Recap: (partial)

- SELECT [DISTINCT] column_name(s)
  FROM table_name(s)
  WHERE conditions
  ORDER BY column_name(s) [ASC/DESC]
Today’s Agenda

- Math in SQL
- Grouping
- Subqueries

Calculations in SQL

- Simple arithmetic
- Five SQL Built-in Functions:
  - COUNT
  - SUM
  - AVG
  - MIN
  - MAX
Simple Arithmetic

- SELECT NbHours* HourlyRate AS Charge FROM FlightEvents

- SELECT Concat(SFirstName, ' ', SLastName) FROM Students

```
<table>
<thead>
<tr>
<th>Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
</tr>
<tr>
<td>400</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>400</td>
</tr>
</tbody>
</table>
```

Aggregate Operators

- SELECT COUNT(*) as nb FROM Students

- SELECT COUNT(DISTINCT SName) FROM Students WHERE SNb > 700

```
(No column name)
John Doe
Brad Johnson
Jessica Smith
Mary Davis
```

- SELECT AVG(Age) FROM Students WHERE SNb LIKE '16____'
Grouping rows

- Find the age of the youngest student for each class year
- SELECT MIN (S.Age)
  FROM Students S
  WHERE S.ClassYear = 2016

<table>
<thead>
<tr>
<th>ClassYear</th>
<th>(no column name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018</td>
<td>21</td>
</tr>
<tr>
<td>2016</td>
<td>17</td>
</tr>
<tr>
<td>2015</td>
<td>18</td>
</tr>
<tr>
<td>2017</td>
<td>20</td>
</tr>
</tbody>
</table>

GROUP-BY Clause

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

- Example:
  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>2018</td>
<td>21</td>
</tr>
<tr>
<td>2016</td>
<td>17</td>
</tr>
<tr>
<td>2015</td>
<td>18</td>
</tr>
<tr>
<td>2017</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 \(\text{table\_names}\)
  - Discard resulting rows if they fail WHERE \(\text{conditions}\)
  - Delete columns that are not in SELECT or GROUP BY(\(\text{column\_names}\) or \(\text{grouping\_columns}\))
  - Remaining rows are partitioned into groups by the value of the columns in \(\text{grouping\_columns}\)
  - One answer row is generated per group
- Note: Does not imply query will actually be evaluated this way!

HAVING Clause

- SELECT [DISTINCT] \(\text{column\_name(s)}\) | \(\text{aggregate\_expr}\)
  FROM \(\text{table\_name(s)}\)
  [WHERE \(\text{conditions}\)]
  GROUP BY \(\text{grouping\_columns}\)
  HAVING \(\text{group\_conditions}\)

- 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(SNb, SName, Email)
- Courses(Cid, CName, Dept)
- Enrolled(SNb, Cid, Semester)

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

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
  **Illegal!**

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

Sub-query
Select students with age higher than average

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

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

Subqueries

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

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

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

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

Subqueries Exercise

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

1. Find alpha for students enrolled in both ‘SY306’ and ‘SI485’
2. Find name of students enrolled in both ‘SY306’ and ‘SI485’
Class Exercise

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

Find the name of students enrolled in ‘SY306’
  - 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'
Efficiency

- Many way to write a SQL query
- Most efficient – use as few tables as possible
- If more than one table needed – use joins
- Less efficient – use sub-queries
- Least efficient – use correlated sub-queries

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]