
MAHARISHI UNIVERSITY of MANAGEMENT
Engaging the Managing Intelligence of Nature
Computer Science Department
CS401 Modern Programming Practices (MPP)

Lecture 7: DDL & DML

Wholeness Statement

SQL is a non-procedural language that can be used by professionals and non-professionals alike. It is both a formal and de facto standard language for defining and manipulating relational databases.

Science & Technology of Consciousness: TM is a simple, effortless mental technique that can be used by anyone, no matter what their lifestyle. It promotes spontaneous fulfillment of desires, by bringing the desires of the individual into accord with Natural Law, without the individual having to know the underlying mechanism.

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

Introduction to SQL

- SQL (Structured Query Language) is a standard language used to interact with relational databases. It is used to store, retrieve, manipulate, and manage data efficiently.

- Why Use SQL?
  - Data Retrieval: Extract data from databases using queries.
  - Data Manipulation: Insert, update, and delete records.
  - Database Management: Define schemas, create and modify tables.
  - Data Control: Manage user access and permissions.

SQL Commands

- SQL commands are divided into five main categories:
  - DDL (Data Definition Language) – Defines and manages database structures.
  - DML (Data Manipulation Language) – Modifies and manipulates data.
  - DCL (Data Control Language) – Controls user access.
  - TCL (Transaction Control Language) – Manages transactions.
  - DQL (Data Query Language) – Used to query data from the database.

SQL Identifier

- A SQL identifier is the name assigned to a table, column, view, etc.
  - The identifier is a string of characters from some set.
  - The string is at most 128 characters long.
  - The standard set of characters consists of capital letters (A,B,..), small letters (a,b,...), digits (0,1,...) and the underscore character (_).
  - An identifier starts with a letter.
  - An identifier contains no spaces (allowed by Access but not by standard SQL).

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

DDL (Data Definition Language)

- DDL commands define and modify database structures such as tables, indexes, and schemas.

Command	Description
CREATE	Creates a new database object (table, view, schema, etc.).
ALTER	Modifies an existing database structure.
DROP	Deletes database objects permanently.
TRUNCATE	Removes all records from a table but keeps its structure.
RENAME	Renames database objects.

CREATE Command - Creating a Database

- The CREATE command is used to create a new database object.
- Creating a Database
  - The CREATE DATABASE statement in SQL is used to create a new database.
  - Basic Syntax

```
CREATE DATABASE database_name;
```

  - Example

```
CREATE DATABASE company_db;
```

CREATE Command - Creating a Table

- The CREATE TABLE statement in SQL is used to define a new table in a database.
- Basic Syntax

```
CREATE TABLE table_name (
    column1 datatype [constraints],
    column2 datatype [constraints],
    column3 datatype [constraints],
    ...
);
```

```
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    salary DECIMAL(10,2) CHECK (salary > 0),
    department_id INT,
    hire_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Data Types and Constraints

Component	Description
INT	Integer data type
VARCHAR(n)	Variable-length string (max n characters)
DECIMAL(p,s)	Fixed-point number with p digits and s decimal places
DATE	Stores date values (YYYY-MM-DD)
PRIMARY KEY	Uniquely identifies each row
NOT NULL	Ensures column cannot have NULL values
CHECK(condition)	Enforces condition on column values
DEFAULT	Assigns a default value if none is provided

ALTER Command

- The ALTER command in SQL is used to modify an existing database object, such as a table, column, or constraint, without deleting it.
- Basic Syntax

```
ALTER TABLE table_name
MODIFY column_name datatype [constraints];
```

- Common operations:
  - Add a Column
  - Modify Column Data Type
  - Rename a Column
  - Drop a Column

Examples

- **Adding a new column:**

```
ALTER TABLE Employees ADD phone_number VARCHAR(15);
```

- **Modifying an existing column:**

```
ALTER TABLE Employees MODIFY salary DECIMAL(12,2) NOT NULL;
```

- **Renaming a Column**

```
ALTER TABLE Employees RENAME COLUMN hire_date TO joining_date;
```

- **Deleting a column:**

```
ALTER TABLE Employees DROP COLUMN phone_number;
```

DROP TABLE Command

- Used to delete a table and all its data permanently.
- Syntax:

```
DROP TABLE table_name;
```

- Example:

```
DROP TABLE Employees;
```

TRUNCATE TABLE Command

- Deletes all records from a table but retains the table structure.
- Faster than DELETE because it does not log individual row deletions.
- Syntax:

```
TRUNCATE TABLE table_name;
```

- Example:

```
TRUNCATE TABLE Employees;
```

RENAME TABLE Command

- Used to rename an existing table.
- Syntax:

```
RENAME TABLE old_table_name TO new_table_name;
```

- Example:

```
RENAME TABLE Employees TO Staff;
```

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

Data Manipulation Language (DML) Statements

DML (Data Manipulation Language) is a subset of SQL (Structured Query Language) used to manage and manipulate data stored in a database. Unlike DDL (Data Definition Language), which focuses on defining and modifying the structure of the database (e.g., creating tables), DML is concerned with working with the data itself.

- **SELECT** – to query data in the database
- **INSERT** – to insert data into a table
- **UPDATE** – to update data in a table
- **DELETE** – to delete data from a table

SELECT Statement

- The purpose of the SELECT statement is to retrieve and display data from one or more database tables.
- It's an extremely powerful and most used command capable of performing Selection, Projection and Join operations in a single statement!

General Form of SELECT Statement

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

- Only SELECT and FROM are mandatory.

Order of the clauses in a SELECT statement cannot be changed.

Example of All Columns, All Rows

- List full details of all staff.

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

- Can use * as an abbreviation for 'all columns':
  - **SELECT * FROM Staff;**

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 of 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;
```

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 of Use of DISTINCT

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

```
SELECT propertyNo
FROM Viewing;
```

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

propertyNo
PA14
PG4
PG4
PA14
PG36

- Use DISTINCT to eliminate duplicates
- DISTINCT can be specified only once in a query.

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

propertyNo
PA14
PG4
PG36

Sequence of Processing in a SELECT Statement

FROM **Specifies table(s) to be used**
(Produces result set which is Cartesian product of tables listed in FROM clause)

WHERE **Filters rows subject to some condition**
(Produces result set consisting of rows that satisfy the given condition)

GROUP BY **Forms groups of rows with same column value**

HAVING **Filters the groups subject to some condition**

SELECT **Specifies which columns are to appear in the output**

ORDER BY **Specifies the order of the output**

Example of Calculated Fields

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

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

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

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

- To name column, use AS clause:

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

staffNo	fName	lName	sal
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 of Comparison Search Condition

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

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

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

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 of Compound Comparison Search Condition

- **List addresses of all branch offices in London or Glasgow.**

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

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

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 of Set Membership (search condition)

List all managers and supervisors.

```
SELECT staffNo, fName, lName, position FROM Staff
WHERE position='Manager' OR position='Supervisor';
```

OR

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

There is a negated version (NOT IN).

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Example of Set Membership (search condition)

- There is a negated version (NOT IN).
- IN does not add much to SQL's expressive power. Could have expressed this as:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position='Manager' OR
      position='Supervisor';
```

- **IN is more efficient when set contains many values.**

Example of Pattern Matching

- SQL has two special pattern matching symbols:
  - % **: sequence of zero or more characters;**
  - _ (underscore)**: any single character – wild card.**

- LIKE '%Glasgow%' means a sequence of characters of any length containing '*Glasgow*'.

Example of Pattern Matching (search condition)

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

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

PrivateOwner

ownerNo	fName	lName	address	telNo	branchNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	B007
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	B003
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	B003
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	B003

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 of NULL Search Condition

- **List details of all viewings on property PG4 where a comment has not been supplied.**

- There are 2 viewings for property PG4, one with and one without a comment.
- Have to test for null explicitly using special keyword *IS NULL*:

SELECT clientNo, viewDate FROM Viewing
WHERE propertyNo = 'PG4' AND comment IS NULL;

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

clientNo	viewDate
CR56	26-May-04

Negated version (IS NOT NULL) can test for non-null values.

Example of Single Column Ordering (Sorting)

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

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

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

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

Example of Multiple Column Ordering

- Four flats in this list - as no minor sort key specified, system arranges these rows in any order it chooses.
- **To arrange in order of rent, specify minor order:**

```
SELECT propertyNo, type, rooms, rent
FROM PropertyForRent
ORDER BY type, rent DESC;
```

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

SELECT Statement - Aggregates

- While retrieving rows and columns from the database, we often want to perform summation or aggregation of data, similar to totals at the bottom of a report.
- 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

contd…

- 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.
- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.
- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.
- Can use DISTINCT before column name to eliminate duplicates.
- DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG.

Example of Use of COUNT(*)

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

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

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

myCount
5

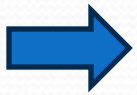
Example of Use of COUNT(DISTINCT)

- How many different properties viewed in May'13?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount
FROM Viewing
WHERE viewDate BETWEEN '1-May-13'
                   AND '31-May-13';
```

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	



myCount
2

Example of 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';
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

myCount	mySum
2	54000.00

Example of Use of MIN, MAX, AVG

- Find minimum, maximum, and average staff salary.

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

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

myMin	myMax	myAvg
9000.00	30000.00	17000.00

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

SELECT Statement - Grouping

- SELECT and GROUP BY are closely integrated. When GROUP BY is used, each item in the SELECT list must be "single-valued per group".
- All column names in SELECT list must appear in GROUP BY clause unless that column name is used only in an aggregate function.
- When GROUP BY is used, the SELECT clause may only contain:
  - column names
  - aggregate functions
  - constants
  - expression involving combinations of the above
- If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.
- If the GROUP BY clause is omitted when an aggregate function is used, then the entire table is considered as one group, and the aggregate function displays a single value for the entire table.

Example of Use of GROUP BY

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

```
SELECT branchNo,
       COUNT(*) AS staffCount,
       SUM(salary) AS sumOfSalary
FROM Staff
GROUP BY branchNo
ORDER BY branchNo;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

	branchNo	staffCount	sumOfSalary
▶	B003	3	54000.00
	B005	2	39000.00
	B007	1	9000.00

Use GROUP BY clause to get sub-totals in reports.

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 they should be contained within an aggregate function.
- Search condition in the HAVING clause always includes at least one aggregate function; otherwise the search condition could be moved to the WHERE clause and applied to individual rows.

Example of 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;
```

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

SELECT Statement - Aggregates

- Aggregate functions "cannot" be used in the WHERE clause. (WHERE sal > AVG(sal) is wrong!)
- **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, then SELECT list **cannot** include some other column as well as that aggregate function.
  - For example, the following is illegal:

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

X

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

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 of 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');
```

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

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

Example of 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';
```

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

Example of 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);
```

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

Example of Subquery with Aggregate Cont.

- 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;
```

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

Lesson Outline

1. Introduction to SQL commands
2. DDL
3. DML
    1. SELECT
    2. Aggregates
    3. Grouping
    4. Subqueries
    5. Join

Multi-Table Queries

- Can use subqueries provided result columns come from same table.
- If result columns come from more than one table then 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).
- 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.

SQL Joins

- SQL joins are used to combine rows from two or more tables based on a related column between them.
- INNER JOIN
  - Returns only rows where there's a match in both tables
- LEFT JOIN (or LEFT OUTER JOIN)
  - Returns all rows from the left table and matched rows from the right table
- RIGHT JOIN (or RIGHT OUTER JOIN)
  - Returns all rows from the right table and matched rows from the left table
- FULL JOIN (or FULL OUTER JOIN)
  - Returns all rows when there's a match in either table

Example of Simple Join (Inner 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;

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

- Only those rows from both tables that have identical values in the clientNo columns (c.clientNo = v.clientNo) are included in result.

Alternative JOIN Construct used in SQL Server

- **SQL provides alternative ways to specify joins:**

```
FROM Client c JOIN Viewing v ON c.clientNo = v.clientNo

FROM Client c INNER JOIN Viewing v
                    ON c.clientNo = v.clientNo
```

Example of Sorting a join

- For each branch, list numbers and names of staff who manage properties, and properties that they manage.

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

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

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

Outer join

- Typically, you create joins that return rows only if they satisfy join conditions; these are called **Inner Joins** and are the default join used when querying.
- However, sometimes you may want to preserve all the rows in one table.
- To display rows in the result that do not have matching values in the join column, use **Outer Join.**
- Advantage of outer join is that the information is preserved; that is, the Outer join preserves tuples that would have been lost by other types of join.

Outer Joins

- Result table has two rows where cities are same.
- There are no rows corresponding to the branch in Bristol and the property in Aberdeen.
- To include unmatched rows in result table, use an Outer join.

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Outer Joins

- In Inner join, if one row of a joined table is unmatched, row is omitted from result table.
- Outer join operations retain rows that do not satisfy the join condition.

- Consider these tables:
- The (inner) join of these two tables:

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

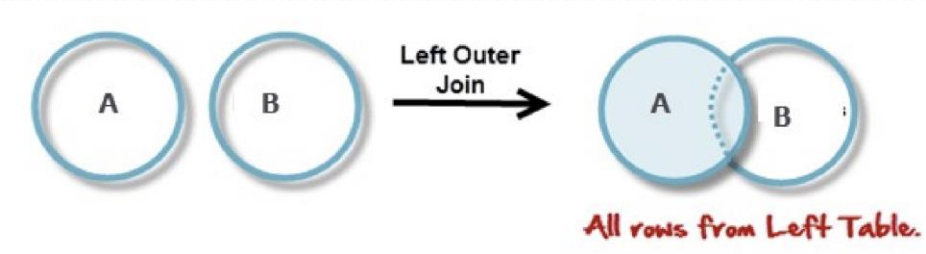
PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

```
SELECT b.*, p.*
FROM Branch1 b, PropertyForRent1 p
WHERE b.bCity = p.pCity;
```

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Left Outer join



- R ⟕ S
  - (Left) outer join is a join in which tuples from R(A) that do not have matching values in common columns of S(B) are also included in result relation.
  - Missing values in the 2nd relation are set to null.
  - One occurrence of each common attribute is eliminated from the result.

Example - Left Outer join

- **Produce a status report on property viewings.**

$$\Pi_{\text{propertyNo, street, city}}(\text{PropertyForRent}) \mathbin{⟕} \text{Viewing}$$

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

propertyNo	street	city	clientNo	viewDate	comment
PA14	16 Holhead	Aberdeen	CR56	24-May-01	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-01	no dining room
PL94	6 Argyll St	London	null	null	null
PG4	6 Lawrence St	Glasgow	CR76	20-Apr-01	too remote
PG4	6 Lawrence St	Glasgow	CR56	26-May-01	
PG36	2 Manor Rd	Glasgow	CR56	28-Apr-01	
PG21	18 Dale Rd	Glasgow	null	null	null
PG16	5 Novar Dr	Glasgow	null	null	null

Example - Left Outer Join

- **List branches and properties that are in same city along with any unmatched branches.**

```
SELECT b.*, p.*
FROM Branch1 b LEFT JOIN
        PropertyForRent1 p ON
                b.bCity = p.pCity;
```

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

Includes those rows of first (left) table unmatched with rows from second (right) table.
Columns from second table are filled with NULLs.

Right Outer Join

- $R \overline{\bowtie} S$
- Keeping all the tuples from the right relation S.
- However, if no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.

Example - Right Outer Join

- **List branches and properties in same city and any unmatched properties.**

```
SELECT b.*, p.*
FROM Branch1 b RIGHT JOIN
  PropertyForRent1 p ON
    b.bCity = p.pCity;
```

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Right Outer join includes those rows of second (right) table that are unmatched with rows from first (left) table.
Columns from first table are filled with NULLs.

Example - Full Outer Join

- **List branches and properties in same city and any unmatched branches or properties.**

```
SELECT b.*, p.*
FROM Branch1 b FULL JOIN
     PropertyForRent1 p ON
     b.bCity = p.pCity;
```

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

- Includes rows that are unmatched in both tables.
- Unmatched columns are filled with NULLs.

MySQL doesn't support Full Join.