3ISY402 – DATABASE SYSTEMS

Lecture 4 –
SQL: Complex DML - Aggregate Functions and Subqueries

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

Leena Gulabivala

Lecture - Objectives

- How to retrieve data from database using SELECT statement to create complex queries:
  - Use aggregate functions.
  - Group data using GROUP BY and HAVING.
  - Use subqueries.

SELECT Statement - Aggregates

- ISO standard defines five aggregate functions:
  - COUNT returns number of values in given column.
  - SUM returns sum of values in given column.
  - AVG returns average of values in given column.
  - MIN returns smallest value in given column.
  - MAX returns largest value in given column.

SELECT Statement - Aggregates

- Each operates on a single column of a table and returns a single value.
- COUNT, MIN, and MAX apply to numeric and non-numeric fields.
- SUM and AVG may be used on numeric fields only.
- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.
- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.
- Can use DISTINCT before column name to eliminate duplicates.

SELECT Statement - Aggregates

- DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG.
- Aggregate functions can be used only in SELECT list and in HAVING clause.
- If SELECT list includes an aggregate function and there is no GROUP BY clause, SELECT list cannot reference a column with an aggregate function. For example, the following is illegal:

```
SELECT staffNo, COUNT(salary)
FROM Staff;
```

Example 1 - Use of COUNT(*)

- How many properties cost more than £350 per month to rent?

```
SELECT COUNT(*) AS myCount
FROM   PropertyForRent
WHERE  rent > 350;
```

myCount
5

Example 2 - Use of COUNT(DISTINCT)

- How many different properties viewed in May '04?

```
SELECT  COUNT(DISTINCT propertyNo) AS
 myCount
FROM     Viewing
WHERE  viewDate BETWEEN '01-May-04'
AND       '31-May-04';
```

myCount
2

Example 3 - Use of COUNT and SUM

- Find number of Managers and sum of their salaries.

```
SELECT COUNT(staffNo) AS myCount,
       SUM(salary) AS mySum
FROM   Staff
WHERE position = 'Manager';
```

myCount	mySum
2	54000.00

Example 4 - Use of MIN, MAX, AVG

- Find the minimum, maximum, and average staff salary.

```
SELECT MIN(salary) AS myMin,
       MAX(salary) AS myMax,
       AVG(salary) AS myAvg
FROM   Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

SELECT Statement - Grouping

- Use GROUP BY clause to get sub-totals.
- SELECT and GROUP BY are closely integrated: each item in SELECT list must be *single-valued per group*, and SELECT clause may only contain:
  - column names
  - aggregate functions
  - constants
  - expression involving combinations of the above.

SELECT Statement - Grouping

- All column names in SELECT list must appear in GROUP BY clause unless name is used only in an aggregate function.
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from the remaining rows satisfying the predicate.
- ISO considers two nulls to be equal for purposes of GROUP BY.

Example 5 - Use of GROUP BY

- Find the number of staff in each branch and their total salaries.

```
SELECT  branchNo,
        COUNT(staffNo) AS myCount,
        SUM(salary) AS mySum
FROM    Staff
GROUP BY branchNo
ORDER BY branchNo;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Restricted Groupings – HAVING clause

- HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table.
- Similar to WHERE, *but WHERE filters individual rows whereas HAVING filters groups.*
- Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

Example 6 - Use of HAVING

- For each branch with more than 1 member of staff, find the number of staff in each branch and the sum of their salaries.

```
SELECT  branchNo,
        COUNT(staffNo) AS myCount,
        SUM(salary) AS mySum
FROM    Staff
GROUP  BY branchNo
HAVING COUNT(staffNo) > 1
ORDER  BY branchNo;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

Subqueries

- Some SQL statements can have a SELECT embedded within them.
- A *subselect* can be used in WHERE and HAVING clauses of an outer SELECT, where it is called a *subquery* or *nested query*.
- Subqueries are executed first, and then the outer queries.
- Subselects may also appear in INSERT, UPDATE, and DELETE statements.

Example 7 - Subquery with Equality

- List the staff who work in the branch at '163 Main St'.

```
SELECT   staffNo, fName, lName, position
FROM     Staff
WHERE    branchNo =
             (SELECT branchNo
             FROM     Branch
             WHERE  street = '163 Main St');
```

Outer Select or Outer-query

Subselect or subquery

Example 7 - Subquery with Equality

- Inner SELECT finds branch number for branch at '163 Main St' ('B003').
- Outer SELECT then retrieves details of all staff who work at this branch.
- Outer SELECT then becomes:

```
SELECT staffNo, fName, lName, position
FROM   Staff
WHERE branchNo = 'B003';
```

Example 7 - Subquery with Equality

staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Example 8 - Subquery with Aggregate

- List all staff whose salary is greater than the average salary, and show by how much.

```
SELECT staffNo, fName, lName, position, salary –
            (SELECT AVG(salary)
             FROM   Staff)  AS SalDiff
FROM   Staff
WHERE salary >
            (SELECT AVG(salary)
             FROM   Staff);
```

Example 8 - Subquery with Aggregate

- Cannot write 'WHERE salary > AVG(salary)'
- Instead, use subquery to find average salary (17000), and then use outer SELECT to find those staff with salary greater than this:

```
SELECT  staffNo, fName, lName, position,
        salary – 17000 AS salDiff
FROM    Staff
WHERE   salary > 17000;
```

Example 8 - Subquery with Aggregate

staffNo	fName	lName	position	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00

Subquery Rules

- ORDER BY clause may not be used in a subquery (although it may be used in outermost SELECT).
- Subquery SELECT list must consist of a single column name or expression, except for subqueries that use EXISTS.
- By default, column names refer to table name in FROM clause of subquery.
- When subquery is an operand in a comparison, subquery must appear on right-hand side.
- A subquery may not be used as an operand in an expression.
- Use IN in the search condition of the main (outer) query when the subquery might return multiple values.

Example 9 - Nested subquery: use of IN

- List properties handled by staff at '163 Main St'.

```
SELECT  propertyNo, street, city, postcode, type,
        rooms, rent
FROM    PropertyForRent
WHERE   staffNo IN
        (SELECT     staffNo
        FROM        Staff
        WHERE       branchNo =
                (SELECT     branchNo
                FROM        Branch
                WHERE       street = '163 Main St'));
```

Example 9 - Nested subquery: use of IN

propertyNo	street	city	postcode	type	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375
PG21	18 Dale Rd	Glasgow	G12	House	5	600

ANY and ALL

- ANY and ALL may be used with subqueries that produce a single column of numbers.
- With ALL, condition will only be true if it is satisfied by *all* values produced by subquery.
- With ANY, condition will be true if it is satisfied by *any* values produced by subquery.
- If subquery is empty, ALL returns true, ANY returns false.
- SOME may be used in place of ANY.

Example 10 - Use of ANY/SOME

- Find staff whose salary is larger than salary of at least one member of staff at branch B003.

```
SELECT  staffNo, fName, lName, position, salary
FROM    Staff
WHERE  salary > SOME
                (SELECT  salary
                 FROM    Staff
                 WHERE  branchNo = 'B003');
```

Example 10 - Use of ANY/SOME

- Inner query produces set {12000, 18000, 24000} and outer query selects those staff whose salaries are greater than any of the values in this set.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Example 11 - Use of ALL

- Find staff whose salary is larger than salary of every member of staff at branch B003.

```
SELECT  staffNo, fName, lName, position, salary
FROM    Staff
WHERE   salary > ALL
                (SELECT     salary
                 FROM       Staff
                 WHERE      branchNo = 'B003');
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00

References and Further Reading:

- T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.
  - Chapter 5 – SQL Data Manipulation.
- T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.
  - Chapter on SQL Data Manipulation.