Session 18 - Miscellaneous Topics

01 October 2024 22:22

LinkedIn : https://www.linkedin.com/in/chauhan-prakash/

DSML Aug24 Beginner 2

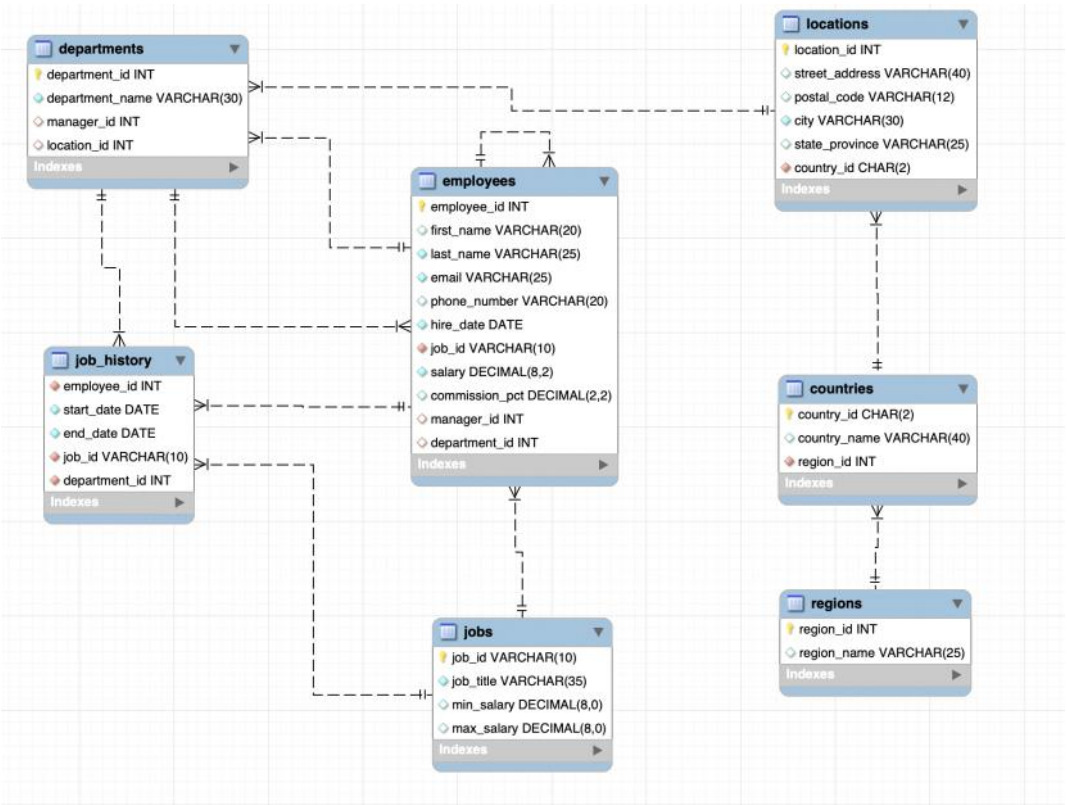
Agenda

1. Problem Statement
2. Correlated Subqueries
3. EXISTS keyword
4. Break & Doubt Resolution
5. ANY and ALL operators
6. Fetch & delete duplicate records
7. GROUP_CONCAT()
8. COALESCE()
9. Impact of the analysis
10. Practice Question

GlobalTech Solutions

GlobalTech Solutions, a leading technology firm with a diverse workforce, has noticed inconsistencies and anomalies in its hu man resources data.
To improve employee satisfaction and streamline HR processes, the HR department aims to conduct a thorough analysis of employ ee data.
This includes identifying salary disparities, commission inconsistencies, managerial structures, and potential data integrity issues.
The primary objective is to gain actionable insights into employee compensation structures, ensure data integrity, and improv e reporting accuracy.

Dataset: LINK



Correlated Subqueries:

Subquery ?

Outer Query (

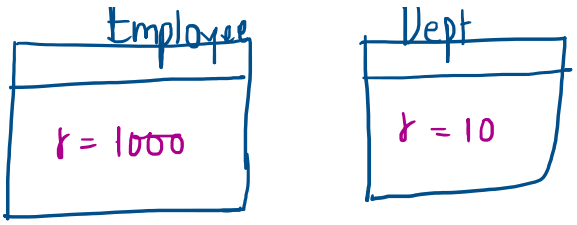
Subquery ?

Outer Query (

Inner Query

)

↑ independent of outer query.



> Outer Query and inner query are correlated ...
> for every iteration of outer query, inner query will also run.

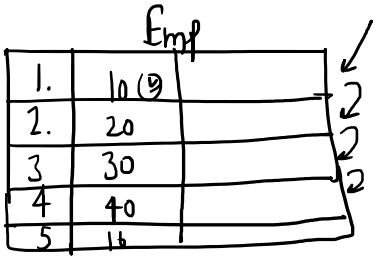
Correlated Subquery =

r = 1000 (Emp)

r = 10 (Dept)

(?)

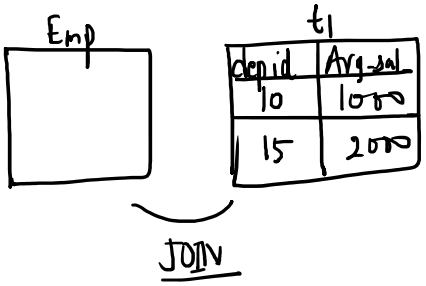
dept	avg_sal
10	1000
20	1500



Q: Retrieve the details of employees whose salary is higher than the average salary of employees in their respective departments.

```
select e.first_name,
       e.last_name,
       e.department_id,
     e.salary
from employees e
where salary > (
    select avg(e1.salary) avg_sal from employees e1
   where e1.department_id = e.department_id
);
```

Using JOIN:



Homework =

Solution:

Q: Retrieve the details of employees whose commission is lower than the average commission of employees in the same job category (job_id).

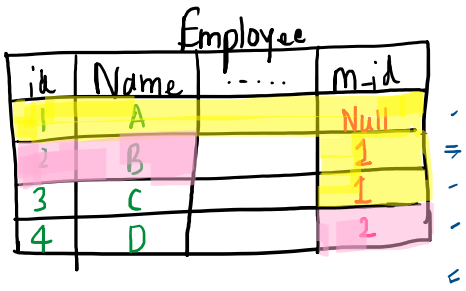
```
select e.first_name,
        e.last_name,
        e.department_id,
      e.commission_pct,
      e.salary
from employees e
where e.commission_pct < (
      select avg(e1.commission_pct) comm_pct from employees e1
   where e1.job_id = e.job_id
);
```

EXISTS keyword:

Q: Retrieve the details of employees who have at least one employee reporting to them.

Columns:

employee_id	int
first_name	text
last_name	text
email	text
phone_number	text
hire_date	text
job_id	text
salary	int
commission_pct	text
manager_id	bigint
department_id	int




```
Select *
from employees e
Where EXISTS (
    Select 1
    from Employees
    where m_id = e.id
)
```

1, A where EXISTS (true)
2, B " (true)
3, C " (false) X

```
select *
from employees e
where EXISTS (
      Select 1 from employees e1
   where e1.manager_id = e.employee_id
);
```

QUIZ:

What does the following SQL query achieve?

Query:

```
SELECT department_name
FROM demo.departments d
WHERE EXISTS (
  SELECT 1
  FROM demo.employees e
```

Employees who are Manager and having salary <= avg(sal) of all emp in their department.

```
WHERE EXISTS (
  SELECT 1
  FROM demo.employees e
  WHERE e.employee_id = d.manager_id
  AND e.salary <= (
    SELECT AVG(salary)
    FROM demo.employees e2
    WHERE e2.department_id = d.department_id
  )
);
```

having salary <= avg(salary) ... in their department.

A
Identifies the departments where the manager is earning less compared to the average earning of an employee from that departm ent.

B
Identifies the departments where the manager's salary is less than or equal to the average salary of employees in that depart ment.

C
Identifies the departments where the average salary of an employee is less than or equal to the manager's salary within that department.

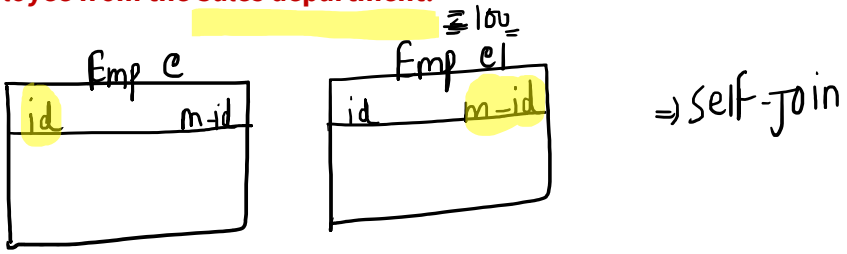
D
Identifies the departments where the manager's salary is greater than or equal to the average salary of employees in that dep artment.

ANY and ALL operators:

These operators are used in combination with comparison operators to compare a value with a set of values returned by a subqu ery.

- The ANY operator returns true if the comparison holds true for **at least one value** in the set.
- The ALL operator returns true if the comparison holds true for **all values** in the set.

Q: Retrieve the details of employees who get a commission equal to the commission percentage of any manager level employee from the Sales department.



Approach 2: ANY

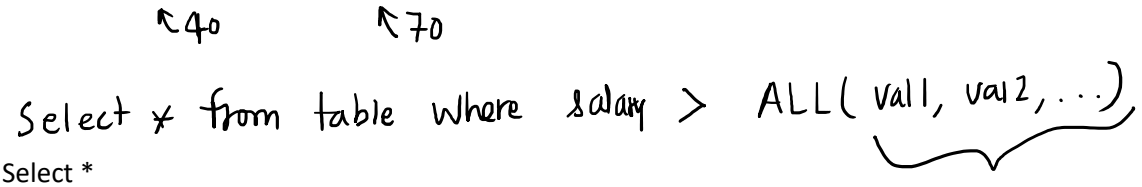
Select * from table1 where col = ANY(val1, val2, ...)

```
Select e.employee_id,
          e.first_name,
      e.last_name
from employees e
where e.commission_pct = ANY (
     select e1.commission_pct
   from employees e1
     where e1.department_id = 80
   and lower(e1.job_id) like "%man%"
```

);

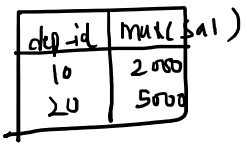
Break Till : 22:27

Q: Retrieve the details of employees who have a higher salary than all employees in both of these departments - Human Resource and Public Relations.




```
Select *
from employees e
where e.salary > ALL (
      select salary
   from employees e1 JOIN departments d
   ON e1.department_id = d.department_id
   where department_name IN ("Human Resource", "Public Relations")
);
```

Approach 2:



Fetching duplicate records

Q: How would you identify if there are any duplicate records present in a table?

```
CREATE TABLE job_hist (
   employee_id INT,
   start_date DATE,
   end_date DATE,
   job_id VARCHAR(20),
   department_id INT
);
```

```
INSERT INTO job_hist (employee_id, start_date, end_date, job_id, department_id) VALUES
(101, '1989-09-21', '1993-10-27', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(102, '1993-01-13', '1998-07-24', 'IT_PROG', 60),
(114, '1998-03-24', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(176, '1998-03-24', '1998-12-31', 'SA_REP', 80),
(176, '1999-01-01', '1999-12-31', 'SA_MAN', 80),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1994-07-01', '1998-12-31', 'AC_ACCOUNT', 90),
(201, '1996-02-27', '1999-12-19', 'MK_REP', 20);
```

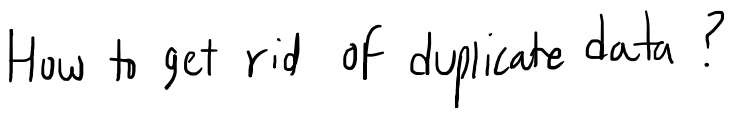
GROUP BY:

```
select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	count(*) dup_cnt
from job_hist
group by employee_id,
		start_date,
	end_date,
		job_id,
	department_id
having count(*) >1 ;
```

WINDOWING:

```
select * from (
select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	row_number() over (
			partition by employee_id,start_date,end_date,job_id,department_id
	) as row_num
from job_hist) t1
where t1.row_num = 1;
```

distinct



→ DISTINCT ⇒

```
CREATE TABLE new_job_hist AS
SELECT DISTINCT employee_id,
		start_date,
	end_date,
		job_id,
	department_id
FROM job_hist;
```

GROUP_CONCAT()

Input:

order_id	customer_id	product
1	101	Product A
2	102	Product B
3	101	Product C
4	103	Product A
5	101	Product B
6	102	Product D
7	104	Product A
8	101	Product D
9	105	Product C
10	103	Product B

Output:

customer_id	ordered_products
101	Product A, Product C, Product B, Product D
102	Product B, Product D
103	Product A, Product B
104	Product A
105	Product C

Q: Let's say we want to concatenate the first_name of employees within each department_id into a single string separated by commas.

```
select e.department_id,
        group_concat(e.first_name order by e.first_name separator ",") emp_list
from employees e
group by e.department_id;
```

```
select e.department_id,
        group_concat(CONCAT(e.first_name," ",e.last_name ) order by e.first_name separator ",") emp_list
from employees e
group by e.department_id;
```

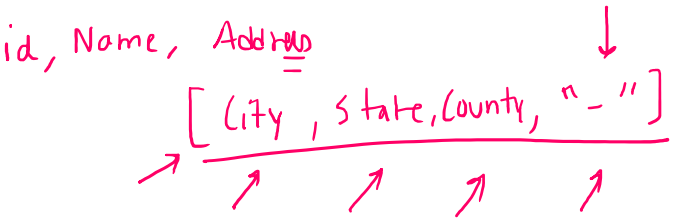
COALESCE()

```
select first_name,
        last_name,
    coalesce(salary,0) as salary
from employees e;
```

Q: Retrieve the names of all employees from our database according to the following guidelines -

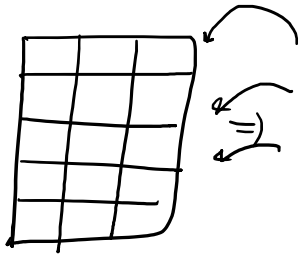
- If the employee's first name is provided, retrieve it.
- If the employee's first name is not available, retrieve their last name.
- If neither the employee's first name nor their last name is specified, set the default value to "UNKNOWN".

```
select coalesce(first_name, last_name, "UNKNOWN") as emp_name
from employees;
```




```
WITH user_platform AS (
SELECT user_id, spend_date,
SUM(CASE WHEN platform = 'mobile' THEN 1 ELSE 0 END) as mobile_use,
SUM(CASE WHEN platform = 'desktop' THEN 1 ELSE 0 END) as desktop_use,
SUM(CASE WHEN platform = 'mobile' THEN amount ELSE 0 END) as mobile_amount,
SUM(CASE WHEN platform = 'desktop' THEN amount ELSE 0 END) as desktop_amount
FROM Spending
GROUP BY user_id, spend_date
),
```

```
SELECT spend_date,
CASE
WHEN mobile_use > 0 AND desktop_use > 0 THEN 'both'
WHEN mobile_use > 0 THEN 'mobile'
ELSE 'desktop'
END as platform,
mobile_amount + desktop_amount as total_amount
FROM user_platform
```



(oalesce (first_name, last_name, "NA")

= ~ ifnull (fist_name, "NA")

QUERIES:

```
use hr;

select e.first_name,
          e.last_name,
          e.department_id,
     e.salary
from employees e
where salary > (
        select avg(e1.salary) avg_sal from employees e1
   where e1.department_id = e.department_id
);

select * from employees limit 1000;

select e.first_name,
          e.last_name,
          e.department_id,
     e.commission_pct,
     e.salary
from employees e
where e.commission_pct < (
        select avg(e1.commission_pct) comm_pct from employees e1
   where e1.job_id = e.job_id
);


select *
from employees e
where EXISTS (
        Select 1 from employees e1
   where e1.manager_id = e.employee_id
);

Select e.employee_id,
                e.first_name,
      e.last_name
from employees e
where e.commission_pct = ANY (
        select e1.commission_pct
   from employees e1
        where e1.department_id = 80
   and lower(e1.job_id) like "%man%"
);

select distinct job_id from employees;



Select *
from employees e
where e.salary > ALL (
        select salary
   from employees e1 JOIN departments d
```

```
   ON e1.department_id = d.department_id
   where department_name IN ("Human Resource", "Public Relations")
);

CREATE TABLE job_hist (
   employee_id INT,
   start_date DATE,
   end_date DATE,
   job_id VARCHAR(20),
   department_id INT
);


INSERT INTO job_hist (employee_id, start_date, end_date, job_id, department_id) VALUES
(101, '1989-09-21', '1993-10-27', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(101, '1993-10-28', '1997-03-15', 'AC_ACCOUNT', 110),
(102, '1993-01-13', '1998-07-24', 'IT_PROG', 60),
(114, '1998-03-24', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(122, '1999-01-01', '1999-12-31', 'ST_CLERK', 50),
(176, '1998-03-24', '1998-12-31', 'SA_REP', 80),
(176, '1999-01-01', '1999-12-31', 'SA_MAN', 80),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1987-09-17', '1993-06-17', 'AD_ASST', 90),
(200, '1994-07-01', '1998-12-31', 'AC_ACCOUNT', 90),
(201, '1996-02-27', '1999-12-19', 'MK_REP', 20);


select * from job_history;

select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	count(*) dup_cnt
from job_hist
group by employee_id,
		start_date,
	end_date,
		job_id,
	department_id
having count(*) >1 ;

select * from (
select employee_id,
		start_date,
	end_date,
		job_id,
	department_id,
	row_number() over (
			partition by employee_id,start_date,end_date,job_id,department_id
	) as row_num
from job_hist) t1
where t1.row_num > 1;

CREATE TABLE new_job_hist AS
SELECT DISTINCT employee_id,
		start_date,
	end_date,
		job_id,
	department_id
FROM job_hist;

select * from new_job_hist;

delete from job_hist;

select * from job_hist;

insert into job_hist (select * from new_job_hist);

drop table new_job_hist;

CREATE TABLE new_job_hist AS
select * from (
select employee_id,
		start_date,
```

```
	end_date,
		job_id,
	department_id,
	row_number() over (
			partition by employee_id,start_date,end_date,job_id,department_id
	) as row_num
from job_hist) t1
where t1.row_num = 1;


select e.department_id,
		group_concat(CONCAT(e.first_name," ",e.last_name ) order by e.first_name separator ",") emp_list
from employees e
group by e.department_id;

select first_name,
		last_name,
	coalesce(salary,0) as salary
from employees e;


select coalesce(first_name, last_name, "UNKNOWN") as emp_name
from employees;
```