascending
Order →

film_id	title	rental_duration
1	KGF	1.5 hrs
2	Kung Fu Panda	2.2 hrs
4	Kantara	2.2 hrs
3	Janghu 007	3.5 hrs



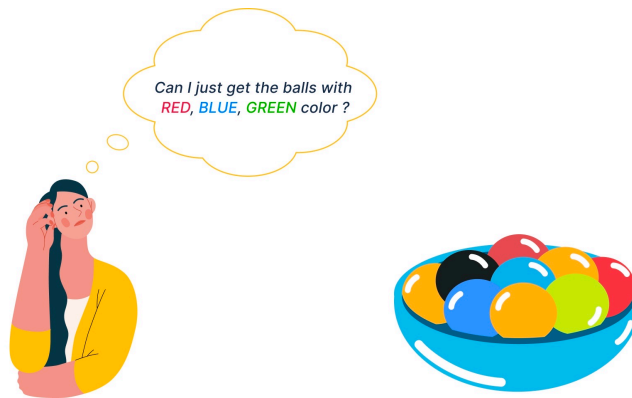
- Order By two column

< / > *Syntax*

```
SELECT *  
FROM film  
ORDER BY title, release_year ;
```



IN Operator



Question : Give data of all the students with batch_id 5 , 2 , 7 , 1 , 3.

< / > *Syntax*

```
SELECT *  
FROM students  
WHERE batch_id = 5  
       or batch_id = 2  
       or batch_id = 7  
       or batch_id = 1  
       or batch_id = 3
```

- Here we can use **IN operator** instead of multiple **OR operator**.

< / > Syntax

SELECT *

FROM table

WHERE value in (list of values)