Chapter 6

SQL Aggregate Operators

Review: SQL Environment

- Data Definition Language (DDL)
  - Commands that define a database, including creating, altering, and dropping tables and establishing constraints
  - CREATE / DROP / ALTER, …
- Data Manipulation Language (DML)
  - Commands that maintain and query a database
  - INSERT, UPDATE, DELETE, SELECT, …
- Data Control Language (DCL)
  - Commands that control a database, including administering privileges and committing data
  - GRANT, ADD, REVOKE

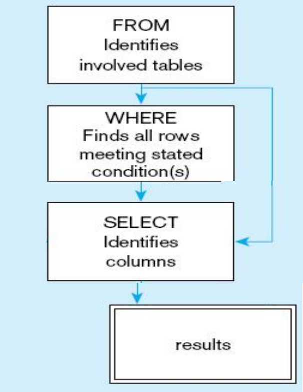
Review: SQL Queries

```
CREATE TABLE table_name (
    field  type  constraints,
    field2 type2,
    CONSTRAINT name ...,
);
```

```
INSERT INTO table (fields)
VALUES (values)
```

```
DELETE FROM table
WHERE conditions
```

```
UPDATE table
SET field = value
WHERE conditions
```



FROM
Identifies involved tables

WHERE
Finds all rows meeting stated condition(s)

SELECT
Identifies columns

results

```
SELECT [DISTINCT] attribute-list
FROM table-list
WHERE conditions
```

SQL Aggregate Operators

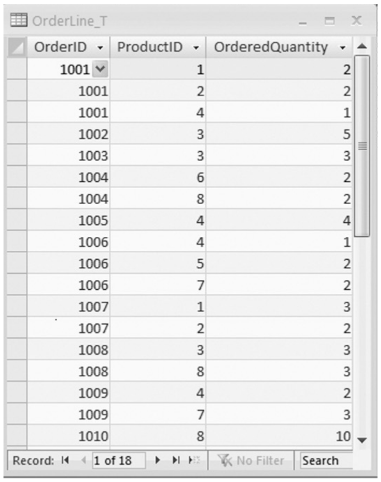
SQL Aggregate Operator: COUNT

- *How many products were on order number 1004?*

```
SELECT COUNT(*)
  FROM OrderLine_T
      WHERE OrderID = 1004;
```

Result: COUNT(*) : 2

- COUNT(*) applied on the set of rows selected by the WHERE clause.



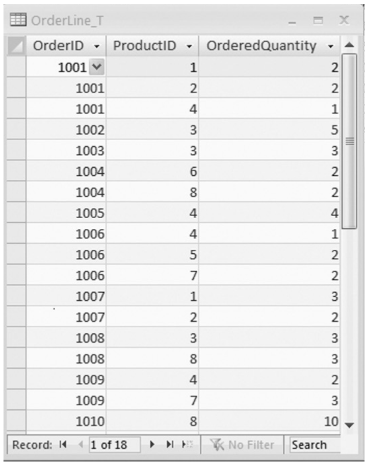
OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

SQL Aggregate Operators

- *How many products were on order number 1004, and what is their ID?*

```
SELECT ProductID, COUNT(*)
   FROM OrderLine_T
     WHERE OrderID = 1004;
```

- *Result*: ERROR



OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

Why ERROR?

- The problem is that ProductID has 2 values for the two rows selected, and COUNT returns one aggregate value.
- Need to ensure that ProductID returns a single value
- Solution: use the **GROUP BY** clause (discussed shortly)

SQL Aggregate Operators

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

```
SELECT SUM(Price*Quantity)
FROM Purchase_T
```

⇩

65 (=20+5+10+30)

```
SELECT SUM(Price*Quantity)
FROM Purchase_T
WHERE Product = ‘Bagel’
```

⇩

50 (=20+30)

Purchase_T

Product	Date	Price	Quantity
Bagel	2013-08-21	1	20
Banana	2013-10-09	1	10
Bagel	2013-10-05	1.50	20
Banana	2013-10-04	0.5	10

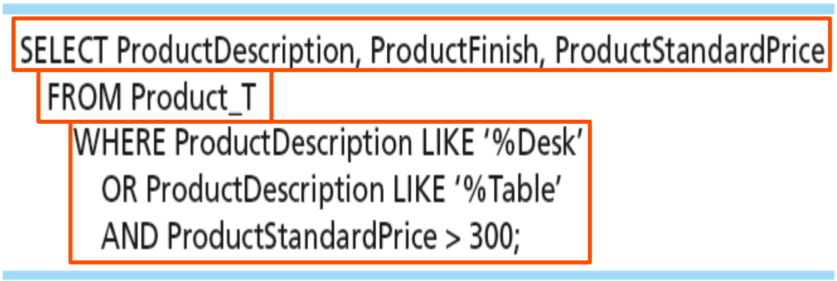
Practice: Exercise #6

Write SQL queries to answer the following question:

3. What is the smallest section number used in the first semester of 2008?

SELECT Example–Boolean Operators

AND, OR, and NOT Operators for customizing conditions in WHERE clause:

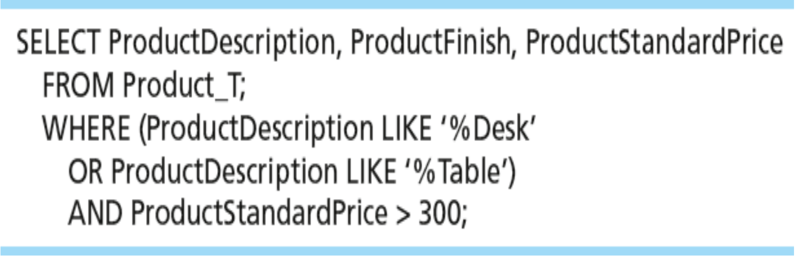


```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
  FROM Product_T
    WHERE ProductDescription LIKE '%Desk'
      OR ProductDescription LIKE '%Table'
      AND ProductStandardPrice > 300;
```

Note: the LIKE operator allows you to compare strings using wildcards. For example, the % wildcard in '%Desk' indicates that all strings that have any number of characters preceding the word "Desk" will be allowed.

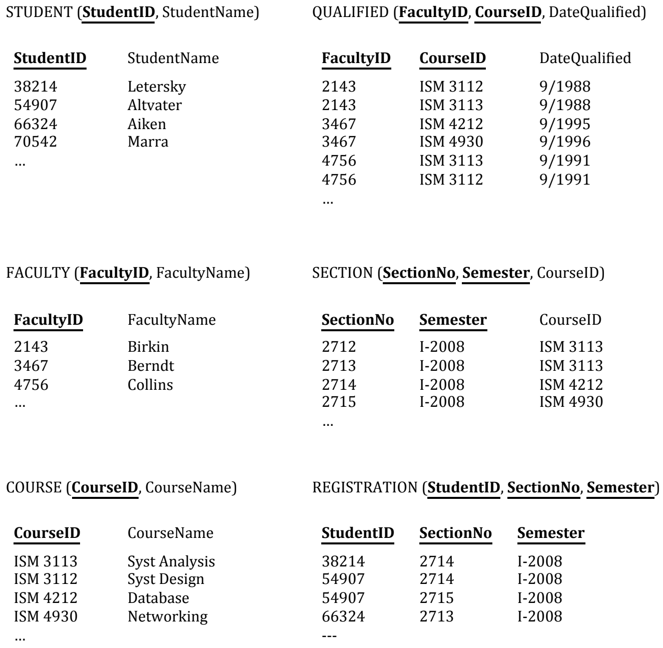
SELECT Example–Boolean Operators

With parentheses…these override the normal precedence of Boolean operators



```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
  FROM Product_T;
  WHERE (ProductDescription LIKE '%Desk'
    OR ProductDescription LIKE '%Table')
    AND ProductStandardPrice > 300;
```

By default, the AND operator takes precedence over the OR operator. With parentheses, you can make the OR take place before the AND.



STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
…	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
…		

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
…	

SECTION (**SectionNo**, **Semester**, CourseID)

SectionNo	**Semester**	CourseID
2712	I-2008	ISM 3113
2713	I-2008	ISM 3113
2714	I-2008	ISM 4212
2715	I-2008	ISM 4930
…		

COURSE (**CourseID**, CourseName)

CourseID	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
…	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

Practice: Exercise #7

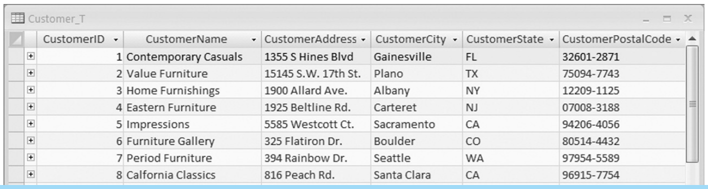
Write SQL queries to answer the following questions:

1. How many students are enrolled in Section 2714 in the first semester of 2008?

2. Which faculty members have qualified to teach a course since 1993? List the faculty ID, course ID, and date of qualification.

SELECT – Sorting Results with the ORDER BY Clause

- List all Customers -- Name, City and State -- ordered first by State, and within a state by Name.



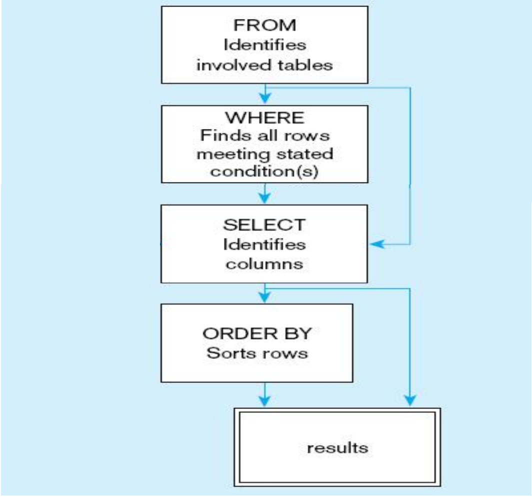
Customer_T

CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	Calfornia Classics	816 Peach Rd.	Santa Clara	CA	96915-7754

```
SELECT CustomerName, CustomerCity, CustomerState
   FROM Customer_T
      WHERE CustomerState IN ('FL', 'TX', 'CA', 'HI')
         ORDER BY CustomerState, CustomerName;
```

IN allows you to include only those rows whose CustomerState value is either FL, TX, CA, or HI.

Fragment of Figure 6-10: SQL statement processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

Practice: Exercise #9

Write SQL queries to answer the following questions:

1. What are the courses included in the Section table? List each course only once.

2. List all students in alphabetical order by StudentName.

3. List the students who are enrolled in each course in Semester I, 2008. Order the students by the sections in which they are enrolled.

4. List the courses available. Order them by course prefix. (ISM is the only prefix shown, but there are many others throughout the university.)

Motivation for Grouping

- So far, we have applied aggregate operators to all rows. What if we want to apply them to only a subset of rows?

Product	Date	Price	Quantity
Bagel	2013-08-21	1	20
Banana	2013-10-09	1	10
Bagel	2013-10-05	1.50	20
Banana	2013-10-04	0.5	10

- Example: *Find the average price for every product purchased after 2013-10-02*
  - We may not know how many (or what) products are
  - Group the rows by Product, then SELECT from each group!

Motivation for Grouping

- *Find the average price for every product purchased after 2013-10-02*

```
SELECT AVG(Price)
FROM Purchase_T
WHERE Date > '2013-10-02'
GROUP BY Product;
```

Purchase_T

Product	Date	Price	Quantity
Bagel	2013-08-21	1	20
Banana	2013-10-09	1	10
Bagel	2013-10-05	1.50	20
Banana	2013-10-04	0.5	10

- Evaluation order:

FROM	select table
WHERE	select rows
GROUP BY	group rows
SELECT	applied on groups

Motivation for Grouping

- *Find the average price for every product purchased after 2013-10-02*

```
SELECT AVG(Price)
FROM Purchase_T
WHERE Date > '2013-10-02'
GROUP BY Product;
```

Purchase_T

Product	Date	Price	Quantity
Bagel	2013-08-21	1	20
Banana	2013-10-09	1	10
Bagel	2013-10-05	1.50	20
Banana	2013-10-04	0.5	10

	Product	Date	Price	Quantity
SELECT AVG(Price)	Banana	2013-10-09	1	10
	Banana	2013-10-04	0.5	10
SELECT AVG(Price)	Bagel	2013-10-05	1.50	20

SELECT– Categorizing Results Using GROUP BY

For use with aggregate functions

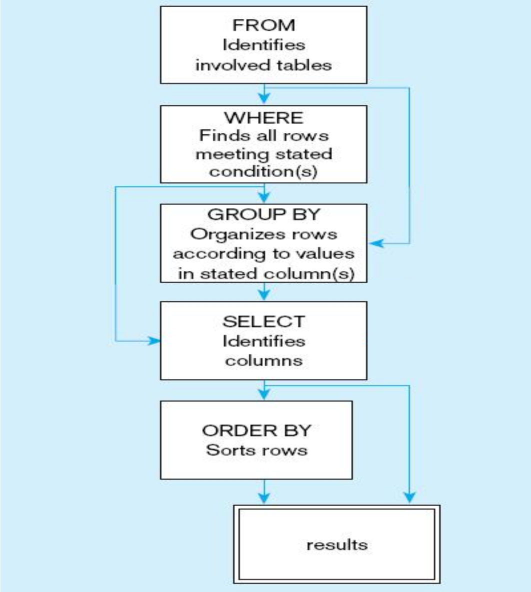
Scalar aggregate: single value returned from SQL query with aggregate function

Vector aggregate: multiple values returned from SQL query with aggregate function (via GROUP BY)

```
SELECT CustomerState, COUNT (CustomerState)
   FROM Customer_T
      GROUP BY CustomerState;
```

Note: you can use single-value fields with aggregate functions if they are included in the GROUP BY clause

Fragment of Figure 6-10: SQL statement processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

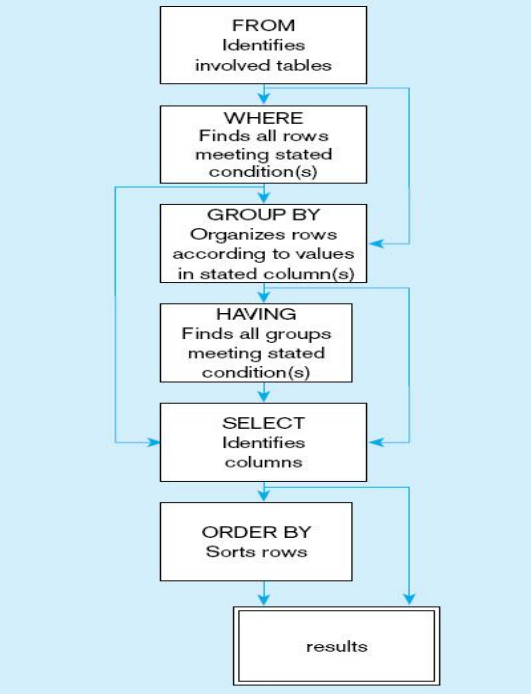
SELECT– Filtering Categories Using HAVING

For use with GROUP BY

```
SELECT CustomerState, COUNT (CustomerState)
  FROM Customer_T
    GROUP BY CustomerState
    HAVING COUNT (CustomerState) > 1;
```

Like a WHERE clause, but it operates on groups (categories), not on individual rows. Here, only those groups with total numbers greater than 1 will be included in final result.

Figure 6-10:
SQL statement
processing order



Practice: Exercise #8

Write SQL queries to answer the following questions:

1. Which students are enrolled in Database and Networking? (Hint: Use SectionNo for each class so you can determine the answer from the Registration table by itself.)

2. Which instructors cannot teach both Syst Analysis and Syst Design?

Using and Defining Views

Views provide users controlled access to tables

Base Table – Table containing the raw data

Dynamic View

- A “virtual table” created dynamically upon request by a user
- No data actually stored; contents materialized when referenced
- Based on SQL SELECT statement on base tables or other views

Materialized View

- Copy or replication of data
- Data actually stored on the disk
- Refreshed periodically to match corresponding base tables

Defining Views

- Useful for presenting different info to different users

Employee_T

SSN	Name	Department	Project	Salary

```
CREATE VIEW Developer_V AS
SELECT Name, Project
FROM Employee_T
WHERE Department = “Development”
```

Developer_V

Name	Project

- Payroll has access to Employee_T, others to Developer_V only

Querying a View

Employee_T

SSN	Name	Department	Project	Salary

- We can later use this view:

Developer_V

Name	Project

```
SELECT *
FROM  Developer_V;
```

```
INSERT INTO Developer_V( Name, Project)
VALUES(‘Mike’, ‘Gadget Design’);
```

```
INSERT INTO Employee_T
VALUES(NULL, ‘Mike’, ‘Development’, ‘Gadget Design’, NULL);
```

Anything Wrong?

Most views are non-updateable.

Views – Another Example

```
CREATE VIEW ExpensiveStuff_V
  AS
    SELECT ProductID, ProductDescription, ProductStandardPrice
      FROM Product_T
        WHERE ProductStandardPrice > 300
        WITH CHECK OPTION;
```

- CHECK OPTION works only for updateable views and prevents updates that would create rows not included in the view

Advantages of Views

- Simplify query commands
- Assist with data security (but don't rely on views for security, there are more important security measures)
- Enhance programming productivity
- Contain most current base table data
- Use little storage space
- Provide customized view for user
- Establish physical data independence

Disadvantages of Views

- Use processing time each time view is referenced
- May or may not be directly updateable

Practice: Exercise #2

Define the following view:

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra