GATE

CRASH COURSE

DS & AI

Database Management System

SQL (Part-2)

Lecture No. 04

By – Vishal Rawtiya Sir

Topics *to be covered*

1. Nested Query (Sub-query)
2. Correlated sub-query
3. AS clause & WITH AS clause

Topic : SQL clauses

ORDER BY:- This clause is used to sort the result in ascending or descending order based on values of attribute specified with ORDER BY clause.

By default order is ascending order.

Student

Sid	Sname	Marks	Branch
S1	A	40	CS
S2	A	20	IT
S3	B	60	CS
S4	A	60	EC
S5	C	40	IT
S6	C	NULL	EC

Query:- From the above Student table retrieve Sids of students who scored maximum Marks.

```
Select Sid
From Student
Where (Marks = Max(Marks))
```

invalid query

We can not use aggregate function directly within the Where clause

Topic : Nested queries

Sub-query Concept :-

* Retrieve Sids of all students from Student table who scored maximum marks.

outer query (Main query)

```
Select Sid
From Student
Where (Marks = (Select Max(Marks) From Student))
```

o/p : 60

inner query (Sub-query (or) Nested query)

o/p =

Sid
S3
S4

Nested Query (Sub-query)

Independent Nested query (Independent Sub-query)

- If inner query can be executed independently, then it is called independent nested query

Correlated Nested query (Correlated Sub-query)

* When execution of inner query requires the value of an attribute from the relation specified in outer query, then it is called Correlated Nested query

Nested Query (Sub-query)

Independent Nested query (Independent Sub-query)

eg:

```
Select S.Sid
From Student S
Where (S.Marks = (Select Max(Marks)
                  From Student))
```

Student renamed in to S

Correlated Nested query (Correlated Sub-query)

eg:

```
Select R.A
From R
Where operator( Select *
                From S
                Where (S.B = R.C))
```

∴ Correlated Sub-query

To evaluate where Condn we need attribute 'C' of relⁿ R

* "Order of Execution w.r.t. independent Nested query".

① Inner query will be executed first and it will produce its output.

and then ② Outer query will execute, it will use the o/p produced by inner query for its execution

Topic : Correlated nested query

```
Select *
From R
Where operator ( Select "List of attributes"
                 From S
                 Where (S.A = R.B) )
```

Outer query

inner query

operator → This operators Will return True or False

- If it returns True, then Where Cond^n is True.
- If it returns false, then Where Cond^n is False.

R.B → attribut from outer rel^n

∴ Correlated query.

(Order of Execution)

V.V.V.V IMP

Topic : Correlated nested query

S

A	-	-	-	-
✓				
✓				

⑥ — Select *

① — From R

R

-	B	-	-	-
✓	✓			
✓				
|	|	|		\

⑤ — Where operator (Select * — ④

From S — ②

Where (S.A = R.B) — ③

Topic : Operators

- Best used operators with independent query are IN, ANY or ALL.
- Best used operator with Correlated query is EXISTS.

IN, ANY, ALL and EXISTS Can be used with any type of Subquery i.e; Independent or Correlated

IN operator :→

IN operator is used to check whether the concerned tuple is present in the set of tuples produce by the inner query or not.

Complement of 'IN' is 'NOT IN'

→ Check whether

X IN { 2, 4, 5, 7, 8 } or not

it is the Concerned tuple

Set of tuples Produced by inner query

if X = 5, then 'IN' operator will return true { as '5' is Present in the set of tuples }

if X = 6, then 'IN' operator will return false { as '6' is not Present in the set of tuples }

- Check whether

X IN {(a,1), (b,1), (b,2), (c,3), (d,3)} or not

(2, b)

if X = (b, 2), then IN returns true

if X = (c, 1), then IN returns false.

Note:- If set of tuples is Empty, i.e. when output Produced by inner query is Empty, then "IN" operator will always return false, and hence 'NOT IN' will always return true.

Consider the following relations.

- Supplier (Sid, Sname, Rating)
- Parts (Pid, Color)
- Catalog (Sid, Pid)

R.A: $\pi_{C.sid}\left(\sigma_{\substack{C.Pid = P.Pid \\ \wedge \\ P.Color = 'Red'}} (C \times P)\right)$

SQL

```
Select distinct Sid
From Catalog C, Parts P
Where (C.Pid = P.Pid AND P.color = 'Red')
```

Query:- Retriere Sids of all the suppliers who have supplied some "red" color parts

```
Select distinct Sid
  From Catalog
  Where Pid IN
        (Select Pid
           From Parts
           Where Color = 'Red')
```

⇒ o/p = {P1, P3}

Catalog (C)

Sid	Pid
S1	P1
S1	P2
S2	P2
S3	P2
S3	P3

Parts (P)

Pid	Color
P1	Red
P2	Green
P3	Red

ANY operator

operator 'ANY' is used along with other Comparison operators.

i.e; <, >, <=, >=, =, <>

Not equal

- Note: "ANY" will return true if and only if at least one tuple in the set of tuples {i.e; in the o/p of inner query} satisfy the Comparison Condition with tuple under Consideration

→ Check whether

Set of tuples Produced by inner query

it is the Concerned tuple → X

X <ANY {2, 4, 5, 7, 8, 9} or not

X=5 ✓ ✓ ✓ ✓

X=15 ✗ ✗ ✗ ✗ ✗ ✗

if X = 5, then "ANY" operator will return true

if X = 15 then "ANY" operator will return false

Note :- ① If inner query result is Empty then operator "ANY" will always return false.

② "IN" is equivalent to "= ANY".

"ALL" operator

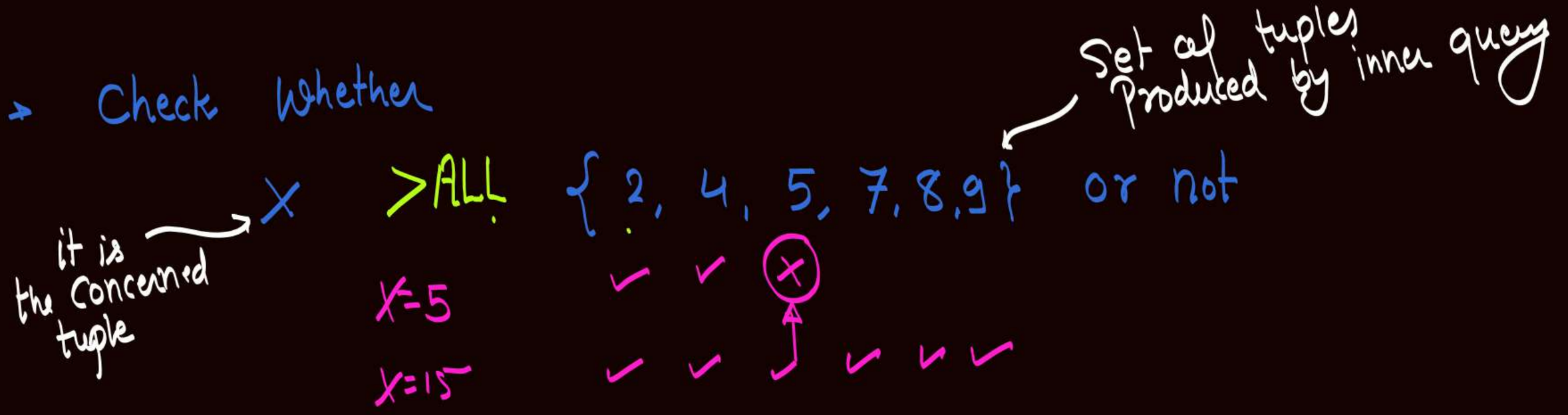
operator 'ALL' is used along with other Comparison operators.

i.e; <, >, <=, >=, =, <>

Not equal

- Note: "ALL" will return false if and only if at least one tuple in the set of tuples {i.e; in the o/p of inner query} fails the Comparison Condition with tuple' under Consideration

→ Check whether



if X = 5, then "ALL" operator will return false

if X = 15 then "ALL" operator will return true

Note :- ① If inner query result is Empty then operator "ALL" will always return "true"

② "NOT IN" is equivalent to "<> ALL".

x NOT IN {2, 3, 5, 7}

x = 4, then "NOT IN" return true

x <> ALL {2, 3, 5, 7}

x = 4, then ALL returns true

EXISTS operator

- If inner query result is not Empty, then "EXISTS" will return True
- If inner query result is Empty, then EXISTS will return False.

i.e; "EXISTS return true if and only if inner query result is not Empty"

Note: Complement of EXISTS is "NOT EXISTS"

What output will be produce by the following query On the relations Catalog Parts

#e.g.

```
SELECT C.sid
FROM Catalog C
WHERE EXISTS ( SELECT *
                FROM Parts P
                WHERE (P.Pid=C.Pid AND P.color='RED'))
```

Catalog

Sid	Pid
S1	P1
S1	P2
S3	P2
S3	P3

Parts

Pid	Color
P1	Red
P2	Green
P3	Red

```
Select C.Sid
    from Catalog C
        Where EXISTS (Select *
                      from Parts P
                      Where (C.Pid = P.Pid AND P.Color = 'Red'))
```

Correlated

Parts

Pid	Color
P_1	Red
P_2	Green
P_3	Red

Catalog

Sid	Pid
S_1	P_1
S_1	P_2
S_3	P_2
S_3	P_3

① ② ③ ④

S_1 ✓
S_1 ✗
S_3 ✗
S_3 ✓

$\Rightarrow$ o/p

Sid
S_1
S_3

Main query

```
Select C.Sid
from Catalog C
Where EXISTS ( Select *
               from Parts P
               Where (C.Pid = P.Pid AND P.Color = 'Red'))
```

① S_1 P_1

P_1 Red ✓

~~P_2 Green~~

~~P_3 Red~~

P_1 Red is selected

∴ Exists return true

⇓

Hence Where Cond^n true

Hence S_1 is selected

Correlated

Parts

Pid	Color	
P_1	Red	✓
P_2	Green	✗
P_3	Red	✗

Catalog

	Sid	Pid
①	S_1	P_1
②	S_1	P_2
③	S_3	P_2
④	S_3	P_3

② S_1 P_2

```
Select C.Sid
    from Catalog C
        Where EXISTS ( Select *
                       from Parts P
                       Where (C.Pid = P.Pid AND P.Color = 'Red'))
```

Parts

Pid	Color	
P_1	Red	✗
P_2	Green	✗
P_3	Red	✗

} Empty

inner query o/p Empty ∴ Where cond^n false ⇓

Hence Sid from 2nd tuple not selected.

Correlated

Catalog

	Sid	Pid
①	S_1	P_1
②	S_1	P_2
③	S_3	P_2
④	S_3	P_3

③ S_3 P_2

```
Select C.Sid
  from Catalog C
    Where EXISTS (Select *
                  from Parts P
                  Where (C.Pid = P.Pid AND P.Color = 'Red'))
```

Parts

Pid	Color	
P_1	Red	✓
P_2	Green	✗
P_3	Red	✓

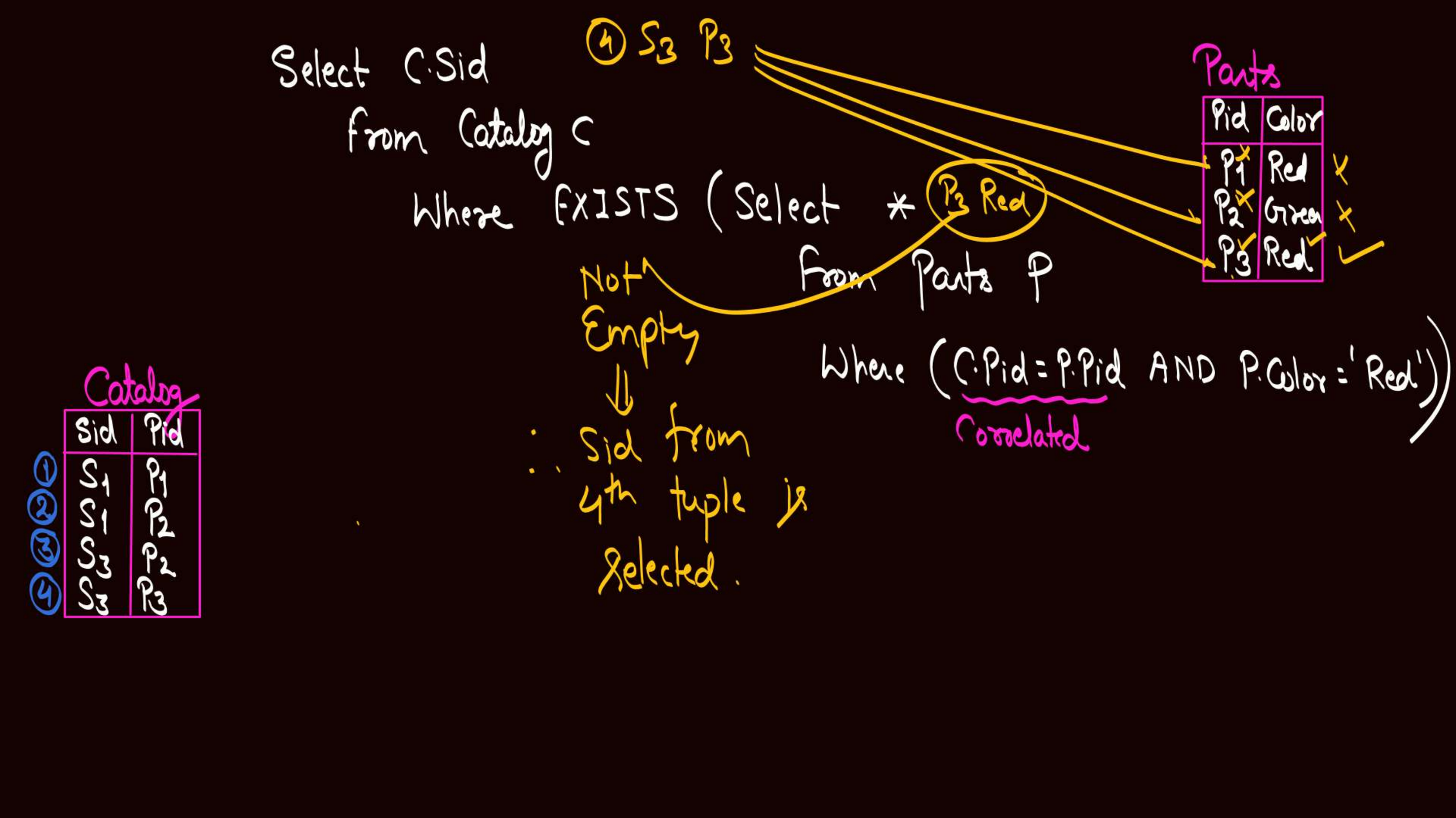
Empty

Empty
↓
∴ Sid from third tuple not selected.

Correlated

Catalog

	Sid	Pid
①	S_1	P_1
②	S_1	P_2
③	S_3	P_2
④	S_3	P_3


Select C.Sid
from Catalog C
Where EXISTS (Select * From Parts P
Where (C.Pid = P.Pid AND P.Color = 'Red'))
④ S3 P3
P3 Red
Not Empty
∴ Sid from 4th tuple is Selected.
Correlated
Parts
Pid Color
P1 Red
P2 Green
P3 Red
Catalog
Sid Pid
S1 P1
S1 P2
S3 P2
S3 P3
① ② ③ ④

#e.g.

```
SELECT C1.sid
FROM Catalog C1
WHERE NOT EXISTS ( SELECT P.Pid
                   FROM Parts P
                   WHERE NOT EXISTS (SELECT C2.Sid
                                     FROM Catalog C2
                                     WHERE(C2.Pid=P.Pid AND  C2.Sid=C1.Sid)))
```

C

S_1	P_1
S_1	P_2
S_2	P_2

P

P_1
P_2

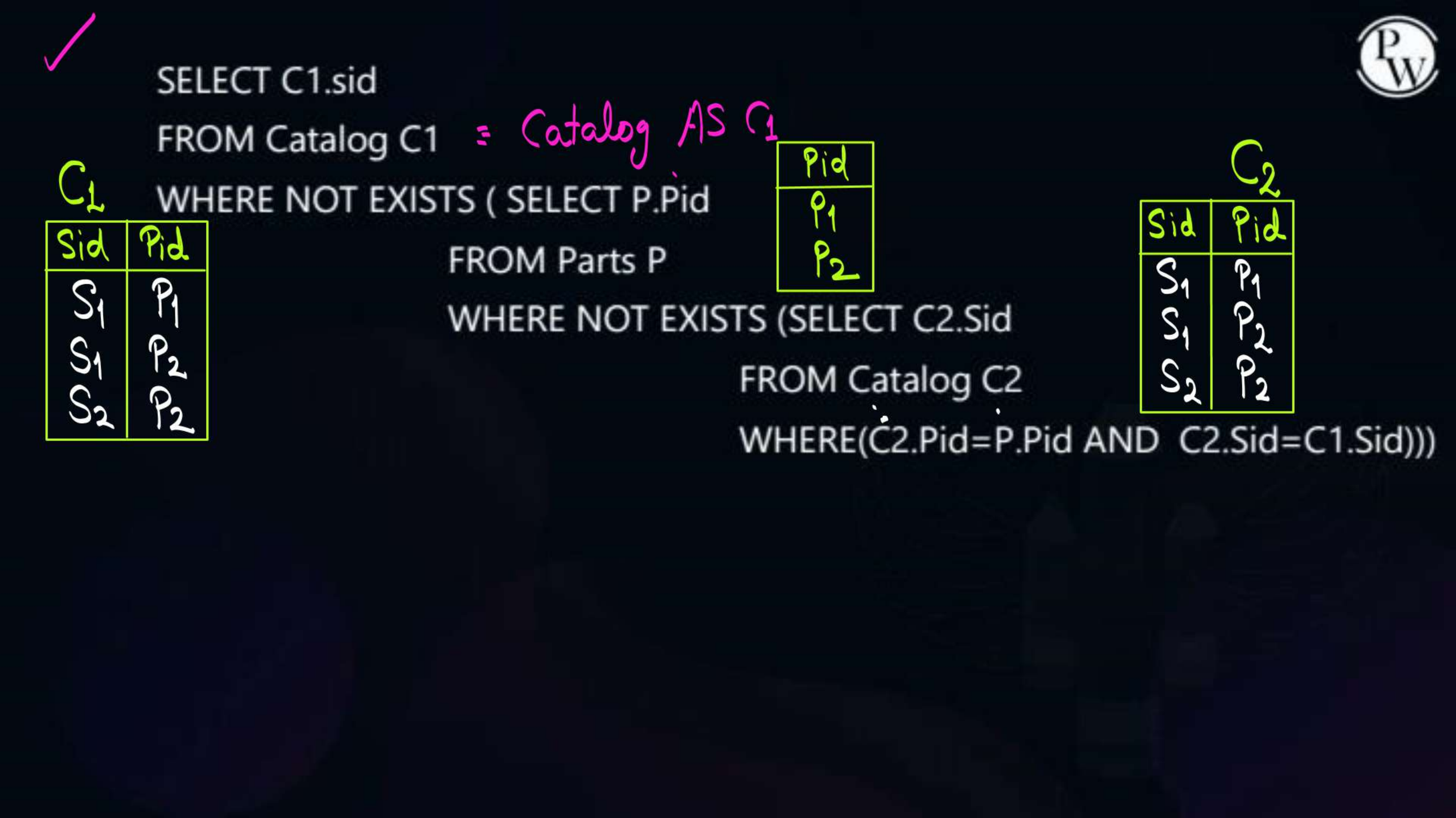
What output is produced by above SQL query:

A. Sids of suppliers who supplied some parts $o/p = \{S_1, S_2\}$

B. Sids of suppliers who supplied only proper subset of parts from all parts $o/p = \{S_2\}$

C. Sids of suppliers who supplied all parts $o/p = \{S_1\}$

D. ~~Sids of suppliers who did not supply any part~~


SELECT C1.sid
FROM Catalog C1 = Catalog AS C1
WHERE NOT EXISTS (SELECT P.Pid
FROM Parts P
WHERE NOT EXISTS (SELECT C2.Sid
FROM Catalog C2
WHERE(C2.Pid=P.Pid AND C2.Sid=C1.Sid)))
C1
Sid Pid
S1 P1
S1 P2
S2 P2
Pid
P1
P2
C2
Sid Pid
S1 P1
S1 P2
S2 P2

Topic : AS clause

{ Using 'AS' clause we can rename almost every thing }

AS clause is used to rename a column or table with an alias.

An alias only exists for the duration of the query execution.

#e.g. Consider a database that has the relation schema EMP (EmpId, EmpName, and DeptName).

An instance of the schema EMP and a SQL query on it are given below:

EmpId	Emp Name	DeptName
1	XYA	AA
2	XYB	AA
3	XYC	AA
4	XYD	AA
5	XYE	AB
6	XYF	AB
7	XYG	AB
8	XYH	AC
9	XYI	AC
10	XYJ	AC
11	XYK	AD
12	XYL	AD
13	XYM	AE

```
SELECT AVG(EC.Num)
FROM EC
WHERE (DeptName, Num) IN
    (SELECT DeptName, COUNT(EmpId) AS EC(DeptName, Num)
     FROM EMP
     GROUP BY DeptName)
```

The output of executing the SQL query is Ans=2.6

DeptName	Count(EmpID)
AA	4
AB	3
AC	3
AD	2
AE	1

$\frac{13}{5} = 2.6$

Topic : WITH clause

{"With" is used along with "As"}

The WITH Clause is mainly used to provide a subquery block [redacted] a name that can be referenced within the main SQL query. or other subquery that follows.

Syntax of WITH:

WITH New_rel"_name(. , . , . . . ,) AS Sub-query block

Here we will define a new schema

List of new names for the attributes

Here we have a sub-query

The output of this sub-query will be assigned a new-name by 'WITH' clause

We may have multiple such sub-query with "WITH" clause.

In the end we will have our main query

Select - - - - - - - - - - - - - - - -

our main query

#e.g. Consider the following database table named water_scheme

water_scheme		
scheme_no	district_name	capacity
1	Ajmer	20
1	Bikaner	10
2	Bikaner	10
3	Bikaner	20
1	Churu	10
2	Churu	20
1	Dungargarh	10

The number of tuples returned by the following SQL query is_____

```
with total(name, capacity) as
        select district_name, sum(capacity)
        from water_schemes
        group by district_name
with total_avg(capacity) as
        select avg(capacity)
        from total
select name from total, total_avg
        where total.capacity >= total_avg.capacity
```


Thank
THANK
Keep Hustling!