# SQL: The Query Language
# Part IV
# Aggregate Queries

**R&G - Chapter 5**

# Simple and Nested SQL Queries

- **Simple SQL Queries**

  SELECT $A_1$, $A_2$, ..., $A_n$
  FROM $r_1$, $r_2$, ..., $r_m$
  WHERE $P$

- **Nested SQL queries**

  SELECT $A_1$, $A_2$, ..., $A_n$
  FROM $r_1$, $r_2$, ..., $r_m$
  WHERE (NOT)IN|(NOT) EXISTS|Op ANY/ALL
  (SELECT $A_1$, $A_2$, ..., $A_n$
  FROM $r_1$, $r_2$, ..., $r_m$
  WHERE $P$)

# Today's plan

- **Aggregate queries**

# Aggregate Function

- **Aggregate functions: significant extension of relational algebra.**
- **Behavior of Aggregate Functions:**
    - Operates on a single column
    - Input: a collection of values.
    - Output: a single value.
    - Used ONLY in the SELECT list and in the HAVING clause.

# Types of SQL Aggregate Functions

- **SUM**
- **AVG**
- **MIN**
- **MAX**
- **COUNT**

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

*single column*

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

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

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

# Input to Aggregate Function

- **SUM and AVG :**
  ➔ Only accept numerical values as input.

- **MIN , MAX and COUNT**
  ➔ Accept both numerical and categorical values at input.

❑ Each function eliminates NULL values and operates on non- null values.

# Aggregate Function Continued…

- Accepts:

  ✓ DISTINCT: consider only distinct values of the argument expression.

  ✓ ALL : consider all values including all duplicates.

Example: SELECT COUNT(DISTINCT column_name)

# Example of DISTINCT and ALL

| sno | salary |
|---|---|
| SL100 | 30000 |
| SL101 | 10000 |
| SL102 | 10000 |
| SL103 | 20000 |

Staff

Query 1:

```
SELECT  AVG(DISTINCT salary) AS avg_sal
FROM  Staff;
```

Result:

| avg_sal |
|---|
| 20000 |

= (30K+10K+20K)/3

Query 2:

```
SELECT  AVG(ALL salary) AS avg_sal
FROM  Staff;
```

Result:

| avg_sal |
|---|
| 17500 |

= (30K+10K+10K+20K)/4

# Example of Aggregate Queries

**Query: Find name and age of the oldest sailor(s)**

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

**Solution 1**

# Example of Aggregate Queries

**Query: Find name and age of the oldest sailor(s)**

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

**Solution 2**

Question: is Solution 2 correct?

# Answer

**Query: Find name and age of the oldest sailor(s)**

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

- MySQL:  return the name of the 1st tuple, and the maximum age
- SQL server and Oracle: return error message.

- Rule: For aggregate queries without GROUP BY clause, DO NOT put non-aggregate attributes and aggregate functions together in SELECT clause.

# More examples

**Find the names of sailors who are older than the oldest sailor of rating 10**

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.age >
        (SELECT  MAX (S2.age)
         FROM  Sailors S2
         WHERE S2.rating =10)
```

# GROUP BY and HAVING

- Sometimes, we want to ask for groups of tuples.
- The query description normally starts with "for each…".
- Consider: *for each rating level, find the age of the youngest sailor.*
  - IF we know that rating values go from 1 to 10; we can write 10 queries that look like this (!):

    For $i$ = 1, 2, … , 10:

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

  - In general, we don't know how many rating levels exist, and what the rating values for these levels are!

# GROUP BY Clause

- To generate values for a column based on groups of rows, use aggregate functions in SELECT statements with the GROUP BY clause

```
SELECT    attr1, attr2, ... attrk, aggregate_function
FROM      tables
[WHERE    qualification]
GROUP BY attr1, attr2, ... attrn;
```

*: the attributes $attr_1, attr_2, \ldots attr_k$ in SELECT clause that do not have any aggregation function must appear in GROUP BY clause (k<=n)

# Tips of GROUP BY Queries

- If the query description starts with "for each **X**".
- Write the GROUP BY queries

```
SELECT    X, ...
FROM      …
[WHERE    …]
GROUP BY X;
```

# Group By Examples

For each rating, find the average age of the sailors

```
SELECT  S.rating,  AVG (S.age)
FROM  Sailors S
GROUP BY  S.rating
```

For each rating, find the age of the youngest sailor with age ≥ 18

```
SELECT  S.rating,  MIN (S.age)
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
```

# Grouping over a join of two relations.

Find the number of reservations for each **red** boat.

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

COUNT(*) AS numres: name the output count values under a column named "numres"

# GROUP BY Examples

For each rating, find the name and age of the oldest sailor

```
SELECT  S.name,  MAX (S.age)
FROM  Sailors S
GROUP BY  S.rating
```

Is this query correct?

This query is not correct, as the non-aggregate attribute name does not appear in GROUP BY Class

How to fix?

# Queries With GROUP BY and HAVING

SELECT $attr_1$, $attr_2$, ... $attr_k$, aggregate_function
FROM ***tables***
[WHERE ***qualification***]
GROUP BY $attr_1$, $attr_2$, ... $attr_n$
HAVING *group-qualification;*

- HAVING clause restricts which group-rows are returned in the result set
- NO GROUP BY, NO HAVING;
- Attributes in *group-qualification* of HAVING clause must be either an aggregate op or appear in the *grouping-list* of GROUP BY clause

# Query: For each age group that has more than 1 sailor, find the lowest rating of sailors

```
SELECT  S.age,  MIN (S.rating)
FROM  Sailors S
GROUP BY  S.age
HAVING  COUNT (*) > 1
```

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45 |
| 31 | lubber | 8 | 45 |
| 71 | zorba | 10 | 18 |
| 64 | horatio | 7 | 35 |
| 29 | brutus | 1 | 33 |
| 58 | rusty | 10 | 35 |

| age | rating |
|---|---|
| 18 | 10 |
| 33 | 1 |
| 35 | 7 |
| 35 | 10 |
| 45 | 7 |
| 45 | 8 |

1: projection

| age | min-rating | count |
|---|---|---|
| 18 | 10 | 1 |
| 33 | 1 | 1 |
| 35 | 7 | 2 |
| 45 | 7 | 2 |

2: grouping

| age | Min-rating |
|---|---|
| 35 | 7 |
| 45 | 7 |

*Answer relation*

Query:
Find the name of sailors who've reserved all boats.

- **Recall our solution of using nested queries**

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

Query: Find the name of sailors who've reserved all boats.

- **Can you do this using GROUP BY and HAVING?**
  - Hint: for each sailor, check whether the number of distinct boats he/she has reserved equals to the total number of (distinct) boats.

```
SELECT  S.name
FROM  Sailors S, reserves R
WHERE  S.sid = R.sid
GROUP BY S.name, S.sid
HAVING  COUNT(DISTINCT R.bid) =
                ( SELECT COUNT (*) FROM Boats)
```

Note: It must have both sid and name in the GROUP BY clause. Why? Can we only use sid? Can we only use sname?