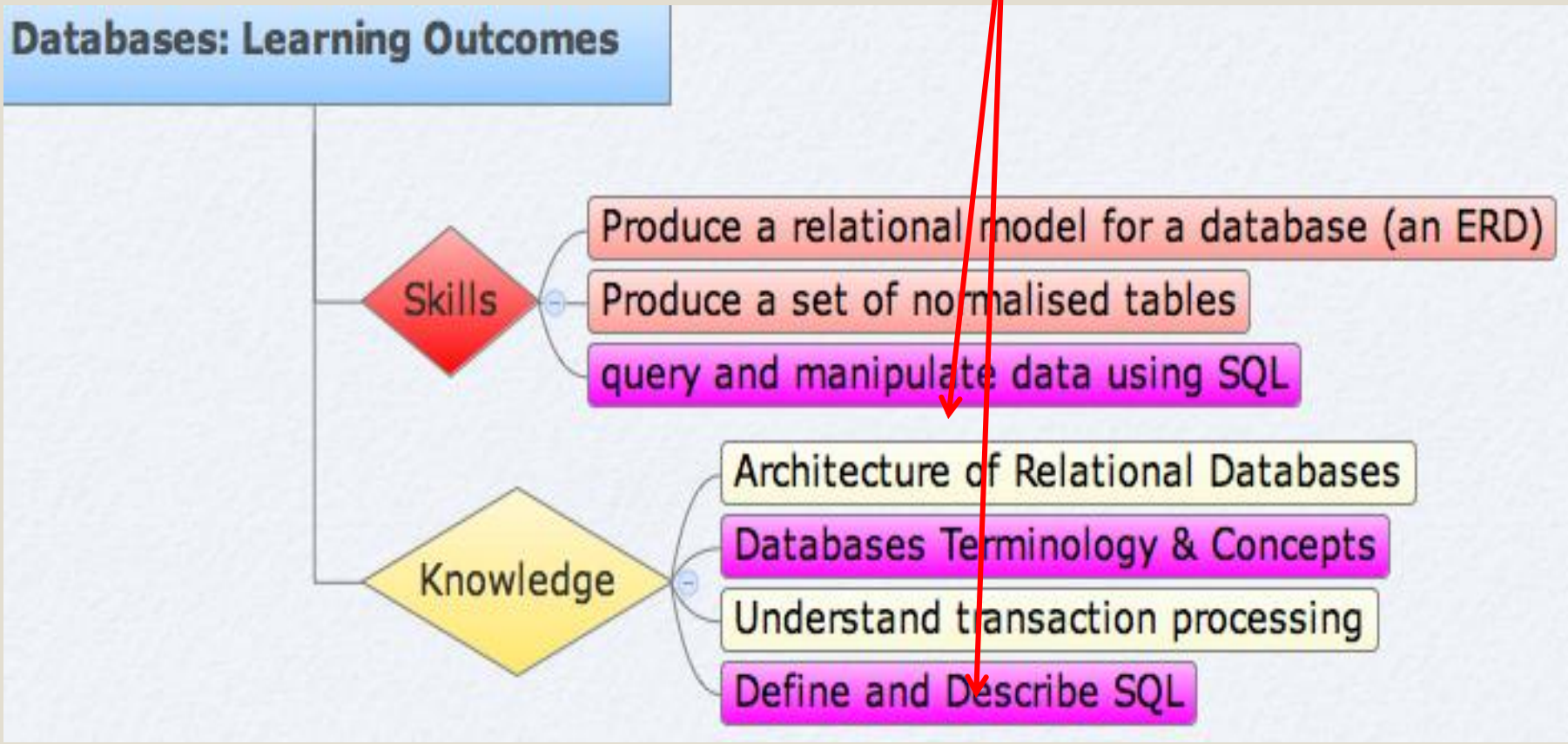
# MODULE: DATABASE FUNDAMENTALS

## Lecture 3

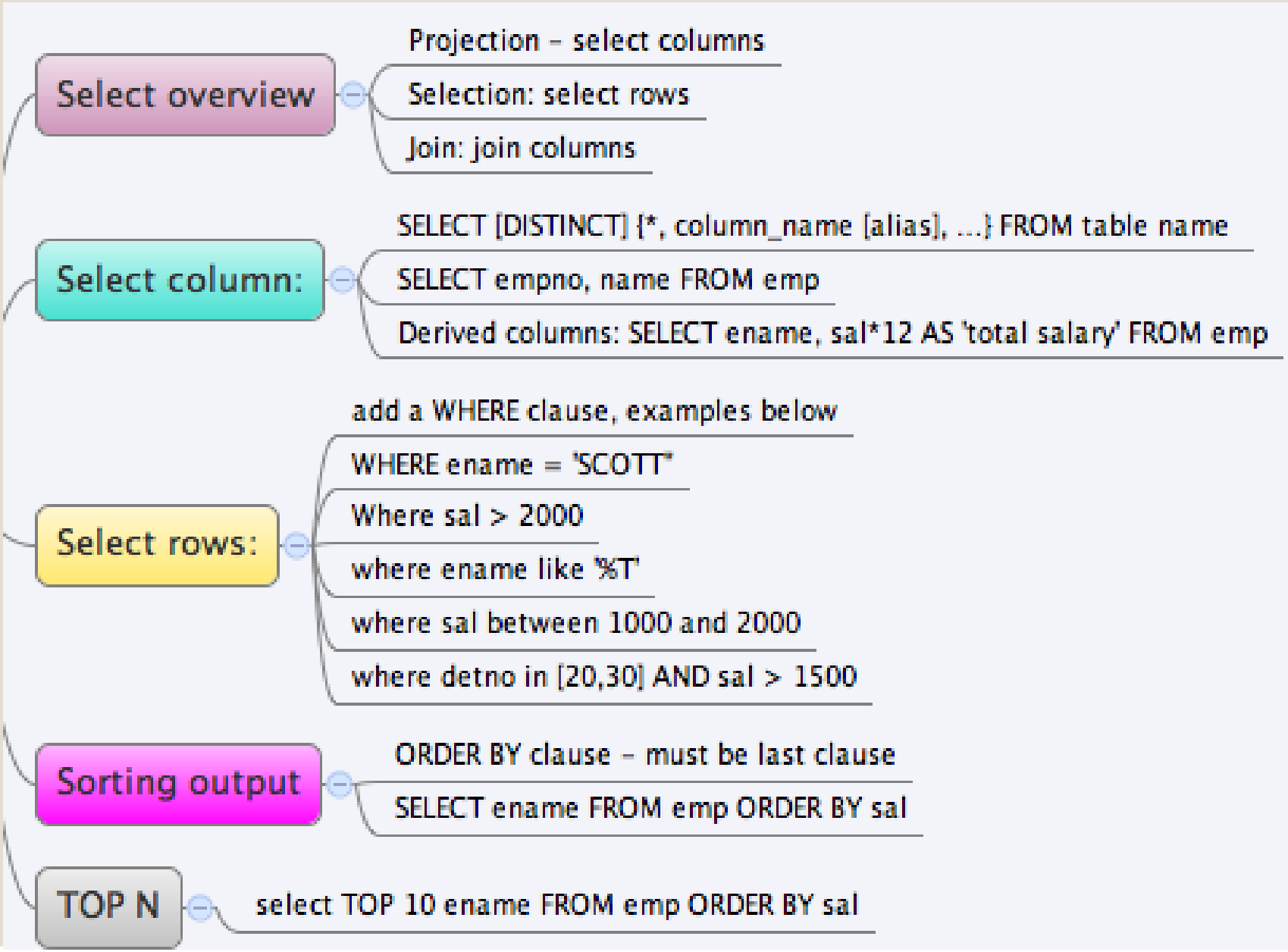
Continue with SQL Select

## Functions

# 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

- **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. sysdate(), 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', 'String') | 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)**
  - SQUARE (9) gives 81
  - SELECT SQUARE (sal) FROM emp will square all salary figures before displaying them
    - Syntax: **SQUARE(column|expr)**

# Single Row Functions

- Date functions
  - SYSDATE()  returns system data
  - DATEDIFF('1980/3/02', '1985/7/22')  gives the number of days between two dates
    - Syntax: **DATEDIFF(date1,date2)**

# Example using single row functions:

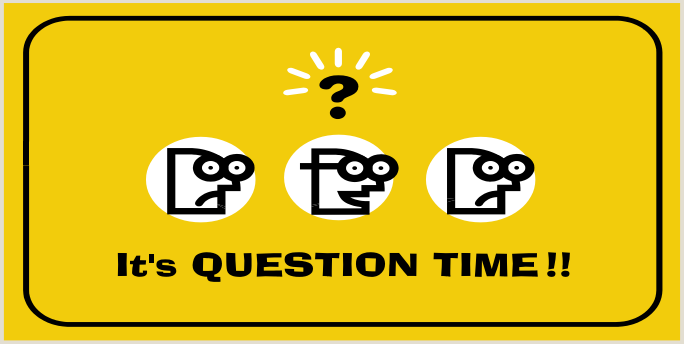
- How many years has Clark being working for the company?

```
SELECT ename, FLOOR(DATEDIFF(sysdate(),
  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 sysdate (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;
```

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

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

| sum(sal) |
|---|
| 29025.00 |

Aggregated data

# 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 |

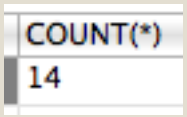
Find the average salary, the highest salary, the lowest salary and the total salary for clerks:

```
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

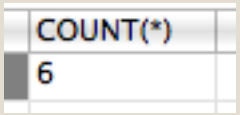
```
SELECT   COUNT(*)
FROM     emp;
```



| 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

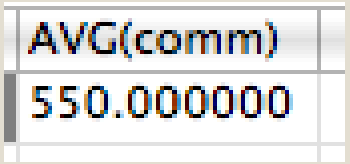
How many employees earn commission?

```
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 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.

- 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

```
SELECTdeptno, 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 salary greater than 2000?

```
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 |

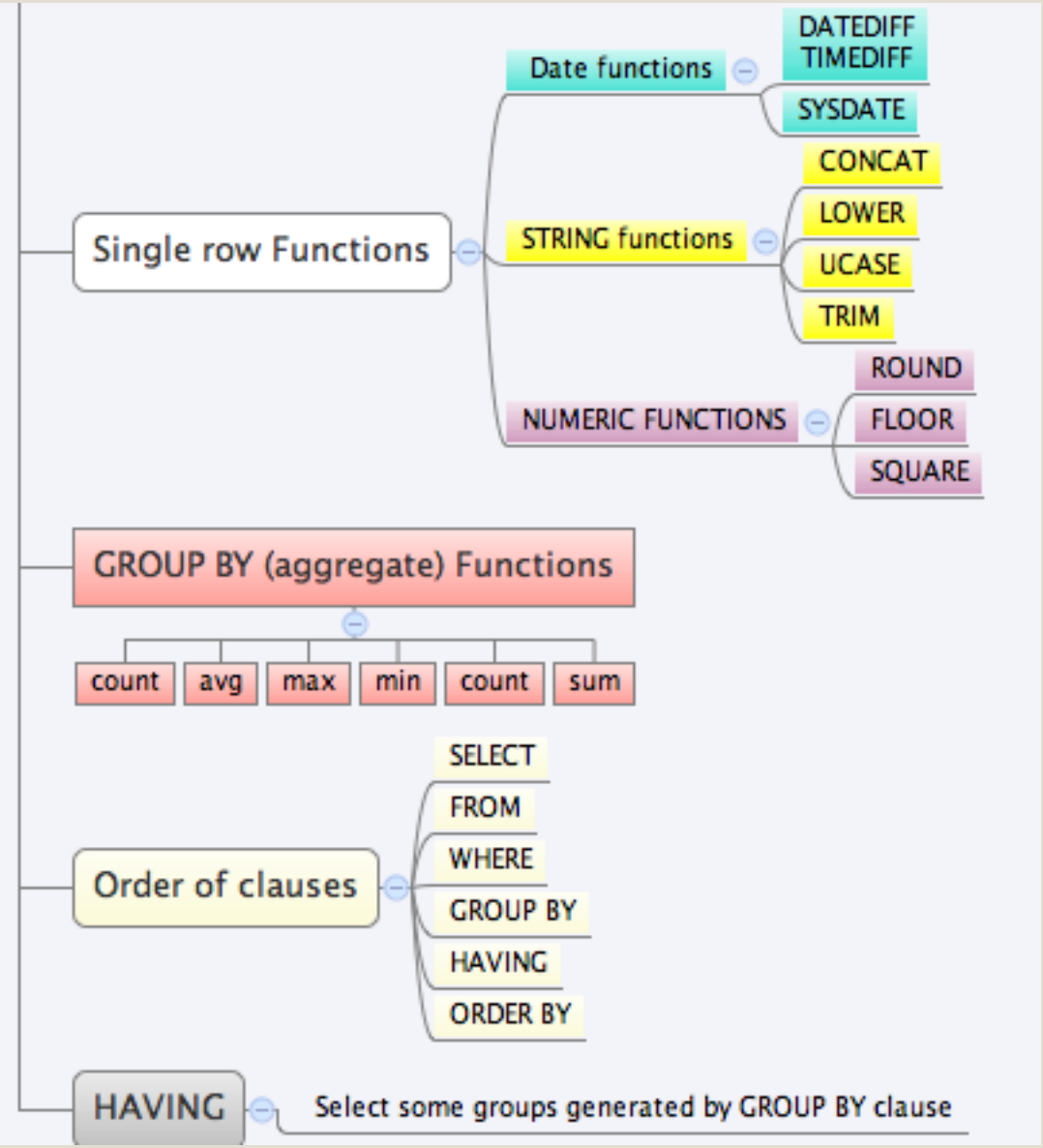
**Query result:**

| 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