Renaming Result Attributes

- It is possible to rename any attribute that appears in the result of a query by adding the qualifier **AS** followed by the desired name.
- Example: Find last name of employee and supervisor

```
SELECT    E.Lname AS Employee_name,
          S.Lname AS Supervisor_name
FROM      EMPLOYEE AS E,   EMPLOYEE AS S
WHERE     E.Super_ssn=S.Ssn;
```

Natural Joins

- In a natural join on two relations R and S, no join condition is specified.
  - An implicit EQUIJOIN condition for each pair of attributes of the **same name** from R and S is created.
  - Each such pair of attributes is included only once in the resulting relation.
  - Reduces the size of queries since you are not explicitly stating join conditions.

Natural Join Example

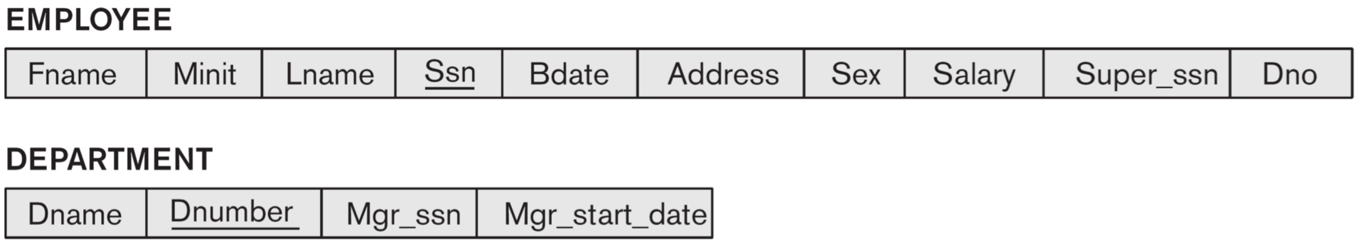
```
SELECT   DISTINCT CourseName
FROM     COURSE, SECTION
WHERE    COURSE.CourseNumber=SECTION.CourseNumber
         AND  Instructor='Anderson';
```

- Using a Natural Join:

```
SELECT   DISTINCT CourseName
FROM     COURSE  NATURAL JOIN  SECTION
WHERE    Instructor='Anderson';
```

Natural Joins

- If the names of the join attributes are not the same in the relations, it is possible to rename the attributes so that they match. You can then apply a NATURAL JOIN.
- Utilize the AS construct to rename a relation and its attributes.
- Example:
  - NATURAL JOIN EMPLOYEE and DEPARTMENT



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

Inner Joins

- The default type of JOIN used by SQL when joining tables is called an INNER JOIN.
- A tuple is included in the result only if a matching tuple exists in the other relation.
  - Example from Slide #2:

```
SELECT    E.Lname AS Employee_name,
          S.Lname AS Supervisor_name
FROM      EMPLOYEE AS E,   EMPLOYEE AS S
WHERE     E.Super_ssn=S.Ssn;
```

- Only employees that have a supervisor are included in the result.
- What if we wanted to see ALL employees? - Need **OUTER JOIN**

Outer Joins

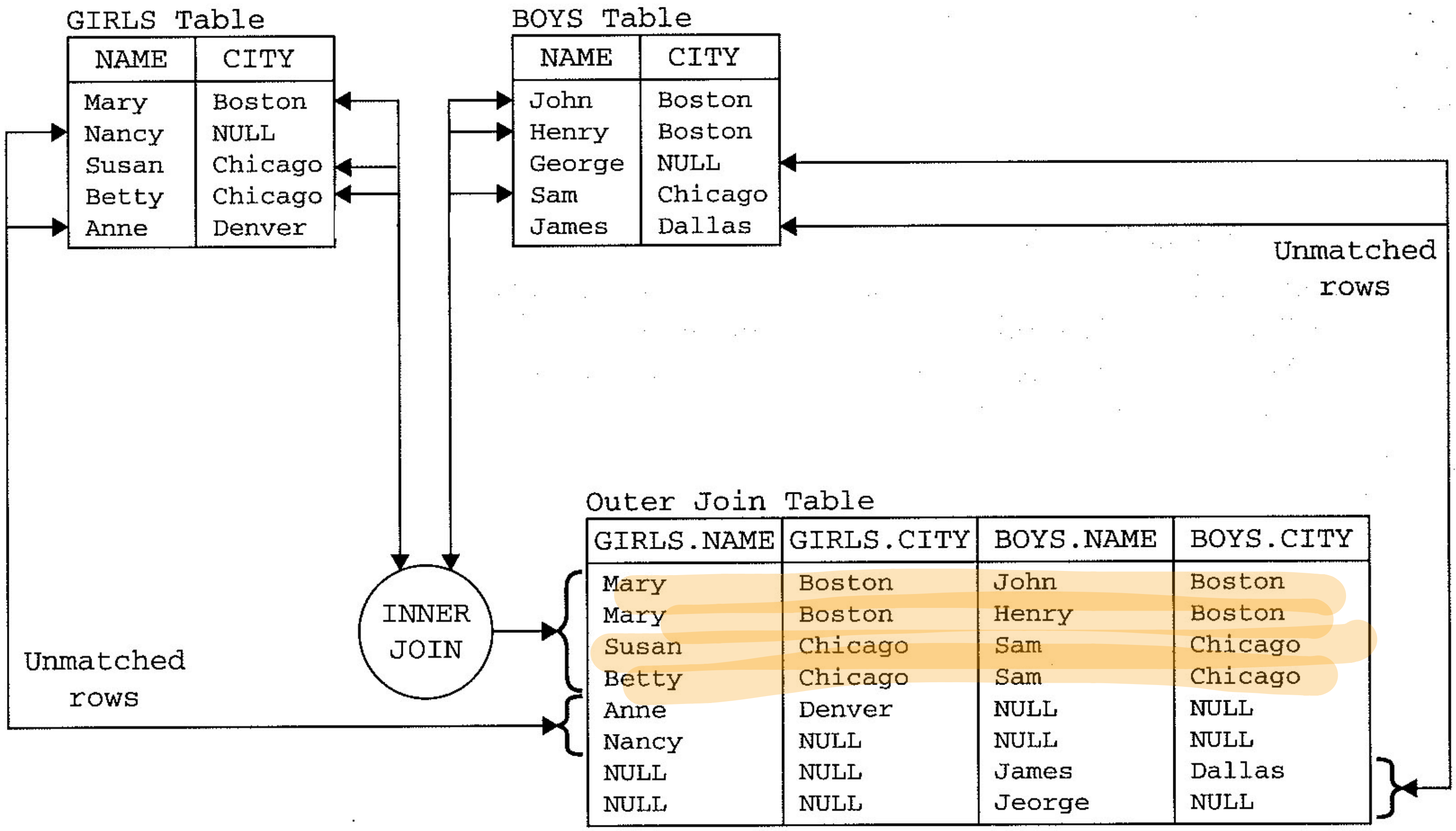
- If you want to include all rows in the relations, regardless of the join conditions, then you need to use an OUTER JOIN.
- Two types: Left Outer Join and Right Outer Join
- Example:

```
SELECT    E.Lname AS Employee_name,
          S.Lname AS Supervisor_name
FROM      (EMPLOYEE AS E   LEFT OUTER JOIN
           EMPLOYEE AS S   ON  E.Super_ssn=S.Ssn);
```

Outer Joins

- Left Outer Joins
  - (ALPHA LEFT OUTER JOIN BETA)
    - Every tuple in the **left** table (ALPHA) of the JOIN must appear in the result.
    - If a tuple does not have a match in the **right** table (BETA), NULL values are used for the **right** table's (BETA's) attributes.
- Right Outer Joins
  - (ALPHA RIGHT OUTER JOIN BETA)
    - Every tuple in the **right** table (BETA) of the JOIN must appear in the result.
    - If a tuple does not have a match in the **left** table (ALPHA), NULL values are used for the **left** table's (ALPHA's) attributes.

Outer Joins - Example



GIRLS Table

NAME	CITY
Mary	Boston
Nancy	NULL
Susan	Chicago
Betty	Chicago
Anne	Denver

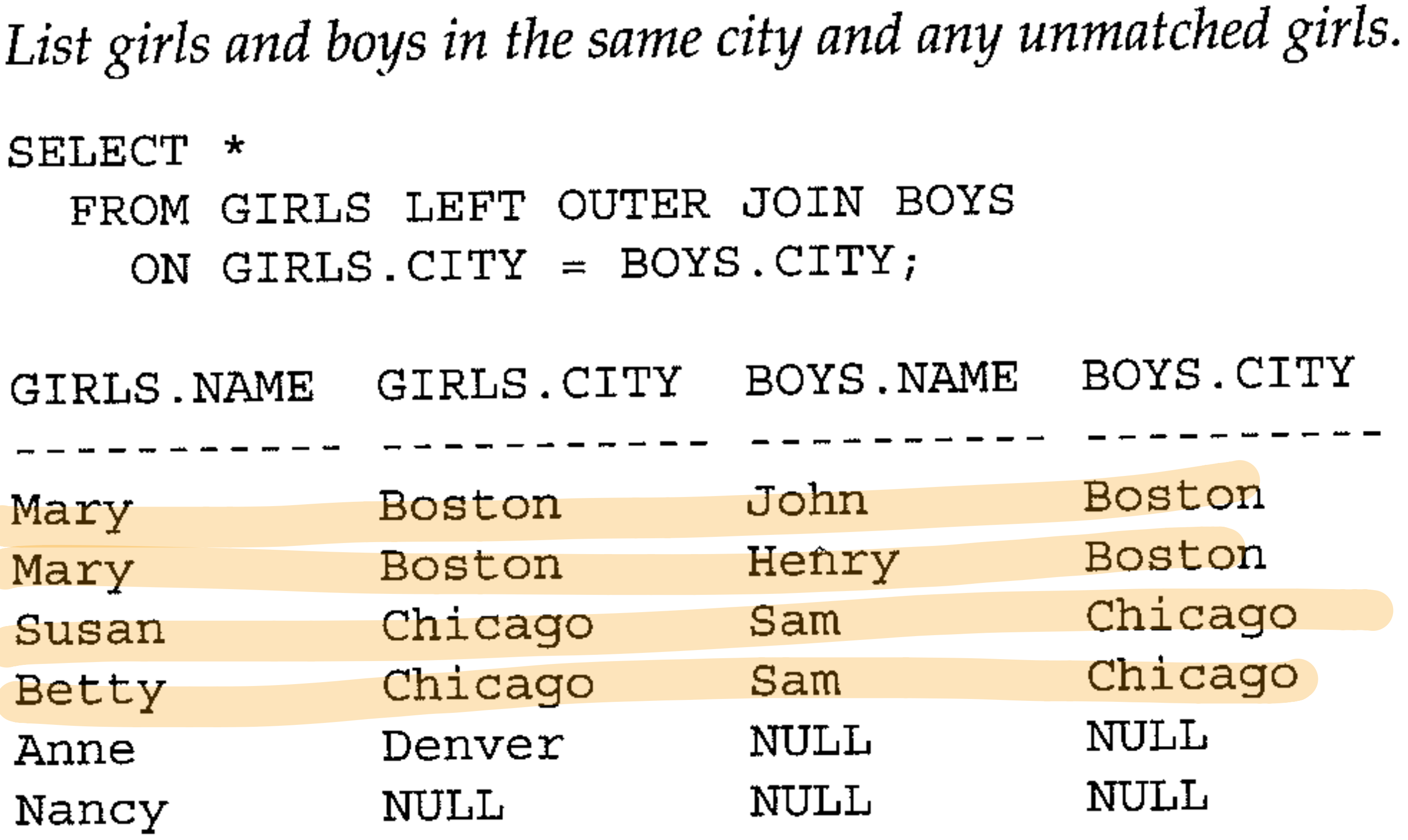
BOYS Table

NAME	CITY
John	Boston
Henry	Boston
George	NULL
Sam	Chicago
James	Dallas

Outer Join Table

GIRLS.NAME	GIRLS.CITY	BOYS.NAME	BOYS.CITY
Mary	Boston	John	Boston
Mary	Boston	Henry	Boston
Susan	Chicago	Sam	Chicago
Betty	Chicago	Sam	Chicago
Anne	Denver	NULL	NULL
Nancy	NULL	NULL	NULL
NULL	NULL	James	Dallas
NULL	NULL	Jeorge	NULL

LEFT Outer Joins

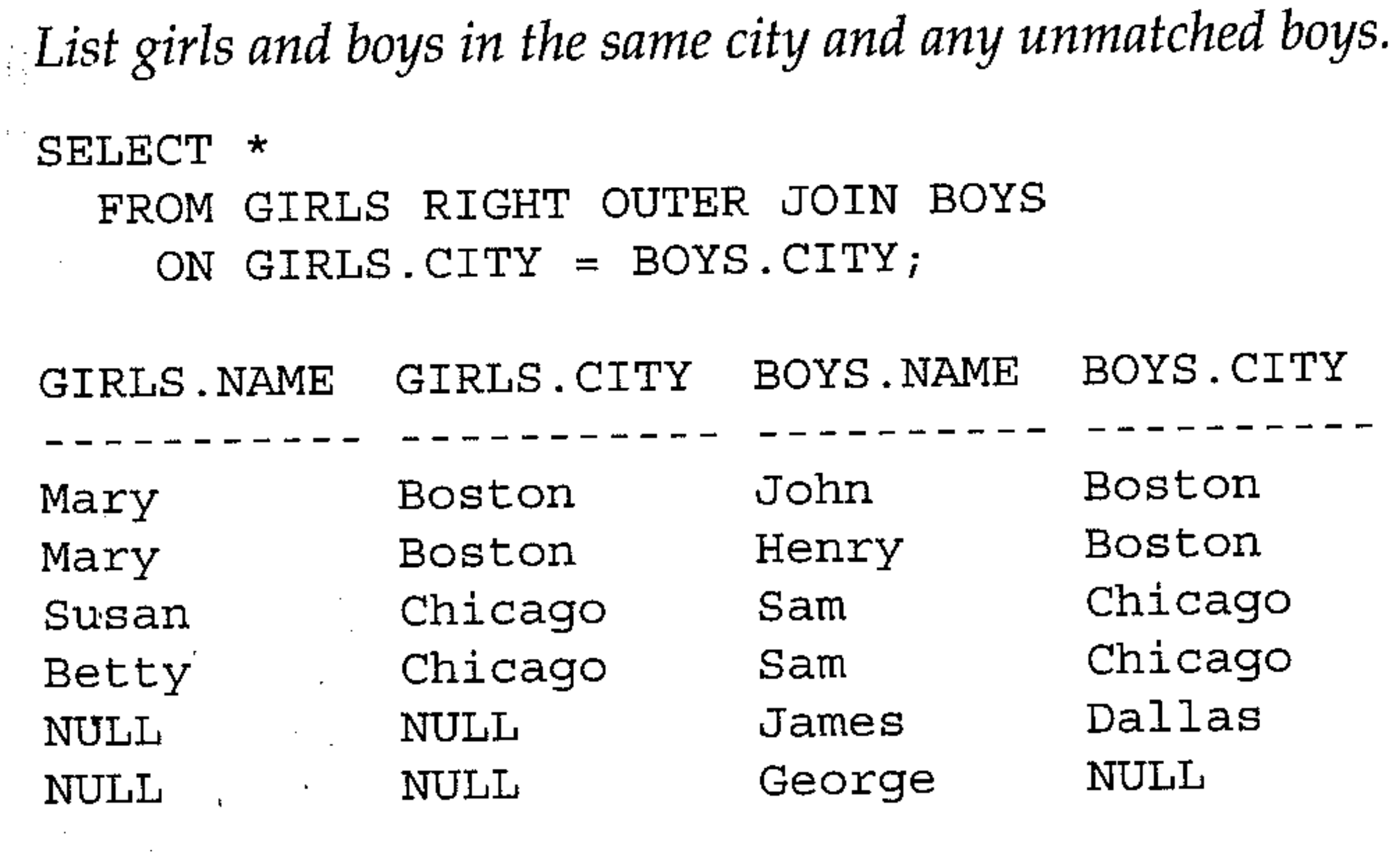


List girls and boys in the same city and any unmatched girls.

```
SELECT *
   FROM GIRLS LEFT OUTER JOIN BOYS
      ON GIRLS.CITY = BOYS.CITY;
```

```
GIRLS.NAME   GIRLS.CITY   BOYS.NAME   BOYS.CITY
----------   ----------   ---------   ---------
Mary         Boston       John        Boston
Mary         Boston       Henry       Boston
Susan        Chicago      Sam         Chicago
Betty        Chicago      Sam         Chicago
Anne         Denver       NULL        NULL
Nancy        NULL         NULL        NULL
```

RIGHT Outer Joins



List girls and boys in the same city and any unmatched boys.

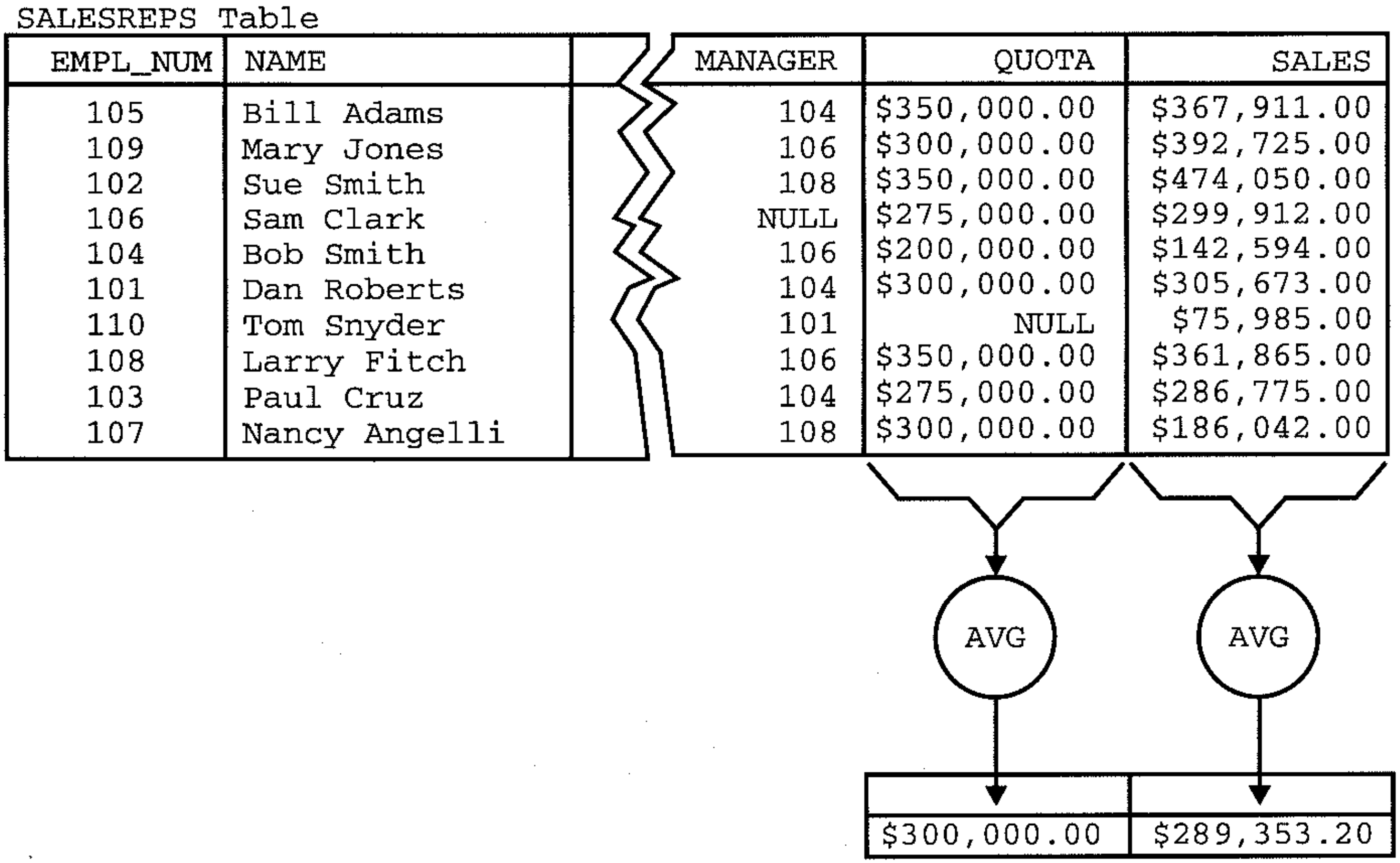
```
SELECT *
  FROM GIRLS RIGHT OUTER JOIN BOYS
    ON GIRLS.CITY = BOYS.CITY;
```

```
GIRLS.NAME   GIRLS.CITY   BOYS.NAME   BOYS.CITY
----------   ----------   ---------   ---------
Mary         Boston       John        Boston
Mary         Boston       Henry       Boston
Susan        Chicago      Sam         Chicago
Betty        Chicago      Sam         Chicago
NULL         NULL         James       Dallas
NULL         NULL         George      NULL
```

Aggregate Functions in SQL

- Aggregate function are used to summarize information from multiple tuples into a single-tuple summary.
- Grouping is used to create subgroups of tuples before summarization.
- SQL Aggregate Functions:
  - COUNT - Returns the number of tuples/values in a result
  - SUM – Returns the summation of a set of values
  - MAX – Returns the maximum value of a set of values
  - MIN – Returns the minimum values of a set of values
  - AVG – Returns the mean of a set of values.

AVG



SALESREPS Table

EMPL_NUM	NAME	MANAGER	QUOTA	SALES
105	Bill Adams	104	$350,000.00	$367,911.00
109	Mary Jones	106	$300,000.00	$392,725.00
102	Sue Smith	108	$350,000.00	$474,050.00
106	Sam Clark	NULL	$275,000.00	$299,912.00
104	Bob Smith	106	$200,000.00	$142,594.00
101	Dan Roberts	104	$300,000.00	$305,673.00
110	Tom Snyder	101	NULL	$75,985.00
108	Larry Fitch	106	$350,000.00	$361,865.00
103	Paul Cruz	104	$275,000.00	$286,775.00
107	Nancy Angelli	108	$300,000.00	$186,042.00

$300,000.00	$289,353.20

Examples:

- Find the sum of all salaries, the maximum, minimum and average salary for all employees.

```
SELECT   SUM(Salary), MAX(Salary),  MIN(Salary), AVG (Salary)
FROM     EMPLOYEE;
```

- Restrict query to just the 'Research' dept. employees.

```
SELECT   SUM(Salary), MAX(Salary), MIN(Salary), AVG (Salary)
FROM     (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE    Dname='Research';
```

Examples

- Retrieve the total number of employees in the company.

```
SELECT    COUNT(*)
FROM      EMPLOYEE;
```

- Retrieve total number of employees in Research dept.

```
SELECT    COUNT(*)
FROM      EMPLOYEE,DEPARTMENT
WHERE     Dno=Dnumber AND Dname='Research';
```

- Count the number of distinct salary values.

```
SELECT    COUNT(DISTINCT Salary)
FROM      EMPLOYEE;
```

What is this query doing?

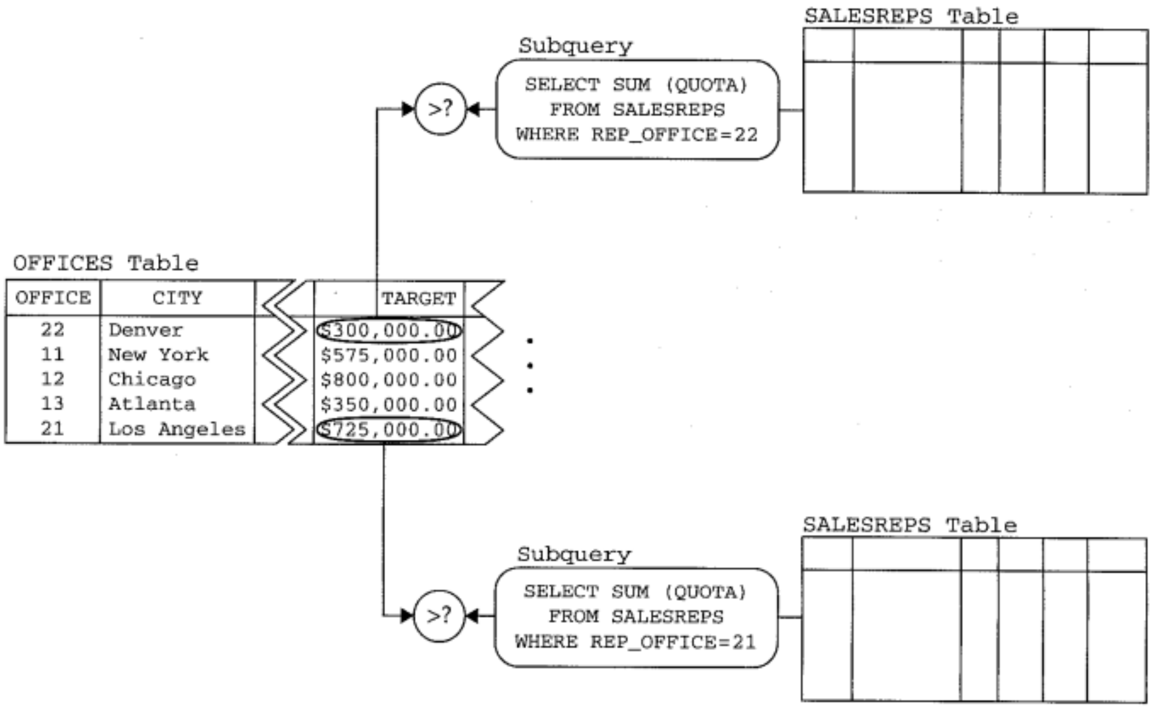
```
SELECT      Lname, Fname
FROM        EMPLOYEE
WHERE       ( SELECT  COUNT(*)
              FROM DEPENDENT
              WHERE Ssn=Essn )  >=2;
```

Select employees with 2 or more dependents.

SUM

List the offices where the sales target for the office exceeds the sum of the salespeople's quotas.

```
SELECT CITY
  FROM OFFICES
 WHERE TARGET > (SELECT SUM(QUOTA)
                  FROM SALESREPS
                 WHERE REP_OFFICE = OFFICE);
```



GROUP BY

- Many times we will want to apply aggregate functions to subgroups of tuples in a relation.
  - *Example: Find the average salary for each department*
  - *Example: How many employees are working on each project?*
- Partition the relation into nonoverlapping groups of tuples
- Each group will consist of tuples that have the same value for the grouping attribute.
- Apply the aggregate function to each group.

GROUP BY

- Add a GROUP BY clause at the end of our query
  - List the grouping attribute(s) in the clause
  - The grouping attribute(s) must also appear in the SELECT clause
- Example: For each dept, retrieve the dept number, num employees in dept and their average salary.

```
SELECT      Dno,  COUNT(*),  AVG(Salary)
FROM        EMPLOYEE
GROUP BY    Dno;
```

GROUP BY

Fname	Minit	Lname	Ssn	. . .	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	. . .	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

GROUP BY

- Example:
  - For each project retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT      Pnumber, Pname, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber, Pname;
```

HAVING

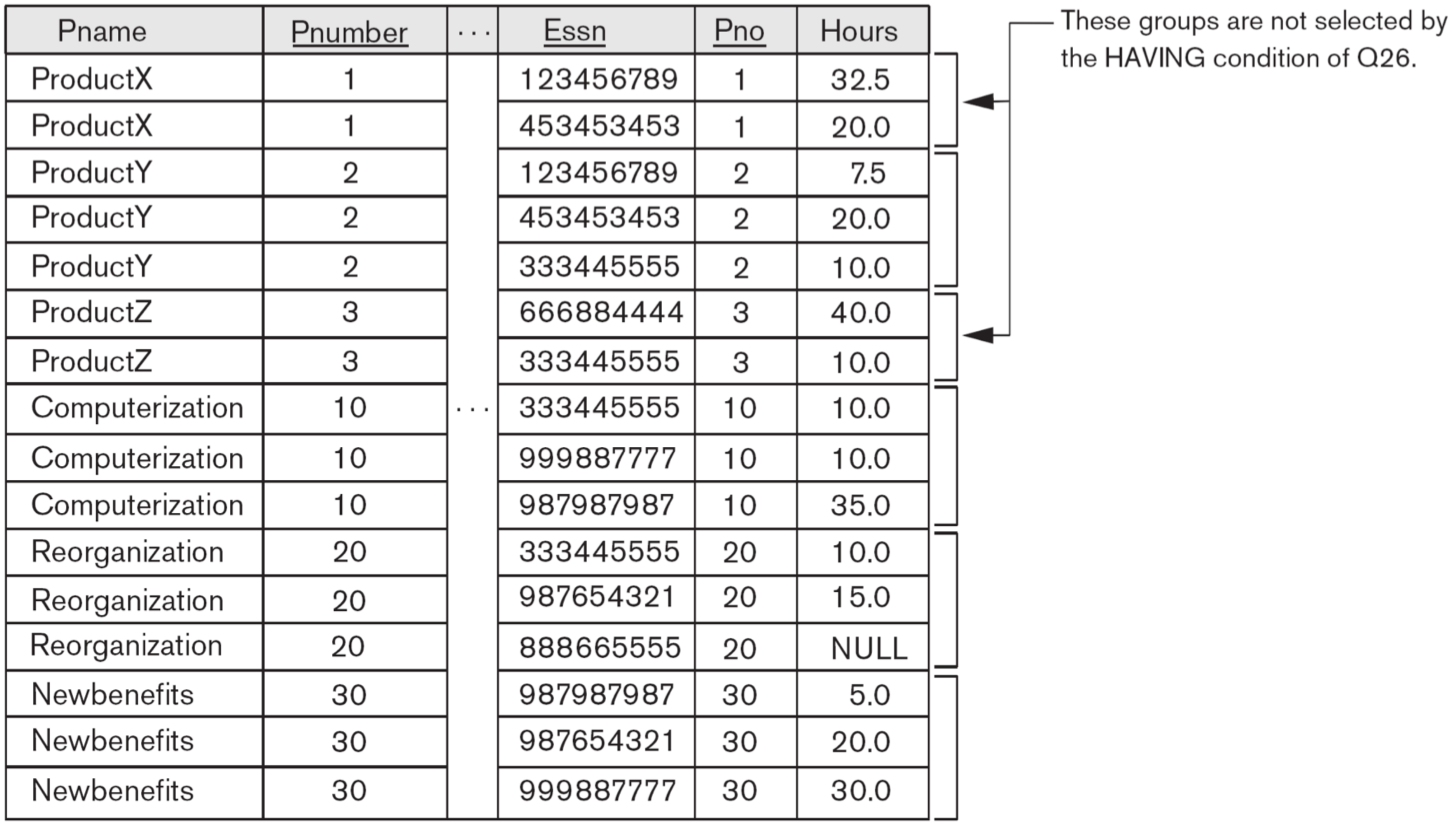
- Retrieve values of aggregate functions only for groups that satisfy certain conditions.
  - HAVING clause can appear in conjunction with a GROUP BY clause.
- HAVING provides a condition on the summary information regarding the group of tuples associated with each value of the grouping attributes.
- Only groups that satisfy the HAVING condition are retrieved in the result of the query.

HAVING - Example

- EXAMPLE:
  - For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT      Pnumber, Pname, COUNT(*)
FROM        PROJECT, WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber, Pname
HAVING      COUNT(*) > 2;
```

HAVING - Example

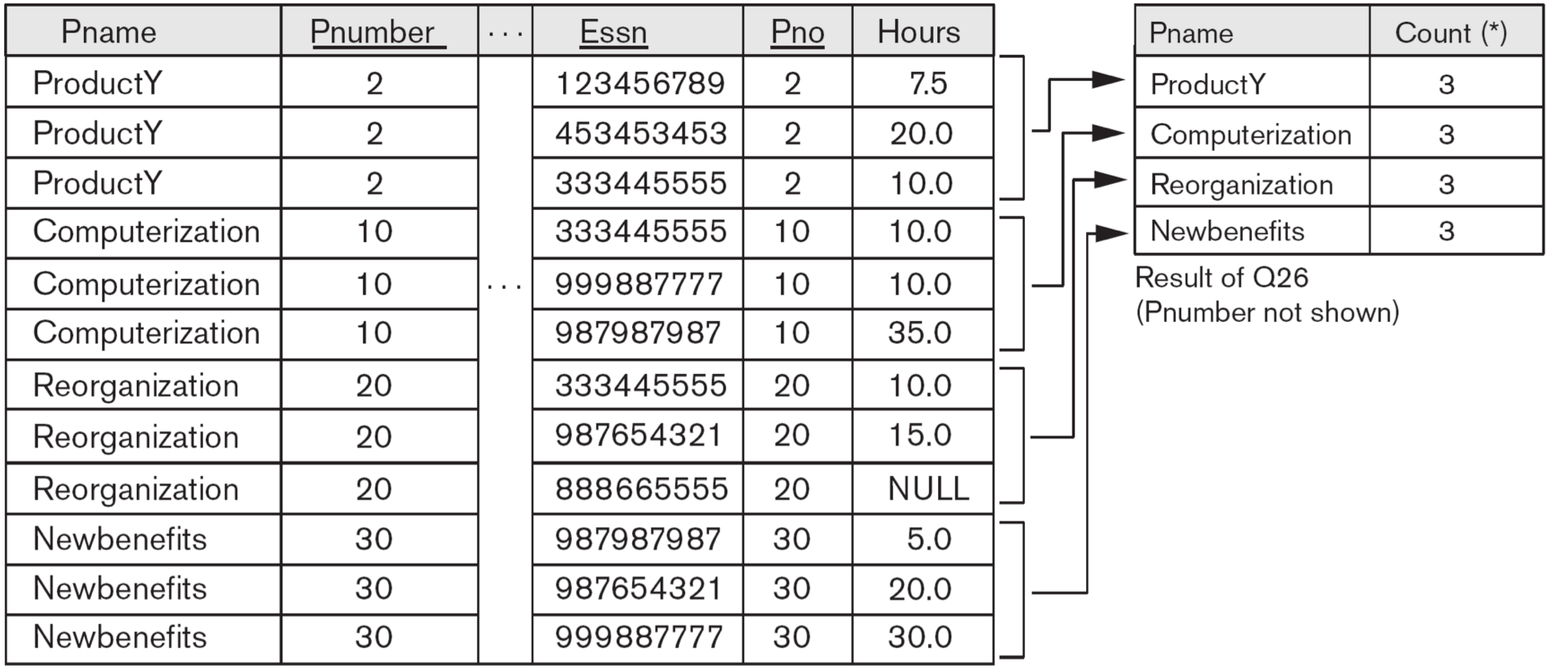


Pname	Pnumber	. . .	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	. . .	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

These groups are not selected by the HAVING condition of Q26.

After applying the WHERE clause but before applying HAVING

HAVING - Example



Pname	Pnumber	. . .	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	. . .	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

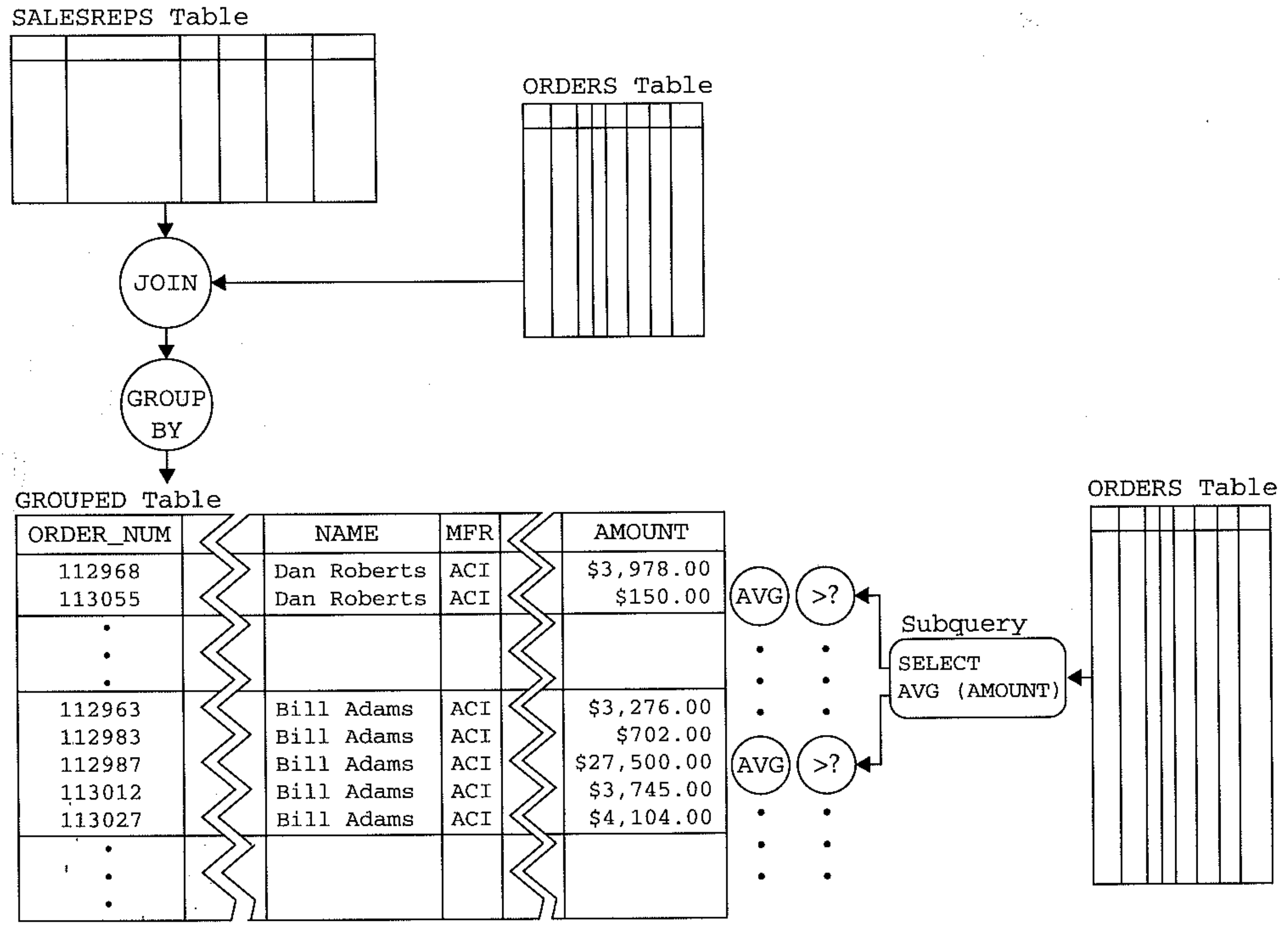
HAVING – Another Example

- List the salespeople whose average order size for products manufactured by ACI is higher than the overall average order size.

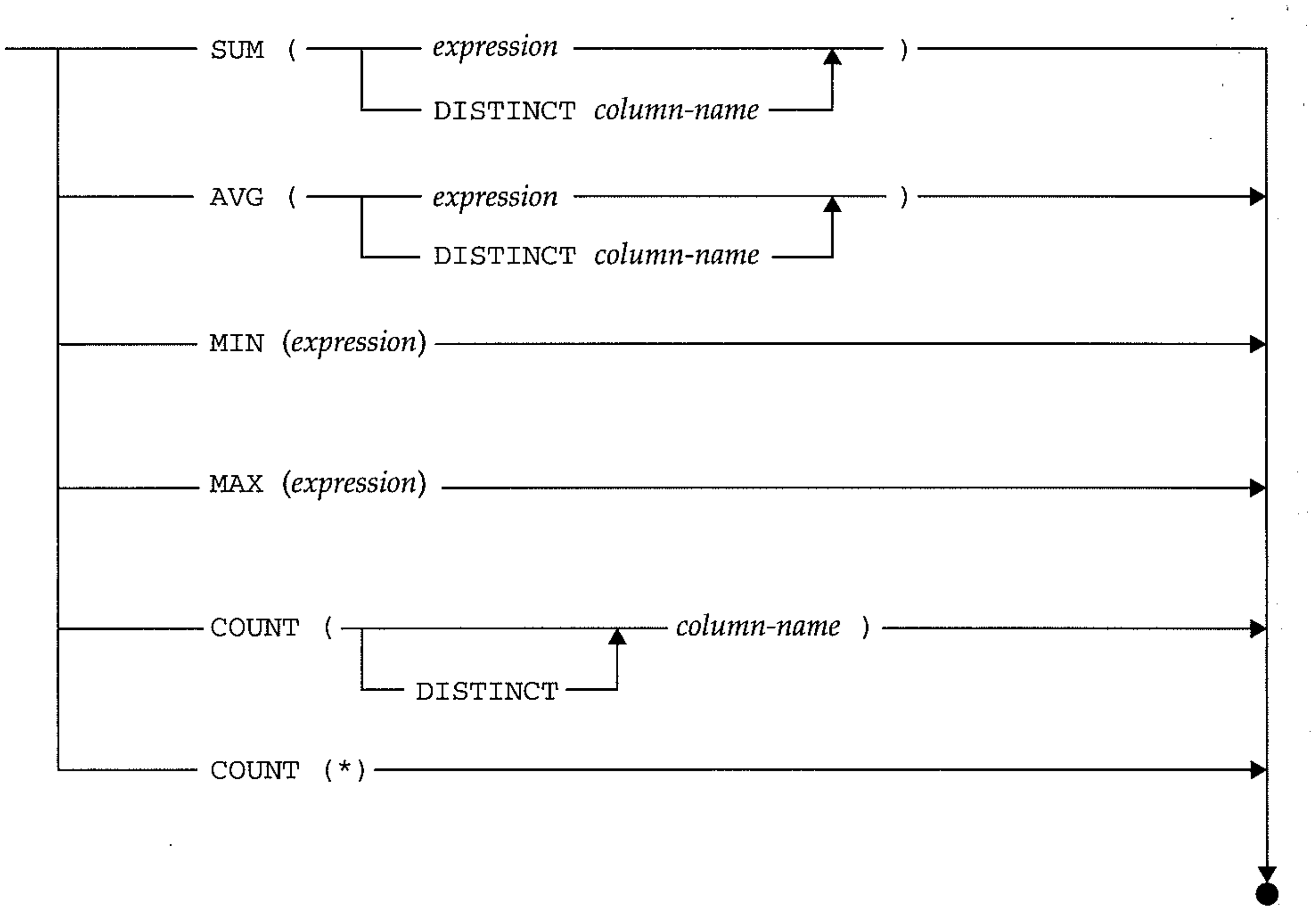
```
SELECT NAME, AVG(AMOUNT)
  FROM SALESREPS, ORDERS
 WHERE EMPL_NUM = REP
   AND MFR = 'ACI'
 GROUP BY NAME
HAVING AVG(AMOUNT) > (SELECT AVG(AMOUNT)
                        FROM ORDERS);

NAME          AVG(AMOUNT)
------------ ------------
Sue Smith      $15,000.00
Tom Snyder     $22,500.00
```

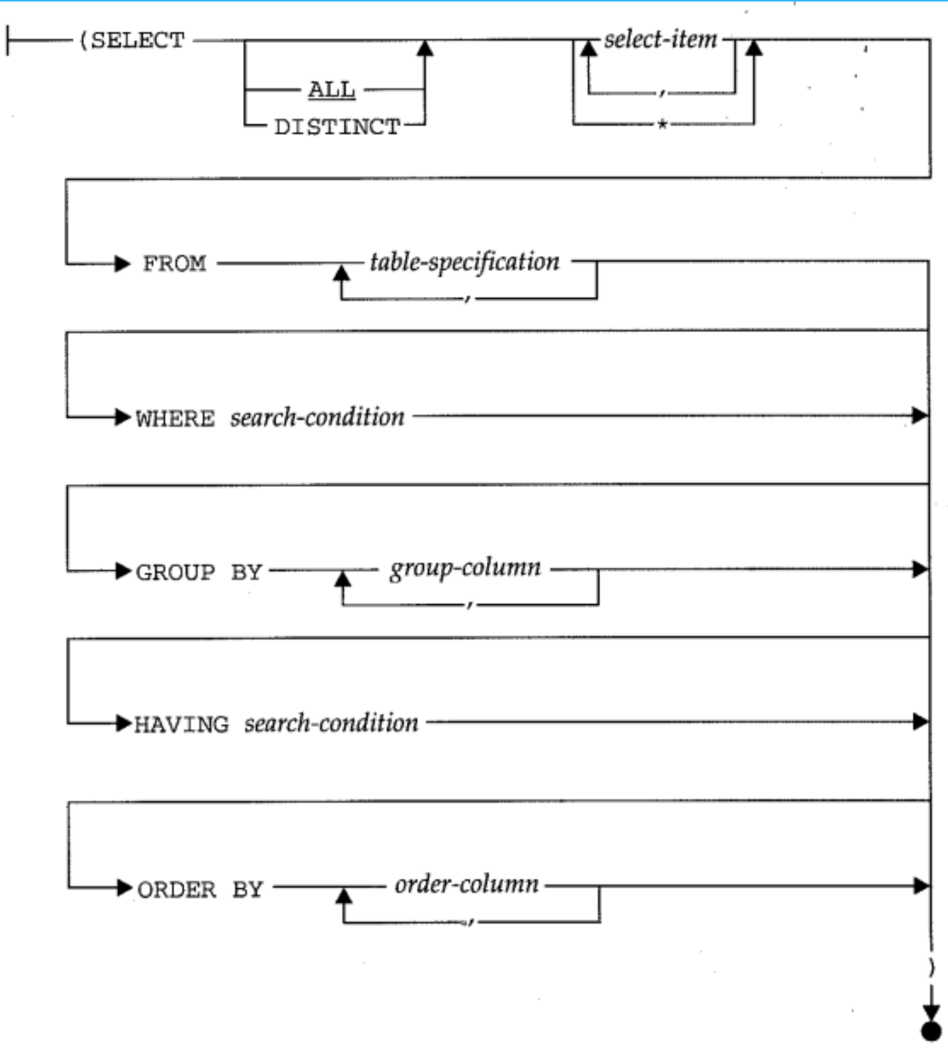
HAVING - Example



Summary – SQL Aggregate Queries



Summary – SQL Queries


(SELECT
ALL
DISTINCT
select-item
,
*
FROM
table-specification
,
WHERE search-condition
GROUP BY
group-column
,
HAVING search-condition
ORDER BY
order-column
,
)