SQL: querying the database

Aggregate functions

Querying sets of rows: the 'GROUP BY' and 'Having' clauses

Subqueries: nesting queries within queries

Aggregate functions

ISO standard defines five aggregate functions:

- COUNT returns number of values in specified column.
- SUM returns sum of values in specified column.
- AVG returns average of values in specified column.
- MIN returns smallest value in specified column.
- MAX returns largest value in specified column.

Aggregate functions

- Each aggregate function 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 for numeric fields only!
  - each aggregate function eliminates nulls first
  - COUNT(*) leaves nulls (and duplicates) in - counts all rows of a table
  - use DISTINCT before column name to eliminate duplicates, eg when using MIN/MAX functions

Aggregate functions

- Aggregates can only be used in SELECT and in HAVING clauses

- If an aggregate function is used in a SELECT clause and there is no GROUP BY clause, SELECT list cannot reference a another column without an aggregate function. For example, the following is illegal:

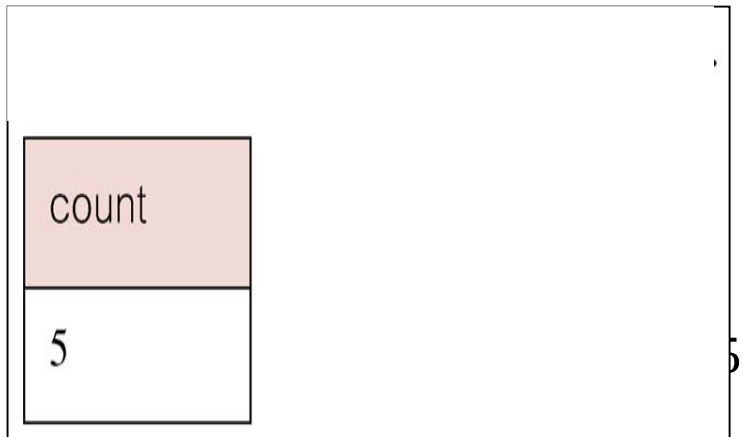
```
SELECT   staffNo, COUNT(salary)
FROM     Staff;
```

The query would tell you how many salaries each employee has!

Use of COUNT(*)

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

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



Use of 'COUNT(DISTINCT)'

- How many different properties were viewed in May '01?

```
SELECT    COUNT(DISTINCT propertyNo) AS count
FROM      Viewing
WHERE     viewDate BETWEEN '1-May-01' AND '31-May-01';
```

count
2

Use of 'COUNT' and 'SUM'

- Find the number of Managers and sum of their salaries.

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

count	sum
2	54000.00

Use of 'MIN', 'MAX', 'AVG'

- Find minimum, maximum, and average staff salary.

```
SELECT    MIN(salary) AS min,
          MAX(salary) AS max,
          AVG(salary) AS avg
FROM      Staff;
```

min	max	avg
9000.00	30000.00	17000.00

Querying sets of rows: the 'GROUP BY' clause

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

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

branchNo	count	sum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

the 'GROUP BY' clause

- Use GROUP BY clause to get sub-totals.
- SELECT and GROUP BY 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.

the 'GROUP BY' clause

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

Conditional grouping: the 'HAVING' clause

- HAVING filters groups (of rows)
  (nb, 'WHERE' filters individual rows)

- Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

Use of 'HAVING'

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

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

branchNo	count	sum
B003	3	54000.00
B005	2	39000.00

Subqueries

- Subquery: a SELECT statement embedded in a query

- It produces:
  - single value;
  - or, single row with selected values;
  - or, a table of selected rows and values
- It can be used in
  - WHERE and HAVING clauses of an outer SELECT, where it is called a subquery or nested query.
  - INSERT, UPDATE, and DELETE statements.

Subquery with Equality

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

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

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.

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

Subquery with Aggregate

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

```
SELECTstaffNo, fName, lName, position,
  salary - (SELECT AVG(salary) FROM
  Staff As SalDiff
FROM   Staff
WHERE salary >
      (SELECT AVG(salary)
       FROM Staff);
```

- **Cannot write 'WHERE salary > AVG(salary)'**
- **use subquery to find value of average salary for outerSelect**

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

Four subquery rules:

- ORDER BY clause may not be used in a subquery
- subquery SELECT list must consist of a single column name or expression (except for subqueries that use EXISTS)
- column names refer to table name in FROM clause of subquery
- When subquery used in a comparison, subquery must appear on right-hand side.

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'));
```

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.

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');
```

◆**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

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

EXISTS and NOT EXISTS

- EXISTS and NOT EXISTS are for use only with subqueries.
- Produce a simple true/false result.
- True if and only if there exists at least one row in result table returned by subquery.
- False if subquery returns an empty result table

EXISTS and NOT EXISTS

- As (NOT) EXISTS checks only for existence or non-existence of rows in subquery result table, subquery can contain any number of columns.

- Common for subqueries following (NOT) EXISTS to be of form:

  (SELECT * ...)

Example of using EXISTS

- Find all staff who work in a London branch.

```
SELECT staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS
        (SELECT *
        FROM Branch b
        WHERE s.branchNo = b.branchNo
        AND city = 'London');
```

staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant

Query using EXISTS

- Note, search condition s.branchNo = b.branchNo is necessary to consider correct branch record for each member of staff.
- If omitted, would get *all* staff records listed out because subquery:

```
SELECT * FROM Branch WHERE city='London'
```

would always be true (they do exist!) and so the outer query would be, in effect:

```
SELECT staffNo, fName, lName, position FROM
   Staff
WHERE true;
```

Query using EXISTS

- Could also write this query using join construct:

```
SELECT staffNo, fName, lName, position
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo AND
                   city = 'London';
```