CMSC 508
Database Theory

Intermediate SQL (I)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003

- Aggregate Functions
  - These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values

select *AVG(salary)* **from** *employees;*

select *AVG(salary)* **from** *employees*
where *department_id = 100;*

select *MAX(salary)* **from** *employees*
where *manager_id* ***is not null****;*

- Aggregate Functions
  - These functions operate on the multiset of values of a column of a relation, and return a value

variance, stddev, median: statistics
count: number of values

select *COUNT(*)* **from** *employees;*

select *COUNT(*)* **from** *employees*
where *first_name = 'John';*

select *COUNT(distinct last_name)* **from** *employees;*

- Aggregate Functions

Combine multiple functions **only** if having same cardinality

select *AVG(salary), MAX(salary), MIN(salary), SUM(salary)*
from *employees* **natural join** *jobs*
where *job_title* **like** *'Sales%'*;

select *MIN(hire_date), MAX(hire_date)*
from *employees*
where *salary* **between** *1500* **and** 2500;

select *AVG(commission_pct)*
from *employees*;

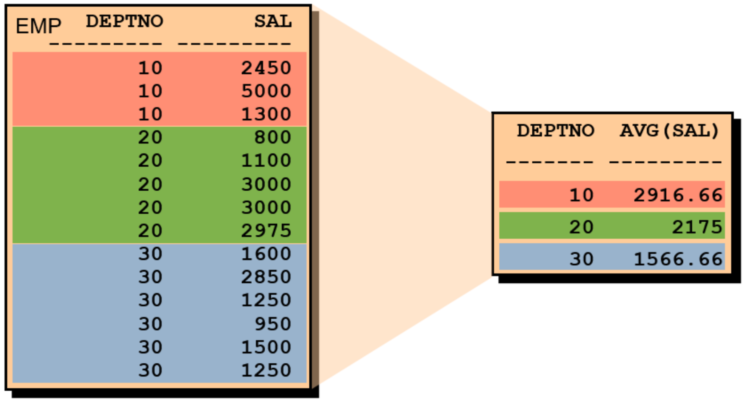
select *AVG(NVL(commission_pct,0))*
from *employees*;

All aggregate operations except **count(*)** ignore tuples with null values on the aggregated attributes

- Aggregate Functions – GROUP BY

```
SELECT    column, FUNCTION(expr)
FROM      table(s)
[WHERE    conditions(s)]
[GROUP BY group_by_expr]
[ORDER BY column];
```

select *department_id, AVG(salary)*
from *employees*
group by *department_id*
order by *department_id*;



EMP

```
 DEPTNO        SAL
--------- ---------
       10      2450
       10      5000
       10      1300
       20       800
       20      1100
       20      3000
       20      3000
       20      2975
       30      1600
       30      2850
       30      1250
       30       950
       30      1500
       30      1250
```

```
 DEPTNO  AVG(SAL)
------- ---------
     10   2916.66
     20      2175
     30   1566.66
```

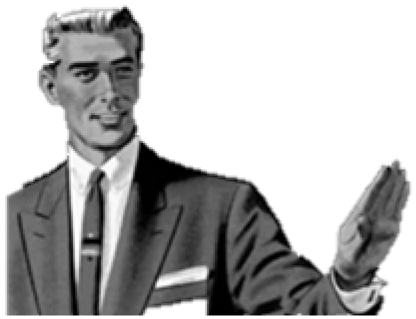
Attributes in **select** clause outside of aggregate functions **must appear** in **group by** list

- Aggregate Functions – GROUP BY

select *department_id, count(*)*
from *employees;*

ORA-00937: not a single-group group function

select *department_id, count(*)*
from *employees*
group by *department_id;*



select *department_name, count(*)*
from *employees*
group by *department_id;*

ORA-00934: group function is not allowed here

select *d.department_name*, count(*)
from *employees e* **join** *departments d*
on *e.department_id = d.department_id*
group by *e.department_id*;

ORA-00979: not a GROUP BY expression

- Aggregate Functions – GROUP BY

select *department_id, AVG(salary)*
from *employees*
where *salary > 5000*
group by *department_id*;

select *department_id, AVG(salary)*
from *employees*
where *AVG(salary) > 5000*
group by *department_id*;

ORA-00934: group function is not allowed here

- Aggregate Functions – HAVING

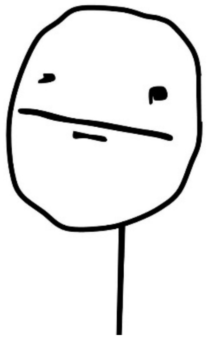
```
SELECT     column(s), FUNCTION(expr)
FROM       table(s)
[WHERE     conditions(s)]
[GROUP BY  group_by_expr]
[HAVING    group_condition]
[ORDER BY  column];
```

  - Predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups

select *department_name, avg(salary)*
from *employees* **join** *departments*
on *employees.department_id = departments.department_id*
group by *department_name*
having *AVG(salary)* > 5000;

- Aggregate Functions – HAVING

```
SELECT     job_title, SUM(salary)
FROM       employees natural join jobs
WHERE      job_title NOT LIKE 'Sales%'
GROUP BY   job_title
HAVING     AVG(salary) > 5000
ORDER BY   SUM(salary) DESC;
```



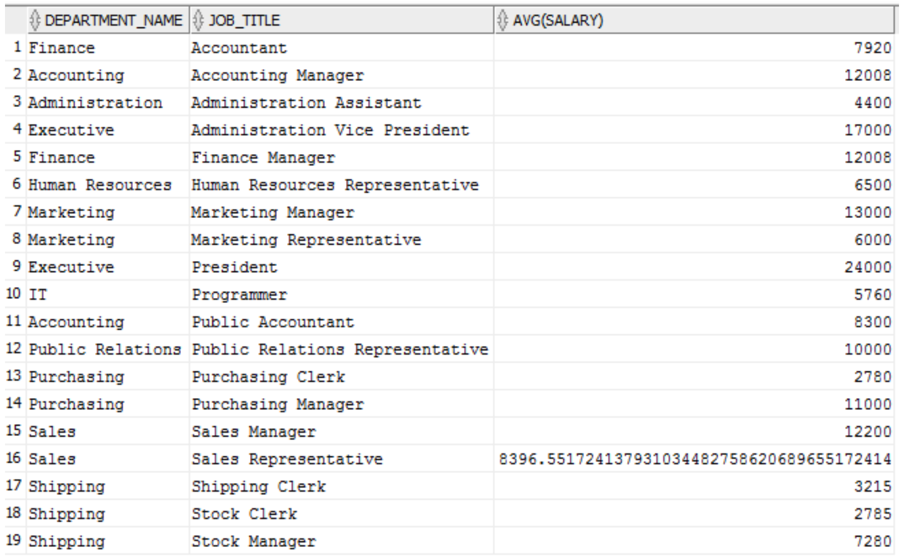
- **Nested** aggregate functions

```
SELECT     MAX(AVG(salary))
FROM       employees
GROUP BY   department_id;

SELECT     AVG(MAX(salary))
FROM       employees
GROUP BY   department_id;
```

- Aggregate Functions – GROUP BY **multiple columns**

```
SELECT  department_name, job_title, AVG(salary)
FROM    employees join departments
        on employees.department_id = departments.department_id
        join jobs
        on employees.job_id = jobs.job_id
GROUP BY department_name, job_title
ORDER BY job_title, department_name;
```

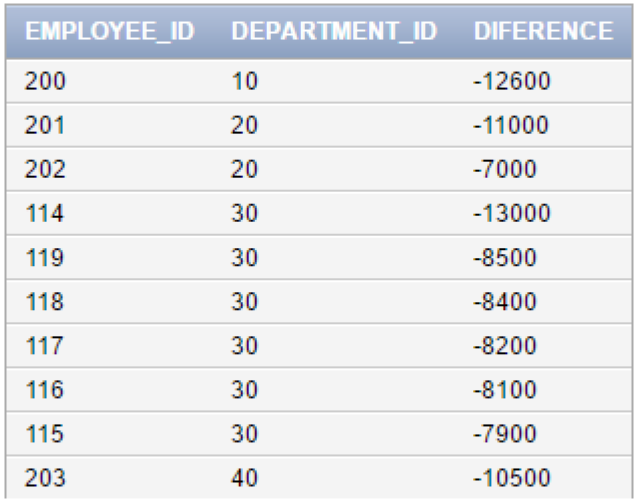


	DEPARTMENT_NAME	JOB_TITLE	AVG(SALARY)
1	Finance	Accountant	7920
2	Accounting	Accounting Manager	12008
3	Administration	Administration Assistant	4400
4	Executive	Administration Vice President	17000
5	Finance	Finance Manager	12008
6	Human Resources	Human Resources Representative	6500
7	Marketing	Marketing Manager	13000
8	Marketing	Marketing Representative	6000
9	Executive	President	24000
10	IT	Programmer	5760
11	Accounting	Public Accountant	8300
12	Public Relations	Public Relations Representative	10000
13	Purchasing	Purchasing Clerk	2780
14	Purchasing	Purchasing Manager	11000
15	Sales	Sales Manager	12200
16	Sales	Sales Representative	8396.551724137931034482758620689655172414
17	Shipping	Shipping Clerk	3215
18	Shipping	Stock Clerk	2785
19	Shipping	Stock Manager	7280

- Exercise
  - Compute for each department the average salary difference between employees and their managers

- Exercise
  - Compute for each department the average salary difference between employees and their managers

```
SELECT   e.employee_id, e.department_id,
(e.salary - m.salary) as difference
FROM     employees e, employees m
WHERE    e.manager_id = m.employee_id
ORDER BY e.department_id;
```

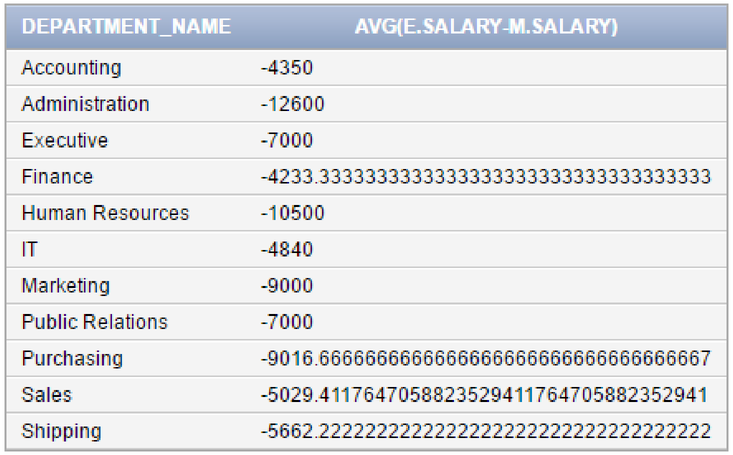


EMPLOYEE_ID	DEPARTMENT_ID	DIFERENCE
200	10	-12600
201	20	-11000
202	20	-7000
114	30	-13000
119	30	-8500
118	30	-8400
117	30	-8200
116	30	-8100
115	30	-7900
203	40	-10500

```
GROUP BY department_id
```

- Exercise
  - Compute for each department the average salary difference between employees and their managers

```
SELECT   d.department_name, AVG(e.salary - m.salary)
FROM     employees e, employees m, departments d
WHERE    e.manager_id = m.employee_id
         and e.department_id = d.department_id
GROUP BY d.department_name
ORDER BY d.department_name;
```



DEPARTMENT_NAME	AVG(E.SALARY-M.SALARY)
Accounting	-4350
Administration	-12600
Executive	-7000
Finance	-4233.3333333333333333333333333333333333
Human Resources	-10500
IT	-4840
Marketing	-9000
Public Relations	-7000
Purchasing	-9016.6666666666666666666666666666666667
Sales	-5029.4117647058823529411764705882352941
Shipping	-5662.2222222222222222222222222222222222

- Exercises
  - Write a query to get the job tile and maximum salary of the employees where maximum salary is greater than or equal to $4000.
  - Write a query to get the average salary for all departments employing more than 10 employees.
  - Find the average salary of departments 50 and 80 only

- Exercises

```
SELECT j.job_title, MAX(e.salary)
FROM employees e join jobs j
on e.job_id = j.job_id
GROUP BY j.job_title
HAVING MAX(e.salary) >=4000;

SELECT department_id, AVG(salary), COUNT(*)
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 10;

SELECT department_id , AVG(salary)
FROM employees
WHERE department_id IN (50, 80)
GROUP BY department_id;
```

CMSC 508
Database Theory

Intermediate SQL (I)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003