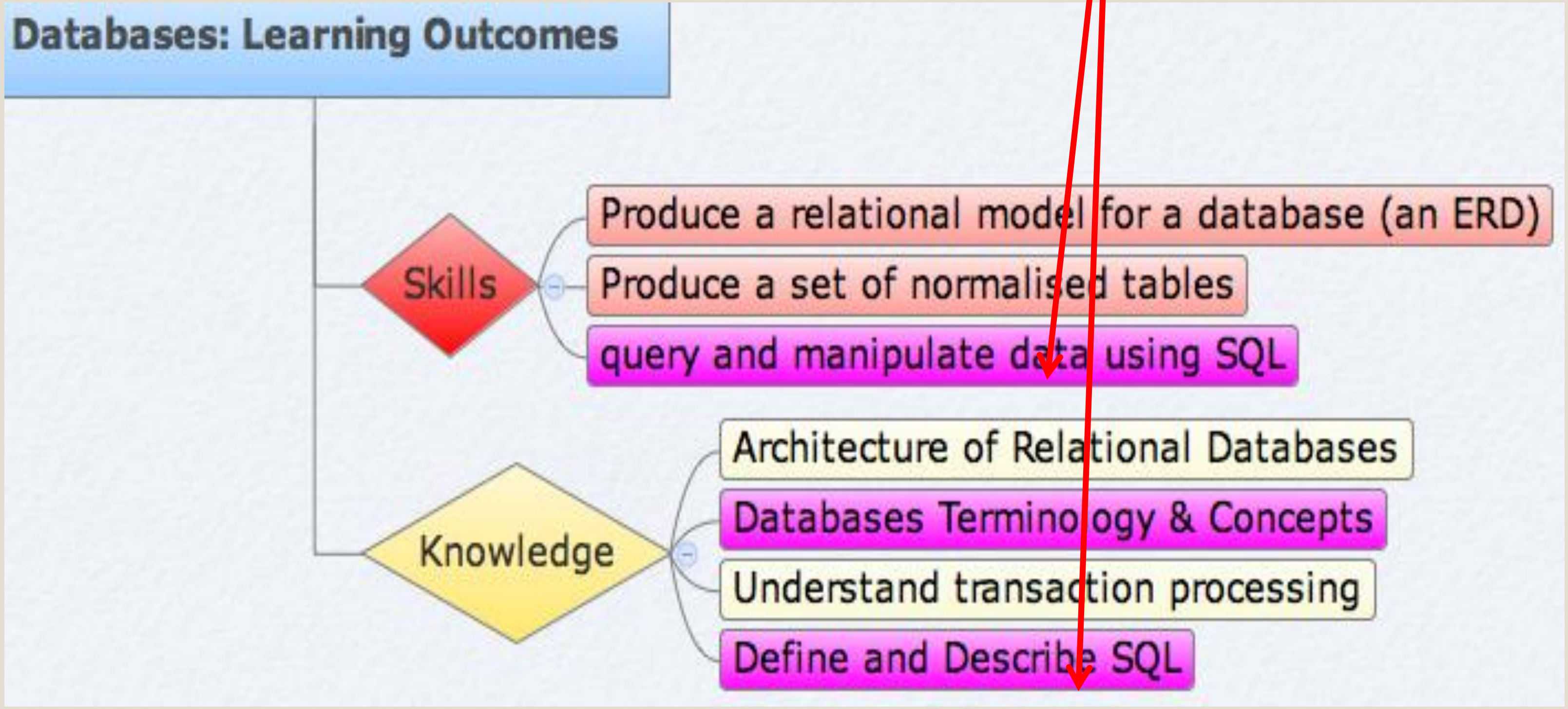
DATABASE FUNDAMENTALS

Lecture 4: Continue with SQL Select

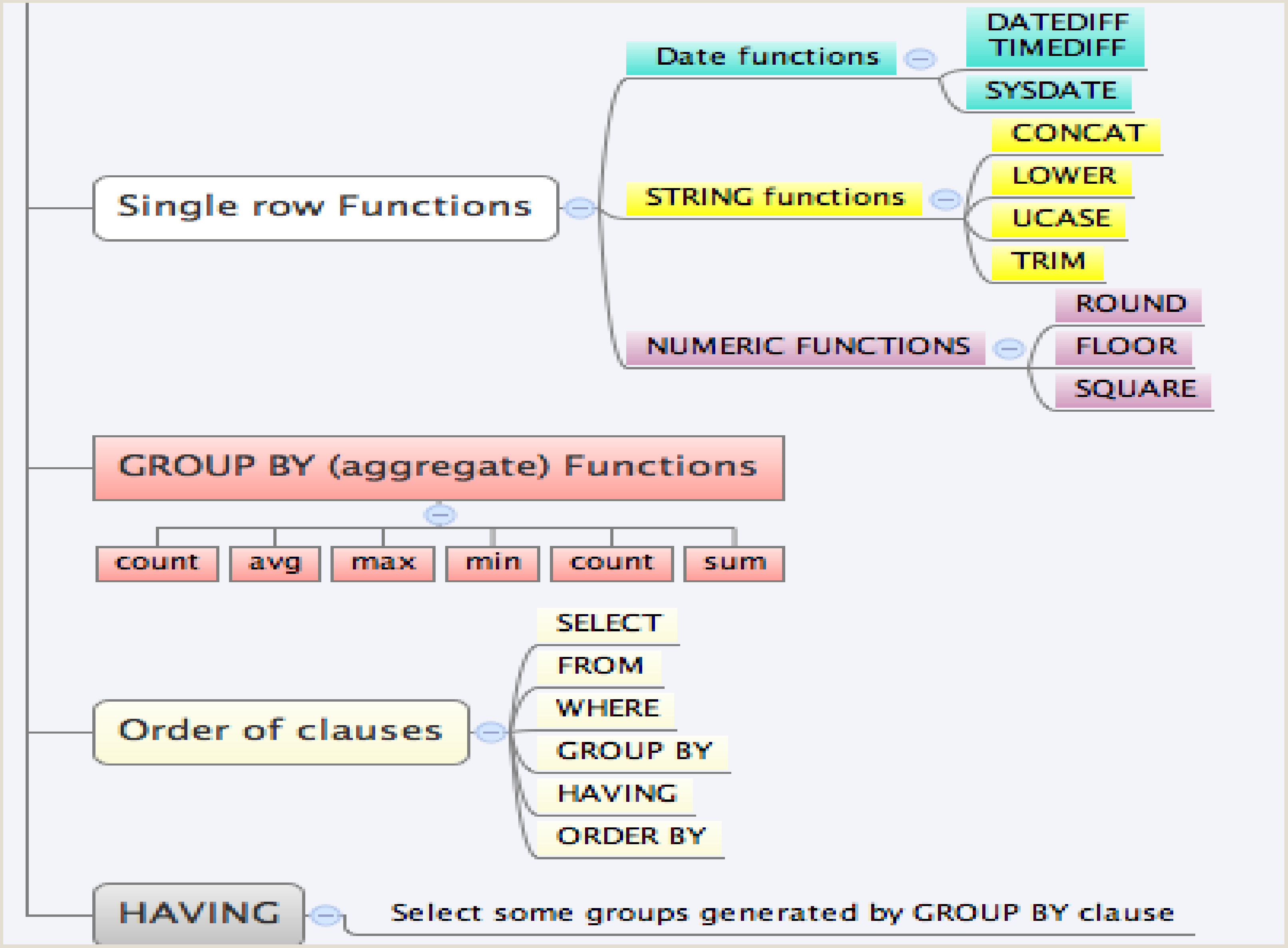
Joins

Subqueries

Learning Outcomes


Databases: Learning Outcomes
Skills
Produce a relational model for a database (an ERD)
Produce a set of normalised tables
query and manipulate data using SQL
Knowledge
Architecture of Relational Databases
Databases Terminology & Concepts
Understand transaction processing
Define and Describe SQL

Recap


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

Objective for this lecture:

Continue with SQL Select clause

- **Join**two or more tables together to answer a query
- **Sub-queries**

SECTION 1

Joining tables together

Joining tables

- **JOINS** is obtaining data from multiple tables
- Where does the following data come from?

```
EMPNO  DEPTNO LOC
----- -------- ---------
 7839       10 NEW YORK
 7698       30 CHICAGO
 7782       10 NEW YORK
 7566       20 DALLAS
 7654       30 CHICAGO
 7499       30 CHICAGO
...
14 rows selected.
```

JOIN

- A JOIN operation **combines two or more tables** generating **one result set** from the information stored in such tables

- One column needs to be the same in each table, usually a **foreign key**, which is the column used to JOIN the two tables.

- Six JOIN keywords:
  - INNER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
  - NON-EQUI JOIN
  - CROSS JOIN (CARTESIAN PRODUCT)

Types of Joins

empId	empName	Deptno		Deptno	Dname
1223	Miller	10		10	Sales
2345	Clark	NULL		30	Marketing
4567	Murphy	30		40	Purchasing

- **Inner Join**: only show rows that are linked across both tables

  Two rows returned:

Miller	Sales
Murphy	Marketing

- **Left outer join**: show all rows from the first table, and only rows that match in the second table.
  - Three rows returned:

Miller	Sales
Clark	NULL
Murphy	Marketing

Types of Joins

empId	empName	Deptno		Deptno	Dname
1223	Miller	10		10	Sales
2345	Clark	NULL		30	Marketing
4567	Murphy	30		40	Purchasing

- **Right outer join**: show all rows from the Second table, and only rows that match in the First table.

Three rows returned:

Miller	Sales
Murphy	Marketing
NULL	Purchasing

- **FULL outer join**: show all rows from both tables.

Four rows returned:

Miller	Sales
Clark	NULL
Murphy	Marketing
NULL	Purchasing

Types of join

- A cross join is generally the result you get when you have made a mistake! It links ALL rows in one table with ALL rows in the other table as follows:

Four rows returned:

Miller	Sales
Miller	Marketing
Miller	Purchasing
Clark	Sales
Clark	Marketing
Clark	Purchasing
Murphy	Sales
Murphy	Marketing
Murphy	Purchasing

Non-equi joins will be explained in a later slide

EXAMPLE: INNER JOIN

```
SELECT emp.empno, dept.deptno, dept.loc
FROM emp INNER JOIN dept
ON emp.deptno = dept.deptno;
```

- Rows in one table can be joined to rows in another table according to common values existing in corresponding tables, **usually primary and foreign keys.**
- **The FROM clause specifies the tables to use and the type of JOIN**
- **INNER JOIN returns all common rows in two or more tables**
- ON clause – specifies what columns to use to finding matching rows
- The column name is prefixed by the table name if the same column name appears in more than one table
- You can have other conditions in the query, such as a WHERE clause, e.g. WHERE dept.deptno>30

Using aliases for table names

Rather than including the full table name in front of column names, you can give each table an alias in the FROM clause, and use the alias in all other clauses. For example:

```
SELECT  emp.empno, dept.deptno, dept.loc
FROM emp INNER JOIN dept
ON          emp.deptno = dept.deptno;
```

is the same as

```
SELECT e.empno, d.deptno, d.loc
FROM emp e INNER JOIN dept d
ON   e.deptno = d.deptno;
```

Cross Join(Cartesian Product)

Warning:

If no join condition is included, or if it is invalid, the DBMS joins every row in the first table with every row in the second table - this is called a **Cartesian product**.

emp

emp_id	**deptno**
001	10
002	10
003	20
004	30

dept

deptno	**dept_name**
10	sales
20	purchasing
30	finance

```
SELECT emp_id,
dept_name
FROM emp INNER
JOIN dept
```

would give:

query result:

emp_id	**dept_name**
001	10
001	20
001	30
002	10
002	20
002	30
003	10
003	20
003	30

Joining more than two tables

CUSTOMER

```
NAME              CUSTID
-----------       ------
JOCKSPORTS           100
TKB SPORT SHOP       101
VOLLYRITE            102
JUST TENNIS          103
K+T SPORTS           105
SHAPE UP             106
WOMENS SPORTS        107
...               ...
9 rows selected.
```

ORDER

```
  CUSTID    ORDID
-------- --------
     101      610
     102      611
     104      612
     106      601
     102      602
     106
     106
...
21 rows s
```

ITEM

```
 ORDID  ITEMID
------ -------
   610       3
   611       1
   612       1
   601       1
   602       1
...
64 rows selected.
```

JOIN CONDITION

FROM customer INNER JOIN order
ON customer. custid = order.custid
INNER JOIN item
ON order.ordid = item.ordid;

- Note: When joining ‘n’ tables, there should be at least ‘n-1’ join conditions.

- If you have 4 tables then you must have at least 3 conditions

Exercises

- List the name and location for each employee
- List the name and location for each employee in Boston
- Give the employee name and department name for each CLERK.

Non-equijoin – using a comparison operator other than =

EMP

```
 EMPNO ENAME        SAL
------ -------- -------
  7839 KING        5000
  7698 BLAKE       2850
  7782 CLARK       2450
  7566 JONES       2975
  7654 MARTIN      1250
  7499 ALLEN       1600
  7844 TURNER      1500
  7900 JAMES        950
...
14 rows selected.
```

SALGRADE

```
GRADE LOSAL   HISAL
----- ----- -------
1       700    1200
2      1201    1400
3      1401    2000
4      2001    3000
5      3001    9999
```

“salary in the EMP table is between low salary and high salary in the SALGRADE table”

```
SELECT e.ename, s.grade
FROM emp e INNER JOIN salgrade s
ON e.sal>= g.losal
AND e.sal<= s.hisal;
```

What would you expect the output to be ?

Examples of Outer Joins

The following examples are based on these two tables:

TEST

TEST_ID	TEST_NAME
001	test1
002	test2
003	test3

CAR

CAR_REG	TEST_ID
91 D 123	002
92 D 456	003
93 D 789	004

Test 1	Wheel Change
Test 2	Clutch Replace
Test 3	Oil Change

Right Outer Join

- Returns **all matching rows in both tables** and also rows in the right table that don't have a corresponding row in the left table
- In the result set, the rows that don't have a corresponding row in the left table contain a NULL value in all columns of the left table
- RIGHT OUTER JOIN is equivalent to RIGHT JOIN, so either can be used

```
SELECT t.test_name, c.car_reg
FROM test t RIGHT OUTER JOIN car c
ON t.test_id = c.test_id;
```

Remember the aliases

Giving

TEST_NAME	CAR_REG
test2	91 D 123
test3	92 D 456
NULL	93 D 789

All the rows in the right table are returned and only the matching values in the left

- All cars are listed
- Cars that don't have a corresponding row in the left table contain a NULL value

- these cars have not been tested

❖ Keyword OUTER is optional

LEFT OUTER JOIN

- Returns all matched rows and rows from the **left table that don't have a corresponding row in the right table**
- The unmatched rows of the result set have NULL values in the columns of the right table.
- A LEFT OUTER join can be turned into a RIGHT INNER JOIN if the order of the tables is changed (the right table becomes the left and vice versa).
- LEFT OUTER JOIN is equivalent to LEFT JOIN, so either can be used

Left Outer Join

```
SELECT t.test_id, t.test_name, c.car_reg
FROM test t LEFT OUTER JOIN car c
ON t.test_id = c.test_id;
```

Giving

Test

Test_id	test_name	car_reg
001	test 1	NULL
002	test 2	91D123
003	test 3	92D456

- All tests are listed but all cars are not, as they have not been tested

FULL OUTER JOIN

- Returns **all rows** that **match the JOIN condition**
- Rows from the left table that don’t have a corresponding row in the right table.

- These rows have NULL values in the columns of the right table

- Rows from the right table that don’t have corresponding row in the left table.

- These rows have NULL values in the columns of the left table.

- FULL OUTER JOIN is equivalent to FULL JOIN, so either can be used

Full Outer Join

- List all tests and all cars

```
SELECT t.test_id, t.test_name, c.car_reg
FROM test t FULL OUTER JOIN car c
ON t.test_id = c.test_id;
```

```
test_id   test_name        car_reg
-------   ----------       ----------
002       test 2            91D123
003       test 3           92D456
NULL      NULL             93D789
001       test 1            NULL
```

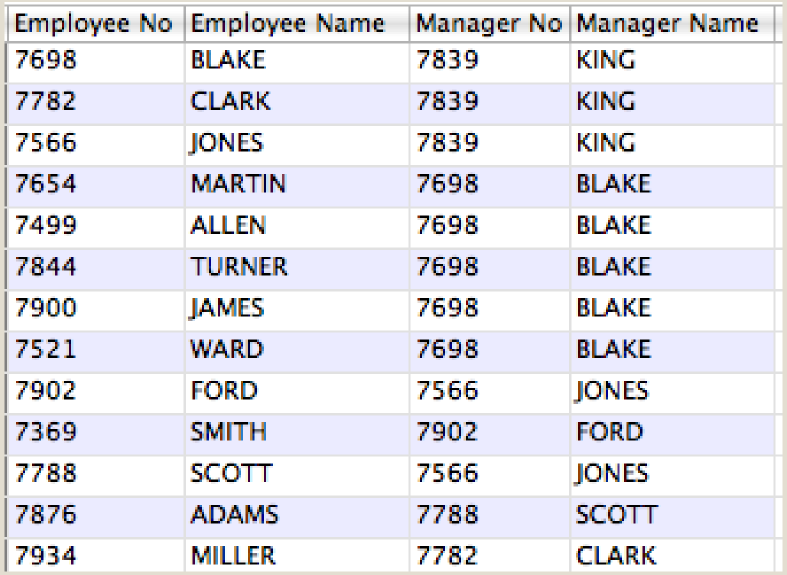
Joining a table to itself – need to use aliases

A different alias for each column

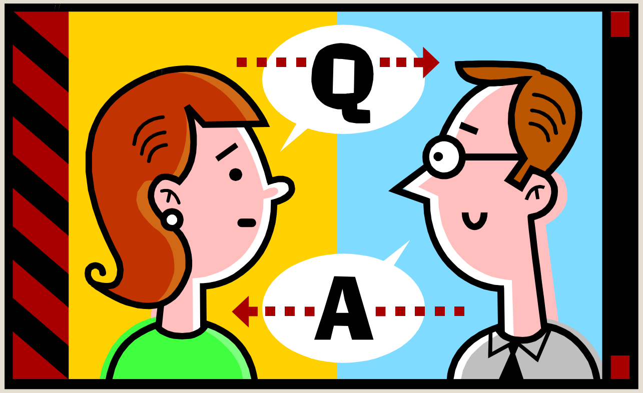
Find the name of the each employee's manager

```
SELECT e1.empno as 'Employee No',
  e1.ename as 'Employee Name',
           e2.empno as 'ManagerNo',
              e2.ename as 'Manager Name'
FROM emp e1
  INNER JOIN emp e2
  ON e1.mgr = e2.empno;
```

Result:



Employee No	Employee Name	Manager No	Manager Name
7698	BLAKE	7839	KING
7782	CLARK	7839	KING
7566	JONES	7839	KING
7654	MARTIN	7698	BLAKE
7499	ALLEN	7698	BLAKE
7844	TURNER	7698	BLAKE
7900	JAMES	7698	BLAKE
7521	WARD	7698	BLAKE
7902	FORD	7566	JONES
7369	SMITH	7902	FORD
7788	SCOTT	7566	JONES
7876	ADAMS	7788	SCOTT
7934	MILLER	7782	CLARK


Q
A

Exercises - In-class

- List all employees working in ‘Dallas’
- Give a unique list of jobs of people based in ‘Chicago’
- List all employees on grade 3
- For all **4** locations, show the employees working in that location.

SECTION 2

Sub-Queries

Sub-Queries

- A Sub-Query is a **query** (i.e. a select statement) **nested within another SQL statement**.
- For Select statements, you can place a subquery in a
  - WHERE clause
  - HAVING clause
  - FROM clause
- You can also use subqueries in
  - CREATE statements
  - UPDATE statements
  - INSERT statements

Sub-query Example

- List all employees whose salary is greater than Jones salary.
- To answer this, you need to know
  - Jones salary - **one** query
  - employees with a salary greater than Jones salary - **second** query

```
SELECT ename
FROM emp
WHERE sal> (SELECT sal
                        FROM emp
                        WHERE ename=‘Jones’);
```

Exercise

- Who is earning more than the average salary?
- Who is earning more than Clark?
- How many people are earning more than Clark?

Points on Subqueries

- A subquery is **always enclosed in brackets**
- When using the following comparison operators, the sub query must return only ONE value
  - =, >, >=, <, <=, <>

- For subqueries that return more than one ROW, use the following operators
  - IN, ANY, ALL
- For subqueries that return more than one COLUMN, list all columns in the WHERE clause:

```
        . . .WHERE (prodid, qty) IN (SELECT prodid, qty
                                      FROM . . . )
```

Example - IN

What employees are paid the same as the lowest salary in a department?

```
SELECT ename, sal, deptno
FROM emp
WHERE sal  IN  ( SELECT MIN(sal)
                 FROM emp
                 GROUP BY deptno);
```

What employees are paid the same as the highest salary in a department?

```
SELECT ename, sal, deptno
FROM emp
WHERE sal IN  ( SELECTMAX(sal)
                FROM emp
                GROUP BY deptno);
```

Using ANY and ALL

- Note:
- > ANY means greater than the smallest value in the list
- < ANY means less than the highest value in the list
- > ALL means greater than the highest value in the list
- < ALL means less than the lowest value in the list
- = ANY is the same as in

Example

List employees who's salary is less than the minimum salary in any department:

```
SELECT ename, sal, deptno
FROM emp
WHERE sal< ANY  ( SELECT MIN(sal)
                   FROM emp
                   GROUP BY deptno);
```

Return a list of three salaries – the minimum salary in each department, i.e. 1300, 800, 950

Less than any of these numbers: 1300, 800, 950
Any value less than 1300 will be less than one of these numbers

Examples

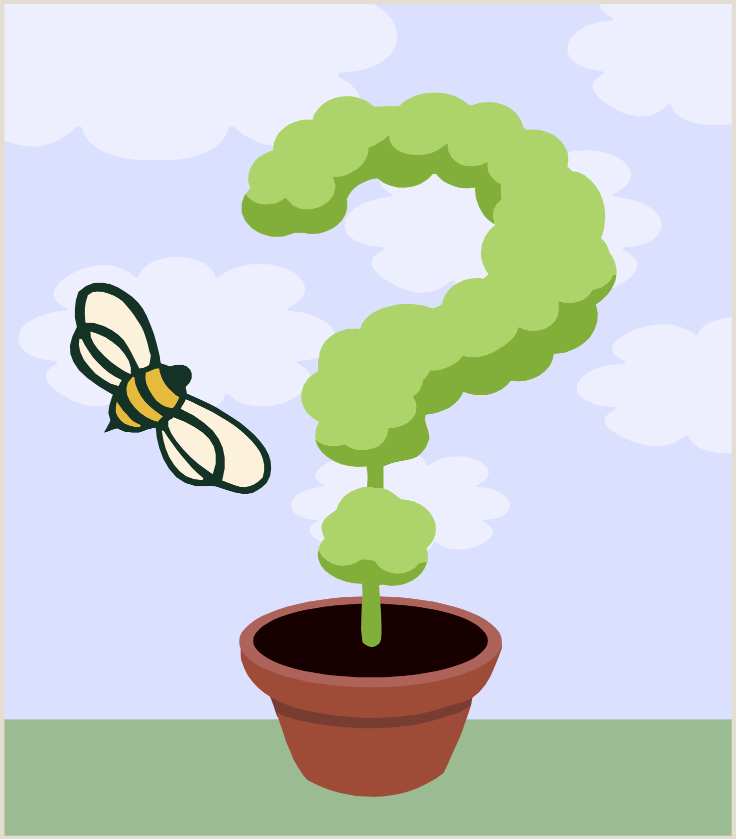
```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE SAL > ALL  ( SELECT MIN(SAL)
                   FROM EMP
                   GROUP BY DEPTNO);
```

Greater than all of these numbers: 1300, 800, 950
Any value greater than 1300 will be greater than all of the numbers in the list

Return a list of three salaries – the minimum salary in each department, i.e. 1300, 800, 950

```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE SAL < ALL  ( SELECT MIN(SAL)
                   FROM EMP
                   GROUP BY DEPTNO);
```

Less than ALL of these numbers: 1300, 800, 950
To be less that each of these numbers, the value must be less than 800.



Exercises - In-class

- What manager has a salary lower than BLAKES
- List all employees working for below the average salary
- List all employees working for below the lowest average salary in a department
- List all employees working for below the highest average salary in a department

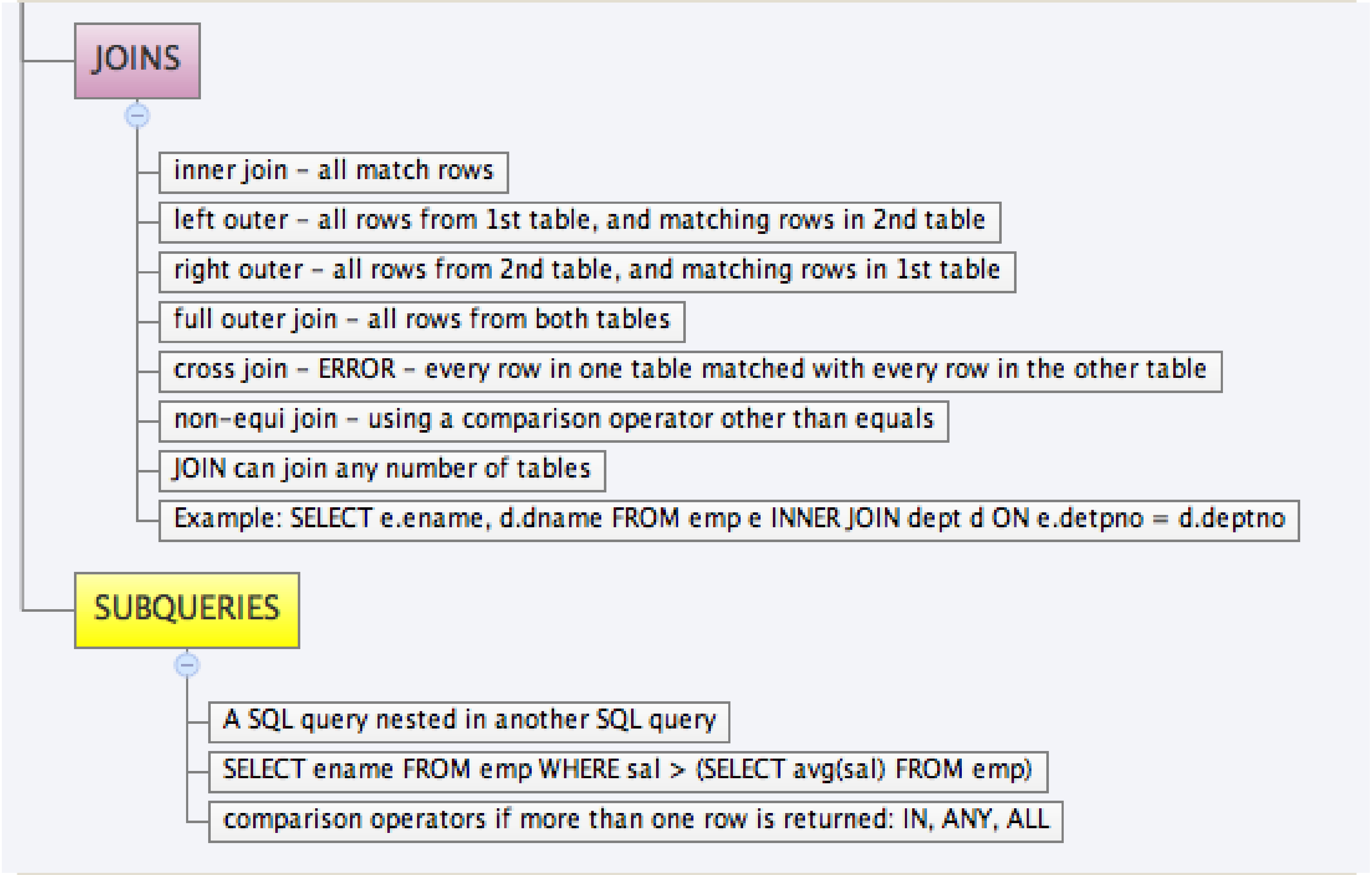
SYNTAX SELECT

```
SELECT [DISTINCT] column_list
FROM table_name [JOIN table name ON
  column_name COMPARISON OPERATOR
  column_name]
[WHERE conditional expression]
[GROUP BY group_by_column_list]
[HAVING conditional expression]
[ORDER BY order_by_column_list]
```

SQL Statement Processing Order

- SELECT – identifies the **columns** to be displayed
- FROM – identifies the **table**(s) involved
- WHERE – Finds **rows** meeting a stated condition
- GROUP BY –Identifies groups to which a **group functionis** to be applied (max, min, avg, sum etc.)
- HAVING – Finds all **groups** meeting a stated conditions
- ORDER BY – **order** in which results are to be displayed

Summary



- JOINS
  - inner join – all match rows
  - left outer – all rows from 1st table, and matching rows in 2nd table
  - right outer – all rows from 2nd table, and matching rows in 1st table
  - full outer join – all rows from both tables
  - cross join – ERROR – every row in one table matched with every row in the other table
  - non-equi join – using a comparison operator other than equals
  - JOIN can join any number of tables
  - Example: SELECT e.ename, d.dname FROM emp e INNER JOIN dept d ON e.detpno = d.deptno
- SUBQUERIES
  - A SQL query nested in another SQL query
  - SELECT ename FROM emp WHERE sal > (SELECT avg(sal) FROM emp)
  - comparison operators if more than one row is returned: IN, ANY, ALL