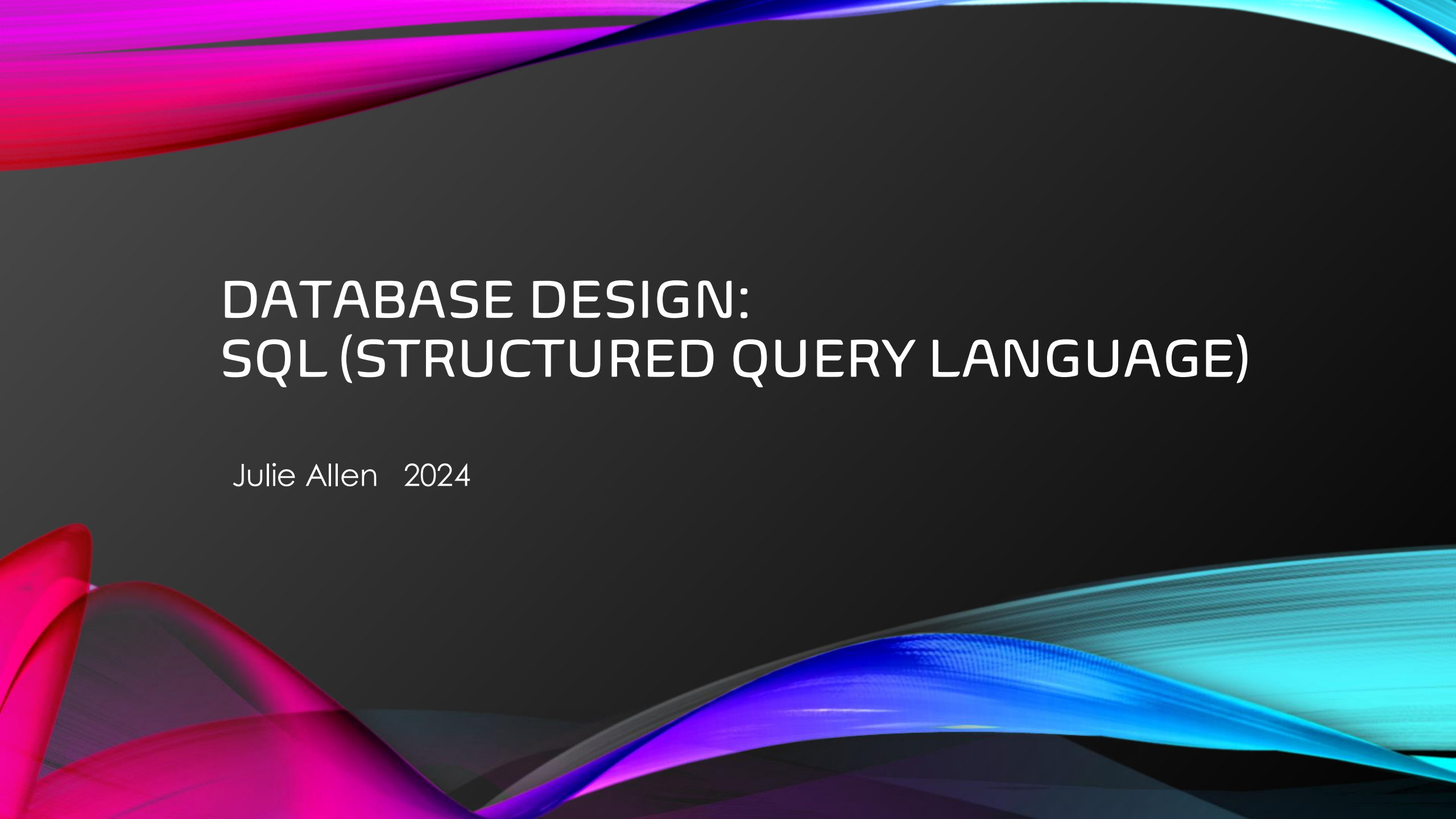




DATABASE DESIGN: SQL (STRUCTURED QUERY LANGUAGE)

Julie Allen 2024



Today's Agenda:

- Creating tables
- Inserting data into tables
- Recap: SELECT – FROM - WHERE
- Joining tables
- SQL functions



RECAP...

EMP AND DEPT TABLES

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	MANCHESTER
50	MARKETING	LUTON

EMP & DEPT: CREATE TABLE SCRIPTS

Create the DEPT Table

```
create table dept(  
  deptno number(2,0),  
  dname  varchar2(14),  
  loc    varchar2(13),  
  constraint pk_dept primary key (deptno)  
);
```

Create the EMP Table

```
create table emp(  
  empno    number(4,0),  
  ename    varchar2(10),  
  job      varchar2(9),  
  mgr      number(4,0),  
  hiredate date,  
  sal      number(7,2),  
  comm     number(7,2),  
  deptno   number(2,0),  
  constraint pk_emp primary key (empno),  
  constraint fk_deptno foreign key (deptno) references dept (deptno)  
);
```

EMP & DEPT: INSERT DATA SCRIPTS

Insert data into the DEPT Table

```
BEGIN

insert into dept
values(10, 'ACCOUNTING', 'LONDON');
insert into dept
values(20, 'RESEARCH', 'PRESTON');
insert into dept
values(30, 'SALES', 'LIVERPOOL');
insert into dept
values(40, 'OPERATIONS', 'STAFFORD');
insert into dept
values(50, 'MARKETING', 'LUTON');
COMMIT;

END;
```

Insert data into the EMP Table

```
BEGIN

insert into emp values(
  7839, 'KING', 'PRESIDENT', null,
  to_date('17-11-2001','dd-mm-yyyy'),
  82000, null, 10
);
insert into emp values(
  7698, 'BLAKE', 'MANAGER', 7839,
  to_date('1-5-2005','dd-mm-yyyy'),
  24000, null, 30
);
insert into emp values(
  7782, 'CLARK', 'MANAGER', 7839,
  to_date('9-6-2009','dd-mm-yyyy'),
  27500, null, 10
);
....
....
....

COMMIT;

END;
```

EMP & DEPT: UPDATE & DELETE

Example:

Add another
department to the
DEPT Table

```
INSERT INTO dept  
VALUES(60, 'HR', 'BLACKPOOL');
```

DEPTNO	DNAME	LOC
60	HR	BLACKPOOL
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	STAFFORD
50	MARKETING	LUTON

Now update the
department number
(deptno) to 70

```
UPDATE dept  
SET deptno = 70  
WHERE deptno = 60
```

DEPTNO	DNAME	LOC
70	HR	BLACKPOOL
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	STAFFORD
50	MARKETING	LUTON

Delete the new
record

```
DELETE FROM dept  
WHERE deptno = 70
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	STAFFORD
50	MARKETING	LUTON

THE SELECT-FROM STATEMENT

The **SELECT-FROM** is the basic statement used to select data from a database.

Selects rows and columns from one or more tables.

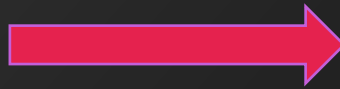
```
SELECT column1, column2, ... FROM table_name;
```

Note the capitalisation of the key words. This isn't compulsory, but it is good convention.

THE SELECT-FROM STATEMENT

To SELECT **all** columns from a table use the ***** symbol...

```
SELECT * FROM emp;
```



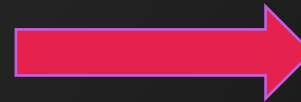
Result

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

THE SELECT-FROM STATEMENT

To SELECT *specific* columns from a table list them after the SELECT keyword, separating each with a comma...

```
SELECT empno, ename FROM emp;
```



EMPNO	ENAME
7839	KING
7698	BLAKE
7782	CLARK
7566	JONES
7788	SCOTT
7902	FORD
7369	SMITH
7499	ALLEN
7521	WARD
7654	MARTIN
7844	TURNER
7876	ADAMS
7900	JAMES
7934	MILLER
3258	GREEN
4422	STEVENS
6548	BARNES

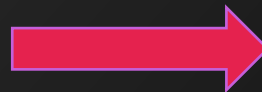
THE WHERE CLAUSE

The **WHERE** Clause is a conditional statement used to filter records. It is used to extract records that fulfil a specified condition and is placed after the SELECT-FROM statement.

Example: Display all employees in department 20...

```
SELECT column1, column2
FROM table_name
WHERE condition;

SELECT * FROM emp WHERE deptno = 20;
```



Result

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20

AND, OR & NOT OPERATORS

The WHERE clause can be combined with **AND**, **OR** and **NOT** operators

The **AND** and **OR** operators are used when we want to filter records based on more than one condition

The **NOT** operator displays a record if the condition is NOT TRUE

AND OPERATOR

The **AND** operator displays a record if ALL the conditions separated by **AND** are TRUE...

```
SELECT column1, column2  
FROM table_name  
WHERE condition1 AND condition2;
```

```
SELECT * FROM emp  
WHERE deptno = 20  
AND job = 'CLERK';
```



Result

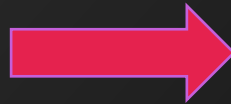
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20

OR OPERATOR

The **OR** operator displays a record if ANY of the conditions separated by **OR** is TRUE...

```
SELECT column1, column2
FROM table_name
WHERE condition1 OR condition2;
```

```
SELECT * FROM emp
WHERE deptno = 20
OR job = 'CLERK';
```



Result

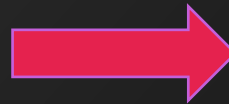
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

NOT OPERATOR

The **NOT** operator displays a record if the condition(s) is NOT TRUE...

```
SELECT column1, column2  
FROM table_name  
WHERE NOT condition;
```

```
SELECT * FROM emp  
WHERE NOT deptno = 20;
```



Result

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

ORDER BY

The ORDER BY statement is used to sort the result-set in ASCENDING or DESCENDING order

To sort records in ascending order use the ASC (this is the default) keyword

To sort records in descending order use the DESC keyword

Numbers are sorted numerically

Text is sorted alphabetically

```
SELECT column1, column2 FROM table_name  
ORDER BY column1, column2 ASC;
```

```
SELECT column1, column2 FROM table_name  
ORDER BY column1, column2;
```

Will both sort the results into ascending order

```
SELECT column1, column2 FROM table_name  
ORDER BY column1, column2 DESC;
```

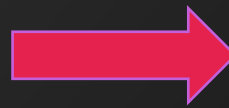
Sorts the results into descending order

ORDER BY: ASC

Result

```
SELECT * FROM emp  
ORDER BY ename;
```

```
SELECT * FROM emp  
ORDER BY ename ASC;
```



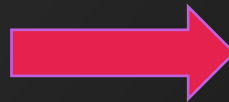
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30

ORDER BY: DESC

Result

```
SELECT column1, column2  
FROM table_name  
ORDER BY column1, column2 ... DESC;
```

```
SELECT * FROM emp  
ORDER BY sal DESC;
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20



JOINING TABLES

JOINING TABLES

The **JOIN** clause is used to combine records from two or more tables, based on a **RELATED** column between them...

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	MANCHESTER
50	MARKETING	LUTON

The **RELATED** column in our **EMP** and **DEPT** table is the **DEPTNO** column

JOINING THE EMP & DEPT TABLES

Let's say we want to retrieve the department name for all employees...

We will need to get the employee details from the EMP table, and the department name from the DEPT table

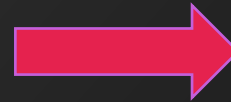
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/2001	82000	-	10
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
4422	STEVENSON	MANAGER	7839	02/14/2019	24750	-	50
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

DEPTNO	DNAME	LOC
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	MANCHESTER
50	MARKETING	LUTON

The **DEPTNO** column in both tables is our link.

JOINING THE EMP & DEPT TABLES

```
SELECT emp.ename, emp.job,  
emp.deptno, dept.dname  
FROM emp  
JOIN dept ON emp.deptno = dept.deptno;
```



ENAME	JOB	DEPTNO	DNAME
KING	PRESIDENT	10	ACCOUNTING
BLAKE	MANAGER	30	SALES
CLARK	MANAGER	10	ACCOUNTING
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
FORD	ANALYST	20	RESEARCH
SMITH	CLERK	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
TURNER	SALESMAN	30	SALES
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
GREEN	SALESMAN	50	MARKETING
STEVENS	MANAGER	50	MARKETING
BARNES	CLERK	50	MARKETING

The query retrieves all records that have a matching value in both tables.

Note that we need to prefix the column name (deptno) with the name of the table to avoid column ambiguity. This is because the columns are named the same in both tables.

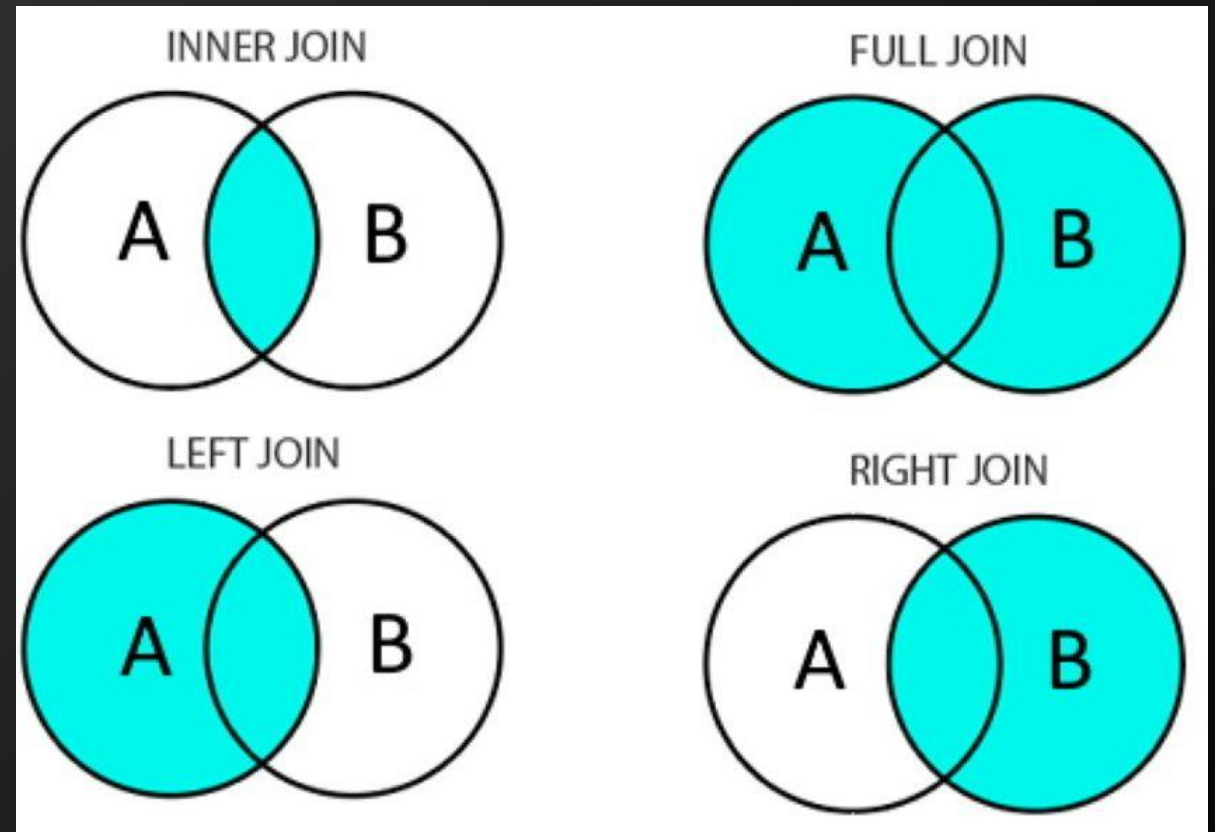
THE TYPES OF JOINS

INNER JOIN: Returns all records that have matching values in both tables

LEFT JOIN: Returns all records from the left table, and matching tables from the right table

RIGHT JOIN: Returns all records from the right table, and matching records from the left table

FULL JOIN: Returns all records when there is a match in either the left or the right table



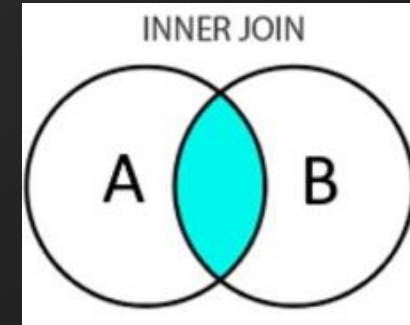
INNER JOIN

Returns rows that have a matching value in both tables. This is the most common join and as such we don't have to specify the 'INNER' keyword...

```
SELECT emp.ename, emp.job,  
emp.deptno, dept.dname  
FROM emp  
INNER JOIN dept ON  
emp.deptno=dept.deptno;
```

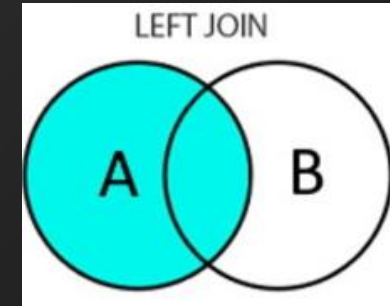
```
SELECT emp.ename, emp.job,  
emp.deptno, dept.dname  
FROM emp  
JOIN dept ON  
emp.deptno=dept.deptno;
```

Both queries return the same result



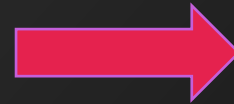
ENAME	JOB	DEPTNO	DNAME
KING	PRESIDENT	10	ACCOUNTING
BLAKE	MANAGER	30	SALES
CLARK	MANAGER	10	ACCOUNTING
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
FORD	ANALYST	20	RESEARCH
SMITH	CLERK	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
TURNER	SALESMAN	30	SALES
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
GREEN	SALESMAN	50	MARKETING

LEFT JOIN



Returns rows from the left table, and matching rows from the right table...

```
SELECT emp.ename, emp.job,  
emp.deptno, dept.dname  
FROM emp  
LEFT JOIN dept ON  
emp.deptno=dept.deptno;
```

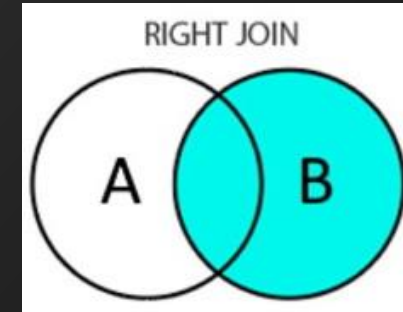


ENAME	JOB	DEPTNO	DNAME
KING	PRESIDENT	10	ACCOUNTING
BLAKE	MANAGER	30	SALES
CLARK	MANAGER	10	ACCOUNTING
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
FORD	ANALYST	20	RESEARCH
SMITH	CLERK	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
TURNER	SALESMAN	30	SALES
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
GREEN	SALESMAN	50	MARKETING
STEVENS	MANAGER	50	MARKETING
BARNES	CLERK	50	MARKETING

The LEFT table is the one mentioned first in the SELECT query (emp). The RIGHT table is the one mentioned in the JOIN statement (dept).

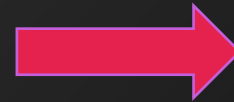
Notice that no records with DEPTNO 40 have been retrieved. Why is this?

RIGHT JOIN



Returns rows from the right table, and matching rows from the left table...

```
SELECT emp.ename, emp.job,  
dept.deptno, dept.dname  
FROM emp  
RIGHT JOIN dept ON  
emp.deptno=dept.deptno;
```



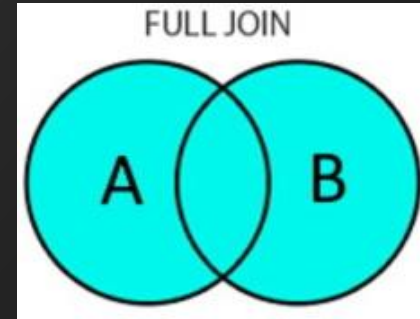
ENAME	JOB	DEPTNO	DNAME
KING	PRESIDENT	10	ACCOUNTING
BLAKE	MANAGER	30	SALES
CLARK	MANAGER	10	ACCOUNTING
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
FORD	ANALYST	20	RESEARCH
SMITH	CLERK	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
TURNER	SALESMAN	30	SALES
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
GREEN	SALESMAN	50	MARKETING
STEVENS	MANAGER	50	MARKETING
BARNES	CLERK	50	MARKETING
-	-	40	OPERATIONS

Now we can see the details of Department 40.

However, there are no employees in this department.

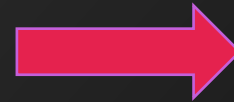
Note: we had to change the prefix for (deptno) to dept instead of emp. Why do you think we had to do this?

FULL JOIN



Also known as FULL OUTER JOIN. Returns all rows from both tables irrespective of the match...

```
SELECT emp.ename, emp.job,  
       dept.deptno, dept.dname  
FROM emp  
FULL JOIN dept ON  
emp.deptno=dept.deptno;
```



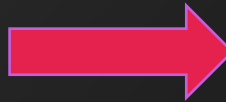
ENAME	JOB	DEPTNO	DNAME
KING	PRESIDENT	10	ACCOUNTING
BLAKE	MANAGER	30	SALES
CLARK	MANAGER	10	ACCOUNTING
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
FORD	ANALYST	20	RESEARCH
SMITH	CLERK	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
TURNER	SALESMAN	30	SALES
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
GREEN	SALESMAN	50	MARKETING
STEVENS	MANAGER	50	MARKETING
BARNES	CLERK	50	MARKETING
-	-	40	OPERATIONS

The result of a FULL JOIN is a combination of a LEFT JOIN and a RIGHT JOIN.

USING KEYWORD

If the related column in each table is named the same we can use the 'USING' keyword in the JOIN statement...

```
SELECT emp.ename, emp.job,  
dept.deptno, dept.dname  
FROM emp  
FULL JOIN dept USING (deptno);
```



ENAME	JOB	DEPTNO	DNAME
KING	PRESIDENT	10	ACCOUNTING
BLAKE	MANAGER	30	SALES
CLARK	MANAGER	10	ACCOUNTING
JONES	MANAGER	20	RESEARCH
SCOTT	ANALYST	20	RESEARCH
FORD	ANALYST	20	RESEARCH
SMITH	CLERK	20	RESEARCH
ALLEN	SALESMAN	30	SALES
WARD	SALESMAN	30	SALES
MARTIN	SALESMAN	30	SALES
TURNER	SALESMAN	30	SALES
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
GREEN	SALESMAN	50	MARKETING
STEVENS	MANAGER	50	MARKETING
BARNES	CLERK	50	MARKETING
-	-	40	OPERATIONS

As our related column (deptno) uses the same name in each table we can use the 'USING' keyword.

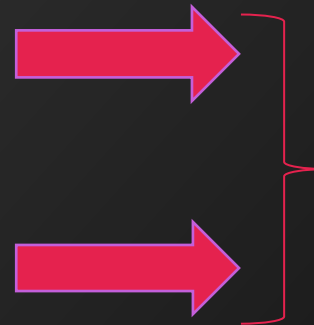
ALIASES

Aliases are used to give a table or a column a temporary name. This is most often done for readability. The name only exists for the duration of the query...

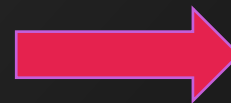
```
SELECT AVG(sal) AS AVERAGE_SALARY  
FROM emp
```

```
SELECT AVG(sal) AVERAGE_SALARY  
FROM emp
```

```
SELECT AVG(sal) "AVERAGE SALARY"  
FROM emp
```



AVERAGE_SALARY
22276.4705882352941176470588235294117647



AVERAGE SALARY
22276.4705882352941176470588235294117647

Using quotes allows you to use spaces in your alias name.
Without the quotes you cannot use spaces in your alias.

JOINING A TABLE TO ITSELF

As well as the main four joins you can join a table to itself as if it were two separate tables...

EMP

Let's say we want to find each employees managers name.

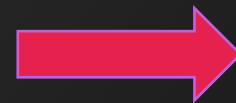
A managers MGR (manager number) is actually the same as their EMPNO (employee number).

Example: Blakes manager has an employee number of 7839. If we look for this number in the EMPNO column we can see that their manager is KING.

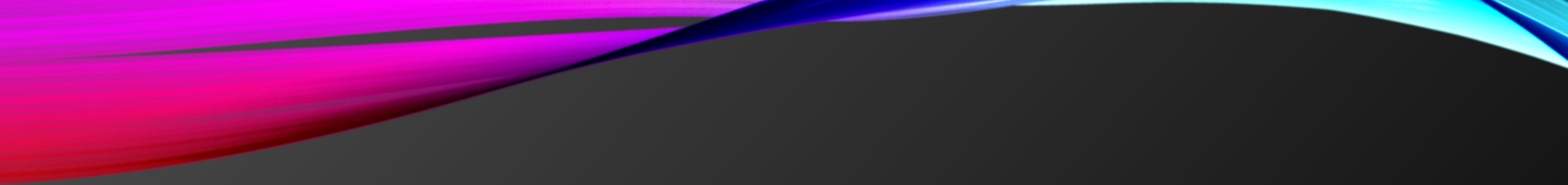
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/2001	82000	-	10
7782	CLARK	MANAGER	7839	06/09/2009	27500	-	10
7934	MILLER	CLERK	7782	01/23/2006	13200	-	10
7566	JONES	MANAGER	7839	04/02/2012	26850	-	20
7788	SCOTT	ANALYST	7566	12/09/2008	19500	-	20
7902	FORD	ANALYST	7566	12/03/2014	21500	-	20
7369	SMITH	CLERK	7902	12/17/2002	13750	-	20
7876	ADAMS	CLERK	7788	01/12/2011	11900	-	20
7698	BLAKE	MANAGER	7839	05/01/2005	24000	-	30
7499	ALLEN	SALESMAN	7698	02/20/2006	19000	6300	30
7521	WARD	SALESMAN	7698	02/22/2018	18500	4500	30
7654	MARTIN	SALESMAN	7698	09/28/2008	15650	3400	30
7844	TURNER	SALESMAN	7698	09/08/2011	18500	0	30
7900	JAMES	CLERK	7698	12/03/1998	11950	-	30
4422	STEVENS	MANAGER	7839	02/14/2019	24750	-	50
3258	GREEN	SALESMAN	4422	07/24/2003	18200	2750	50
6548	BARNES	CLERK	4422	01/16/2018	11950	-	50

JOINING A TABLE TO ITSELF

```
SELECT WORKER.ename, MANAGER.ename AS MANAGER
FROM emp WORKER, emp MANAGER
WHERE WORKER.mgr = MANAGER.empno
```



WORKER	MANAGER
BLAKE	KING
CLARK	KING
JONES	KING
STEVENS	KING
ALLEN	BLAKE
WARD	BLAKE
MARTIN	BLAKE
TURNER	BLAKE
JAMES	BLAKE
MILLER	CLARK
SCOTT	JONES
FORD	JONES
ADAMS	SCOTT
SMITH	FORD
GREEN	STEVENS
BARNES	STEVENS



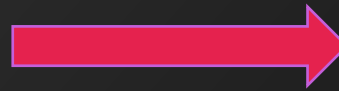
SQL FUNCTIONS

SQL has a wide range of built-in functions that can be applied to data.

STRING FUNCTIONS

Converts each string value in the specified column to lowercase...

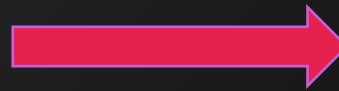
```
SELECT LOWER (ename) FROM emp;
```



king
blake
clark
jones
scott

Finds the length of each string value in the specified column...

```
SELECT LENGTH (ename) FROM emp  
WHERE ename = 'JONES';
```



5

LOWER
UPPER
LENGTH
INITCAP
SUBSTR
LTRIM
RTRIM
LPAD
RPAD
Etc.

AGGREGATE FUNCTIONS

Aggregate functions return a SINGLE result based on a GROUP of rows...

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

Returns the maximum salary from all the salaries

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

Returns the minimum salary from all the salaries

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

Calculates the sum of all salaries

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

Counts the number of rows in the EMP table

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

Calculates the average salary

AGGREGATE FUNCTIONS

A common error everyone gets when they start with aggregate functions...

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

Returns a single value

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

Will produce an error.
Why do you think that is?



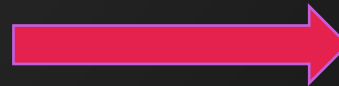
Error at line 1/8: ORA-00937: not a single-group group function

GROUP BY STATEMENT

The GROUP BY statement divides rows returned by a query into groups...

Example 1: Give me a list of the different jobs:

```
SELECT job FROM emp  
GROUP BY job;
```



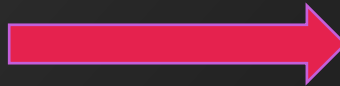
JOB
ANALYST
CLERK
SALESMAN
MANAGER
PRESIDENT

GROUP BY STATEMENT

GROUP BY statement are commonly used with aggregate functions
(SUM, COUNT, AVG, MIN, MAX)

Example 1: Let's revisit our error from earlier:

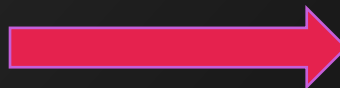
```
SELECT deptno, SUM(sal) FROM emp;
```



Error at line 1/8: ORA-00937: not a single-group group function

If we add the GROUP BY statement to our query...
we can now get a sum of the salaries for each department:

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

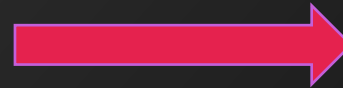


DEPTNO	SUM(SAL)
50	54900
30	107600
10	122700
20	93500

GROUP BY STATEMENT

Example 2: How many employees are in each department:

```
SELECT deptno, COUNT(ename)  
FROM emp  
GROUP BY deptno;
```



DEPTNO	COUNT(ENAME)
50	3
30	6
10	3
20	5

RESOURCES

- [W3 Schools - SQL Tutorials](#)
- [Oracle Documentation](#)