Dr Joseph Vella

Dept. of Computer Information Systems

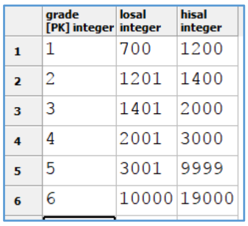
Introduction to Databases

SQL – Doing Joins in PostgreSQL (tested with version 9.6)

We are using SCOTT schema. If need be, download SCOTT schema (SQL constructs to create schema, tables, constraints and data) and run it.

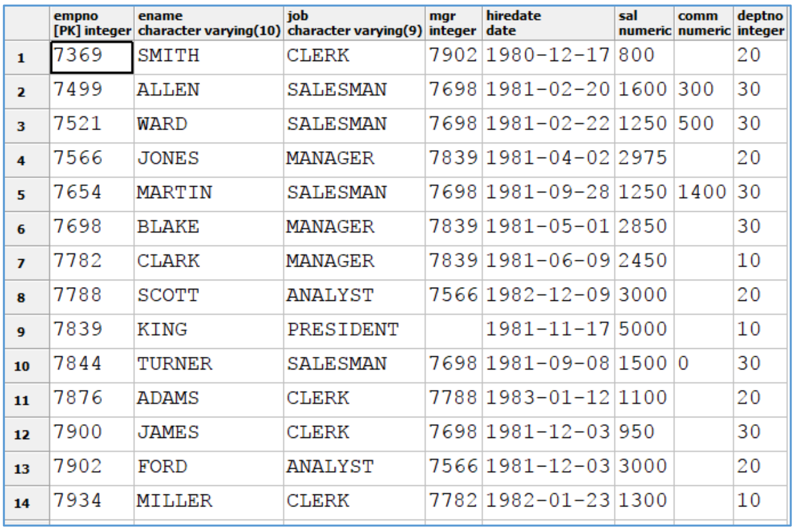
Please familiarise yourself with the schema tables and their relative content.

Table: salgrade



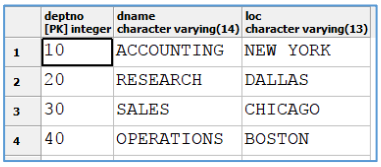
	grade [PK] integer	losal integer	hisal integer
1	1	700	1200
2	2	1201	1400
3	3	1401	2000
4	4	2001	3000
5	5	3001	9999
6	6	10000	19000

Table: emp



	empno [PK] integer	ename character varying(10)	job character varying(9)	mgr integer	hiredate date	sal numeric	comm numeric	deptno integer
1	7369	SMITH	CLERK	7902	1980-12-17	800		20
2	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
3	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
4	7566	JONES	MANAGER	7839	1981-04-02	2975		20
5	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	1981-05-01	2850		30
7	7782	CLARK	MANAGER	7839	1981-06-09	2450		10
8	7788	SCOTT	ANALYST	7566	1982-12-09	3000		20
9	7839	KING	PRESIDENT		1981-11-17	5000		10
10	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
11	7876	ADAMS	CLERK	7788	1983-01-12	1100		20
12	7900	JAMES	CLERK	7698	1981-12-03	950		30
13	7902	FORD	ANALYST	7566	1981-12-03	3000		20
14	7934	MILLER	CLERK	7782	1982-01-23	1300		10

Table: dept

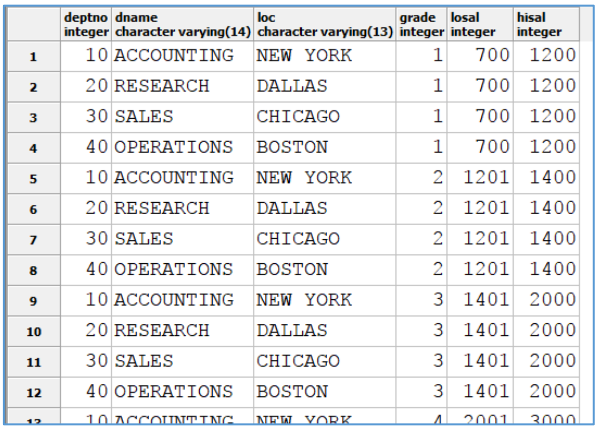


	deptno [PK] integer	dname character varying(14)	loc character varying(13)
1	10	ACCOUNTING	NEW YORK
2	20	RESEARCH	DALLAS
3	30	SALES	CHICAGO
4	40	OPERATIONS	BOSTON

No join: All combination of tuples from the listed tables.

```
SELECT *

  FROM scott.dept d CROSS JOIN scott.salgrade s;
```



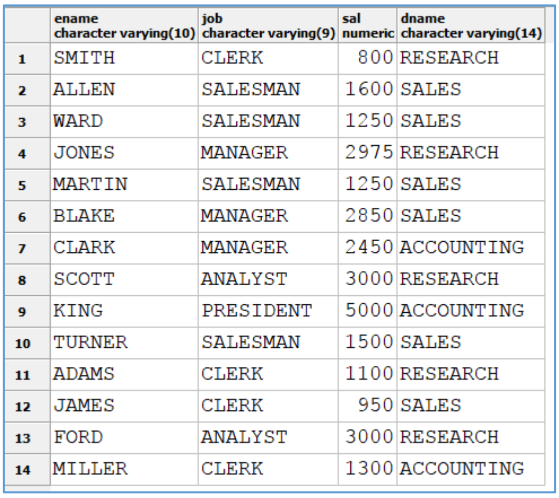
	deptno integer	dname character varying(14)	loc character varying(13)	grade integer	losal integer	hisal integer
1	10	ACCOUNTING	NEW YORK	1	700	1200
2	20	RESEARCH	DALLAS	1	700	1200
3	30	SALES	CHICAGO	1	700	1200
4	40	OPERATIONS	BOSTON	1	700	1200
5	10	ACCOUNTING	NEW YORK	2	1201	1400
6	20	RESEARCH	DALLAS	2	1201	1400
7	30	SALES	CHICAGO	2	1201	1400
8	40	OPERATIONS	BOSTON	2	1201	1400
9	10	ACCOUNTING	NEW YORK	3	1401	2000
10	20	RESEARCH	DALLAS	3	1401	2000
11	30	SALES	CHICAGO	3	1401	2000
12	40	OPERATIONS	BOSTON	3	1401	2000
13	10	ACCOUNTING	NEW YORK	4	2001	3000

Output consists of 24 rows in total.s

Natural Join: Match tuples from two tables on two corresponding, or common, attributes.

```
SELECT e.ename, e.job, e.sal, d.dname

  FROM scott.emp e INNER JOIN scott.dept d ON e.deptno=d.deptno;
```

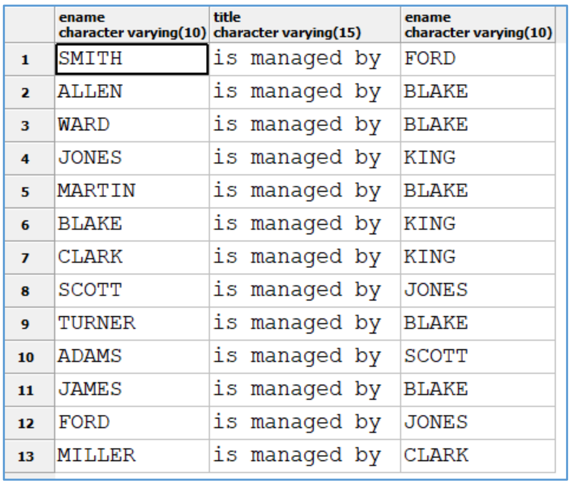


	ename character varying(10)	job character varying(9)	sal numeric	dname character varying(14)
1	SMITH	CLERK	800	RESEARCH
2	ALLEN	SALESMAN	1600	SALES
3	WARD	SALESMAN	1250	SALES
4	JONES	MANAGER	2975	RESEARCH
5	MARTIN	SALESMAN	1250	SALES
6	BLAKE	MANAGER	2850	SALES
7	CLARK	MANAGER	2450	ACCOUNTING
8	SCOTT	ANALYST	3000	RESEARCH
9	KING	PRESIDENT	5000	ACCOUNTING
10	TURNER	SALESMAN	1500	SALES
11	ADAMS	CLERK	1100	RESEARCH
12	JAMES	CLERK	950	SALES
13	FORD	ANALYST	3000	RESEARCH
14	MILLER	CLERK	1300	ACCOUNTING

Self Join: Match tuples from two tables on two corresponding, or common, attributes. But the two tables are really the same one – very important to note the table alias mechanism.

```
SELECT e.ename,'is managed by'::character varying(15) as title, m.ename

  FROM scott.emp e INNER JOIN scott.emp m ON e.mgr=m.empno;
```



	ename character varying(10)	title character varying(15)	ename character varying(10)
1	SMITH	is managed by	FORD
2	ALLEN	is managed by	BLAKE
3	WARD	is managed by	BLAKE
4	JONES	is managed by	KING
5	MARTIN	is managed by	BLAKE
6	BLAKE	is managed by	KING
7	CLARK	is managed by	KING
8	SCOTT	is managed by	JONES
9	TURNER	is managed by	BLAKE
10	ADAMS	is managed by	SCOTT
11	JAMES	is managed by	BLAKE
12	FORD	is managed by	JONES
13	MILLER	is managed by	CLARK

Note that employee King is missing (in fact we have 13 not 14 rows). That's because our King as no manager (he's the boss!?).

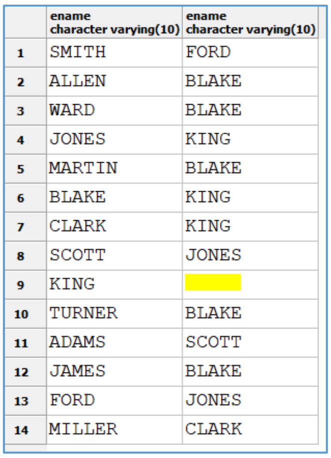
Outer Join:

Match all combinations of tuples from two tables that are equal on their common attribute, in addition to this (i.e. inner join) all the tuples in either table that have no matching tuples in the other. We have one side (i.e. left, right) and full outer joins.

-- left outer join (because we want all tuples from emp as an employee).

```
SELECT e.ename,m.ename

  FROM scott.emp e LEFT OUTER JOIN scott.emp m ON e.mgr=m.empno;
```



	ename character varying(10)	ename character varying(10)
1	SMITH	FORD
2	ALLEN	BLAKE
3	WARD	BLAKE
4	JONES	KING
5	MARTIN	BLAKE
6	BLAKE	KING
7	CLARK	KING
8	SCOTT	JONES
9	KING	🟨
10	TURNER	BLAKE
11	ADAMS	SCOTT
12	JAMES	BLAKE
13	FORD	JONES
14	MILLER	CLARK

Compare this output (especially with the null (yellow highlight)) to the output of the previous self-join. We have all the employees now (i.e. 14).

-- **full outer join**

-- say we want to generate a set of evens (2 to 99) and tiplets (3 to 99)

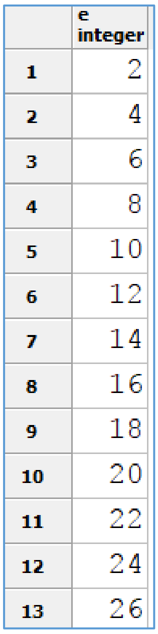
```
SELECT evens.*
  FROM generate_series(2,99,2) evens(e);


SELECT triplets.*
  FROM generate_series(3,99,3) triplets(t);
```

-- we want a list of all even and triplets i two columns (and is a number is both even and a triplet we list then both on the same line)

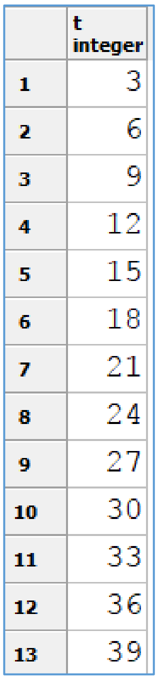
```
SELECT *
FROM (SELECT * FROM generate_series(2,99,2) ) evens(e)
      FULL OUTER JOIN
     (SELECT * FROM generate_series(3,99,3) ) triplets(t)
      ON evens.e=triplets.t;
```

Evens:



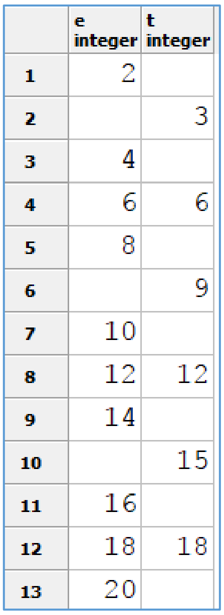
	e integer
1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18
10	20
11	22
12	24
13	26

Triplets:



	t integer
1	3
2	6
3	9
4	12
5	15
6	18
7	21
8	24
9	27
10	30
11	33
12	36
13	39

Outer Join of 2s & 3s:



	e integer	t integer
1	2	
2		3
3	4	
4	6	6
5	8	
6		9
7	10	
8	12	12
9	14	
10		15
11	16	
12	18	18
13	20	

Union Operator:

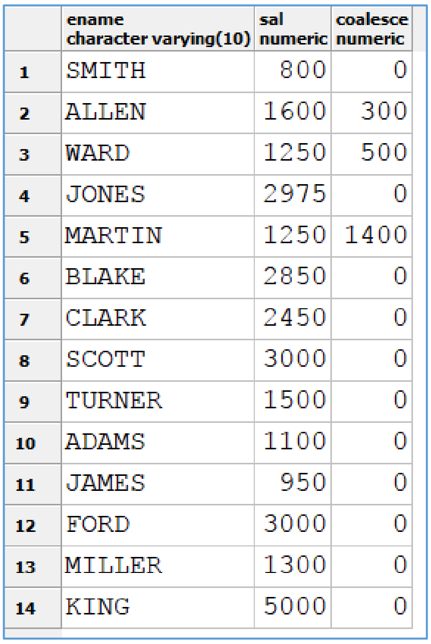
Look at the following wages bill for a month.

```
SELECT e.ename, e.sal, coalesce(e.comm,0)
  FROM scott.emp e;
```

-- but we want a line for sal and a line for comm

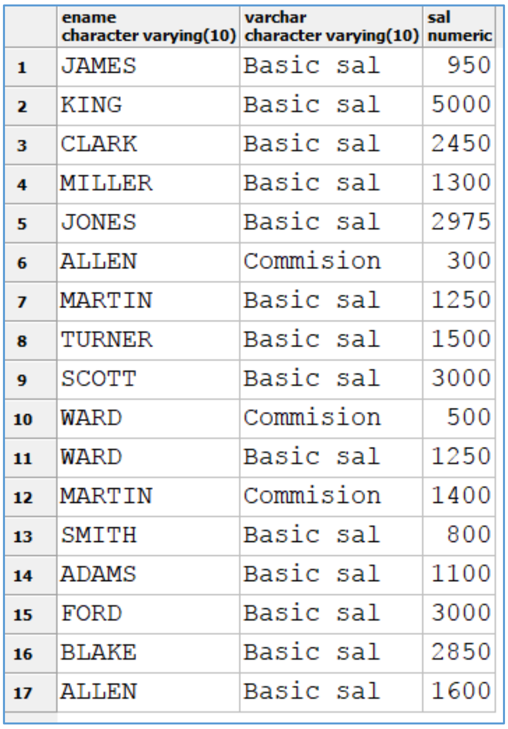
```
SELECT e.ename, 'Basic sal'::character varying(10), e.sal
  FROM scott.emp e
UNION
SELECT e.ename, 'Commision'::character varying(10), e.comm
  FROM scott.emp e
 WHERE e.comm is not null
   AND e.comm > 0;
```

Salary and Commission on same line.



	ename character varying(10)	sal numeric	coalesce numeric
1	SMITH	800	0
2	ALLEN	1600	300
3	WARD	1250	500
4	JONES	2975	0
5	MARTIN	1250	1400
6	BLAKE	2850	0
7	CLARK	2450	0
8	SCOTT	3000	0
9	TURNER	1500	0
10	ADAMS	1100	0
11	JAMES	950	0
12	FORD	3000	0
13	MILLER	1300	0
14	KING	5000	0

Salary and Commission entries on separate lines

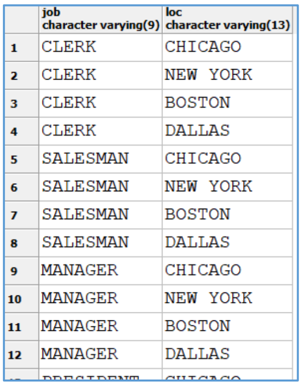


	ename character varying(10)	varchar character varying(10)	sal numeric
1	JAMES	Basic sal	950
2	KING	Basic sal	5000
3	CLARK	Basic sal	2450
4	MILLER	Basic sal	1300
5	JONES	Basic sal	2975
6	ALLEN	Commision	300
7	MARTIN	Basic sal	1250
8	TURNER	Basic sal	1500
9	SCOTT	Basic sal	3000
10	WARD	Commision	500
11	WARD	Basic sal	1250
12	MARTIN	Commision	1400
13	SMITH	Basic sal	800
14	ADAMS	Basic sal	1100
15	FORD	Basic sal	3000
16	BLAKE	Basic sal	2850
17	ALLEN	Basic sal	1600

Union vs. Product

Example: Give a permutation of all job and locations (product).

```
SELECT *
FROM (SELECT DISTINCT e.job as job FROM scott.emp e) as ee CROSS JOIN
     (SELECT DISTINCT d.loc as loc FROM scott.dept d) as dd;
```



	job character varying(9)	loc character varying(13)
1	CLERK	CHICAGO
2	CLERK	NEW YORK
3	CLERK	BOSTON
4	CLERK	DALLAS
5	SALESMAN	CHICAGO
6	SALESMAN	NEW YORK
7	SALESMAN	BOSTON
8	SALESMAN	DALLAS
9	MANAGER	CHICAGO
10	MANAGER	NEW YORK
11	MANAGER	BOSTON
12	MANAGER	DALLAS
	PRESIDENT	CHICAGO

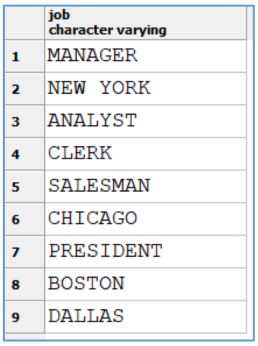
There are twenty tuples in total.

Example: Give a list of all job and locations (union).

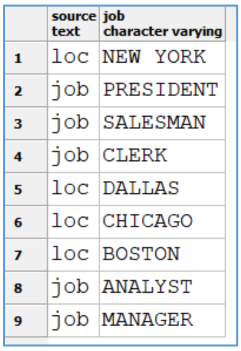
```
SELECT DISTINCT e.job as job FROM scott.emp e
UNION
SELECT DISTINCT d.loc as loc FROM scott.dept d;
```

Or to keep track from where the data derived (which expression).

```
SELECT DISTINCT 'job' as source, e.job as job FROM scott.emp e
UNION
SELECT DISTINCT 'loc' as source, d.loc as loc FROM scott.dept d;
```



	job character varying
1	MANAGER
2	NEW YORK
3	ANALYST
4	CLERK
5	SALESMAN
6	CHICAGO
7	PRESIDENT
8	BOSTON
9	DALLAS



	source text	job character varying
1	loc	NEW YORK
2	job	PRESIDENT
3	job	SALESMAN
4	job	CLERK
5	loc	DALLAS
6	loc	CHICAGO
7	loc	BOSTON
8	job	ANALYST
9	job	MANAGER