Yet More SQL SELECT

Database Systems Lecture 9 Natasha Alechina

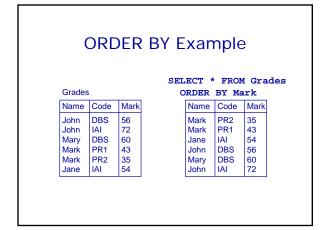
In This Lecture

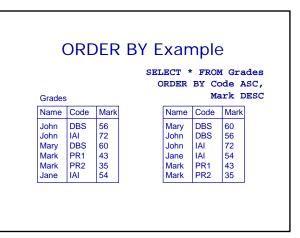
- Yet more SQL
 - ORDER BY
 - Aggregate functions
 - GROUP BY and HAVING
 - UNION etc.
- For more information
 - Connoly and Begg Chapter 5
 - Ullman and Widom Chapter 6.4

SQL SELECT Overview

ORDER BY

- The ORDER BY clause sorts the results of a query
 You can sort in
 - ascending (default) or descending order
 Multiple columns can
 - be given
- SELECT <columns> FROM <tables> WHERE <condition> ORDER BY <cols> [ASCENDING | DESCENDING | ASC | DESC]



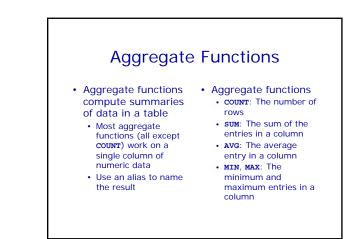


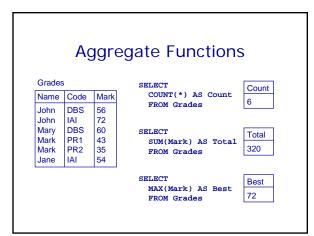
Constants and Arithmetic

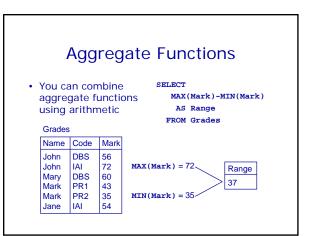
• As well as column names, you can select constants, compute arithmetic expressions and evaluate functions in a **SELECT** statement SELECT Mark/100 FROM Grades

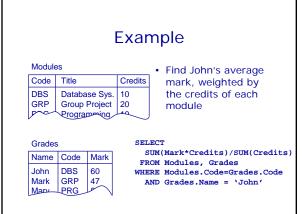
SELECT Salary + Bonus FROM Employee

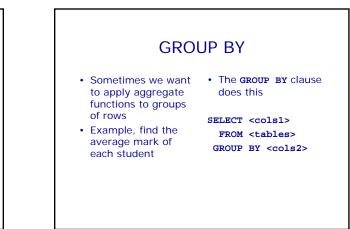
SELECT 1.175*Price FROM Products



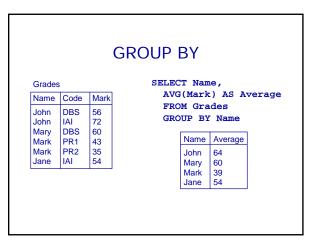


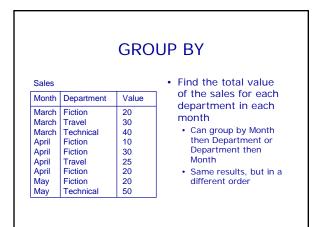




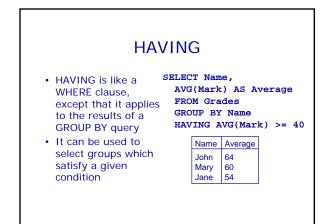


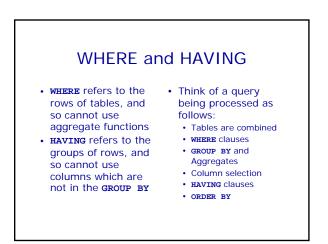






		GRO	JP B	(
SELECT Month, Department, SUM(Value) AS Total FROM Sales GROUP BY Month, Department			SUM	SELECT Month, Department, SUM(Value) AS Total FROM Sales GROUP BY Department, Month		
Month	Department	Total	Month	Department	Total	
April	Fiction	60	April	Fiction	60	
April	Travel	25	March	Fiction	20	
March	Fiction	20	May	Fiction	20	
March	Technical	40	March	Technical	40	
March	Travel	30	May	Technical	50	
May	Fiction	20	April	Travel	25	
May	Technical	50	March	Travel	30	
vicy	reennear	00	Warch	Havei	50	





UNION, etc.

- UNION, INTERSECT, and EXCEPT
 - These treat the tables as sets and are the usual set operators of union, intersection, and difference
 - We'll concentrate on
 UNION
 - Oracle has MINUS
 instead of EXCEPT
- They all combine the results from two select statements
- The results of the two selects must have the same columns and data types

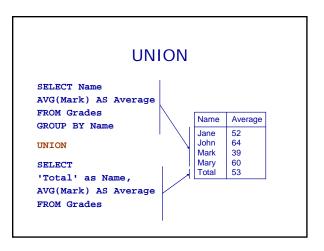
UNION · Find, in a single query, the average Grades Name Code Mark mark for each Jane student, and the IAI 52 John John DBS IAI 56 72 average mark overall PR1 43 35 Mark PR2 Mark Mary DBS 60

UNION

• The average for each • The average overall student:

SELECT Name, AVG(Mark) AS Average FROM Grades GROUP BY Name

- SELECT 'Total' AS Name, AVG(Mark) AS Average FROM Grades
- Note this has the same columns as the average by student



A Final Example

- Examiners' reports • We want a list of
 - students and their average markFor first and second years the average is
 - years the average is for that year
 For finalists it is 40% of the second year
 - of the second year plus 60% of the final year average.
- We want the results
 Sorted by year then
 - average mark (High to low) then last name, first name, and finally ID
 - To take into account the number of credits each module is worth
 Produced by a single query

Tables for the Example

Student

ID First Last Year

Grade ID Code Mark YearTaken

Module Code Title Credits

We'll Need a UNION

- · Finalists are treated differently
 - · Write one query for the finalists
 - · Write a second query
 - UNION for the first and
 - second years · Use a UNION to join them together
- <QUERY FOR OTHERS>

<QUERY FOR FINALISTS>

We'll need to Join the Tables

- · Both of the subaueries need information from all
 - the tables · The student ID, name and year
 - The marks for each module and the year
 - taken The number of credits for each module
- · This is a natural join operation
 - · We could use a NATURAL JOIN statement, and hope that our version of SQL can do it
 - Safer to just use a WHERE clause

The Query So Far

SELECT <some information> FROM Student, Module, Grade WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code AND <student is in third year>

UNION

SELECT < some information> FROM Student, Module, Grade WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code AND <student is in first or second year>



```
•We need to retrieve

    Compute average

     mark, weighted 40-60
```

```
across years 2 and 3
• First year marks need
```

- to be ignored • The ID, Name, and
- Year are needed as they are used for ordering
- The average is hard We don't have any statement to separate years 2 and 3 easily
 - We can exploit the fact that 40 = 20*2and 60 = 20*3, so
 - YearTaken and the weighting have a simple relationship

Information for Finalists

```
SELECT Year, Student.ID, Last, First,
       SUM((20*YearTaken/100)*Mark*Credits)/120
         AS AverageMark
 FROM Student, Module, Grade
 WHERE Student.ID = Grade.ID
  AND Module.Code = Grade.Code
  AND YearTaken IN (2,3)
  AND Year = 3
 GROUP BY Year, Student.ID, First, Last
```

Information for Other **Students**

- Other students are easier than finalists · We just need to average their marks where YearTaken and Year are the same
 - · As before we need the ID, Name, and Year for ordering

Information for Other Students

SELECT Year, Student.ID, Last, First, SUM(Mark*Credits)/120 AS AverageMark FROM Student, Module, Grade WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code AND YearTaken = Year AND Year IN (1,2) GROUP BY Year, Student.ID, First, Last

The Final Query

SELECT Year, Student.ID, Last, First, SUM((20*YearTaken/100)*Mark*Credits)/120 AS AverageMark FROM Student, Module, Grade WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code AND YearTaken IN (2,3) AND Year = 3 GROUP BY Year, Student.ID, First, Last UNION SELECT Year, Student.ID, Last, First, SUM(Mark*Credits)/120 AS AverageMark FROM Student, Module, Grade

SUM(Mark*Credits)/120 AS Averagemark FROM Student, Module, Grade WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code AND YearTaken = Year AND Year IN (1,2) GROUP BY Year, Student.ID, First, Last

ORDER BY Year desc, AverageMark desc, First, Last, ID

Next Lecture

- Missing Information
 - NULLs and three-valued logic
 - NULLs and the relational model
 - OUTER JOINs
 - Default values
- For more information
 - Ullman and Widom 6.1.5, 6.1.6, 6.3.8

Yet More SQL SELECT