SQLBolt.

code with harry handbook.

① SQL Bolt.
② ~~fff~~ sql-practice.
③ leetcode
④ lets-code

where clause:

→ select column. from table_name
where condition AND/OR another_condition.

Operator	Condition
=, !=, <, <=, >, >=	standard numerical operators.
between, and (inclusive)	number is within range of 2 values. Ex: col_name between 1.5 and 10.5
Not between, and	number is not within range of 2 values. Ex: col_name not between 1 and 10.
in	number exists in list Ex: col_name in (2,4,6)
not in	number does not exist in a list
=	case sensitive (exact string comparision).
!= or <>	
like	case insensitive exact string comparision
not like	" " " " inequality
%	used anywhere in a string to match a sequence of zero or more characters (only with like or not like) Ex: col_name like "%AT%" (matches "AT", "ATTIC", "CAT", "BATS")
_	used anywhere in a string to match a single character. (only with like or not like) Ex: col_name LIKE "AN_" (matches "AND", but not "AN")

Filtering & sorting Query results.

→ DISTINCT :- removes duplicate rows
→ GROUP BY
→ ORDER BY - sort.
→ LIMIT
→ OFFSET

shows only a part of results.

① DISTINCT :- director (or) year

select distinct year from movies.

gives only unique director names or years

② ORDER BY.

select * from movies order by year DESC

③ LIMIT and OFFSET

Syntax :- SELECT col1, col2 from table order by column LIMIT no.of.rows OFFSET start_position;

Ex: select title from movies order by year desc
limit 5 offset 10;
skips first 10

gives 5 movies starting from 11th movies.

Inner Join :

→ join is same as inner join
→ It will return matched rows from both tables.

Syntax: SELECT col_names ,, from my_table join another_table on mytable.id = another_table.otherid where conditions. order ...

Outer Join
- Left Outer Join or Left Join
- Right Outer Join or Right Join
- Full Outer Join or Full Join

→ Same syntax as join but in place of join we will write right join, left join, full join
i.e., select col_names... from mytable left join/ right join / full join another on mytable.id = another.other id.

→ In left join :- includes all rows of left table & only matching rows of right table.

→ full join : include both tables.

NULLS.

→ In SQL, null means "no value" or "unknown"
→ It is not same as 0, false or an empty string
→ NULL is completely empty - the data missing
→ fns like avg(), count() & conditions like '=' dont work normally with null.

Syntax to check NULL :-
where col is NULL
" " is not NULL

Aggregate fns.

Syntax: select agg-func(col_name) as agg_desc,... from my_table where condn ;

count(*), count(column), sum(column), avg(col)
min(col) max(col)

group by:-

→ group by is used to group rows that have the same value in one or more ~~rows~~ cols.

Syntax: select col, agg_fn(other_col) from table where con. group by col

→ grouping happens after filtering with where.

ex: select rating, avg(sales) as avg_sales from movies group by rating

→ we can also group by 2 cols.

→ So, even if u write group by before where, SQL still apply where first, then do grouping. So to apply filters on group we use having clause.

Having

Ex. select rating, count(*) as movie_count from movies group by rating having count(*) > 2;

=

Query order of execution.

1) from and joins.
2) where
3) groupby
4) having
5) select
6) distinct
7) orderby
8) limit / offset.

insert into table

→ insert into movies (title, year) values ('Toy', 2012);
→ remaining null or default values
→ insert into movies values ('Toy', 2012, 8.7, 90);
→ insert into movies (title, year) values ('Toy');
↓
error.

update table rows

Syntax: update table_name set col1 = value1,
col2 = value2 where condition;

→ if u dont write where condition all rows are effected

delete:

once gone, its gone.
unless u have backup or a transaction,
↓
works only before u commit; once commited u can't roll back.

Syntax: delete from mytable where condition;

Create a table.

Table datatypes: integer, int, boolean, float, double, real, character (num_chars), varchar(num_chars), text, data, datetime, blob (binary data)

Constraints: primary key, autoincrement, unique, notnull, check (expression), foreign key, default

Ex: create table table_name (id int primary key autoincrement, name varchar);

Alter Table:

↳ change the design.

→ rename a col - alter table movies rename old to new
→ add a new col - alter table movies add column dir text
→ change col datatype - alter table movies alter column year type year
→ drop (remove) col - alter table movies drop column rating

Drop table:

drop table table_name.

=0=

SubQuery.

It is full select statement nested inside another sql query

→ subquery must be enclosed in paranthesis.
→ some implementation don't allow subqueries to use limit or offset.

① Subquery in a where clause. (Single value).

Ex: select * from sales_associates where
salary > (select avg(revenue) from
sales_associates);

↳ inner query gives avg then outer query filters it

② Existence tests :- (Subquery using IN (list of values))

this type of subquery returns a list, and you check if a value exists in that list.

Ex; select name from customers where id
in (select from orders where order_date >=
'2024-01-01')

returns name of all customers who placed orders in 2024

③ Correlated Subquery:

Here, a value from outer query inside inner query. that means the inner subquery runs once per row in the outer query.

Ex: select * from students s where marks > (select avg(marks) from students where class = s.class);

Table:

name	class	marks
Alice	10A	80
Bob	10A	60
Charlie	10B	70
David	10B	50.

O/P:

name	class	mark
Alice	10A	80
charli	10B	70

$\frac{80+60}{2}$ = 70 - 10A 10B - 60.

Alice 80>70 ✓ char 70>60 ✓

Bob 60>70 ✗ David 50>60 ✗

=

Another way

select customer_id, total_spent from (select customer_id, sum(total_amount) as total_spent from orders group by customer_id) as customer_totals where total_spent > 600.

Set Operators.

Syntax: select col, anr_col from mytable
union / union all / intersect / except
(select other_col from another_table
order by col desc limit n;

→ first executes select stmts and the order them or apply limit on output that is generated after applying set operations.