International University, VNU-HCMC

School of Computer Science and Engineering

Lecture 6: SQL part 2

Instructor: Nguyen Thi Thuy Loan

nttloan@hcmiu.edu.vn, nthithuyloan@gmail.com
https://nttloan.wordpress.com/

Recap: Lecture 3

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

- SQL Data types
- Create database, tables
- Create constraints

  – Reading material: [RG] Chapters 3 and 5
  – Additional reading for practice: [GUW] Chapter 6

Acknowledgement

- The following slides have been created adapting the instructor material of the [RG] book provided by the authors Dr. Ramakrishnan and Dr. Gehrke.
- Other slides are referenced from Dr. Sudeepa Roy, Duke University and Prof. Jeffrey D. Ullman.

Purpose of the Lecture

- Extend students' understanding of SQL querying beyond basic commands.
- Introduce advanced SQL concepts, including joins, grouping and aggregation, nested queries, and set operations.
- Explain the semantics and conceptual evaluation strategy of SQL queries.
- Enable students to write complex queries involving multiple relations, conditions, and aggregations.

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Warm-up Question

- Given multiple related tables in a database (e.g., Students and Enrolled)
- How would you retrieve information that combines data from multiple tables, such as students who received a grade of "A" in a course?

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Outline

- More SQL
  - Extended algebra
  - Semantic
  - Joins
  - Group by and aggregates
  - Nested queries

The Extended Algebra

δ = eliminate duplicates from bags.

τ = sort tuples.

γ = grouping and aggregation.

Outerjoin: avoids "dangling tuples" = tuples that do not join with anything.

Duplicate Elimination

- R1 := δ(R2).
- R1 consists of one copy of each tuple that appears in R2 one or more times.

Example: Duplicate Elimination

R =

A	B
1	2
3	4
1	2

δ(R) =

A	B
1	2
3	4

Sorting

- R1 := τ_L (R2).
  - *L* is a list of some of the attributes of R2.
- R1 is the list of tuples of R2 sorted first on the value of the first attribute on *L*, then on the second attribute of *L*, and so on.
  - Break ties arbitrarily.
- τ is the only operator whose result is neither a set nor a bag.

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Example: Sorting

R = (

A	B
1	2
3	4
5	2

)

$\tau_B(R)$ = [(5,2), (1,2), (3,4)]

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Aggregation Operators

- Aggregation operators are not operators of relational algebra.
- Rather, they apply to entire columns of a table and produce a single result.
- The most important examples: SUM, AVG, COUNT, MIN, and MAX.

Example: Aggregation

R =

A	B
1	3
3	4
3	2

SUM(A) = 7
COUNT(A) = 3
MAX(B) = 4
AVG(B) = 3

Grouping Operator

- R1 := γ_L (R2). L is a list of elements that are either:
  1. Individual (*grouping*) attributes.
  2. AGG(A), where AGG is one of the aggregation operators and A is an attribute.
     - An arrow and a new attribute name rename the component.

Applying $\gamma_L(R)$

- Group *R* according to all the grouping attributes on list *L*.
  - That is: form one group for each distinct list of values for those attributes in *R*.
- Within each group, compute AGG(*A*) for each aggregation on list *L*.
- Result has one tuple for each group:
  1. The grouping attributes and
  2. Their group's aggregations.

Example: Grouping/Aggregation

R =

A	B	C
1	2	3
4	5	6
1	2	5

$\gamma_{A,B,AVG(C)->X}$ (R) = ??

First, group *R* by *A* and *B*:

A	B	C
1	2	3
1	2	5
4	5	6

Then, average *C* within groups:

A	B	X
1	2	4
4	5	6

Outerjoin

- Suppose we join $R \bowtie_C S$.
- A tuple of *R* that has no tuple of *S* with which it joins is said to be *dangling*.
  - Similarly for a tuple of *S*.
- Outerjoin preserves dangling tuples by padding them NULL.

Example: Outerjoin

R =

A	B
1	2
4	5

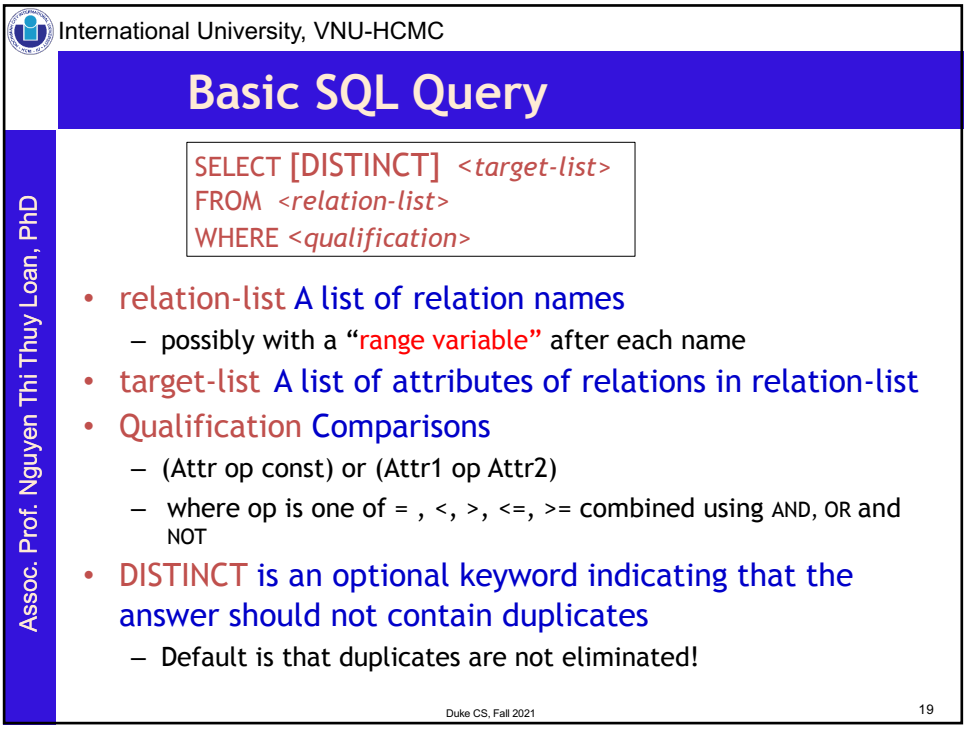
S =

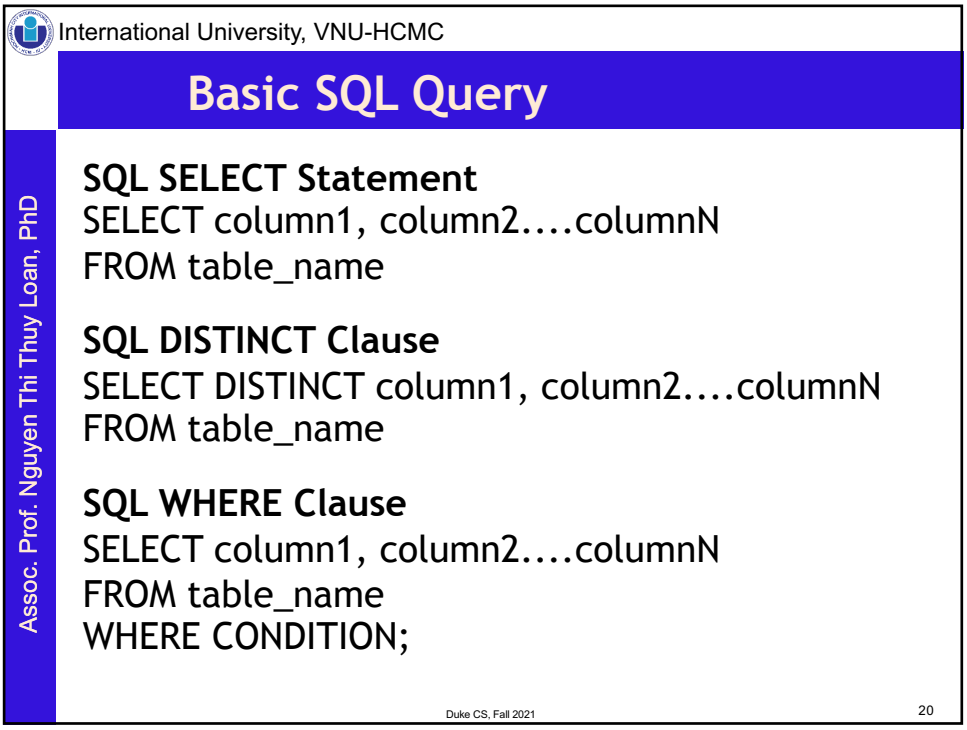
B	C
2	3
6	7

(1,2) joins with (2,3), but the other two tuples are dangling.

R OUTERJOIN S =

A	B	C
1	2	3
4	5	NULL
NULL	6	7


International University, VNU-HCMC
Basic SQL Query
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
SELECT [DISTINCT] <target-list>
FROM <relation-list>
WHERE <qualification>
• relation-list A list of relation names
– possibly with a "range variable" after each name
• target-list A list of attributes of relations in relation-list
• Qualification Comparisons
– (Attr op const) or (Attr1 op Attr2)
– where op is one of = , <, >, <=, >= combined using AND, OR and NOT
• DISTINCT is an optional keyword indicating that the answer should not contain duplicates
– Default is that duplicates are not eliminated!
Duke CS, Fall 2021
19


International University, VNU-HCMC
Basic SQL Query
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
SQL SELECT Statement
SELECT column1, column2....columnN
FROM table_name
SQL DISTINCT Clause
SELECT DISTINCT column1, column2....columnN
FROM table_name
SQL WHERE Clause
SELECT column1, column2....columnN
FROM table_name
WHERE CONDITION;
Duke CS, Fall 2021
20

Basic SQL Query

SQL AND/OR Clause

```
SELECT column1, column2....columnN
FROM table_name
WHERE CONDITION-1 {AND|OR} CONDITION-2;
```

SQL IN Clause

```
SELECT column1, column2....columnN
FROM table_name
WHERE column_name IN (val-1, val-2,...val-N);
```


Basic SQL Query

SQL BETWEEN Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name BETWEEN val-1 AND val-2;
```

SQL LIKE Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

Basic SQL Query

SQL ORDER BY Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

SQL GROUP BY Clause

```
SELECT Count(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```


Basic SQL Query

SQL COUNT Clause

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

SQL HAVING Clause

```
SELECT Count(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithmetic function condition);
```

Conceptual Evaluation Strategy


```
SELECT [DISTINCT] <target-list>
FROM <relation-list>
WHERE <qualification>
```

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of <*relation-list*>
  - Discard resulting tuples if they fail <*qualifications*>
  - Delete attributes that are not in <*target-list*>
  - If DISTINCT is specified, eliminate duplicate rows
- This strategy is probably the least efficient way to compute a query!
  - An optimizer will find more efficient strategies to compute the same answers

Conceptual Evaluation Strategy

Find sailor names who've reserved a boat #103

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Step 1: Form **cross product** of Sailor and Reserves

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/96
22	dustin	7	45	58	103	11/12/96
31	lubber	8	55	22	101	10/10/96
31	lubber	8	55	58	103	11/12/96
58	rusty	10	35	22	101	10/10/96
58	rusty	10	35	58	103	11/12/96

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Conceptual Evaluation Strategy


```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Step 2: Discard tuples that do not satisfy <qualification>

sid	sname	rating	age	sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45~~	~~22~~	~~101~~	~~10/10/96~~
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35	58	103	11/12/96

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

International University, VNU-HCMC

Conceptual Evaluation Strategy

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Step 3: Select the specified attribute(s)

sid	sname	rating	age	sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45~~	~~22~~	~~101~~	~~10/10/96~~
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35	58	103	11/12/96

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

A Note on "Alias"

- You can rename a table or a column temporarily by giving another name known as Alias
- Really needed only if the same relation appears twice in the FROM clause
  - sometimes used as a short-name
- The previous query can also be written as:

```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE S.sid=R.sid AND bid=103
OR
SELECT  sname
FROM    Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid
        AND bid=103
```

It is good style, however, to use alias (range variables) always!

Find sailor ids who've reserved at least one boat

```
SELECT  ????
FROM Sailors S, Reserves R
WHERE  S.sid=R.sid
```

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/19
58	103	11/12/19

Find sailor ids who've reserved at least one boat


```
SELECT  S.sid
FROM Sailors S, Reserves R
WHERE  S.sid=R.sid
```

- Would adding DISTINCT to this query make a difference?

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/19
58	103	11/12/19

Find sailor ids who've reserved at least one boat


```
SELECT  S.sid
FROM Sailors S, Reserves R
WHERE  S.sid=R.sid
```

- Would adding DISTINCT to this query make a difference?
  - Note that if there are multiple bids for the same sid, you get multiple output tuples for the same sid
  - Without distinct, you get them multiple times
- What is the effect of replacing *S.sid* by *S.sname* in the SELECT clause?
  - Would adding DISTINCT to this variant of the query make a difference even if one sid reserves at most one bid?

Sailor

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Reserves

sid	bid	day
22	101	10/10/19
58	103	11/12/19

Example

- To find all 18 year old students, we can write:

all attributes

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	19	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```


Querying Multiple Relations

- What does the following query compute?
- Given the following instances of Enrolled and Students:

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade='A'
```

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get: ??

Querying Multiple Relations

- What does the following query compute?
- Given the following instances of Enrolled and Students:

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade='A'
```

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get:

S.name	E.cid
Smith	Topology112

Joins

- Condition/Theta-Join
- Equi-Join
- Natural-Join
- (Left/Right/Full) Outer-Join

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/19
58	103	11/12/19

Condition/Theta Join

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid=R.sid and age >= 40
```

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

Form cross-product, discard rows that do not satisfy the condition

sid	sname	rating	age	sid	bid	day
22	dustin	7	45	22	101	10/10/19
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/19~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/19~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/19~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/19~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~58~~	~~103~~	~~11/12/19~~

sid	bid	day
22	101	10/10/19
58	103	11/12/19

Equi Join

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

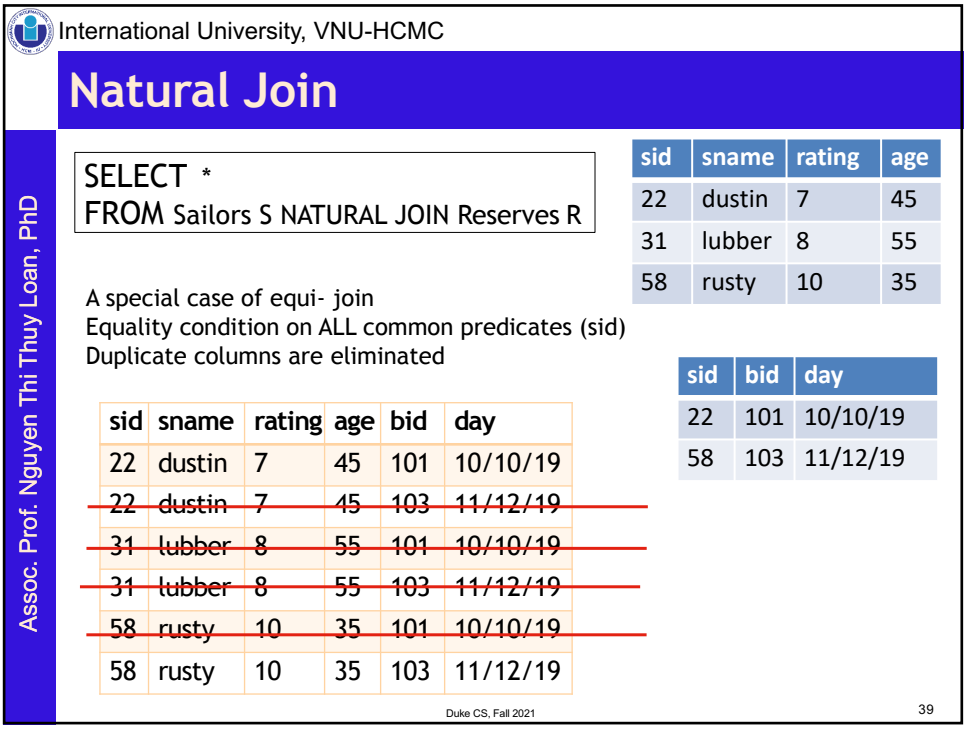
```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid=R.sid and age = 45
```

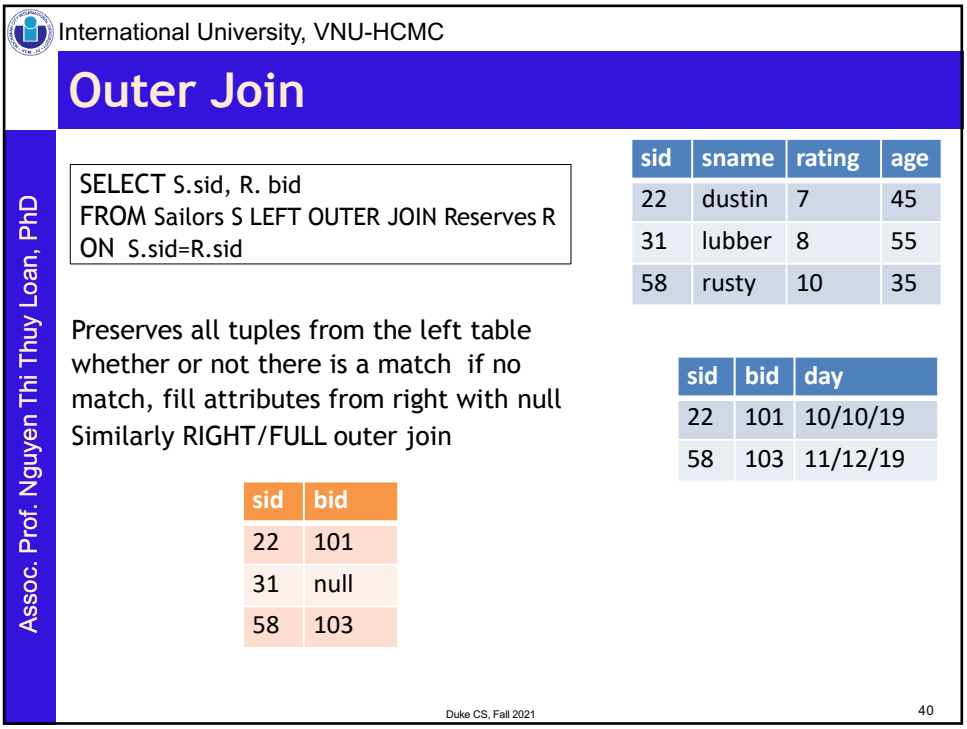
sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

A special case of theta join
Join condition only has equality predicate =

sid	sname	rating	age	sid	bid	Day
22	dustin	7	45	22	101	10/10/19
~~22~~	~~dustin~~	~~7~~	~~45~~	~~58~~	~~103~~	~~11/12/19~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~22~~	~~101~~	~~10/10/19~~
~~31~~	~~lubber~~	~~8~~	~~55~~	~~58~~	~~103~~	~~11/12/19~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~22~~	~~101~~	~~10/10/19~~
~~58~~	~~rusty~~	~~10~~	~~35~~	~~58~~	~~103~~	~~11/12/19~~

sid	bid	day
22	101	10/10/19
58	103	11/12/19


International University, VNU-HCMC
Natural Join
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
SELECT *
FROM Sailors S NATURAL JOIN Reserves R
A special case of equi- join
Equality condition on ALL common predicates (sid)
Duplicate columns are eliminated
sid sname rating age
22 dustin 7 45
31 lubber 8 55
58 rusty 10 35
sid bid day
22 101 10/10/19
58 103 11/12/19
sid sname rating age bid day
22 dustin 7 45 101 10/10/19
22 dustin 7 45 103 11/12/19
31 lubber 8 55 101 10/10/19
31 lubber 8 55 103 11/12/19
58 rusty 10 35 101 10/10/19
58 rusty 10 35 103 11/12/19
Duke CS, Fall 2021
39


International University, VNU-HCMC
Outer Join
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
SELECT S.sid, R. bid
FROM Sailors S LEFT OUTER JOIN Reserves R
ON S.sid=R.sid
Preserves all tuples from the left table whether or not there is a match if no match, fill attributes from right with null
Similarly RIGHT/FULL outer join
sid sname rating age
22 dustin 7 45
31 lubber 8 55
58 rusty 10 35
sid bid day
22 101 10/10/19
58 103 11/12/19
sid bid
22 101
31 null
58 103
Duke CS, Fall 2021
40

Expressions and Strings


```
SELECT S.age, age1=S.age-5, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching
- *Find triples (of ages of sailors and two fields defined by expressions) for sailors*
  - *whose names begin and end with B and contain at least three characters*
- LIKE is used for string matching. '_' stands for any one character and '%' stands for 0 or more arbitrary characters
  - You will need these often

Like Operator

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

- ```
  SELECT column1, column2, ...
  FROM table_name
  WHERE columnN LIKE pattern;
  ```
  ```

Example

- `SELECT * FROM Customers WHERE CustomerName LIKE 'a%';`
- `SELECT * FROM Customers WHERE CustomerName LIKE '%a';`
- `SELECT * FROM Customers WHERE CustomerName LIKE '_u%';`
- `SELECT * FROM Customers WHERE ContactName LIKE 'a%o';`

UNION Clause/Operator

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.
UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows

To use this UNION clause, each SELECT statement must have

- The same number of column
- The same data type and
- Have them in the same order

But they need not have to be in the same length.

Syntax of a UNION


```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION/ UNION ALL

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```


INTERSECT Clause/Operator

The SQL INTERSECT returns only common rows returned by the two SELECT statements.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

INTERSECT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```


International University, VNU-HCMC

EXCEPT Clause/Operator

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

The SQL EXCEPT returns only rows, which are not available in the second SELECT statement.

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

EXCEPT

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```



International University, VNU-HCMC

SQL set and bag operations

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

- UNION, EXCEPT, INTERSECR
  - Set semantics
  - Exactly like set $\cup$, -, and $\cap$ in relational algebra.
- UNION ALL, EXCEPT ALL, INTERSECT ALL
  - Bag semantics
  - Think of each row as having implicit count (the number of times it appears in the table)
  - Bag union: sum up the counts from two tables
  - Bag difference: proper-subtract the two counts
  - Bag intersection: take the minimum of the two counts.

Examples of bag operations

Bag1

fruit
apple
apple
orange

Bag2

fruit
apple
orange
orange

```
(SELECT * FROM Bag1)
UNION ALL
(SELECT * FROM Bag2);
```

fruit
apple
apple
orange
apple
orange
orange

```
(SELECT * FROM Bag1)
EXCEPT ALL
(SELECT * FROM Bag2);
```

fruit
apple

```
(SELECT * FROM Bag1)
INTERSECT ALL
(SELECT * FROM Bag2);
```

fruit
apple
orange

Example: Find sid's of sailors who've reserved a red or a green boat

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND
(B.color='red' OR B.color='green')
```

- Assume a Boats relation
- If we replace OR by AND in the first version, what do we get?
- Also available: EXCEPT (What do we get if we replace UNION by EXCEPT?)

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
        AND B.color='red'
UNION
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
        AND B.color='green'
```

Sailors (sid,sname,rating,age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

INTERSECT Clause/Operator

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Find sname's of sailors who've reserved a red and a green boat

Sailors (sid,sname,rating,age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

INTERSECT Clause/Operator

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Find sid's of sailors who've reserved a red and a green boat

```
SELECT S.sid
FROM Sailors S, Boats B1, Reserves R1,
     Boats B2, Reserves R2
WHERE S.sid=R1.sid AND R1.bid=B1.bid
  AND S.sid=R2.sid AND R2.bid=B2.bid
  AND (B1.color='red' AND B2.color='green')
```

- INTERSECT: Can be used to compute the intersection of any two union-compatible sets of tuples.
  - Included in the SQL/92 standard, but some systems don't support it

Key field!

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
      AND B.color='red'
INTERSECT
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
      AND B.color='green'
```

Nested Queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries must be enclosed within parentheses.

In and Exists Operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

- The EXISTS operator is used to test for the existence of any record in a subquery.
- The EXISTS operator returns TRUE if the subquery returns one or more records.

In syntax


```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);

SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```


Exists syntax


```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE
condition);
```

IN subqueries

User(uid, name, age, pop)

- x IN $(subquery)$ checks if x is in the result of $subquery$
- Example: users (all columns) at the same age as (some) Long

Let's first try without sub-queries

```
SELECT * FROM
User
WHERE age IN (SELECT age
                FROM User
                WHERE name = 'Long');
```

You can use NOT IN too

EXISTS subqueries

User(uid, name, age, pop)

- EXISTS $(subquery)$ checks if the result of $subquery$ is non-empty
- Example: users at the same age as (some) Bart

```
SELECT *
FROM User AS u
WHERE EXISTS (SELECT * FROM User
                WHERE name = 'Bart'
                AND age = u.age);
```

- This happens to be a correlated subquery - a subquery that references tuple variables in surrounding queries

You can use NOT EXISTS too

- How about the previous one with "IN"?

Find names of sailors who've reserved boat #103

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
                FROM Reserves R
                WHERE R.bid=103)
```

- A very powerful feature of SQL
  - a WHERE/FROM/HAVING clause can itself contain an SQL query
- To find sailors who've not reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a nested loops evaluation
  - For each Sailors tuple, check the qualification by computing the subquery

Nested Queries with Correlation

Find names of sailors who've reserved boat #103:

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
              FROM Reserves R
              WHERE R.bid=103 AND S.sid=R.sid)
```

- EXISTS is another set comparison operator, like IN
- Illustrates why, in general, subquery must be re-computed for each Sailors tuple

Nested Queries with Correlation

Find names of sailors who've reserved boat #103:

```
SELECT  S.sname
FROM Sailors S
WHERE     UNIQUE (SELECT  R.bid
                  FROM  Reserves R
                   WHERE R.bid=103 AND S.sid=R.sid)
```

- If UNIQUE is used, and * is replaced by *R.bid*, finds sailors with at most one reservation for boat #103
  - UNIQUE checks for duplicate tuples

Another example

User(uid, name, pop)
Member(uid, gid)
Group(gid, name)

```
• SELECT * FROM User u
  WHERE EXISTS
      (SELECT * FROM Member m
       WHERE m.uid = u.uid
       AND EXISTS
          (SELECT * FROM Member n
           WHERE n.uid = u.uid
           AND n.gid <> m.gid));
```

- What does this query return?
- Users who join at least two groups

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE
- Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL, *op* IN
  - where op : >, <, =, <=, >=
- Find sailors whose rating is greater than that of some sailor called Horatio

- similarly ALL

```
SELECT *
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                      FROM Sailors S2
                      WHERE S2.sname='Horatio')
```


Aggregate Operators

Check yourself:
What do these queries compute?

```
SELECT COUNT (*)
FROM Sailors S
```

```
COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)
```

single column

```
SELECT AVG (S.age)
FROM Sailors S
WHERE S.rating=10
```

```
SELECT S.sname
FROM Sailors S
WHERE S.rating= (SELECT MAX(S2.rating)
                 FROM Sailors S2)
```

```
SELECT COUNT (DISTINCT S.rating)
FROM Sailors S
WHERE S.sname='Bob'
```

```
SELECT AVG ( DISTINCT S.age)
FROM Sailors S
WHERE S.rating=10
```

Motivation for Grouping

- So far, we've applied aggregate operators to all (qualifying) tuples
  - Sometimes, we want to apply them to each of several groups of tuples
- Consider: Find the age of the youngest sailor for each rating level
  - In general, we don't know how many rating levels exist, and what the rating values for these levels are!
  - Suppose we know that rating values go from 1 to 10; we can write 10 queries that look like this (need to replace i by num):

For *i* = 1, 2, ... , 10:

```
SELECT MIN (S.age)
FROM Sailors S
WHERE S.rating = i
```


Queries With GROUP BY and HAVING


```
SELECT [DISTINCT] target
FROM relation-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

First go over the examples in the following slides. Then come back to this slide and study yourself

- The target-list contains
  - (i) attribute names
  - (ii) terms with aggregate operations (e.g., MIN (S.age))
- The attribute list (i) must be a subset of grouping-list
  - Intuitively, each answer tuple corresponds to a group, and these attributes must have a single value per group
  - Here a group is a set of tuples that have the same value for all attributes in grouping-list

Conceptual Evaluation

- The cross-product of relation-list is computed
- Tuples that fail qualification are discarded
- `Unnecessary' fields are deleted
- The remaining tuples are partitioned into groups by the value of attributes in grouping-list
- The group-qualification is then applied to eliminate some groups
- Expressions in group-qualification must have a single value per group
  - In effect, an attribute in group-qualification that is not an argument of an aggregate op also appears in grouping-list
  - like "…GROUP BY bid, sid HAVING bid = 3"
- One answer tuple is generated per qualifying group

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Sailors instance:

sid	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING COUNT (*) > 1
```

Sailors instance:

sid	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Answer relation:

rating	minage
3	25.5
7	35.0
8	25.5

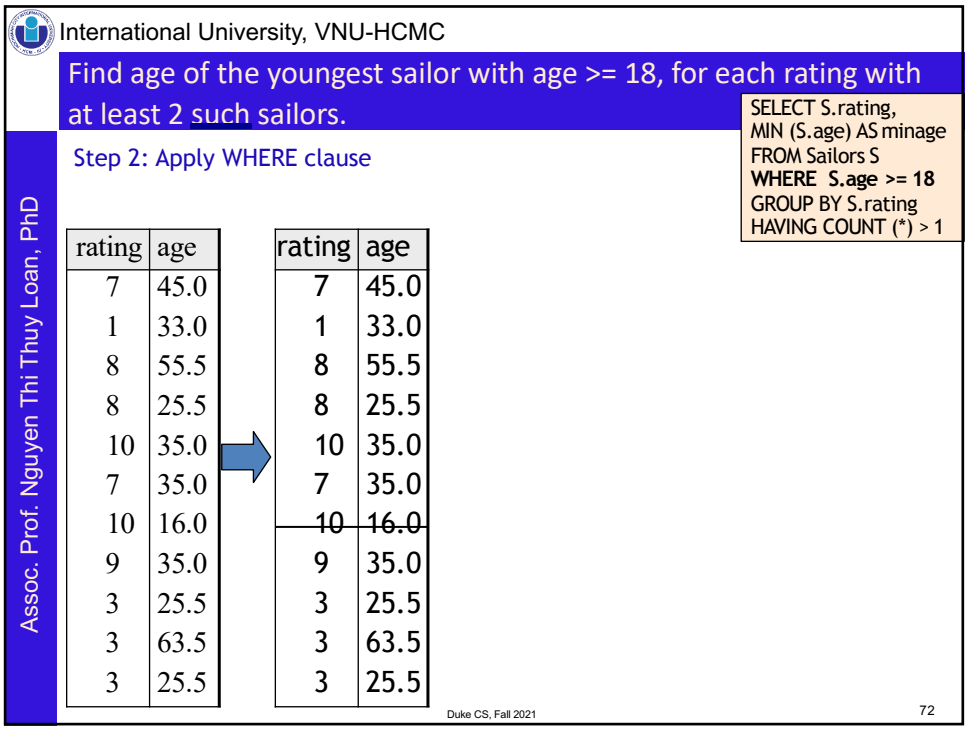
Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

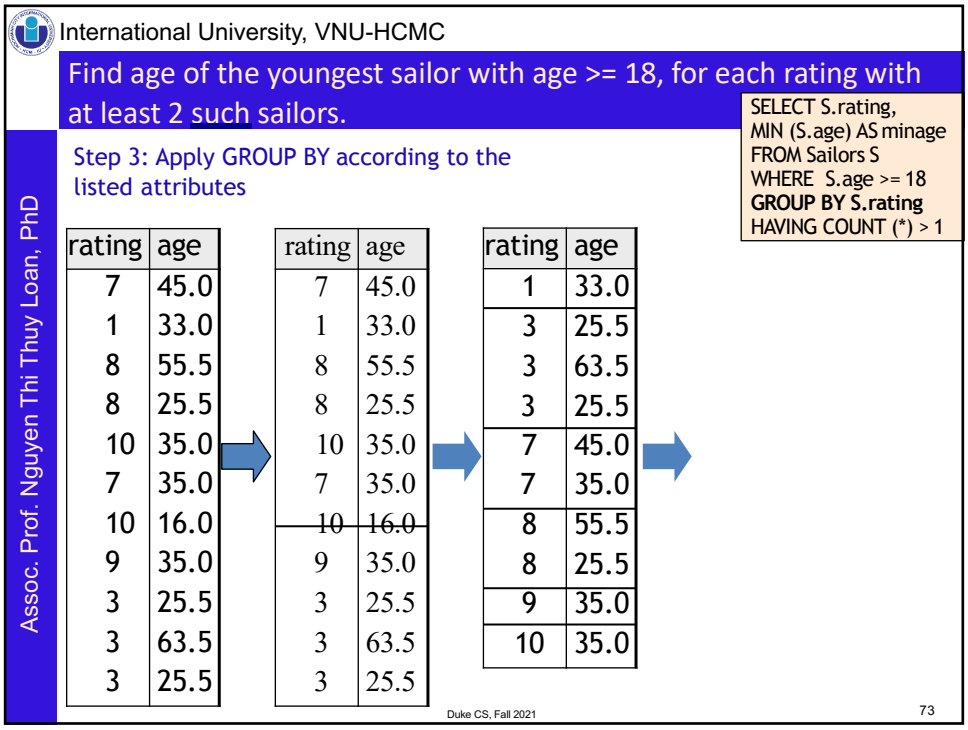
Assoc. Prof. Nguyen Thi Thuy Loan, PhD

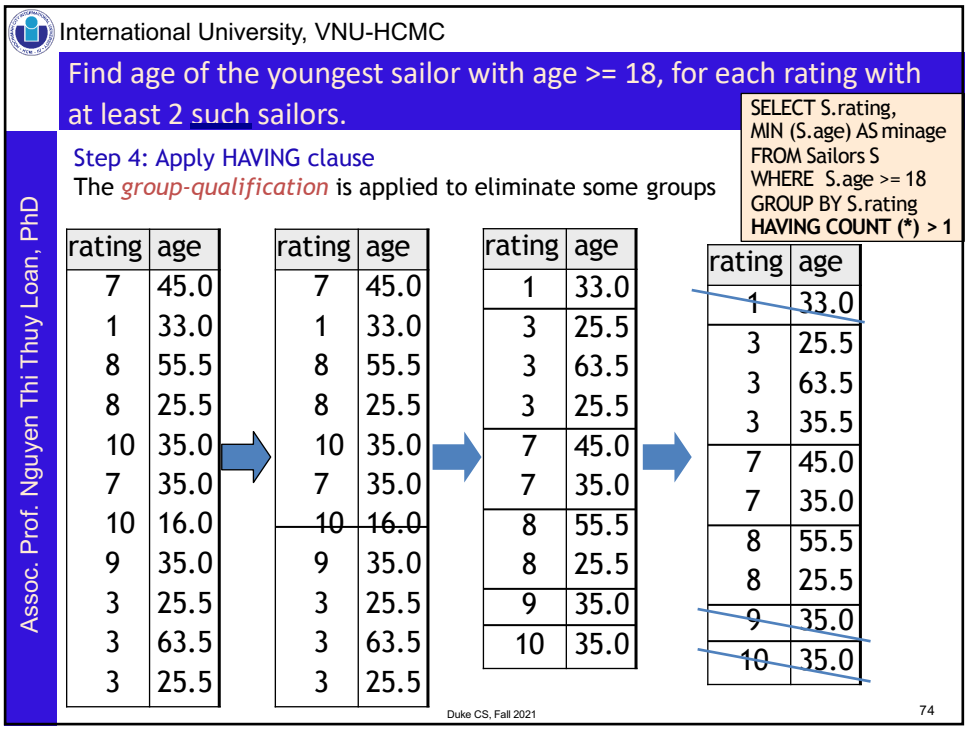
```
SELECT S.rating,
MIN (S.age) AS minage
FROM Sailors S
WHERE  S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

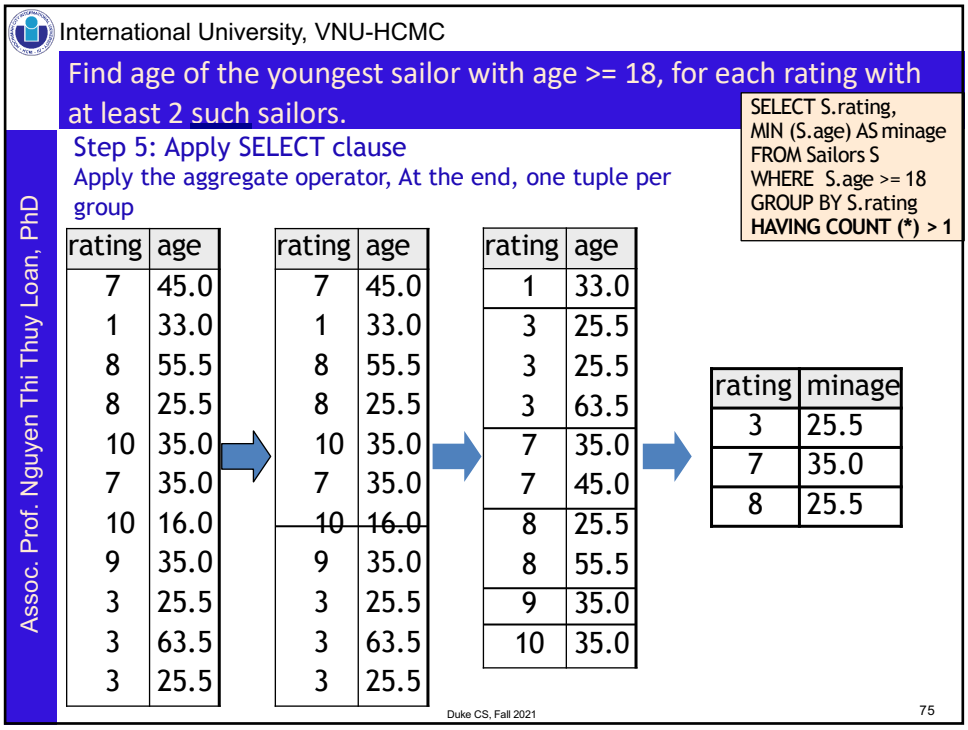
Step 1: Form the cross product: FROM clause
(some attributes are omitted for simplicity)

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5


International University, VNU-HCMC
Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.
Step 2: Apply WHERE clause
SELECT S.rating, MIN (S.age) AS minage FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating HAVING COUNT (*) > 1
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
Duke CS, Fall 2021
72


International University, VNU-HCMC
Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.
Step 3: Apply GROUP BY according to the listed attributes
SELECT S.rating, MIN (S.age) AS minage FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating HAVING COUNT (*) > 1
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
1 33.0
3 25.5
3 63.5
3 25.5
7 45.0
7 35.0
8 55.5
8 25.5
9 35.0
10 35.0
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
Duke CS, Fall 2021
73


International University, VNU-HCMC
Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.
SELECT S.rating, MIN (S.age) AS minage FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating HAVING COUNT (*) > 1
Step 4: Apply HAVING clause
The group-qualification is applied to eliminate some groups
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
1 33.0
3 25.5
3 63.5
3 25.5
7 45.0
7 35.0
8 55.5
8 25.5
9 35.0
10 35.0
rating age
1 33.0
3 25.5
3 63.5
3 35.5
7 45.0
7 35.0
8 55.5
8 25.5
9 35.0
10 35.0
Duke CS, Fall 2021
74


International University, VNU-HCMC
Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.
SELECT S.rating, MIN (S.age) AS minage FROM Sailors S WHERE S.age >= 18 GROUP BY S.rating HAVING COUNT (*) > 1
Step 5: Apply SELECT clause
Apply the aggregate operator, At the end, one tuple per group
Assoc. Prof. Nguyen Thi Thuy Loan, PhD
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
7 45.0
1 33.0
8 55.5
8 25.5
10 35.0
7 35.0
10 16.0
9 35.0
3 25.5
3 63.5
3 25.5
rating age
1 33.0
3 25.5
3 25.5
3 63.5
7 35.0
7 45.0
8 25.5
8 55.5
9 35.0
10 35.0
rating minage
3 25.5
7 35.0
8 25.5
Duke CS, Fall 2021
75

Top Clause

The SQL TOP clause is used to fetch a TOP N number or X percent records from a table.

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE [condition];
```

MySQL syntax

```
SELECT column_name(s)
FROM table_name
WHERE [condition]
LIMIT number;
```


Demo database

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT TOP 3 * FROM Customers;
```

Order by syntax


```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

```
SELECT *
FROM Customers
ORDER BY Country;
```


Case syntax


```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    …
     WHEN conditionN THEN resultN
     ELSE result
END;
```

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Demo Database

OrderDetails table

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Case example

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The
quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The
quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

Example

- The following SQL will order the customers by City. However, if City is NULL, then order by Country

```
SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
    WHEN City IS NULL THEN Country
    ELSE City
END);
```


"WITH" clause - very useful!

- You will find "WITH" clause very useful!

```
WITH Temp1 AS
        (SELECT ..... ..),
        Temp2 AS
        (SELECT ..... ..)
SELECT X, Y
FROM TEMP1, TEMP2
WHERE....
```

User(uid, name, age, pop)

- Can simplify complex nested queries

Example: users at the same age as (some) Long

```
WITH LongAge AS
  (SELECT age
  FROM User
   WHERE name = 'Long')
SELECT U.uid, U.name, U.age, U.pop
FROM User U, LongAge L
WHERE U.age = L.age
```

WITH clause not really needed for this query!

WITH CLAUSE


```
WITH Cus_CTE (sid, sname, sage, searning)
AS
-- Define the CTE query.
(
    SELECT  id, name, age, salary AS earning
    FROM CUSTOMERS
    WHERE ADDRESS IS NOT NULL
)
-- Define the outer query referencing the CTE name.
SELECT searning, COUNT(sname) AS Totalname
FROM Cus_CTE
GROUP BY searning
ORDER BY searning
```

Customer(id, name, age, salary, address)

Rewriting INTERSECT Queries Using IN

Find sid's of sailors who've reserved both a red and a green boat:

```
SELECT  S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
        AND S.sid IN (SELECT  S2.sid
                FROM Sailors S2, Boats B2, Reserves R2
                    WHERE S2.sid=R2.sid AND R2.bid=B2.bid
                      AND  B2.color='green')
```

- Similarly, **EXCEPT** queries re-written using **NOT IN**.
- To find names (not sid's) of Sailors who've reserved both red and green boats, just replace S.sid by S.sname in SELECT clause

Division in SQL

Find sailors who've reserved all boats.

- Option 1:
- Option 2: Let's do it the hard way, without EXCEPT:

option 1

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS
        ((SELECT B.bid
          FROM Boats B)  EXCEPT  (SELECT  R.bid
                                  FROM Reserves R
                                  WHERE    R.sid=S.sid))
```

option 2

```
SELECT  S.sname     Sailors S such that ...
FROM Sailors S
WHERE    NOT EXISTS (SELECT  B.bid  there is no boat B...
                     FROM Boats B
                     WHERE   NOT EXISTS (SELECT  R.bid   ...without ...
                                         WHERE  R.bid=B.bid
                                           AND R.sid=S.sid))
...a Reserves tuple showing S reserved B
```


Example

- Find the names of students who've enrolled a Database and AI (Artificial Intelligence) course.
- Assume a Student database:

Students (sid, sname, address, age)
Enrolled (sid, cid, year, semester)
Courses (cid, cname, credit)

Find name and age of the oldest sailor(s)

- The first query is illegal!
  - Recall the semantic of GROUP BY
- The third query is equivalent to the second query
  - and is allowed in the SQL/92 standard, but is not supported in some systems

```
SELECT S.sname, MAX (S.age)
FROM Sailors S
```

```
SELECT S.sname, S.age
FROM Sailors S
WHERE S.age =
        (SELECT MAX (S2.age)
        FROM Sailors S2)
```

```
SELECT S.sname, S.age
FROM Sailors S
WHERE (SELECT MAX (S2.age)
        FROM Sailors S2) = S.age
```


Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors and with every sailor under 60.

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

```
SELECT S.rating,
        MIN (S.age)  AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1 AND
EVERY (S.age <60)
```

rating	minage
7	35.0
8	25.5

What is the result of changing EVERY to ANY?

Find age of the youngest sailor with age >= 18, for each rating with at least 2 sailors between 18 and 60.

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18 AND S.age <= 60
GROUP BY    S.rating
HAVING COUNT (*) > 1
```

Sailors instance:

sid	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Answer relation:

rating	minage
3	25.5
7	35.0
8	25.5

For each red boat, find the number of reservations for this boat

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

For each red boat, find the number of reservations for this boat


```
SELECT B.bid, COUNT (*) AS scount
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
GROUP BY B.bid
```

- Grouping over a join of three relations.
- What do we get if we remove B.color='red' from the WHERE clause and add a HAVING clause with this condition?
- What if we drop Sailors and the condition involving S.sid?

Find age of the youngest sailor with age > 18, for each rating with at least 2 sailors (of any age)

Sailors (sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

International University, VNU-HCMC

Find age of the youngest sailor with age > 18, for each rating with at least 2 sailors (of any age)

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age > 18
GROUP BY S.rating
HAVING 1 < (SELECT COUNT (*)
            FROM Sailors S2
            WHERE S.rating=S2.rating)
```

- Shows HAVING clause can also contain a subquery.
- Compare this with the query where we considered only ratings with 2 sailors over 18!
- What if HAVING clause is replaced by:
  - HAVING COUNT(*) >1

International University, VNU-HCMC

Find those ratings for which the average age is the minimum over all ratings

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

- Aggregate operations cannot be nested!

```
SELECT S.rating
FROM Sailors S                          WRONG
WHERE S.age = (SELECT MIN (AVG (S2.age))
               FROM Sailors S2)
```

- Correct solution (in SQL/92):

```
SELECT Temp.rating, Temp.avgage
FROM (SELECT S.rating, AVG (S.age) AS avgage
      FROM Sailors S
      GROUP BY S.rating) AS Temp
WHERE Temp.avgage = (SELECT MIN (Temp.avgage)
                     FROM Temp)
```

Triggers

- Trigger: procedure that starts automatically if specified changes occur to the DBMS
- Three parts:
  - Event (activates the trigger)
  - Condition (tests whether the triggers should run)
  - Action (what happens if the trigger runs)

```
CREATE TRIGGER youngSailorUpdate  AFTER INSERT
 ON SAILORS REFERENCING NEW TABLE NewSailors
  FOR EACH STATEMENT
  INSERT INTO YoungSailors(sid, name, age, rating)
        SELECT sid, name, age, rating
        FROM NewSailors N
        WHERE N.age <= 18
```


Nulls and Views in SQL

Null Values

- Field values in a tuple are sometimes
  - unknown, e.g., a rating has not been assigned, or
  - inapplicable, e.g., no spouse's name
  - SQL provides a special value null for such situations.

Standard Boolean 2-valued logic

- True = 1, False = 0
- Suppose X = 5
  - (X < 100) AND (X >= 1) is T ∧T = T
  - (X > 100) OR (X >= 1) is F ∨T = T
  - (X > 100) AND (X >= 1) is F ∧T = F
  - NOT(X = 5) is ¬T = F
- Intuitively,
  - T = 1, F = 0
  - For V1, V2 ∈{1, 0}
  - V1 ∧V2 = MIN (V1, V2)
  - V1 ∨V2 = MAX(V1, V2)
  - ¬(V1) = 1 - V1

International University, VNU-HCMC

2-valued logic does not work for nulls

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

- Suppose rating = null, X = 5
- Is rating>8 true or false?
- What about AND, OR and NOT connectives?
  - (rating > 8) AND (X = 5)?
- What if we have such a condition in the WHERE clause?

International University, VNU-HCMC

3-Valued Logic For Null

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

- TRUE (= 1), FALSE (= 0), UNKNOWN (= 0.5)
  - unknown is treated as 0.5
- Now you can apply rules from 2-valued logic!
  - For V1, V2 ∈{1, 0, 0.5}
  - V1 ∧V2 = MIN (V1, V2)
  - V1 ∨V2 = MAX(V1, V2)
  - ¬(V1) = 1 - V1
- Therefore,
  - NOT UNKNOWN = UNKNOWN
  - UNKNOWN OR TRUE = TRUE
  - UNKNOWN AND TRUE = UNKNOWN
  - UNKNOWN AND FALSE = FALSE
  - UNKNOWN OR FALSE = UNKNOWN

New issues for Null

- The presence of null complicates many issues. E.g.:
  - Special operators needed to check if value IS/IS NOT NULL
  - Be careful!
  - "WHERE X = NULL" does not work!
  - Need to write "WHERE X IS NULL"
- Meaning of constructs must be defined carefully
  - e.g., WHERE clause eliminates rows that don't evaluate to true
  - So not only FALSE, but UNKNOWNs are eliminated too
  - very important to remember!
- But NULL allows new operators (e.g. outer joins)
- Arithmetic with NULL
  - all of +, -, *, / return null if any argument is null
- Can force "no nulls" while creating a table
  - sname char(20) NOT NULL
  - primary key is always not null

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
- SELECT count(*) from R1?
- SELECT count(rating) from R1?

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
- SELECT count(*) from R1?
- SELECT count(rating) from R1?
- Answer: 3 for both

International University, VNU-HCMC

Aggregates with NULL

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
- SELECT count(*) from R1?
- SELECT count(rating) from R1?
- Answer: 3 for both

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

- What do you get for
- SELECT count(*) from R2?
- SELECT count(rating) from R2?

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
- SELECT count(*) from R1?
- SELECT count(rating) from R1?
- Answer: 3 for both

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

- What do you get for
- SELECT count(*) from R2?
- SELECT count(rating) from R2?
- Answer: First 3, then 2

Aggregates with NULL

- COUNT, SUM, AVG, MIN, MAX (with or without DISTINCT)
  - Discards null values first
  - Then applies the aggregate
  - Except count(*)
- If only applied to null values, the result is null

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

- SELECT sum(rating) from R2?

sid	sname	rating	age
22	dustin	null	45
31	lubber	null	55
58	rusty	null	35

R3

- SELECT sum(rating) from R3?

Aggregates with NULL

- COUNT, SUM, AVG, MIN, MAX (with or without DISTINCT)
  - Discards null values first
  - Then applies the aggregate
  - Except count(*)
- If only applied to null values, the result is null

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

- SELECT sum(rating) from R2?
- Answer: 17

sid	sname	rating	age
22	dustin	null	45
31	lubber	null	55
58	rusty	null	35

R3

- SELECT sum(rating) from R3?
- Answer: null

Overview: General Constraints

- Useful when more general ICs than keys are involved
- There are also ASSERTIONS to specify constraints that span across multiple tables
- There are TRIGGERS too: procedure that starts automatically if specified changes occur to the DBMS
  - see additional slides at the end

```
CREATE TABLE Sailors
    ( sid INTEGER,
    sname CHAR(10),
    rating INTEGER,
    age REAL,
    PRIMARY KEY (sid),
    CHECK (rating >= 1
        AND rating <= 10 )
```

```
CREATE TABLE Reserves
    ( sname CHAR(10),
    bid INTEGER,
    day DATE,
    PRIMARY KEY (bid,day),
    CONSTRAINT noInterlakeRes
    CHECK (`Interlake' <>
            ( SELECT B.bname
            FROM Boats B
            WHERE B.bid=bid)))
```

VIEWS

Views, which are a type of virtual tables. Views can be created from a single table, multiple tables or another view.

```
CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS
WHERE age <=30

SELECT * FROM CUSTOMERS_VIEW
```


VIEWS

Views can be dropped using the DROP VIEW command

```
DROP VIEW view_name

DROP VIEW CUSTOMERS_VIEW;
```

Views and Security: Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s)

- The above view hides customers “salary” from CUSTOMERS

International University, VNU-HCMC

UPDATING A VIEW

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

A view can be updated under certain conditions which are given below –

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.

International University, VNU-HCMC

Can create a new table from a query on other tables too

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

SELECT… INTO.... FROM.... WHERE

```
SELECT S.name, E.grade
INTO YoungActiveStudents
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age<21
```

Summary

- SQL has a huge number of constructs and possibilities
  - You need to learn and practice it on your own
  - Given a problem, you should be able to write a SQL query and verify whether a given one is correct
- Pay attention to NULLs

International University, VNU-HCMC

Assoc. Prof. Nguyen Thi Thuy Loan, PhD

Thank you for your attention!