

Week 4 Tutoring

CSE 180





Aggregates

- Used for computing summary results over a table
- Applied on **scalar** values
 - Exception of COUNT
- Types of aggregates:
 - COUNT()
 - SUM()
 - AVG()
 - MIN()
 - MAX()



COUNT

- COUNT(*): returns total number of records (tuples)
- COUNT(STUDENT): return number of **non null** values over the column student.
- COUNT(DISTINCT STUDENT): return number of distinct **non null** values over the column student



SUM

- SUM(STUDENT): Sum all non null values over the column student.
- SUM(DISTINCT STUDENT): Sum all distinct non null values over the column student



AVG

- `AVG(GRADE)`: return the average of non null values over the column grade.
- `AVG(DISTINCT GRADE)`: return average of distinct non null values over the column grade



MIN/MAX

- MIN(GRADE): return the minimum value of non null values over the column grade.
- MAX(GRADE): return the maximum value of non null values over the column grade.



Empty Results

SUM, AVG, MIN, MAX on an empty result (no tuples) is NULL.

COUNT of an empty result is 0.

GROUP BY

A group-by clause takes the tuples from the FROM clause and divides them into groups based on a certain conditions.

```
SELECT working_area, COUNT(*)  
FROM agents  
GROUP BY working_area;
```

agents

AGENT_NAME	WORKING_AREA
Alex	London
Subbarao	Bangalore
Benjamin	Hampshair
Ramasundar	Bangalore
Alford	New York
Ravi Kumar	Bangalore
Santakumar	Chennai
Lucida	San Jose
Anderson	Brisban
Mukesh	Mumbai
McDen	London
Ivan	Toronto

the working_area have
been grouped and appearing
once

WORKING_AREA	COUNT(*)
San Jose	1
Toronto	1
London	2
Hampshair	1
New York	1
Brisban	1
Bangalore	3
Chennai	1
Mumbai	1

result



Query Notation

SELECT [**DISTINCT**] c1, c2, ..., cm AGGOP(...)

FROM R1, R2, ..., Rn

[**WHERE** condition]

[**GROUP BY** <list grouping attributes>]

[**ORDER BY** < list of attributes [**ASC** | **DESC**]>]



Important Notes

- If SELECT clause has aggregates AGGOP, then c_1, c_2, \dots, c_m must come from the list of grouping attributes
- Aggregates can't appear in WHERE clauses.
- The non-aggregate columns in the SELECT clause must come from the
- attributes in the GROUP BY clause.



HAVING

the HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.



So what's the difference between these two?

```
SELECT e.execName, SUM(m.length)
FROM MovieExec e, Movies m
WHERE m.producerC# = e.cert#
GROUP BY e.execName
HAVING MIN(m.movieYear) < 1930;
```

```
SELECT e.execName, SUM(m.length)
FROM MovieExec e, Movies m
WHERE m.producerC# = e.cert#
AND
    (SELECT MIN(m1.movieYear)
     FROM m1.movieYear
     WHERE m1.producerC# = e.cert#) < 1930
GROUP BY e.execName;
```



```
SELECT e.execName, SUM(m.length)
FROM MovieExec e, Movies m
WHERE m.producerC# = e.cert#
GROUP BY e.execName
HAVING MIN(m.movieYear) < 1930;
```

Notice that there are multiple minimum values generated for different groups..

```
SELECT e.execName, SUM(m.length)
FROM MovieExec e, Movies m
WHERE m.producerC# = e.cert#
AND
  (SELECT MIN(m1.movieYear)
   FROM m1.movieYear
   WHERE m1.producerC# = e.cert#) < 1930
GROUP BY e.execName;
```

The only minimum value is generated before these tuples are grouped. So this query won't work right for our purpose.