University of Salahaddin
College of Engineering
Software & Informatics Engineering Department

SQL
2ND STAGE

Lecturer:
Hanan Kamal

Objectives

- Purpose and importance of SQL.
- How to retrieve data from database using SELECT and:
  - Use compound WHERE conditions.
  - Sort query results using ORDER BY.
  - Use aggregate functions.
  - Group data using GROUP BY and HAVING.
  - Use subqueries.

Objectives

- Join tables together.
- Perform set operations (UNION, INTERSECT..).

- How to update database using INSERT, UPDATE, and DELETE.

Objectives of SQL

- **SQL is a transform-oriented language with 2 major components:**
  - **A DDL for defining database structure.**
  - **A DML for retrieving and updating data.**

Writing SQL Commands

- **Use extended form of notation:**

- Upper-case letters represent reserved words.

- Lower-case letters represent user-defined words.

- | indicates a *choice* among alternatives.

- Curly braces indicate a *required element*.

- Square brackets indicate an *optional element*.

- … indicates *optional repetition* (0 or more).

Create Table

```
CREATE TABLE Staff(staffNo VARCHAR(5),
                   fName VARCHAR(15),
                   lName VARCHAR(15),
                   salary DECIMAL(7,2),
                   sex CHAR
                   );
```

SQL Data types

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit†	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	

SELECT Statement

- **Order of the clauses cannot be changed.**
- **Only SELECT and FROM are mandatory.**

Example 5.1 All Columns, All Rows

List full details of all staff.

```
SELECT staffNo, fName, lName, address,
        position, sex, DOB, salary, branchNo
FROM Staff;
```

- **Can use * as an abbreviation for ‘all columns’:**

```
SELECT *
FROM Staff;
```

Example 5.1 All Columns, All Rows

Table 5.1 Result table for Example 5.1.

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Example 5.2 Specific Columns, All Rows

Produce a list of salaries for all staff, showing only staff number, first and last names, and salary.

```
SELECT staffNo, fName, lName, salary
FROM Staff;
```

Example 5.2 Specific Columns, All Rows

Table 5.2 Result table for Example 5.2.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Example 5.3 Use of DISTINCT

List the property numbers of all properties that have been viewed.

```
SELECT propertyNo
FROM Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

Example 5.3 Use of DISTINCT

- **Use DISTINCT to eliminate duplicates:**

```
SELECT DISTINCT propertyNo
FROM Viewing;
```

propertyNo
PA14
PG4
PG36

Example 5.4 Calculated Fields

Produce list of monthly salaries for all staff, showing staff number, first/last name, and salary.

```
SELECT staffNo, fName, lName, salary/12
FROM Staff;
```

Table 5.4 Result table for Example 5.4.

staffNo	fName	lName	col4
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

Example 5.4 Calculated Fields

- **To name column, use AS clause:**

```
SELECT staffNo, fName, lName, salary/12
        AS monthlySalary
FROM Staff;
```

Example 5.5 Comparison Search Condition

List all staff with a salary greater than 10,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > 10000;
```

Table 5.5 Result table for Example 5.5.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Example 5.6 Compound Comparison Search Condition

List addresses of all branch offices in London or Glasgow.

```
SELECT *
FROM Branch
WHERE city = 'London' OR city = 'Glasgow';
```

Table 5.6 Result table for Example 5.6.

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Example 5.7 Range Search Condition

List all staff with a salary between 20,000 and 30,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000;
```

- BETWEEN test includes the endpoints of range.
- None btween
  - WHERE salary>=20000 AND salary <= 30000;

Example 5.7 Range Search Condition

Table 5.7 Result table for Example 5.7.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

Example 5.8 Set Membership

List all managers and supervisors.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position IN ('Manager', 'Supervisor');
```

- **There is a negated version (NOT IN).**

```
WHERE position='Manager' OR
                position='Supervisor';
```

Example 5.9 Pattern Matching

Find all owners with the string 'Glasgow' in their address.

```
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE '%Glasgow%';
```

Table 5.9 Result table for Example 5.9.

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Example 5.9 Pattern Matching

- **SQL has two special pattern matching symbols:**
  - **%: sequence of zero or more characters;**
  - **_ (underscore): any single character.**
- **LIKE '%Glasgow%' means a sequence of characters of any length containing '*Glasgow*'.**

Example 5.11 Single Column Ordering

List salaries for all staff, arranged in descending order of salary.

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```

Example 5.11 Single Column Ordering

Table 5.11 Result table for Example 5.11.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

SELECT Statement - Aggregates

- 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.

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, but 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.

SELECT Statement - Aggregates

- **COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.**
- **Can use DISTINCT before column name to eliminate duplicates.**
- **DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG.**

SELECT Statement - Aggregates

- **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 out with an aggregate function. For example, the following is illegal:**

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

Example 5.13 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

SELECT Statement - Grouping

- **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.**

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 remaining rows satisfying predicate.

Example 5.17 Use of GROUP BY

Find 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;
```

Example 5.17 Use of GROUP BY

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

Example 5.18 Use of HAVING

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

SELECT Statement Summary

```
SELECT [DISTINCT | ALL]
  {* | [columnExpression [AS newName]] [,...] }
FROM                    TableName [alias] [, ...]
[WHERE      condition]
[GROUP BY  columnList]  [HAVING  condition]
[ORDER BY  columnList]
```

SELECT Statement Summary (cont.)

FROM	Specifies table(s) to be used.
WHERE	Filters rows.
GROUP BY	Forms groups of rows with same column value.
HAVING	Filters groups subject to some condition.
SELECT	Specifies which columns are to appear in output.
ORDER BY	Specifies the order of the output.

- Order of the clauses cannot be changed.
- Only SELECT and FROM are mandatory.

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*.**
- **Subselects may also appear in INSERT, UPDATE, and DELETE statements.**

Example 5.19 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');
```

Example 5.20 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 5.20 Subquery with Aggregate

Table 5.20 Result table for Example 5.20.

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

Multi-Table Queries

- Can use subqueries provided result columns come from same table.
- If result columns come from more than one table must use a join.
- To perform join, include more than one table in FROM clause.
- Use comma as separator and typically include WHERE clause to specify join column(s).

Multi-Table Queries

- Also possible to use an alias for a table named in FROM clause.
- Alias is separated from table name with a space.
- Alias can be used to qualify column names when there is ambiguity.

Example 5.24 Simple Join

List names of all clients who have viewed a property along with any comment supplied.

```
SELECT c.clientNo, fName, lName,
        propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

Example 5.24 Simple Join

- **Only those rows from both tables that have identical values in the clientNo columns (c.clientNo = v.clientNo) are included in result.**
- **Equivalent to equi-join in relational algebra.**

Table 5.24 Result table for Example 5.24.

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

Example 5.26 Three Table Join

For each branch, list staff who manage properties, including city in which branch is located and properties they manage.

```
SELECT b.branchNo, b.city, s.staffNo, fName, lName,
        propertyNo
FROM Branch b, Staff s, PropertyForRent p
WHERE b.branchNo = s.branchNo AND
        s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

Example 5.26 Three Table Join

Table 5.26 Result table for Example 5.26.

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

- **Alternative formulation for FROM and WHERE:**

```
FROM (Branch b JOIN Staff s USING branchNo) AS
        bs JOIN PropertyForRent p USING staffNo
```

INSERT

INSERT INTO TableName [(columnList)]
VALUES (dataValueList)

- ***columnList*** **is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.**
- **Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.**

Example 5.35 INSERT … VALUES

Insert a new row into Staff table supplying data for all columns.

```
INSERT INTO Staff
VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M', Date'1957-05-25', 8300, 'B003');
```

Example 5.36 INSERT using Defaults

Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO Staff (staffNo, fName, lName,
                   position, salary, branchNo)
 VALUES ('SG44', 'Anne', 'Jones',
         'Assistant', 8100, 'B003');
```

- Or

```
 INSERT INTO Staff
 VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL,
         NULL, 8100, 'B003');
```

INSERT … SELECT

- Second form of INSERT allows multiple rows to be copied from one or more tables to another:

```
INSERT INTO TableName [ (columnList) ]
   SELECT ...
```

UPDATE

```
UPDATE TableName
SET columnName1 = dataValue1
        [, columnName2 = dataValue2...]
[WHERE searchCondition]
```

- ***TableName*** **can be name of a base table or an updatable view.**
- **SET clause specifies names of one or more columns that are to be updated.**

UPDATE

- **WHERE clause is optional:**
  - **if omitted, named columns are updated for all rows in table;**
  - **if specified, only those rows that satisfy *searchCondition* are updated.**
- **New *dataValue(s)* must be compatible with data type for corresponding column.**

Example 5.38/39 UPDATE All Rows

Give all staff a 3% pay increase.

```
UPDATE Staff
SET salary = salary*1.03;
```

Give all Managers a 5% pay increase.

```
UPDATE Staff
SET salary = salary*1.05
WHERE position = 'Manager';
```

Example 5.40 UPDATE Multiple Columns

Promote David Ford (staffNo='SG14') to Manager and change his salary to £18,000.

```
UPDATE Staff
SET position = 'Manager', salary = 18000
WHERE staffNo = 'SG14';
```

DELETE

DELETE FROM TableName
[WHERE searchCondition]

- ***TableName*** **can be name of a base table or an updatable view.**
- ***searchCondition*** **is optional; if omitted, all rows are deleted from table. This does not delete table. If** ***search_condition*** **is specified, only those rows that satisfy condition are deleted.**

Example 5.41/42 DELETE Specific Rows

Delete all viewings that relate to property PG4.

```
DELETE FROM Viewing
WHERE propertyNo = 'PG4';
```

Delete all records from the Viewing table.

```
DELETE FROM Viewing;
```

Thank you for your attention

Questions????