**LIKE**

name LIKE 'Y%' -- start with Y

'%x%' -- contains x

'%land' -- ends with land

'%a%a%a%' -- contains 3 or more a

'\_t%' -- has t as the 2nd letter

'%o\_\_o%' -- two 'o' char separated by two other char e.g. Lesotho and Moldova

'\_\_\_%e' -- ending with “e” that consists of at least four characters

round(value, 2) -- round to 2dp

round(value, -3) -- round to nearest 1000

**Row construct**

select \*

from Lectures

where day > 3

or ((day = 3) and (hour > 11));

EQUIVALENT TO

select \*

from Lectures

where row(day,hour) > row(3,11);

-- For each lecturer, find the time of his/her earliest lecture during the week.

select pname, day, hour from Lectures L

where row(day, hour) <= all (

select day, hour

from Lectures L2

where L2.pname = L.pname);

-- order by multiple columns

SELECT id, first\_name, last\_name, salary FROM employee

ORDER BY salary DESC, last\_name;

-- decreasing salary followed by increasing last name

2ND HIGHEST

SELECT MAX(Salary)

FROM Employee

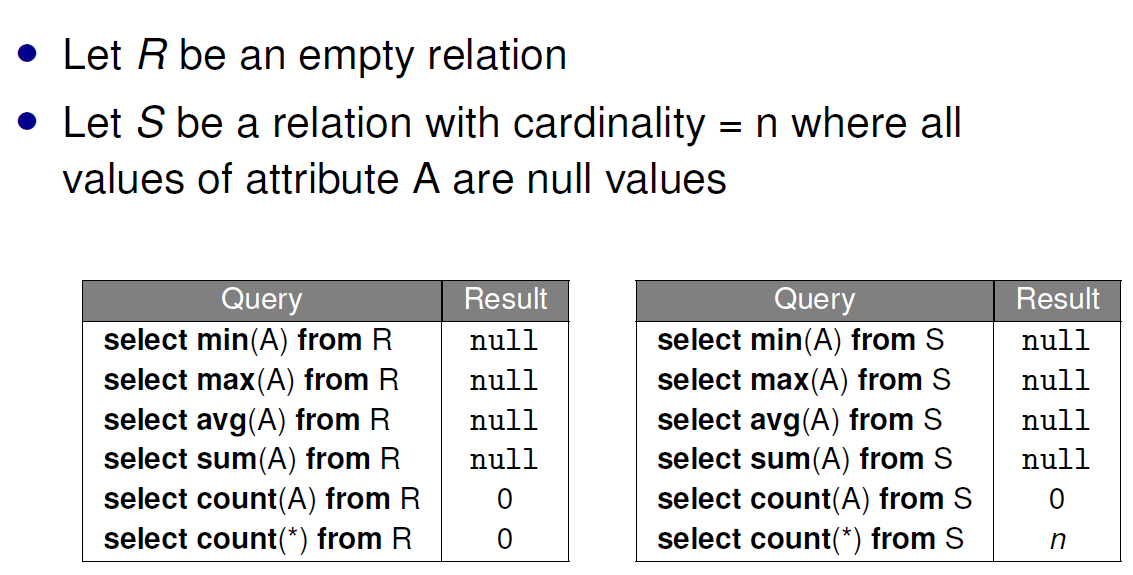
WHERE Salary < (SELECT MAX(Salary) FROM Employee)

If the set being evaluated by the SQL NOT IN condition contains any values that are null, then the outer query here will return an empty set, e.g.

SELECT \* FROM runners

WHERE id NOT IN (SELECT winner\_id FROM races)

-- if winner\_id contains null, whole query will be empty



**Conditional Expression**

select name, case

when marks >= 70 then ’A’

when marks >= 60 then ’B’

when marks >= 50 then ’C’

else ’D’

end as grade

from Scores;

**Coalesce**

Returns the first non-null value in its arguments, if all null, returns null value

coalesce(col, [value]) returns value if col is null

**NULLIF**

nullif(v1, v2) returns null if v1 == v2, else return v1

**Mock Midterms Q3**

create or replace view v3 (sid) as

with Weeks as (

select distinct week

from Presenters

)

select S.sid

from Students S

where (select max(week) from Weeks) >= 3

and exists (

select 1

from Weeks W

where W.week + 2 <= (select max(week) from Weeks)

and not exists (

select week

from Weeks W2

where W2.week >= W.week

and W2.week <= W.week + 2

intersect

select week

from Presenters P

where P.sid = S.sid

)

);

-- update with case

update Salary

set sex = case sex when 'm' then 'f' else 'm' end;

Rank

- The scores should be ranked from the highest to the lowest.

- If there is a tie between two scores, both should have the same ranking.

- After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

select score, (select count(distinct(score)) from scores WHERE score >= s.score ) as 'rank'

from scores s

order by score desc;

**Assignment Q7 (num of students admitted x num of courses offered x total enrolment)**

select d.did, d.faculty, (

select count(\*)

from Students s

where s.did = d.did and year = 2021

) as num\_admitted, (

select count(\*)

from Offerings natural join Courses

where did = d.did and year = 2021

) as num\_offering, (

select count(\*)

from transcripts t

where t.cid in (select cid from courses c where c.did = d.did)

and t.year = 2021

) as total\_enrollment

from departments d;

select did, faculty, coalesce(num\_admitted,0), coalesce(num\_offering,0), coalesce(total\_enrollment,0)

from Departments natural left join

(

select did, count(sid) as num\_admitted

from Students

where year = 2021

group by did

) as X

natural left join (

select did, count( \* ) as num\_offering

from Courses natural join Offerings

where year = 2021

group by did

) as Y

natural left join (

select did, count(\*) as total\_enrollment

from Courses natural join Offerings natural join Transcripts

where year = 2021

group by did

) as Z;

**Assignment Q9 (top scorer for all courses enrolled)**

with top\_scores as (

select cid, year, semester, max(marks) as highest

from transcripts

group by cid, year, semester

)

select sid, year, semester

from Transcripts natural join top\_scores

group by sid, year, semester

having sum(case when marks = highest then 0 else 1 end) = 0;

select distinct sid, year, semester

from Transcripts T

where not exists (

select 1

from Offerings O

where year = T.year

and semester = T.semester

and exists (

select 1

from Transcripts T1, Transcripts T2

where T1.year = T2.year

and T1.semester = T2.semester

and T1.cid = T2.cid

and T1.year = T.year

and T1.semester = T.semester

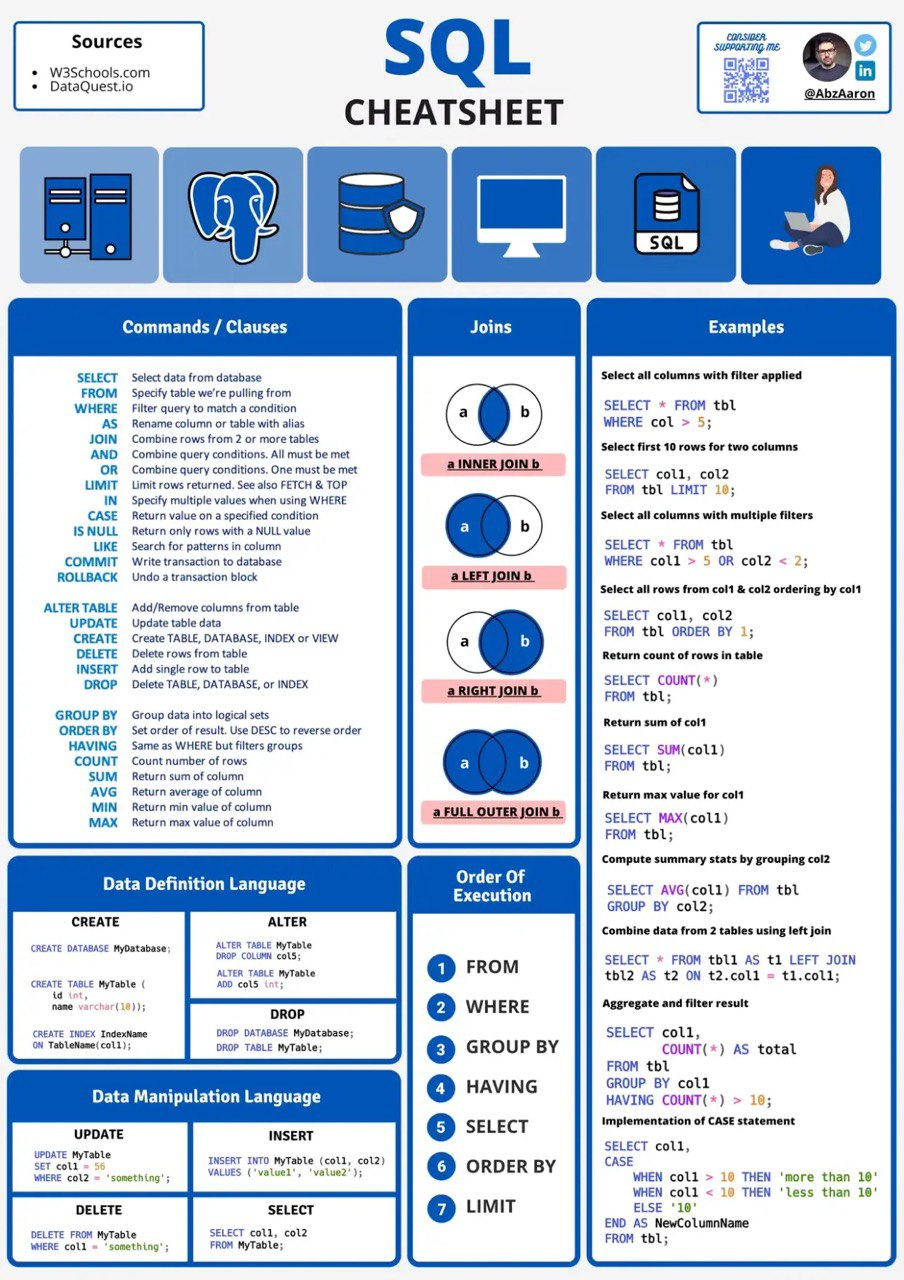
and T1.cid = O.cid

and T1.sid = T.sid

and T1.marks < T2.marks

)

);

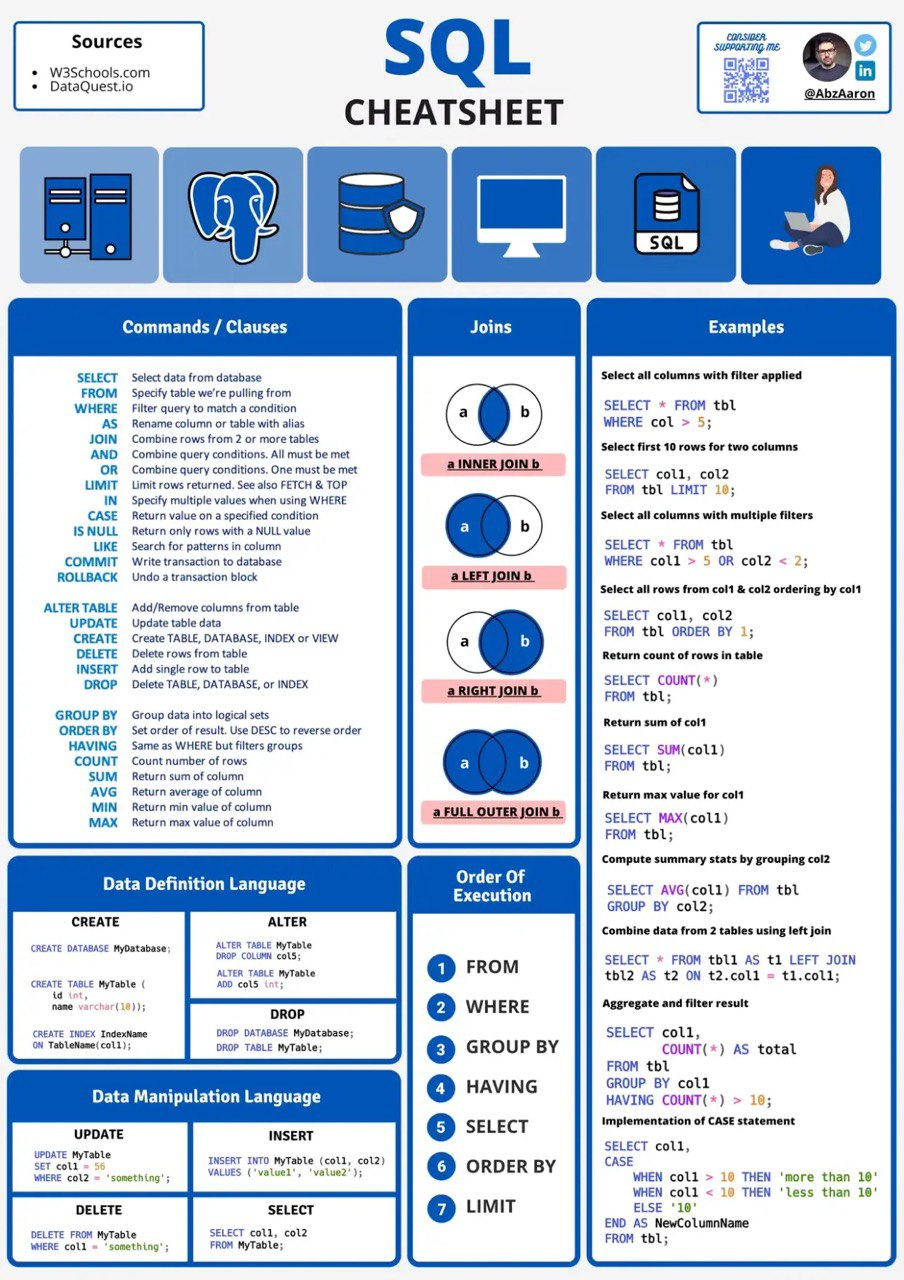


COUNT(\*) / CAST(c.population AS DECIMAL)

-- division by integer gives math floor

select ’Price of ’ || pizza || ’ is ’ || round(price / 1.3) || ’ USD’ as menu

-- CONCAT



AY21/22 Semester 2

Ang Koon Hwee, Wu Xiao Yun