Database Fundamentals

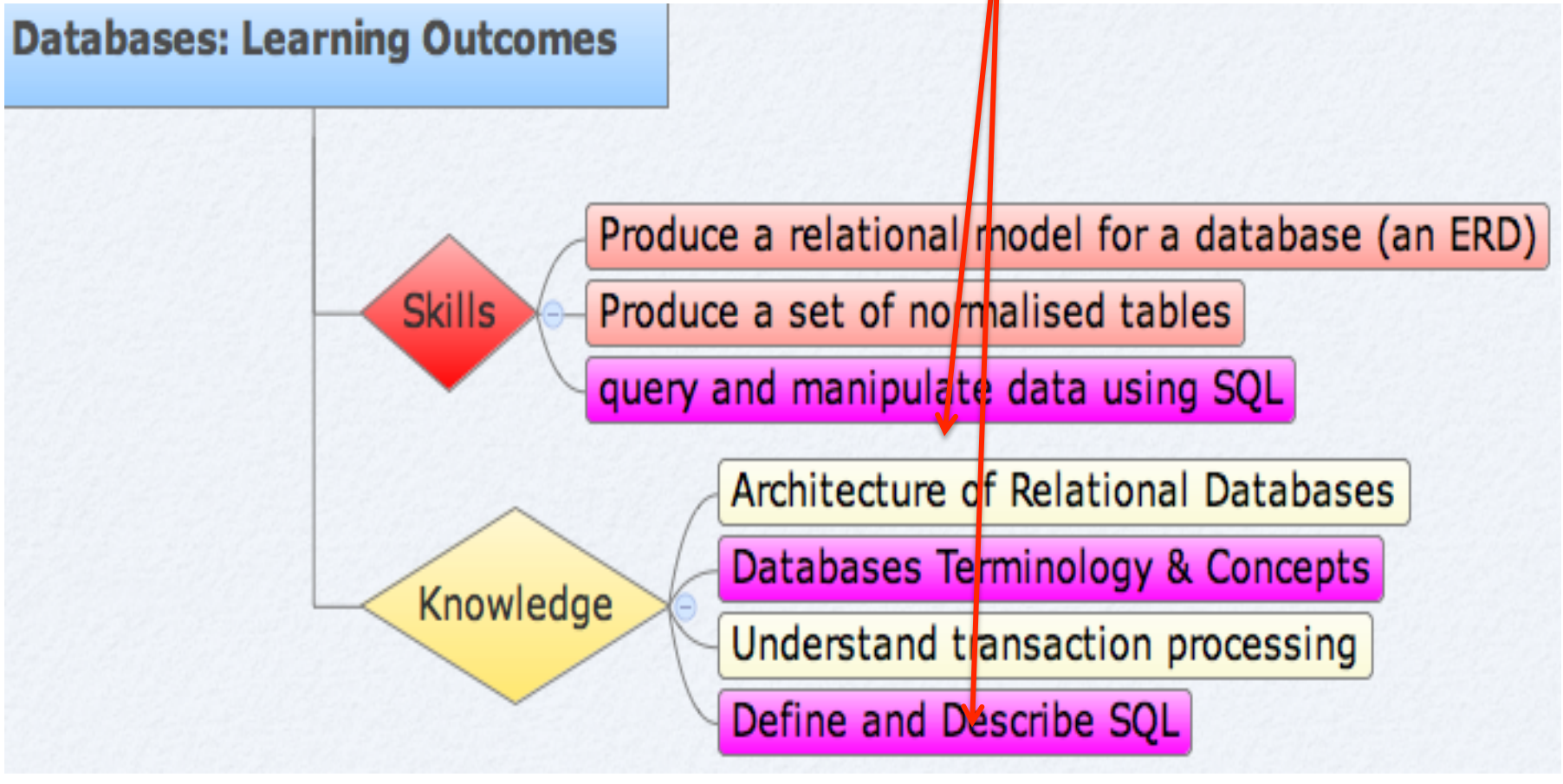
Lecture 3 (More SQL)

Lecturer : Dr Irene Murtagh

Room :A15

Email: irene.murtagh@tudublin.ie

Learning Outcomes



Recap

- **Select overview**
  - Projection – select columns
  - Selection: select rows
  - Join: join columns
- **Select column:**
  - SELECT [DISTINCT] {*, column_name [alias], ...} FROM table name
  - SELECT empno, name FROM emp
  - Derived columns: SELECT ename, sal*12 AS 'total salary' FROM emp
- **Select rows:**
  - add a WHERE clause, examples below
  - WHERE ename = 'SCOTT'
  - Where sal > 2000
  - where ename like '%T'
  - where sal between 1000 and 2000
  - where detno in [20,30] AND sal > 1500
- **Sorting output**
  - ORDER BY clause – must be last clause
  - SELECT ename FROM emp ORDER BY sal
- **TOP N**
  - select TOP 10 ename FROM emp ORDER BY sal

Objective for this lecture:

- Continue with SQL Select clause (jump back to lecture 2 notes from slide 44 onwards)
- **Functions** in SQL
  - Single Row functions
  - Group (Aggregate) Functions
- Combining clauses done to date

Exercise – using a different table.
Table name: student

StudentID	Surname	Course	Age	County
B00012345	Murphy	BN002	19	Dublin 15
B00034593	Doyle	BN104	34	Meath
B00043894	Hope	BN002	27	Dublin 7
B00345640	Keily	BN103	19	Dublin 15

- Which students are enrolled in the Certificate in Computing (BN002)?
- We need to send a mail shot to students not living in Dublin 15. List all such students.
- Are any students under 18?

Wildcard searches using LIKE
(for strings - varchar)

- Use LIKE operator to find patterns in strings (varchar). Typically look for a specific pattern in the values of the rows in a given table
- Condition can contain
  1. characters or number
  2. % denotes a string of any length (0 or more)
  3. _ (underscore) denotes any single character
  4. search for a character in a range [a–e] (any character between a and e)
  5. search for a set of characters [abc] (character must be either a, b or c)

Get all employees whose name <u>begins</u> with the letter S

```
SELECT ename
FROM emp
WHERE ename LIKE 'S%';
```

Get all employees whose name <u>ends</u> with the letter S

```
SELECT ename
FROM emp
WHERE ename LIKE '%S';
```

Get all employees whose name <u>contains</u> the character sequence ae

```
SELECT ename
FROM emp
WHERE ename LIKE '%ae%';
```

Find all employees whose name begins with any character and the last four characters are ***anet*** ***(5 letters in name)***

```
SELECT ename
FROM emp
WHERE ename LIKE '_anet';
```

Find all employees whose name begins either with J or S

```
SELECT ename
FROM emp
WHERE ename LIKE '[js]%';
```

https://dev.mysql.com/doc/refman/5.1/en/pattern-matching.html

Limitations of MySQL

- Note: currently MySQL only supports using % and _ with the like operator, but not character ranges such as [a-e]. To do more complex pattern matching with MySQL, you need to use the REGEXP operator instead of LIKE. This will interpret patterns based on Java's Regular Expression (REGEX) syntax. This is outside the scope of the module, but as an example, a query looking for all employee names beginning with J or S would be written as:

```
Select ename
From emp
Where ename REGEXP '^[JS]'
```

Exercises

- List all employees working in a department whose number starts with the digits '76'
- Which employees have the letter A in their name?
- Which employees have names that start with a letter from the first half of the alphabet (A to M)?
- Which names start with the letters B or C?

Exercise – Product table of furniture available in IKEA

ProductID	Description	Range	Aisle	Section
10203097	TV Bench	BENNO	18	A
30133942	TV Bench with Panel	BENNO	18	B
70104438	TV Bench	**GREVBÄCK**	18	F
60105339	Corner TV Bench	**LACK**	18	D
10065958	Coffee Table	LACK	15	H
07305310	DVD Tower	BENNO	2	J

- Based on the table above, write queries for the following:
  - List all products in the BENNO range
  - List all products in Aisle's 15 or higher
  - List all products in sections A, B and C
  - List all the types of TV Benches available

Combining conditions

Where clauses can include more than one condition, conditions can be joined using logical operators: AND, OR

Operators are evaluated in the following order - Rules of Precedence

1. Comparison Operators
2. NOT
3. AND
4. OR

Use parentheses / brackets to override rules

Example – combining conditions

Display the employee number, name, position and salary of all employees earning at least 1100 per month employed as clerks.

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal>=1100
AND job='CLERK';
```

Rows meeting both search conditions are returned.

Example – combining conditions

List the name, job and salary for staff who are employed either as salesmen or presidents and who earn more than 1500 a month.

```
SELECT ename, job, sal
FROM    emp
WHERE job='SALESMAN'
OR      job='PRESIDENT'
AND     sal>1500;
```

Example: Combining conditions

- Return all employees whose name begins with ‘b’ and are not employed as clerks, presidents or managers

```
SELECT empno, ename, job
FROM emp
WHERE ename LIKE ‘b%’
AND job NOT IN (‘clerk’, ‘manager’, ‘president’);
```

Exercise

- Display the name, job, and salary for all employees whose job is Clerk or Analyst and their salary is not equal to $1000, $3000, or $5000.

Derived Columns

- You can derive a new column from existing numeric or date fields using arithmetic expressions as shown below. This does not make any change to the database table itself.

SELECT ename, sal, (sal *12) +100 AS 'Annual Salary'

FROM emp;

```
ENAME              SAL  Annual Salary
---------- --------- -----------
KING              5000       60100
BLAKE             2850       34300
CLARK             2450       29500
JONES             2975       35800
MARTIN            1250       15100
ALLEN             1600       19300
...
```

Operator Precedence:

*, /, +, -

Renaming Column Headings

- In the last slide, the new column was display under a column heading of 'Total Salary'.
- The display name of a column can be changed in three ways:

SELECT ename AS NAME, sal SALARY, sal*12 AS 'Total Salary' FROM emp;

1. With AS

2. Without AS

3. If the new name includes a blank, then it must be in quotes

NAME	SALARY	Total Salary
---	---	---
KING	5000	60000
BLAKE	2850	34200
CLARK	2450	29400
JONES	2975	35700

Adding text to the output

- Characters, numbers or dates can be outputted as part of each row:

```
SELECT ename,' is a ', job AS 'Employee Details'
FROM   emp;
```

ename	is a	Employee Details
KING	is a	PRESIDENT
BLAKE	is a	MANAGER
CLARK	is a	MANAGER
JONES	is a	MANAGER
MARTIN	is a	SALESMAN
ALLEN	is a	SALESMAN
TURNER	is a	SALESMAN
JAMES	is a	CLERK

Sorting rows for output

- ORDER BY clause SORTS output, and should be the **last** clause in a SELECT statement

```
SELECT [DISTINCT] {*, column_name [alias], …}
FROM table
[WHERE condition(s)];
[ORDER BY {column, expression} [ASC | DESC]];
```

The default sort method is Ascending.

Sorting rows for output

Display each employee's name, position, department and the date they were hired. Sort the output in ascending order according to hiredate.

```
SELECT ename, job, deptno, hiredate
FROM   emp
ORDER BY   hiredate;
```

```
ENAME      JOB          DEPTNO HIREDATE
---------- --------- --------- ---------
SMITH      CLERK            20 17-DEC-80
ALLEN      SALESMAN         30 20-FEB-81
...
14 rows selected.
```

Sorting rows for output

Display each employee's name, position, department and the date they were hired. Sort the by date, with the most recently employed listed first.

```
SELECT ename, job, deptno, hiredate
FROM  emp
ORDER BY hiredate DESC;
```

```
ENAME      JOB           DEPTNO HIREDATE
---------- --------- --------- ---------
ADAMS      CLERK             20 12-JAN-83
SCOTT      ANALYST           20 09-DEC-82
MILLER     CLERK             10 23-JAN-82
JAMES      CLERK             30 03-DEC-81
FORD       ANALYST           20 03-DEC-81
KING       PRESIDENT         10 17-NOV-81
MARTIN     SALESMAN          30 28-SEP-81
...
14 rows selected.
```

Using multiple expressions in the ORDER BY clause

Display each employee's name, position, department and the date they were hired. Sort by job in alphabetical order with the most recently employed in each job category listed first.

```
SELECT ename, job,
   deptno, hiredate
FROM emp
ORDER BY job ASC,
   hiredate DESC;
```

ename	job	deptno	hiredate
SCOTT	ANALYST	20	1982-12-0
FORD	ANALYST	20	1981-12-0
ADAM	CLERK	20	1983-01-1
MILLE	CLERK	10	1982-01-2
JAMES	CLERK	30	1981-12-0
SMITH	CLERK	20	1980-12-1
CLARK	MANAGER	10	1981-06-0
BLAKE	MANAGER	30	1981-05-0
JONES	MANAGER	20	1981-04-0

How will the results of the following query be ordered?

```
SELECT ename, job, deptno, sal
FROM   emp
ORDER BY deptno DESC, sal;
```

ename	job	deptno	sal
JAMES	CLERK	30	950
WARD	SALESMAN	30	1250
MARTIN	SALESMAN	30	1250
TURNER	SALESMAN	30	1500
ALLEN	SALESMAN	30	1600
BLAKE	MANAGER	30	2850
SMITH	CLERK	20	800
ADAMS	CLERK	20	1100
JONES	MANAGER	20	2975
FORD	ANALYST	20	3000
SCOTT	ANALYST	20	3000
MILLER	CLERK	10	1300
CLARK	MANAGER	10	2450
KING	PRESIDENT	10	5000

Limit (or Top N) clause

- LIMIT is used to limit the number of rows returned by a query as follows

```
SELECT ename, sal
FROM emp
ORDER BY sal DESC
LIMIT 10;
```

Exercises

1. Give a list of the name and salary of each employee earning more that $1500, and working in department 10 or 30. Label the columns 'Employee' and 'Monthly Salary'
2. Show a list of unique salary values from the employee table in descending sequence
3. Show the top 3 salary values.
4. List each employees name, department, an their total earnings for the year where total earnings is (sal*12 + comm). Name the columns 'Employee', 'Department' and 'Yearly Earnings'

Summary

- **Select overview**
  - Projection – select columns
  - Selection: select rows
  - Join: join columns
- **Select column:**
  - SELECT [DISTINCT] {*, column_name [alias], ...} FROM table name
  - SELECT empno, name FROM emp
  - Derived columns: SELECT ename, sal*12 AS 'total salary' FROM emp
- **Select rows:**
  - add a WHERE clause, examples below
  - WHERE ename = 'SCOTT'
  - Where sal > 2000
  - where ename like '%T'
  - where sal between 1000 and 2000
  - where detno in [20,30] AND sal > 1500
- **Sorting output**
  - ORDER BY clause – must be last clause
  - SELECT ename FROM emp ORDER BY sal
- **TOP N**
  - select TOP 10 ename FROM emp ORDER BY sal

Objective for this lecture:

Continue with SQL Select clause

- **Functions** in SQL
  - Single Row functions
  - Group (Aggregate) Functions
- Combining clauses done to date

FUNCTIONS in SQL

- There are a number of functions available in SQL for manipulating data items to get the desired output from data in the database.

- These functions fall into two categories:
  - Single row functions – which work on ONE cell at a time
  - Multiple row functions (GROUP functions) – merge (group) a number of rows into one.

Single Row Functions

There are a range of functions available which can be used in SQL queries.

- Some function names are **NOT** consistent across different vendors, and so should be used sparingly.

1. **Numeric Functions**: e.g. round(), power(), log(), log2(), etc.
   - A full list of the MySQL numeric functions can be found at: http://dev.mysql.com/doc/refman/5.0/en/numeric-functions.html

2. **String Functions**: e.g. trim(), concat(), length(), substr()
   - A full list of the MySQL string functions can be found at: http://dev.mysql.com/doc/refman/5.0/en/string-functions.html

3. **Date & Time Functions**: e.g. curdate(), dateDiff(), timeDiff(), Date_Add() etc.
   - A full list of the MySQL string functions can be found at: http://dev.mysql.com/doc/refman/5.0/en/date-and-time-functions.html

Single Row Functions

- String / Character Functions – for manipulating strings (varchar)

Function	Result
CONCAT('Good', 'String')	GoodString
LEFT('String',3)	Str
FIND_IN_SET('r', 'S,t,r,l,n,g')	3
REPLACE('abcde', 'bc','oo')	aoode
LOWER('ABCD')	abcd
UCASE('abc')	ABC
TRIM(' dfgdf ')	dfgdf

Single Row Functions

- Number functions
  - ROUND(15.126) gives 15
  - SELECT ROUND(sal) FROM emp will display round all salary figures for display.
    - Syntax: **ROUND(column|expr)**

  - FLOOR (123.7), gives 123 (round down)
  - SELECT FLOOR (sal) FROM emp will round down all salary figures for display.
    - Syntax: **FLOOR(column|expr)**

  - POW(9,2) gives 81
  - SELECT POW(sal,2) FROM emp will square all salary figures before displaying them
    - Syntax: **POW(column|expr,y)**

Single Row Functions

- Date functions
  - CURDATE () returns current system data
  - DATEDIFF('2015-12-25', '2015-09-29'); gives the number of days between two dates
    - Syntax: **DATEDIFF(date1,date2)**

Example using single row functions

- How many years has Clark being working for the company?

```
SELECT ename, FLOOR((DATEDIFF(curdate(),
   hiredate))/365)
FROM emp
WHERE TRIM(ename) = 'Clark';
```

Remove leading and trailing blank spaces from ename before comparing it to 'Clark'

DATEDIFF calculates the number of DAYS between curdate (today's date) and hiredate. Divide by 365 to get years, and round down (FLOOR)

Exercises - In class

- Write a query which displays all employee names in uppercase, and the job name in lower case

- Calculate to the nearest year how many years each employee has worked with the company

Summarising data on tables

Called by three names:

Aggregating data

Group functions

Multiple row functions

Group Functions

- One of the key benefits of the SQL language is that it enables you to generate **summaries of the data** stored in the database, for example:
  - What's the total profit made by shops in Dublin?
  - How many **creme eggs** were sold in Dublin this year? And how does that compare to the total sales for this time last year?

- Group functions **return a result** after carrying out a function on a group of rows in the table

Group Functions

- What is the total salary paid out to employees?

```
SELECT      sum(sal)
FROM  emp;
```

This query will add up all the salaries in the emp table and return the result as follows:

sum(sal)
29025.00

Aggregated data

```
SAL
------
   800
  1600
  1250
  2975
  1250
  2850
  2450
  3000
  5000
  1500
  1100
   950
  3000
  1300
```

Types of Group Functions

- **AVG**([DISTINCT|ALL] n)
  - Returns the average value ignoring null values
- **COUNT**([DISTINCT|ALL] expr)
  - Returns the number of values – does not count null values
- **SUM**([DISTINCT|ALL] n)
  - Returns the sum of the values, ignoring null values

- **MAX**([DISTINCT|ALL] expr)
  - Returns the maximum value, ignoring null values
- **MIN**([DISTINCT|ALL] expr)
  - Returns the minimum value, ignoring null values

Examples – aggregating ALL rows in the table

What is the highest salary in the company?

```
SELECT MAX(sal)
FROM emp
```

max(sal)
5000.00

What is the minimum salary paid month?

```
SELECT MIN(sal)
FROM emp
```

min(sal)
800.00

What is the companies average commission?

```
SELECT AVG(comm)
FROM emp
```

avg(comm)
550.000000

```
SELECT ROUND(AVG(comm)) as Avg_Comm
FROM emp
```

avg_comm
550

What date was the first employee hired, and what date was the most recent employee hired?

```
SELECT MIN(hiredate), MAX(hiredate)
FROM emp;
```

min(hiredate)	max(hiredate)
1980-12-17	1983-01-12

Examples – aggregating SOME rows in the table

What is the highest salary among the salesmen?

As queries get more complicated, you need to build them up in stages.

What information do you need to answer this query?

1. Limit the query to just the salaries of sales people:

   Select sal from emp where job = 'Salesman'

2. From this list, select the highest salary, giving:

```
SELECT MAX(sal)
FROM emp
WHERE job = 'Salesman'
```

MAX(sal)
1600.00

Examples – aggregating SOME rows in the table

What is the total salary bill paid each month for employees hired in 1981?

```
SELECT SUM(sal)
FROM emp
WHERE hiredate like "1981%"
```

sum(sal)
22825.00

Find the average salary, the highest salary, the lowest salary and the total salary for clerks:

```
SELECT  AVG(sal), MIN(sal), MAX(sal), SUM(sal)
FROM emp
WHERE job ="Clerk";
```

avg(sal)	max(sal)	min(sal)	sum(sal)
1037.500	1300.00	800.00	4150.00

Return the number of rows in the Employee table

```
SELECT   COUNT(*)
FROM     emp;
```

COUNT(*)
14

How many employees are in department 30?

```
SELECT  COUNT(*)
FROM    emp
WHERE   deptno = 30;
```

COUNT(*)
6

Note:

- Count, max and min can be used on any data type
- Other functions can only be used on numeric data

How many employees earn commission?

```
SELECT COUNT(comm)
FROM emp;
```

COUNT(comm)
4

What is the average amount of commission earned by staff in department 30?

```
SELECT    AVG(comm)
FROM      emp
WHERE     deptno = 30;
```

AVG(comm)
550.000000

DISTINCT with aggregate function

- The DISTINCT keyword can be used in any aggregate function to **consider repeating values just once.**

- Find how many different job positions the employee table has:

```
SELECT count(DISTINCT job)
FROM emp;
```

count(DISTINCT job)
5

Exercise

- Calculate the total amount paid out in commission.
- How many employees have a salary greater than £1500?
- What is the average salary?
- What is the average salary rounded to the nearest whole number?
- How many distinct salary amounts are there?

GROUP BY clause

- The GROUP BY clause is used to group rows in a result set, generating a summary row for each group of data.
- All (non-aggregated) columns specified in the SELECT column list must also be specified in the GROUP BY clause
- However, columns specified in the GROUP BY clause don't have to be in the SELECT column list.

Note: this is the case in standard SQL. MySQL extends the GROUP BY clause so that the SELECT list can refer to non-aggregated columns named in the GROUP BY clause.

- What is the total salary paid out in each department?

```
SELECT      deptno, SUM(sal)
FROM        emp
GROUP BY    deptno;
```

deptno	SUM(sal)
10	8750.00
20	10875.00
30	9400.00

What is the average salary paid for each job type?

```
SELECT job, avg(sal)
FROM emp
GROUP BY  job;
```

job	avg(sal)
ANALYST	3000.000000
CLERK	1037.500000
MANAGER	2758.333333
PRESIDEN	5000.000000
SALESMAN	1400.000000

Find the number of employees per job title.

```
SELECT job, count(*)
FROM emp
GROUP BY  job;
```

job	count(*)
ANALYST	2
CLERK	4
MANAGER	3
PRESIDENT	1
SALESMAN	4

Find the average salary for each job title within in each department

```
SELECT        deptno, job, avg(sal)
FROM  emp
GROUP BY deptno, job;
```

deptno	job	avg(sal)
10	CLERK	1300.000000
10	MANAGER	2450.000000
10	PRESIDEN	5000.000000
20	ANALYST	3000.000000
20	CLERK	950.000000
20	MANAGER	2975.000000
30	CLERK	950.000000
30	MANAGER	2850.000000
30	SALESMAN	1400.000000

Exercise

- What is the minimum salary in each department?
- Grouping employees by the manager they report to, what is the maximum salary paid in each group?

HAVING clause

- If there is a WHERE clause in a query, it must be specified BEFORE the GROUP BY clause.
- WHERE clause specifies which rows to include/ exclude in the query. The DBMS evaluates the WHERE clause BEFORE a GROUP BY clause, so only selected rows are included when calculated the group function.
- **To exclude rows AFTER aggregating the data you use a HAVING clause**

Examples

- Retrieve the number of departments with more than 3 employees

```
SELECT deptno, COUNT(*)
FROM emp
GROUP BY deptno
HAVING COUNT(*) > 3;
```

deptno	COUNT(*)
20	5
30	6

There are only 2 people in department 10, so it's not included

- Which job roles have an average salar

```
SELECT deptno, avg(sal)
FROM emp
GROUP BY deptno
HAVING avg(sal)> 2000;
```

deptno	avg(sal)
10	2916.666667
20	2175.000000

Average salary in dept 30 is < 2000 so it's not included

Exercise

- Which department(s) have a minimum salary less than 1000?

Putting it all together

- For all employees with 'S' in their name, show the total salary by department, provided that total is more than 2000. Show highest value first.

```
SELECT deptno, sum(sal)
FROM emp
WHERE ename LIKE "%S%"
GROUP BY deptno
HAVING sum(sal) > 2000
ORDER BY sum(sal) DESC;
```

deptno	sum(sal)
20	7875.00

Another example with different tables . . . .. .

- For all non-Dublin based customers, show the total spent by each customer for all customers that spent more than £10,000. Show highest spenders first

```
SELECT customer_id, sum(order_total)
FROM order
WHERE customer_county NOT LIKE 'Dublin%'
GROUP BY customer_id
HAVING sum(order_total) > 10,000
ORDER BY sum(order_total) DESC;
```

Order Table:

Order_ number	Customer _id	Customer_c ounty	Order_ total
001	46	Dublin	600
002	48	Cork	13,000
003	45	Limerick	7,000
004	46	Dublin	12,000
005	45	Limerick	5,000
006	50	Cavan	500
007	50	Cavan	2,000

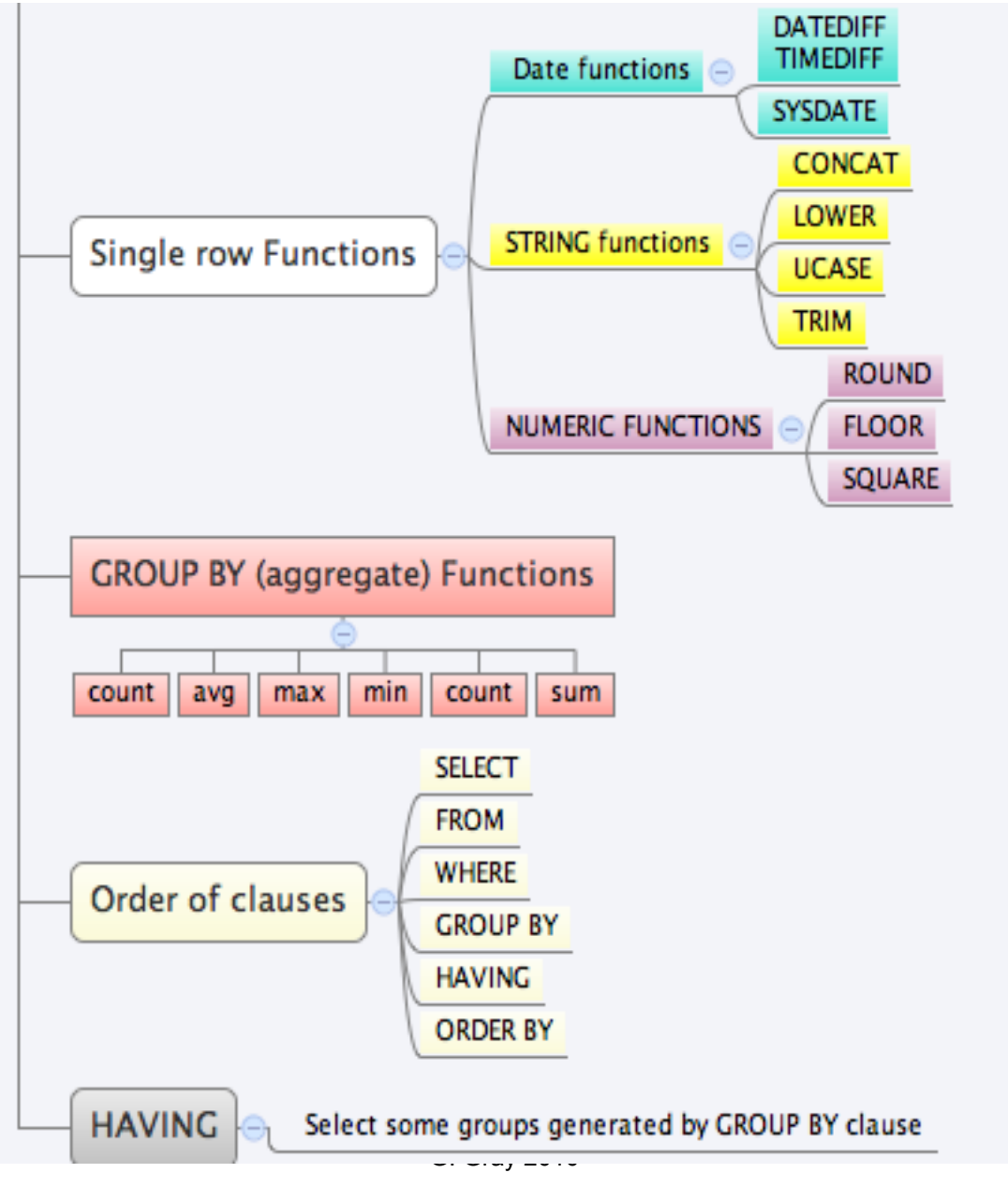
Query result:

Customer_ID	Sum(Order_total)
48	13,000
45	12,000

Note the order of the clauses!!

```
SELECT columnlist
FROM tablename
WHERE condition
GROUP BY
HAVING group condition
ORDER BY          ;
```

Summary


Single row Functions
Date functions
DATEDIFF
TIMEDIFF
SYSDATE
STRING functions
CONCAT
LOWER
UCASE
TRIM
NUMERIC FUNCTIONS
ROUND
FLOOR
SQUARE
GROUP BY (aggregate) Functions
count
avg
max
min
count
sum
Order of clauses
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
HAVING
Select some groups generated by GROUP BY clause

Learning Outcomes