Recap: (partial)

- SELECT \([DISTINCT]\) column\_name(s) / aggregate\_expr
  
  FROM table\_name(s)
  
  WHERE conditions
  
  ORDER BY column\_name(s) [ASC/DESC]
Today’s Agenda

- Grouping
- Sub-queries

Grouping rows

- Find the age of the youngest student for each class year
- SELECT MIN (S.Age)
  FROM Students S
  WHERE S.ClassYear = 2012

| (no column name) | 21 |
GROUP-BY Clause

- **SELECT [DISTINCT] column_name(s) | aggregate_expr**
  FROM table_name(s)
  [WHERE conditions]
  GROUP BY grouping_columns

- **Example:**
  ```sql
  SELECT ClassYear, MIN(Age)
  FROM Students
  GROUP BY ClassYear
  ```

<table>
<thead>
<tr>
<th>ClassYear</th>
<th>(no column name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>21</td>
</tr>
<tr>
<td>2012</td>
<td>17</td>
</tr>
<tr>
<td>2011</td>
<td>18</td>
</tr>
<tr>
<td>2013</td>
<td>20</td>
</tr>
</tbody>
</table>

Conceptual Evaluation

- Semantics of an SQL query defined as follows:
  - Compute the cross-product of tables in FROM (table_names)
  - Discard resulting rows if they fail WHERE conditions
  - Delete columns that are not in SELECT or GROUP BY(column_names or grouping-columns)
  - Remaining rows are partitioned into groups by the value of the columns in grouping-columns
  - One answer row is generated per group

- **Note:** Does not imply query will actually be evaluated this way!
HAVING Clause

- SELECT [DISTINCT] column_name(s) | aggregate_expr
  FROM table_name(s)
  [WHERE conditions]
  GROUP BY grouping_columns
  HAVING group_conditions

- GROUP BY groups the rows
- HAVING restricts the groups presented in the result

Example- HAVING

- SELECT ClassYear, MIN(Age)
  FROM Students
  WHERE MajDeptName = 'ComSci'
  GROUP BY ClassYear
  HAVING COUNT(*) > 20
Conceptual Evaluation

- SQL query semantics:
  - Compute the cross-product of `table_names`
  - Discard resulting rows if they fail `conditions`
  - Delete columns that are not specified in `SELECT, GROUP BY`
  - Remaining rows are partitioned into groups by the value of the columns in `grouping-columns`
  - One answer row is generated per group
  - Discard resulting groups that do not satisfy `group_conditions`

Example

- `SELECT Class, MIN(Age)`
- `FROM Students`
- `WHERE MajDeptName = 'ComSci'`
- `GROUP BY Class`
- `HAVING COUNT(*) > 2`
Class Exercise

- Students(\textit{SNb}, \textit{SName}, \textit{Email})
- Courses(\textit{Cid}, \textit{CName}, \textit{Dept})
- Enrolled(\textit{SNb}, \textit{Cid}, \textit{Semester})

- List all course names, and the number of students enrolled in the course

Sub-queries

- Aggregate Operators Limitations
  - Return only one row
  - Not in WHERE clause
Select age of oldest student

- SELECT MAX (Age)
  FROM Students S

Select oldest students and their age

- SELECT S.SName, MAX (Age)
  FROM Students S

- SELECT S.SName, S.Age
  FROM Students S
  WHERE S.AGE = (SELECT MAX(Age)
  FROM Students)

Correct!

Illegal!
Select students with age higher than average

- SELECT *
  FROM Students
  WHERE Age > AVG(Age)

  Illegal!

- SELECT *
  FROM Students
  WHERE Age > (SELECT AVG(Age)
  FROM Students)

  Correct!

Subqueries

- SELECT *
  FROM Students
  WHERE Age > (SELECT AVG(Age)
  FROM Students)

  Second select is a subquery (or nested query)
  You can have subqueries in FROM or HAVING clause also
Subqueries in FROM Clause

- Find name of students enrolled in both 'IT360' and 'IT334'

```sql
SELECT FName + ' ' + LName AS StudentName
FROM Students,
     (SELECT Alpha
      FROM Enroll
      WHERE CourseID = 'IT360'
      AND Alpha IN
      (SELECT Alpha
       FROM Enroll
       WHERE CourseID = 'IT334')
     ) AS ResultAlphaTable
WHERE Students.Alpha = ResultAlphaTable.Alpha
```

Subqueries Exercise

- Students(`Alpha`, LName, FName, Class, Age)
- Enroll(`Alpha`, CourseID, Semester, Grade)

1. Find alpha for students enrolled in both ‘IT360’ and ‘IT334’
2. Find name of students enrolled in both ‘IT360’ and ‘IT334’
Class Exercise

- Students($\text{Alpha}$, LName, FName, Class, Age)
- Enroll($\text{Alpha}$, CourseID, Semester, Grade)

- Find the name of students enrolled in ‘IT360’
  - Usual way
  - Use subqueries

Class Exercise

- What does this query compute:
- SELECT FName, LName
  FROM Students S, Enroll E1, Enroll E2
  WHERE S.Alpha = E1.Alpha
    AND S.Alpha = E2.Alpha
    AND E1.CourseID = ‘IT360’
    AND E2.CourseID = ‘IT344’
SELECT Summary

- SELECT [DISTINCT] column_name(s) | aggregate_expr
  FROM table_name(s)
  WHERE row_conditions
  GROUP BY grouping_columns
  HAVING group_conditions
  ORDER BY column_name(s) [ASC/DESC]