



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

Introduction to SQL

**SQL Commands (DDL, DML, DCL, TCL, DQL): Types, Syntax,
and Examples**

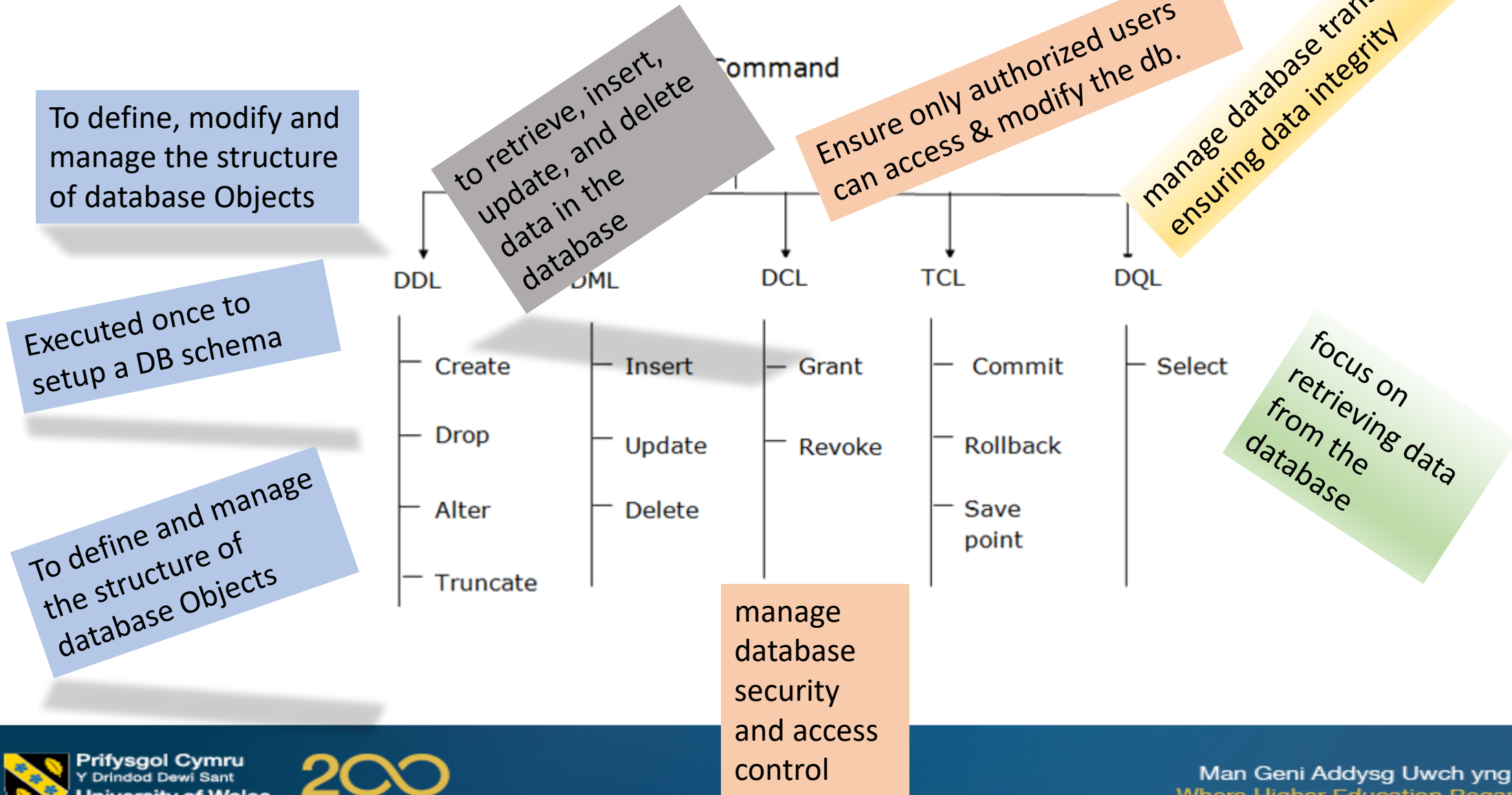
*Man Geni Addysg Uwch yng Nghymru
Where Higher Education Began in Wales*

www.uwtsd.ac.uk

Objectives

- Alter table to include primary key and foreign key
- How to retrieve data from the database using the SELECT statement.
- How to build SQL statements that:
 - use the WHERE clause to retrieve rows that satisfy various conditions;
 - sort query results using ORDER BY;
 - use the aggregate functions of SQL;
 - group data using GROUP BY;
 - use subqueries;

Categorization of SQL Commands



Activity

- CREATE DATABASE called dream home
- Create the following tables :
 - Branch (branchNo, street, city, postcode)
 - Staff (staffNo, fName, IName, position, sex, DOB, salary, branchNo)
 - PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo,branchNo)
 - Client (clientNo, fName, IName, telNo, prefType, maxRent, eMail)
 - PrivateOwner (ownerNo, fName, IName, address, telNo, eMail, password)
 - Viewing (clientNo, propertyNo, viewDate, comment)

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

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

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

PrivateOwner

ownerNo	fName	lName	address	telNo	eMail	password
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	jkeogh@lhh.com	*****
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	cfarrel@gmail.com	*****
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	tinam@hotmail.com	*****
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	tony.shaw@ark.com	*****

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	

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-13
CR56	B003	SG37	11-Apr-12
CR74	B003	SG37	16-Nov-11
CR62	B007	SA9	7-Mar-12

DDL Commands in SQL with Examples

SQL Command	Code Snippet	Output
CREATE TABLE	CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), Department VARCHAR(50));	New “Employees” table created with specified columns.
ALTER TABLE	ALTER TABLE Employees ADD Email VARCHAR(100);	“Email” column added to the “Employees” table.
DROP TABLE	DROP TABLE Employees;	“Employees” table and its data deleted.

DML Commands in SQL with Examples

SQL Command	Code Snippet	Output
SELECT	SELECT FirstName, LastName FROM Employees WHERE Department = 'Sales';	Retrieves the first and last names of employees in the “Sales” department.
INSERT	INSERT INTO Employees (FirstName, LastName, Department) VALUES ('John', 'Doe', 'HR');	New employee record added to the “Employees” table.
UPDATE	UPDATE Employees SET Salary = Salary * 1.1 WHERE Department = 'Engineering';	Salary of employees in the “Engineering” department increased by 10%.
DELETE	DELETE FROM Employees WHERE Department = 'Finance';	Employees in the “Finance” department deleted.

Literals

- Literals are constants that are used in SQL statements.
- All nonnumeric data values must be enclosed in single quotes;
- all numeric data values need not be enclosed in single quotes.

SQL CONSTRAINTS

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified

To add a primary key or foreign key to an existing table, you can use the **ALTER TABLE** statement in SQL.

Here are the **basic syntax** for adding a primary key and a foreign key:

Add Primary Key:

```
ALTER TABLE table_name
```

```
ADD CONSTRAINT constraint_name PRIMARY KEY (PK_column);
```

Add Foreign Key:

```
ALTER TABLE child_table
```

```
ADD CONSTRAINT constraint_name
```

```
FOREIGN KEY (child_column)
```

```
REFERENCES parent_table(parent_column);
```

To add a primary key or foreign key to an existing table, you can use the **ALTER TABLE** statement in SQL.

Here are the **examples** for adding a primary key and a foreign key:

Add Primary Key:

```
CREATE TABLE Orders ( OrderID INT, ProductName VARCHAR(50), Quantity INT, CustomerID INT );
```

-- Add a primary key to the 'Orders' table

```
ALTER TABLE Orders ADD CONSTRAINT PK_Orders PRIMARY KEY (OrderID);
```

Table name

Constraint name

Column name

To add a primary key or foreign key to an existing table, you can use the **ALTER TABLE** statement in SQL.

Here are the **examples** for adding a primary key and a foreign key:

Add Foreign Key:

-- Assume we have an existing 'Customers' table with a primary key on 'CustomerID'

```
CREATE TABLE Customers ( CustomerID INT PRIMARY KEY, CustomerName VARCHAR(50), Email VARCHAR(50) );
```

Orders

OrderID	ProductName	Quantity	CustomerID
---------	-------------	----------	------------

-- Add a foreign key to the 'Orders' table that references the 'CustomerID' column in 'Customers'

```
ALTER TABLE Orders
```

```
ADD CONSTRAINT FK_Orders_Customers
```

```
FOREIGN KEY (CustomerID)
```

```
REFERENCES Customers(CustomerID);
```

Exercise

Retrieve all columns, all rows

Query: List full details of all staff.

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

Or

```
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

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

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

staffNo	fName	IName	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

Use of DISTINCT

Query: List the property numbers of all properties that have been viewed

SELECT propertyNo **FROM** Viewing;

SELECT DISTINCT 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

PG36

Calculated fields

Query: Produce a list of monthly salaries for all staff, showing the staff number, the first and last names, and the salary details.

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

staffNo	fName	IName	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

Staff

staffNo	fName	IName	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

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

Row selection (WHERE clause)

- *Comparison:* Compare the value of one expression to the value of another expression.
- *Range:* Test whether the value of an expression falls within a specified range of values.
- *Set membership:* Test whether the value of an expression equals one of a set of values.
- *Pattern match:* Test whether a string matches a specified pattern.
- *Null:* Test whether a column has a null (unknown) value.

In SQL, the following simple comparison operators are available:

= equals

<> is not equal to (ISO standard) != is not equal to (allowed in some dialects)

< is less than <= is less than or equal to

> is greater than >= is greater than or equal to

The rules for evaluating a conditional expression are:

- an expression is evaluated left to right;
- subexpressions in brackets are evaluated first;
- NOTs are evaluated before ANDs and ORs;
- ANDs are evaluated before ORs.

The use of parentheses is always recommended, in order to remove any possible ambiguities.

Comparison search condition

Query: List all staff with a salary greater than £10,000

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

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

Compound comparison search condition

Query: List the addresses of all branch offices in London or Glasgow.

```
SELECT *
```

```
FROM Branch
```

```
WHERE city = 'London' OR city = 'Glasgow';
```

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

Range search condition (BETWEEN/NOT BETWEEN)

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

```
SELECT staffNo, fName, IName, position, salary
```

```
FROM Staff
```

```
WHERE salary BETWEEN 20000 AND 30000;
```

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

Set membership search condition (IN/NOT IN)

Query: List all managers and supervisors.

SELECT staffNo, fName, IName, position

FROM Staff

WHERE position IN ('Manager', 'Supervisor');

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

SELECT staffNo, fName, IName, position

FROM Staff

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

Pattern match search condition (LIKE/NOT LIKE)

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

- The % percent character represents any sequence of zero or more characters (*wildcard*).
- The _ underscore character represents any single character.

For example:

- address LIKE 'H%' means the first character must be *H*, but the rest of the string can be anything.
- address LIKE 'H___' means that there must be exactly four characters in the string, the first of which must be an *H*.
- address LIKE '%e' means any sequence of characters, of length at least 1, with the last character an *e*.
- address LIKE '%Glasgow%' means a sequence of characters of any length containing *Glasgow*.
- address NOT LIKE 'H%' means the first character cannot be an *H*.

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

ownerNo	fName	IName	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

NULL search condition (IS NULL/IS NOT NULL)

Query: List the details of all viewings on property PG4 where a comment has not been supplied.

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-13

Sorting Results (ORDER BY Clause)

Single-column ordering

Query: Produce a list of salaries for all staff, arranged in descending order of salary.

```
SELECT staffNo, fName, IName, salary
```

```
FROM Staff
```

```
ORDER BY salary DESC;
```

staffNo	fName	IName	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

Multiple column ordering

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

Note: the result is ordered first by property type,
in ascending alphabetic order (ASC being the default setting),
and within property type, in descending order of rent.

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

Using the SQL Aggregate Functions

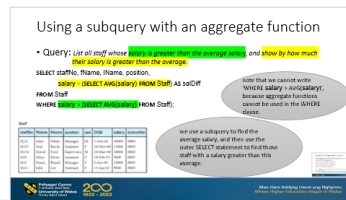
In addition to retrieving rows and columns from the database, we often want to perform some form of summation or **aggregation** of data, similar to the totals at the bottom of a report.

The ISO standard defines five **aggregate functions**:

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

Note: **COUNT(*)** is a special use of COUNT that counts all the rows of a table, regardless of whether nulls or duplicate values occur.

Further reading: Connolly,T.,&begg,C.(2005). *Database systems: a practical approach to design, Implementation and management*, Boston, MA: pearson Addison-Wesley(pg No 207)



Grouping Results (GROUP BY Clause)

Grouped Query: Find the number of staff working in each branch and the sum of their salaries

```
SELECT branchNo, COUNT(staffNo) AS myCount, SUM(salary) AS mySum
```

```
FROM Staff GROUP BY branchNo ORDER BY branchNo;
```

Conceptually, SQL performs the query as follows:

branchNo	staffNo	salary		COUNT(staffNo)	SUM(salary)
B003	SG37	12000.00	}	3	54000.00
B003	SG14	18000.00			
B003	SG5	24000.00			
B005	SL21	30000.00	}	2	39000.00
B005	SL41	9000.00			
B007	SA9	9000.00	}	1	9000.00

we could also express the previous query as:

```
SELECT branchNo, (SELECT COUNT(staffNo) AS myCount
FROM Staff s
WHERE s.branchNo = b.branchNo),
(SELECT SUM(salary) AS mySum
FROM Staff s
WHERE s.branchNo = b.branchNo)
FROM Branch b
ORDER BY branchNo;
```

Grouping Results (GROUP BY Clause)

Grouped Query: Find the number of staff working in each branch and the sum of their 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

Restricting groupings (HAVING clause)

- The HAVING clause is designed for use with the GROUP BY clause to restrict the **groups** that appear in the final result table.
- The WHERE clause filters individual rows going into the final result table,
- whereas HAVING filters **groups** going into the final result table.
- **Grouped Query:** For each branch office *with more than one member of staff*, find the number of staff working 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

exercise 1 • SELECT propertyNO, type, rent FROM propertyForRent; • SELECT * FROM branch; • SELECT fname, lname, telNo from client;	-- exercise 2 • SELECT propertyNo, street, city, type, rent AS monthlyRent, <u>rent *12</u> AS annualRent FROM propertyForRent; Calculated fields
-- exercise 3 • SELECT CONCAT(lName, ", ", fName) AS fullName FROM staff;	-- exercise 4 • SELECT * FROM propertyForRent WHERE rent <= 400; • SELECT clientNo, fName,lName FROM client WHERE preftype = "house" ; • SELECT * FROM viewing WHERE viewDate > "2015-05-15" ;
-- exercise 5 • SELECT fName, lName, branchNo FROM staff WHERE position = "assistant" AND salary = 9000; • SELECT * FROM propertyForRent WHERE RENT < 400 AND CITY ="Glasgow" ; • --SELECT * FROM propertyForRent WHERE RENT < 400 AND CITY = "Glasgow" ; • SELECT * FROM viewing WHERE viewDate >= '2015-04-01' AND viewDate <= '2015-04-31';	---exercise 6 • SELECT * FROM viewing ORDER BY ViewDate DESC; • SELECT * FROM viewing ORDER BY clientNo, ViewDate;

Subqueries

```
SELECT branchNo, (SELECT COUNT(staffNo) AS myCount  
FROM Staff s  
WHERE s.branchNo = b.branchNo),  
(SELECT SUM(salary) AS mySum  
FROM Staff s  
WHERE s.branchNo = b.branchNo)  
FROM Branch b  
ORDER BY branchNo;
```

Using a subquery with equality

Query: *List the staff who work in the branch at '163 Main St'.*

```
SELECT staffNo, fName, IName, position
FROM Staff
WHERE branchNo = (SELECT branchNo
```

```
FROM Branch
```

```
WHERE street = '163 Main St');
```

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

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

Output:



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	IName	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

Using a subquery with an aggregate function

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

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

note that we cannot write
'WHERE salary > AVG(salary)',
because aggregate functions
cannot be used in the WHERE
clause.

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

we use a subquery to find the
average salary, and then use the
outer SELECT statement to find those
staff with a salary greater than this
average.

Using the SQL Aggregate Functions

In addition to retrieving rows and columns from the database, we often want to perform some form of summarization or aggregation of data. Before we look at the details of the various aggregate functions, let's first look at the SQL standard names for aggregate functions.

- COUNT – returns the number of rows in a specified column
- SUM – returns the sum of the values in a specified column
- AVG – returns the average of the values in a specified column
- MIN – returns the smallest value in a specified column
- MAX – returns the largest value in a specified column

Note: COUNT(*) is a special case of COUNT that counts all the rows of a table, regardless of whether null or duplicate values occur.

Further reading: [Database Systems: A Practical Approach to Design, Implementation and Management](#), Boston, MA: Pearson Addison-Wesley Inc. 2003

Using a subquery with an aggregate function

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

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

The subquery returns the average salary as £17,000.



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

Using a subquery with an aggregate function

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

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	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00

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

Output:



Further reading

- Chapter 6 from Connolly,T.,&begg,C.(2005). Database systems: a practical approach to design, Implementation and management, Boston, MA: pearson Addison-Wesley

Next Session

- join tables together;
- perform set operations (UNION, INTERSECT, EXCEPT).