JOIN OPERATION IN DATABASE (QUERYING MULTIPLE TABLES)

Learning Outcomes

By the end of this session student should be able to:

Understand and know what is Relational Algebra (RA) for Join Operation
Transform from RA to SQL or vice versa

In this exercise, we will practice some functions provided in Oracle for DML - JOIN Operation.

JOINING DATABASE TABLES

The ability to combine, or join, tables on common attributes is perhaps the most important distinction between a relational database and other databases. A join is performed when data are retrieved from more than one table at a time.

For the minute of 7:40 - 11.13 of this video, it explained about CARTESIAN PRODUCT or CROSS-JOIN

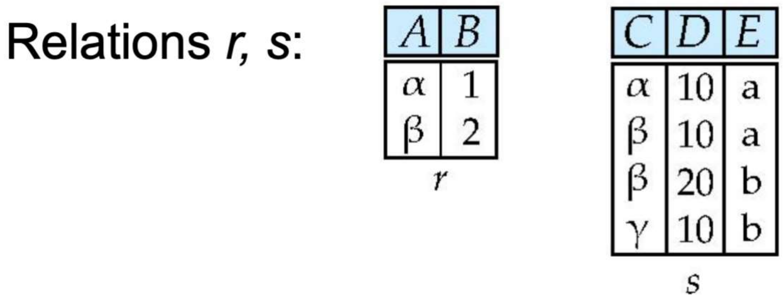
CARTESIAN PRODUCT (CROSS-JOIN)

RELATIONAL ALGEBRA (RA)

Notation:

```
r X s
```

It defines a relation that is the concatenation of every tuple of relation r with every tuple of relation s.



Relations *r*, *s*:

A	B
α	1
β	2

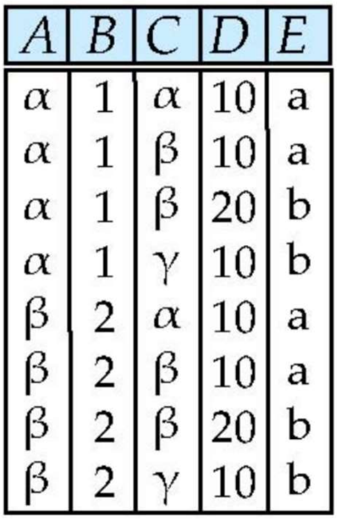
r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

For example, if tuple r has 2 columns and 2 rows, and Tuple s has 3 columns and 4 rows, then the new tuple produced from the cartesian product of relation r X s is 5 columns and 8 rows as shown below.

r x s:

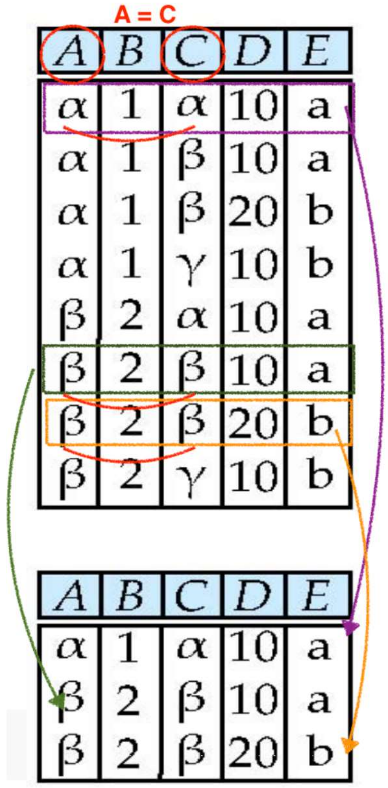


A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

In this example, assume that attributes of r(R) and s(S) are disjoint. (That is, R ∩ S = ∅) (each field of both tables has a different name)

We can build expressions using multiple operations. For example,

$\sigma_{A=C}$(r x s)



A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

For the exercise below, we will use PIZZA SQL script which can be download **HERE**

After, upload the SQL scripts, we should see the following tables:

– Person (name, age, gender)

– Frequents (name, pizzeria)

– Eats (name, pizza)

– Serves (pizzeria, pizza, price)

Results from a multi-table query that does not have a WHERE clause is a Cartesian product. The product results in a huge output which normally is not very useful.

An example of a Cartesian Product based on Pizza Schema is

Person X Eats

Similar like

```
πname, age, gender (Person) X  πname, pizza (Eats)
```

This operation creates a relation combining two relations, concatenating every tuple in one relation with every tuple in another relation. In simple terms, if the **Person** table has 9 records with 3 attributes and the **Eats** table has 20 records with 2 attributes, this creates an output with 180 records (9 * 20) and 5 attributes (3 + 2).

SQL

The SQL statement for the above example is shown below

```
SELECT * FROM Person, Eats
```

Or

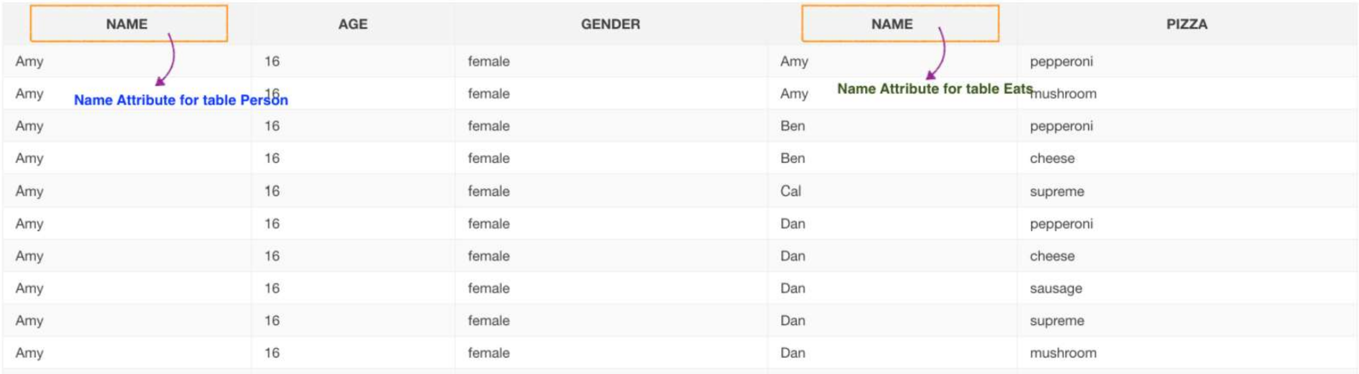
```
SELECT * FROM Person CROSS JOIN Eats
```

CARTESIAN PRODUCT AND SELECTION

Query:

Find all pizzas eaten by at least one female over the age of 20.

For this query two tables are required which are **Person** and **Eats**. So we will do **CROSS JOIN** operation as above. From results (180 records, with 5 columns) we found that there is a conflict where both tables have a field called NAME.


Name Attribute for table Person
Name Attribute for table Eats

NAME	AGE	GENDER	NAME	PIZZA
Amy	16	female	Amy	pepperoni
Amy	16	female	Amy	mushroom
Amy	16	female	Ben	pepperoni
Amy	16	female	Ben	cheese
Amy	16	female	Cal	supreme
Amy	16	female	Dan	pepperoni
Amy	16	female	Dan	cheese
Amy	16	female	Dan	sausage
Amy	16	female	Dan	supreme
Amy	16	female	Dan	mushroom

(Note: this table produce 180 rows of records)

We can use selection operation to extract those tuples where **Person.Name = Eats.Name**.

RA:

```
σ Person.name = Eats.name  (π name, age, gender (Person)) X ( π name, pizza (Eats))
```

OR

```
σ Person.name = Eats.name  (Person X Eats)
```

SQL:

```
SELECT * FROM PERSON
CROSS JOIN EATS
WHERE PERSON.NAME = EATS.NAME
```

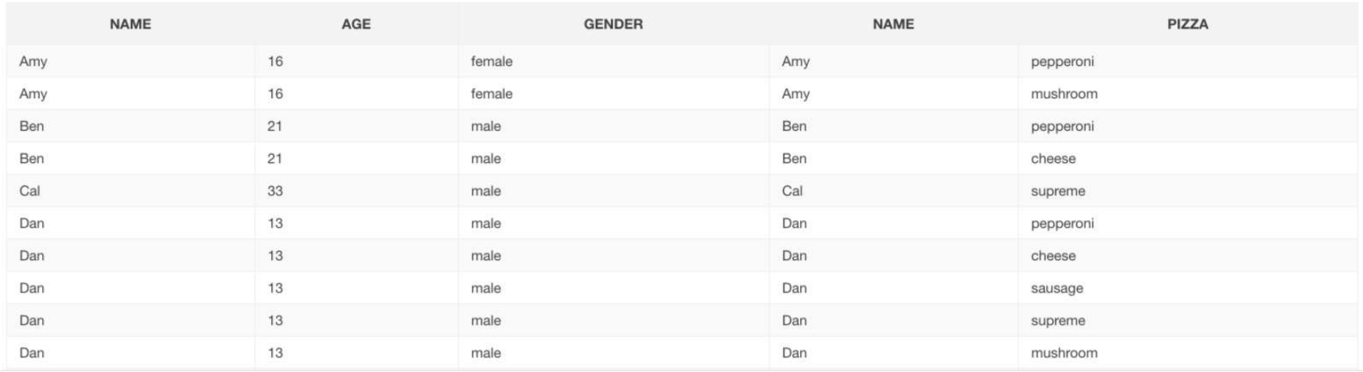
OR

```
SELECT *
FROM PERSON, EATS
WHERE PERSON.NAME = EATS.NAME
```

OR

```
SELECT * FROM PERSON P, EATS E
WHERE P.NAME = E.NAME
```

which yield below table



NAME	AGE	GENDER	NAME	PIZZA
Amy	16	female	Amy	pepperoni
Amy	16	female	Amy	mushroom
Ben	21	male	Ben	pepperoni
Ben	21	male	Ben	cheese
Cal	33	male	Cal	supreme
Dan	13	male	Dan	pepperoni
Dan	13	male	Dan	cheese
Dan	13	male	Dan	sausage
Dan	13	male	Dan	supreme
Dan	13	male	Dan	mushroom

(Note: this table produce 20 rows of records)

Notice that there are 5 columns and 20 records (rows).

To produce the result based on the query, we can write the other condition as shown below.

RA:

$$\sigma_{\text{Person.name = Eats.name ^ Person.gender = 'female' ^ Person.age >=20}} (\pi_{\text{name, age, gender}} (\text{Person})) \text{ X } (\pi_{\text{name, pizza}}(\text{Eats}))$$

OR

$$\sigma_{\text{Person.name = Eats.name ^ Person.gender = 'female' ^ Person.age >=20}} (\text{Person X Eats})$$

SQL:

```
SELECT *
FROM PERSON
CROSS JOIN EATS
WHERE PERSON.NAME = EATS.NAME
AND PERSON.GENDER = 'female'
AND PERSON.AGE >=20
```

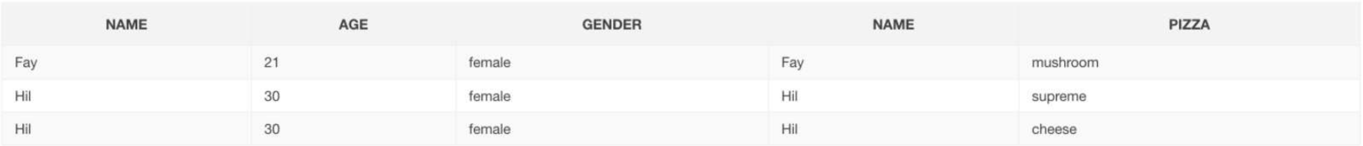
OR

```
SELECT *
FROM PERSON, EATS
WHERE PERSON.NAME = EATS.NAME
AND PERSON.GENDER = 'female'
AND PERSON.AGE >=20
```

OR

```
SELECT *
FROM PERSON P, EATS E
WHERE P.NAME = E.NAME
AND P.GENDER = 'female'
AND P.AGE >=20
```

RESULTS:



NAME	AGE	GENDER	NAME	PIZZA
Fay	21	female	Fay	mushroom
Hil	30	female	Hil	supreme
Hil	30	female	Hil	cheese

(Note: this table produce 3 rows of records)

Notice that there are 5 columns and 3 records (rows).

THETA-JOIN - Θ-JOIN

Theta join is called condition join. Meaning it joins two relations with the condition. The condition may use one of the comparison operators such as (<, ≤, >, ≥, =, ≠).

If the join condition uses an *equal operator (=)* then the operation is called **EQUIJOIN**.

Join in RA is written using a **'bow tie'** ($\bowtie$).

Example:

R = (A, B, C, D)

S = (E, B, D)

– Result schema = (A, B, C, D, E)

– r $\bowtie$ s is defined as:

$\pi_{r.A,\ r.B,\ r,C,\ r.D,\ s.E}$ ($\sigma_{r.B\ =\ s.B\ \wedge\ r.D\ =\ s.D}$ (r X s))

RELATIONAL ALGEBRA (RA)

Notation:

Can rewrite Theta join using basic Selection and Cartesian product operations.

$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$

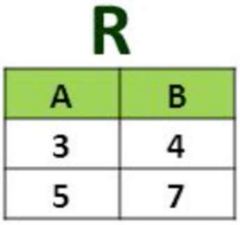
OR

$R \bowtie_{F} S = \sigma_{F}(R \times S)$

Defines a relation that contains tuples satisfying the predicate **F** or **θ** from the Cartesian product of R and S. The predicate **F** or **θ** is of the form R.ai θ S.bi where θ may be one of the comparisons operators (<, ≤, >, ≥, =, ≠).

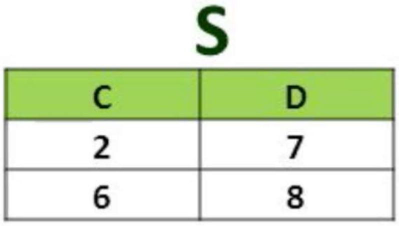
Example of Theta-Join and Greater than operator (>)

Relation R and S



R

A	B
3	4
5	7



S

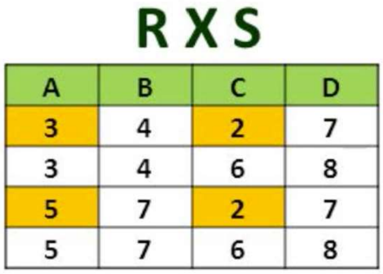
C	D
2	7
6	8

Suppose we want to find an A value in R relation that is greater than the value of C in S relation.

RA

$R \bowtie_{A > C} S = \sigma_{A > C}(R \times S)$

The operation starts by computing cartesian product R X S. And then selects those rows which satisfy that condition (A > C).

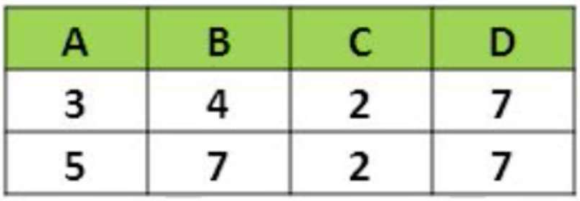


R X S

A	B	C	D
3	4	2	7
3	4	6	8
5	7	2	7
5	7	6	8

As a result, we get the Theta-Join table as shown below.

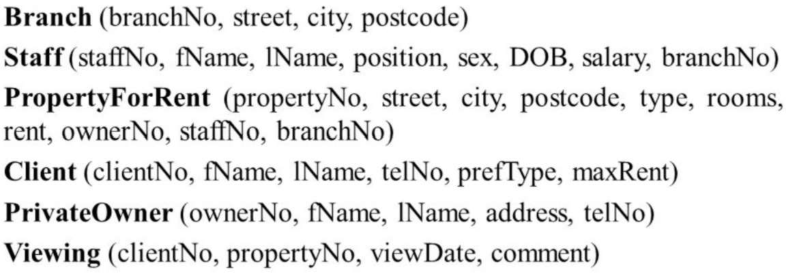
$$R \bowtie_{A>C} S$$



A	B	C	D
3	4	2	7
5	7	2	7

Most DBMS implemented theta join as a basic operation to combining relation. The exercise below is referring to our relational schema and example table as shown below.

RELATIONAL SCHEMA



Branch (branchNo, street, city, postcode)

Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)

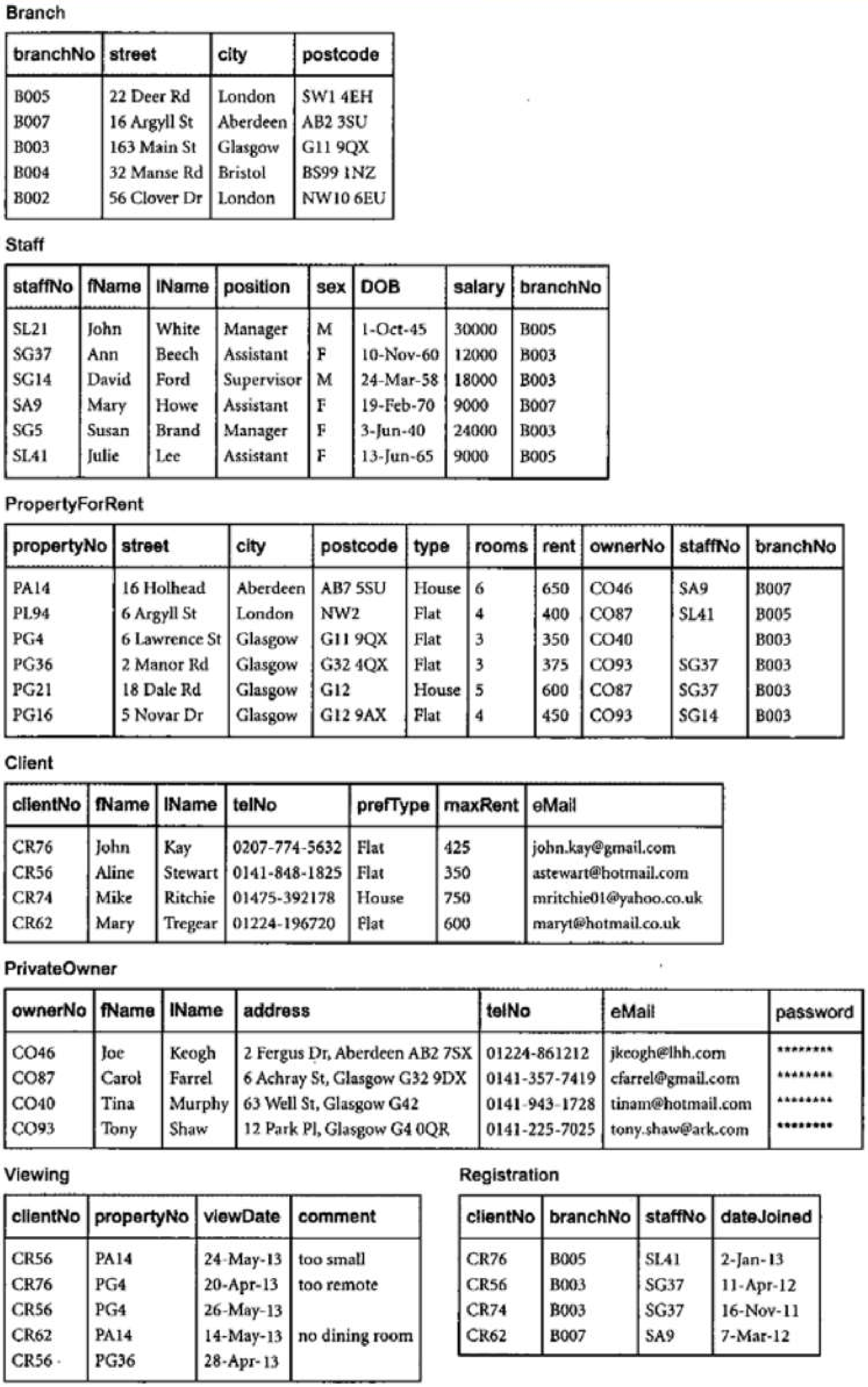
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Client (clientNo, fName, lName, telNo, prefType, maxRent)

PrivateOwner (ownerNo, fName, lName, address, telNo)

Viewing (clientNo, propertyNo, viewDate, comment)

EXAMPLE TABLE OF DREAMHOME DATABASE



Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

PrivateOwner

ownerNo	fName	lName	address	telNo	eMail	password
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	jkeogh@lhh.com	********
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	cfarrel@gmail.com	********
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	tinam@hotmail.com	********
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	tony.shaw@ark.com	********

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-13
CR56	B003	SG37	11-Apr-12
CR74	B003	SG37	16-Nov-11
CR62	B007	SA9	7-Mar-12

Query:

Find all the staff info including their address (street, city, postcode) where they worked?

In order to the information, we need to combine each **Staff** tuple and **Branch** tuple where **BrancNo** matches between both tables. In this case, the join condition is **Branch.BranchNO = Staff.BranchNo**

RA:

$$\textbf{Branch} \bowtie_{\text{Branch.BranchNo = Staff.BranchNo}} \textbf{Staff}$$

$$= \sigma_{\text{Branch.BranchNo = Staff.BranchNo}} (\textbf{Branch X Staff})$$

SQL:

```
SELECT *
FROM BRANCH
JOIN STAFF
ON BRANCH.BRANCHNO = STAFF.BRANCHNO
```

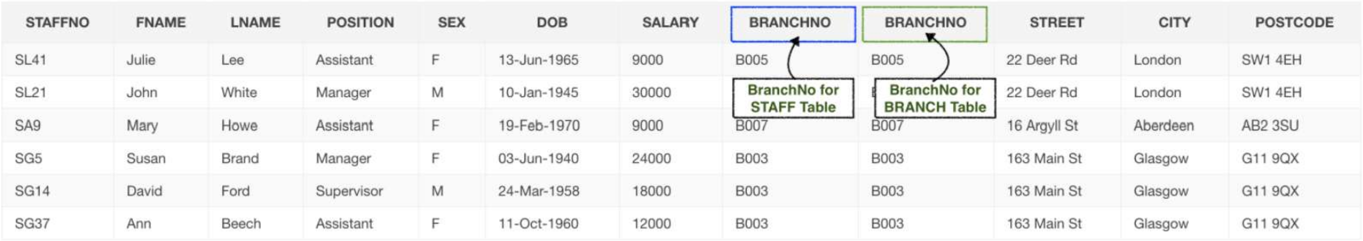
OR

```
SELECT *
FROM BRANCH, STAFF
WHERE BRANCH.BRANCHNO = STAFF.BRANCHNO
```

OR

```
SELECT *
FROM BRANCH B, STAFF S
WHERE B.BRANCHNO = S.BRANCHNO
```

RESULTS:



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO	BRANCHNO	STREET	CITY	POSTCODE
SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005	B005	22 Deer Rd	London	SW1 4EH
SL21	John	White	Manager	M	10-Jan-1945	30000			22 Deer Rd	London	SW1 4EH
SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007	B007	16 Argyll St	Aberdeen	AB2 3SU
SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003	B003	163 Main St	Glasgow	G11 9QX
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003	B003	163 Main St	Glasgow	G11 9QX
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	B003	163 Main St	Glasgow	G11 9QX

The difference between **Condition join (Theta Join)** and **Natural Join** is Natural join will display BranchNo once in the table. Theta-join is similar to **Cartesian Product with Selection Condition**. This operation starts with compute cartesian product BRANCH X STAFF and then selects those rows which satisfy that condition. As the condition above uses an equal operator (BranchNo.BranchNo = Staff.BranchNo), it's called EQUIJOIN.

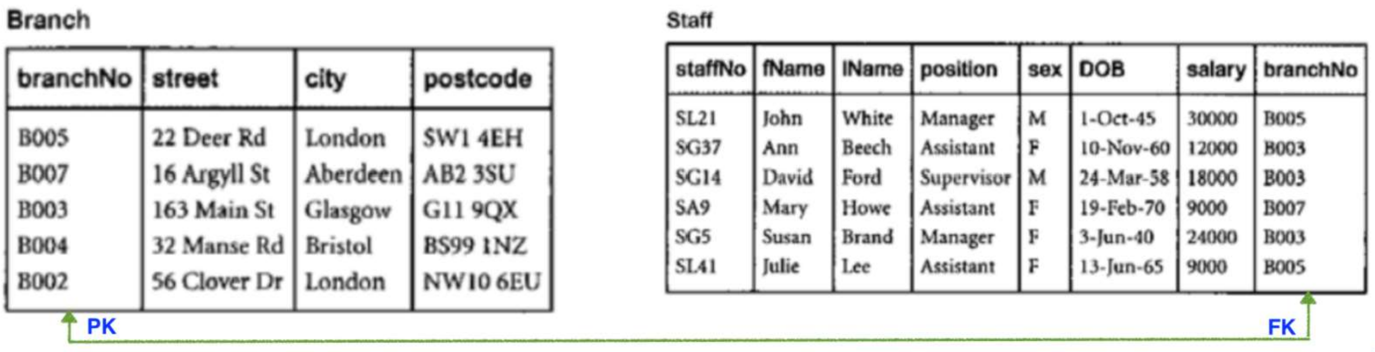
EQUIJOIN OR INNER JOIN

Equijoin is when we have a condition with an equality operator (=) that is when two relations connected with the primary key and foreign key.

Notation:

$$\textbf{R} \bowtie_{\text{R.Primarykey = S.Foreignkey}} \textbf{S} = \sigma_{\text{R.Primarykey = S.Foreignkey}} (\textbf{R X S})$$

Relation BRANCH & STAFF



Branch

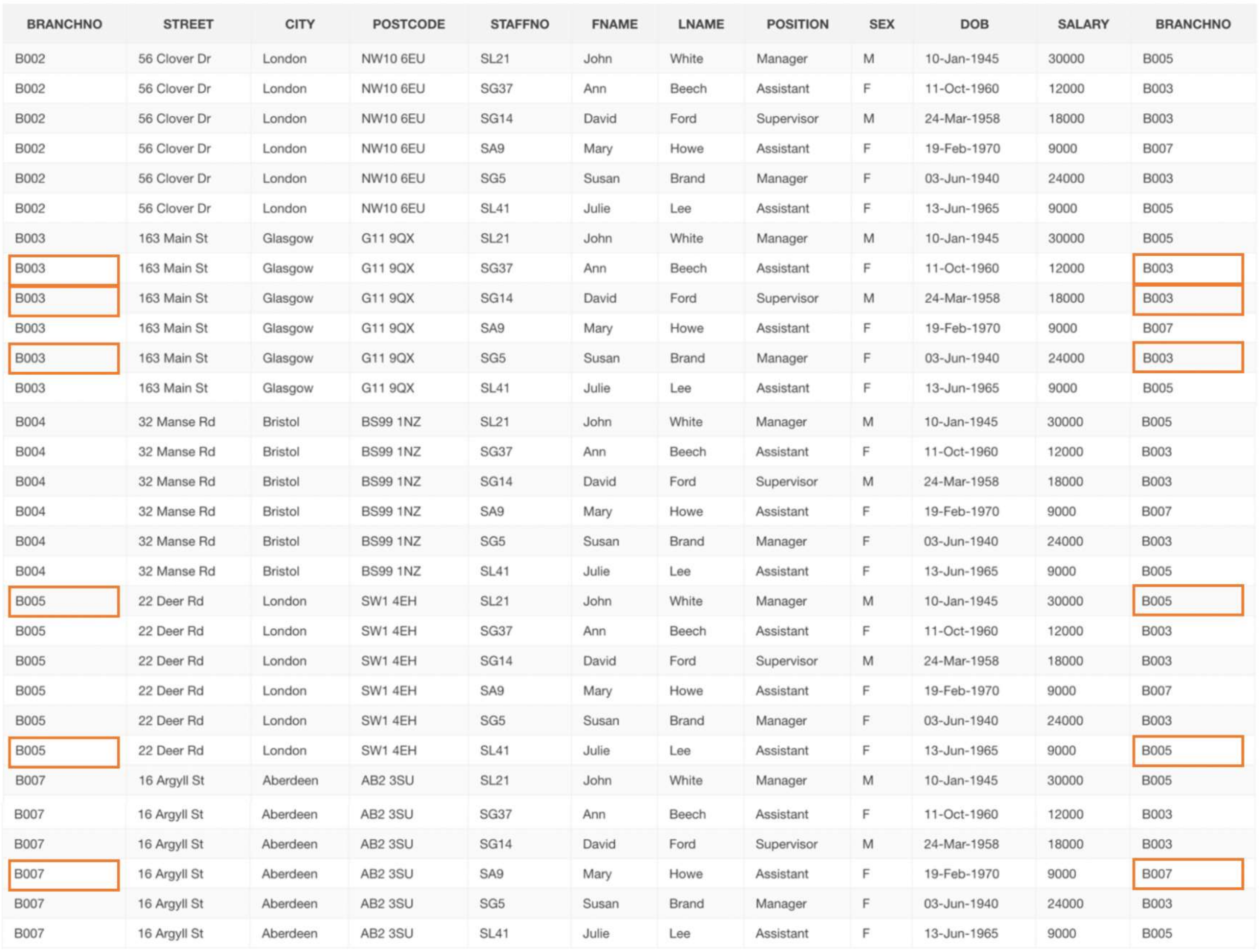
branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

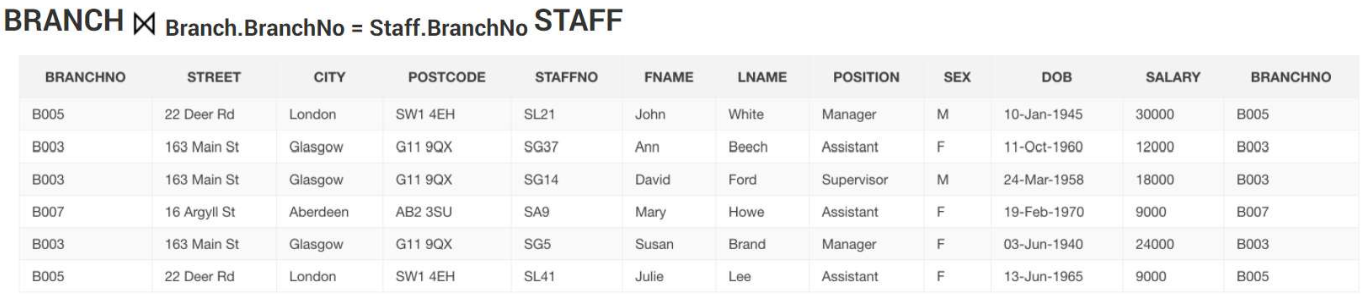
BRANCH X STAFF (CARTESIAN PRODUCT BRANCH X STAFF)

Branch Table has 5 rows with 4 columns and the Staff table has 6 rows with 8 columns. After Cartesian Product, we will get 30 rows with 12 columns as shown below.



BRANCHNO	STREET	CITY	POSTCODE	STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO
B002	56 Clover Dr	London	NW10 6EU	SL21	John	White	Manager	M	10-Jan-1945	30000	B005
B002	56 Clover Dr	London	NW10 6EU	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
B002	56 Clover Dr	London	NW10 6EU	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
B002	56 Clover Dr	London	NW10 6EU	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
B002	56 Clover Dr	London	NW10 6EU	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
B002	56 Clover Dr	London	NW10 6EU	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005
B003	163 Main St	Glasgow	G11 9QX	SL21	John	White	Manager	M	10-Jan-1945	30000	B005
B003	163 Main St	Glasgow	G11 9QX	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
B003	163 Main St	Glasgow	G11 9QX	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
B003	163 Main St	Glasgow	G11 9QX	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
B003	163 Main St	Glasgow	G11 9QX	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
B003	163 Main St	Glasgow	G11 9QX	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005
B004	32 Manse Rd	Bristol	BS99 1NZ	SL21	John	White	Manager	M	10-Jan-1945	30000	B005
B004	32 Manse Rd	Bristol	BS99 1NZ	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
B004	32 Manse Rd	Bristol	BS99 1NZ	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
B004	32 Manse Rd	Bristol	BS99 1NZ	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
B004	32 Manse Rd	Bristol	BS99 1NZ	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
B004	32 Manse Rd	Bristol	BS99 1NZ	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005
B005	22 Deer Rd	London	SW1 4EH	SL21	John	White	Manager	M	10-Jan-1945	30000	B005
B005	22 Deer Rd	London	SW1 4EH	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
B005	22 Deer Rd	London	SW1 4EH	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
B005	22 Deer Rd	London	SW1 4EH	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
B005	22 Deer Rd	London	SW1 4EH	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
B005	22 Deer Rd	London	SW1 4EH	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005
B007	16 Argyll St	Aberdeen	AB2 3SU	SL21	John	White	Manager	M	10-Jan-1945	30000	B005
B007	16 Argyll St	Aberdeen	AB2 3SU	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
B007	16 Argyll St	Aberdeen	AB2 3SU	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
B007	16 Argyll St	Aberdeen	AB2 3SU	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
B007	16 Argyll St	Aberdeen	AB2 3SU	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
B007	16 Argyll St	Aberdeen	AB2 3SU	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005

After that, those rows which satisfy the condition (Branch.BranchNo = Staff.BranchNo) will be selected as the output as shown below.



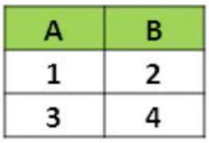
BRANCH ⋈ Branch.BranchNo = Staff.BranchNo STAFF

BRANCHNO	STREET	CITY	POSTCODE	STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO
B005	22 Deer Rd	London	SW1 4EH	SL21	John	White	Manager	M	10-Jan-1945	30000	B005
B003	163 Main St	Glasgow	G11 9QX	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
B003	163 Main St	Glasgow	G11 9QX	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
B007	16 Argyll St	Aberdeen	AB2 3SU	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
B003	163 Main St	Glasgow	G11 9QX	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
B005	22 Deer Rd	London	SW1 4EH	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005

NATURAL JOIN

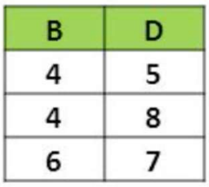
Natural join is a special case of equijoin. It enforces equality on all attributes with the same name. In addition, it eliminates one copy of duplicate attributes or columns.

R



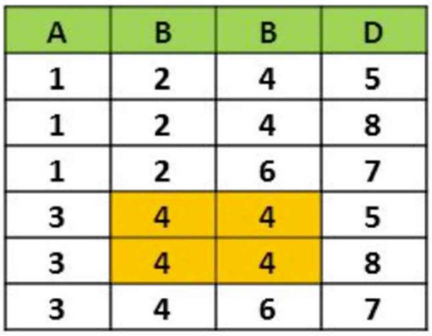
A	B
1	2
3	4

S



B	D
4	5
4	8
6	7

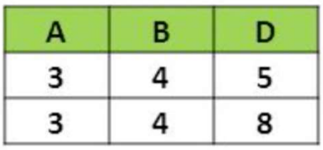
R X S



A	B	B	D
1	2	4	5
1	2	4	8
1	2	6	7
3	4	4	5
3	4	4	8
3	4	6	7

$R \bowtie S =$

$\pi\ A,R.B,D\ (R \bowtie_{R.B=S.B} S)$



A	B	D
3	4	5
3	4	8

From the Pizza example in the Cartesian Product section, we can see that both table Person and Eats has an attribute called Names.

We can eliminate one of the fields by using Natural Join. In this case, after the Person table is combined with the Eats table using Natural Join, we may get only one column NAME as both values are equal from both tables.

Example of Natural Join for based on Pizza Schema

RA

σ (Person ⋈ Eats)

SQL

```
SELECT *
FROM Person
NATURAL JOIN Eats
```

Another example from **DreamHome** tables, we can see the implementation of Natural Join for BRANCH table and STAFF table. It requires 2 relations/ tables with common column (PK & FK). Same like other joins, it starts from computing the Cartesian Product and then combines the tuples (columns) which have equal values in attributes with the same name.

Query:

Find all the staff info including their address (street, city, postcode) where they worked?

RA

σ (BRANCH $\bowtie$ STAFF)

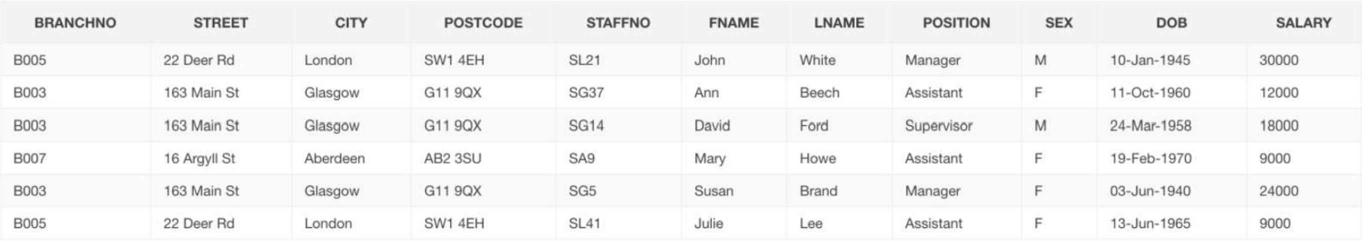
SQL

```
SELECT *
FROM BRANCH
NATURAL JOIN STAFF
```

OR

```
SELECT *
FROM BRANCH
JOIN STAFF
USING (BRANCHNO)
```

RESULTS:



BRANCHNO	STREET	CITY	POSTCODE	STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY
B005	22 Deer Rd	London	SW1 4EH	SL21	John	White	Manager	M	10-Jan-1945	30000
B003	163 Main St	Glasgow	G11 9QX	SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000
B003	163 Main St	Glasgow	G11 9QX	SG14	David	Ford	Supervisor	M	24-Mar-1958	18000
B007	16 Argyll St	Aberdeen	AB2 3SU	SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000
B003	163 Main St	Glasgow	G11 9QX	SG5	Susan	Brand	Manager	F	03-Jun-1940	24000
B005	22 Deer Rd	London	SW1 4EH	SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000

The difference between this result with the previous result in the Equijoin operation is we will get 6 rows with 12 columns. However, in Natural Join we get 6 rows with 11 columns as the duplicate column (BranchNo) has been eliminated.

NATURAL JOINS WITH CONDITION

Query:

Find all pizzas eaten by at least one female over the age of 20.

RA:

$\sigma_{gender = 'female' \wedge age >=20}$ (Person $\bowtie$ Eats)

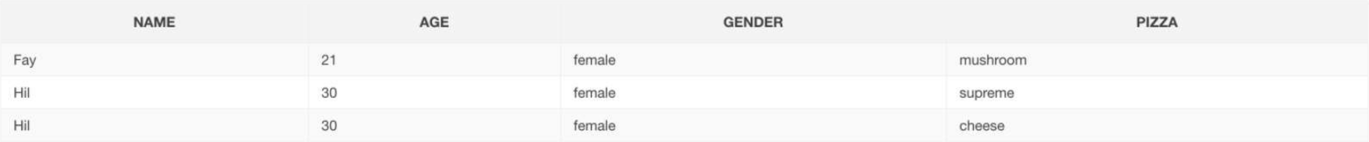
SQL:

```
SELECT  *
FROM PERSON
NATURAL JOIN EATS
WHERE GENDER = 'female'
AND
AGE >=20
```

OR

```
SELECT *
FROM PERSON P
NATURAL JOIN EATS E
WHERE P.GENDER = 'female'
AND
P.AGE >=20
```

RESULTS:



NAME	AGE	GENDER	PIZZA
Fay	21	female	mushroom
Hil	30	female	supreme
Hil	30	female	cheese

(Note: this table produce 3 rows of records)

Notice that there are 4 columns and 3 records (rows).

If there is a requirement to join those relations with another table SERVES, we just need to add another bow tie and table name.

RA

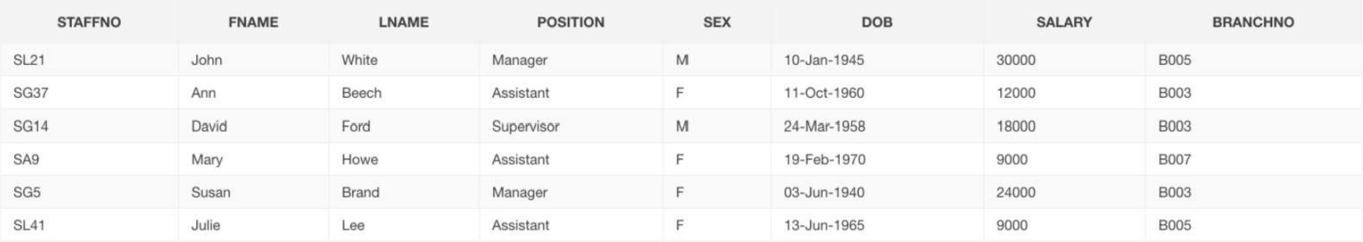
σ (Person ⋈ (Eats ⋈ Serves))

SQL

```
SELECT *
FROM Person
NATURAL JOIN Eats
NATURAL JOIN Serves
```

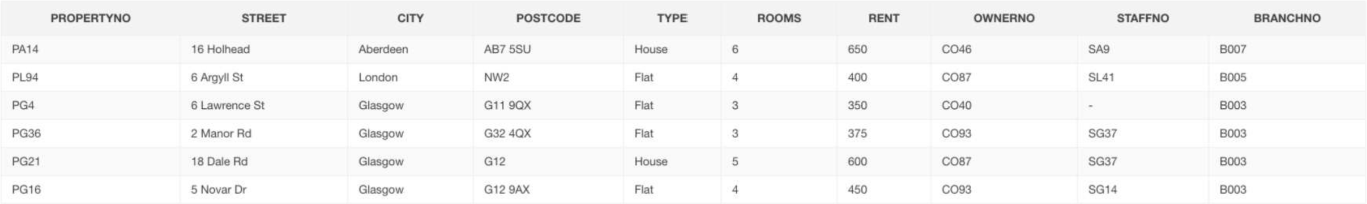
Suppose we have 2 tables as such below.

STAFF TABLE



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO
SL21	John	White	Manager	M	10-Jan-1945	30000	B005
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005

PROPERTYFORRENT TABLE



PROPERTYNO	STREET	CITY	POSTCODE	TYPE	ROOMS	RENT	OWNERNO	STAFFNO	BRANCHNO
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	-	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

σ (STAFF ⋈ PROPERTYFORRENT)

```
SELECT * FROM STAFF NATURAL JOIN PROPERTYFORRENT
```

For these two tables (STAFF and PROPERTYFORRENT), there are two columns (tuples) that have similar common attributes which are STAFFNO and BRANCHNO. The natural join will automatically rename one copy of each common attribute (column) before the cartesian product and use a projection to eliminate these double attributes at the end.

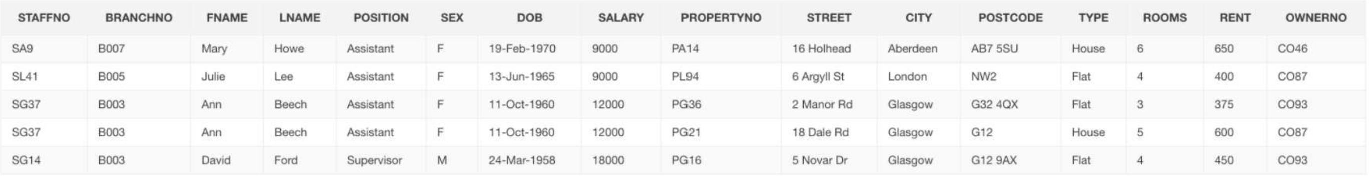
STAFF (**STAFFNO, BRANCHNO**, FNAME, LNAME...)

PROPERTYFORRENT(PROPERTYNO, CITY.., **STAFFNO, BRANCHNO**)

STAFF ⋈ PROPERTYFORRENT = $\pi_{\text{Staff.StaffNO, Staff.BranchNo, FName, LName, PropertyNo, City}}$ (STAFF

⋈ Staff.StaffNo = Propertyforrent.StaffNo ^ Staff.BranchNo = Propertyforrent.BranchNo PROPERTYFORRENT)

RESULT



STAFFNO	BRANCHNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	PROPERTYNO	STREET	CITY	POSTCODE	TYPE	ROOMS	RENT	OWNERNO
SA9	B007	Mary	Howe	Assistant	F	19-Feb-1970	9000	PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46
SL41	B005	Julie	Lee	Assistant	F	13-Jun-1965	9000	PL94	6 Argyll St	London	NW2	Flat	4	400	CO87
SG37	B003	Ann	Beech	Assistant	F	11-Oct-1960	12000	PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93
SG37	B003	Ann	Beech	Assistant	F	11-Oct-1960	12000	PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87
SG14	B003	David	Ford	Supervisor	M	24-Mar-1958	18000	PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93

After the elimination of 2 tuples (StaffNo and BranchNo) from Propertyforrent table, instead of 18 tuples (columns), we only have 16 tuples (columns).

However, one of the drawbacks of Natural Join is that it loses information about the tuples they do not match in both of the relation STAFF and PROPERTYFORRENT.

For STAFF information, we lost the information about StaffNo SG5 (Susan Brand) and StaffNo SL21 (John White). And for the PROPERTYFORRENT information, we lost the information about PropertyNo PG4, since they do not match in both relations.

OUTER JOIN

Outer join is an extension of join.

OUTER JOIN preserves dangling tuples by padding them with a NULL value in the result. It avoids the loss of information.

There are three (3) types of Outer Join.

1) Left Outer Join (⟕)

It takes all the tuples in the left relation that do not match with any tuple in the right. Fill tuple with NULL values ('-') for all other attributes from the right relations and add them to the result of the Natural Join.

For example, if we perform R ⟕ S then all the information from the left relation (R) present in the result.

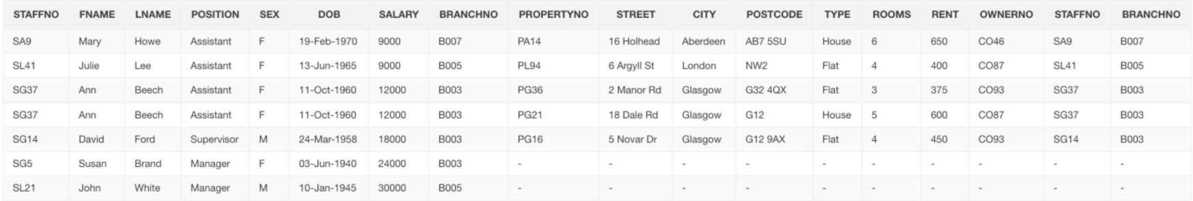
RA

STAFF ⟕ Staff.StaffNo = Propertyforrent.StaffNo ^ Staff.BranchNo = Propertyforrent.BranchNo PROPERTYFORRENT

SQL

```
SELECT * FROM Staff
LEFT OUTER JOIN Propertyforrent
ON Staff.StaffNo = Propertyforrent.StaffNo
AND Staff.BranchNo = Propertyforrent.BranchNo
```

RESULT



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO	PROPERTYNO	STREET	CITY	POSTCODE	TYPE	ROOMS	RENT	OWNERNO	STAFFNO	BRANCHNO
SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007	PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005	PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003	PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003	-	-	-	-	-	-	-	-	-	-
SL21	John	White	Manager	M	10-Jan-1945	30000	B005	-	-	-	-	-	-	-	-	-	-

2) RIGHT OUTER JOIN (⟖)

In this example, we can identify the Staff who is not assigning any property to be handled yet.

It fills tuple from the right relation that does not match any other from the left relation. Fill tuples with NULL values ('-') for all other attributes from the left relations and add them to the result of the Natural Join.

For example, if we perform R ⟖ S then all the information from the right relation (S) present in the result.

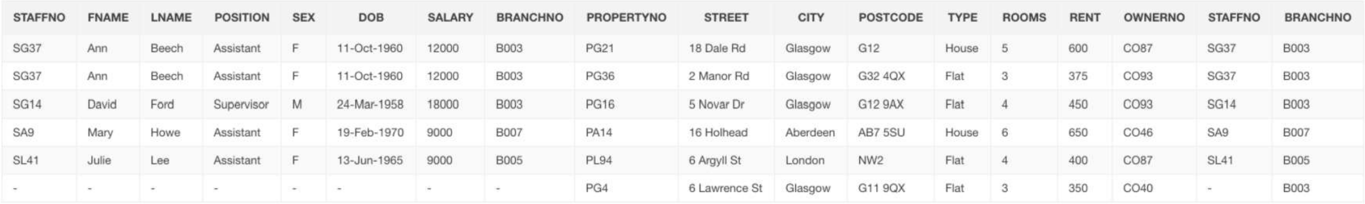
RA

STAFF ⟖ **Staff.StaffNo = Propertyforrent.StaffNo ^ Staff.BranchNo = Propertyforrent.BranchNo** **PROPERTYFORRENT**

SQL

```
SELECT * FROM Staff
RIGHT OUTER JOIN Propertyforrent
ON Staff.StaffNo = Propertyforrent.StaffNo
AND Staff.BranchNo = Propertyforrent.BranchNo
```

RESULT



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO	PROPERTYNO	STREET	CITY	POSTCODE	TYPE	ROOMS	RENT	OWNERNO	STAFFNO	BRANCHNO
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003	PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007	PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005	PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
-	-	-	-	-	-	-	-	PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	-	B003

In this example, we can identify the property which is not assigning to any staff yet (i.e. PG4).

3) Full Outer Join (⟗)

It does both of those left and right outer join. Fill tuples from the left relation that did not match any from the right relation as well as tuples from the left relation.

For example, if we perform R ⟗ S then all the information from the right and left relation (R) is present in the result.

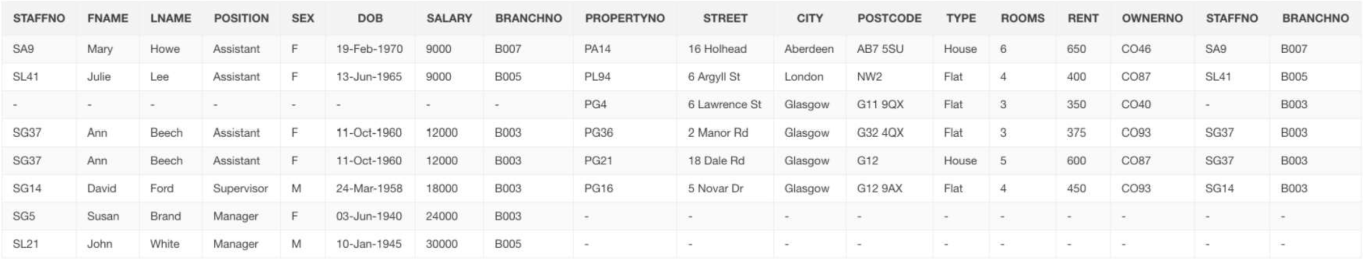
RA

STAFF ⟗ **Staff.StaffNo = Propertyforrent.StaffNo ^ Staff.BranchNo = Propertyforrent.BranchNo** **PROPERTYFORRENT**

SQL

```
SELECT * FROM Staff
FULL OUTER JOIN Propertyforrent
ON Staff.StaffNo = Propertyforrent.StaffNo
AND Staff.BranchNo = Propertyforrent.BranchNo
```

RESULT



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO	PROPERTYNO	STREET	CITY	POSTCODE	TYPE	ROOMS	RENT	OWNERNO	STAFFNO	BRANCHNO
SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007	PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005	PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
-	-	-	-	-	-	-	-	PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	-	B003
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003	PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003	PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003
SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003	-	-	-	-	-	-	-	-	-	-
SL21	John	White	Manager	M	10-Jan-1945	30000	B005	-	-	-	-	-	-	-	-	-	-

In this example, we can identify both Staff and Property No which still has no data assigned.

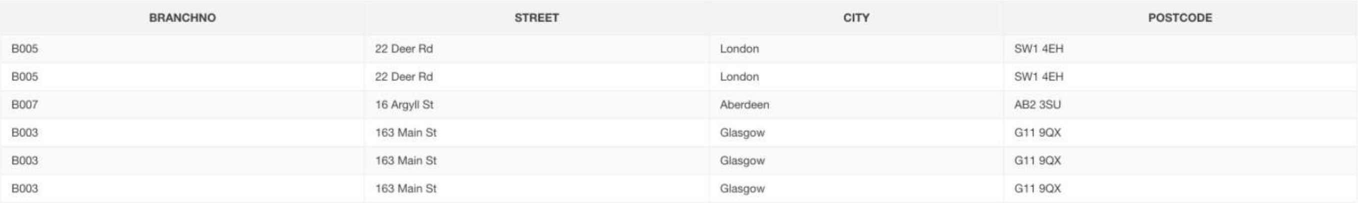
SEMI JOIN (⋉)

A semijoin returns rows that match an EXISTS subquery without duplicating rows from the left side of the predicate when multiple rows on the right side satisfy the criteria of the subquery. A semi-join is a join where the result only contains columns from one of the joined tables. Usually, it is used for reducing communication costs. Semi-joins are written using EXISTS or IN. The difference between Semi-Join and conventional join is that rows in the first table will be returned at most once. Even if the second table contains two matches for a row in the first table, only one copy of the row will be returned.

Query written in a conventional join is:

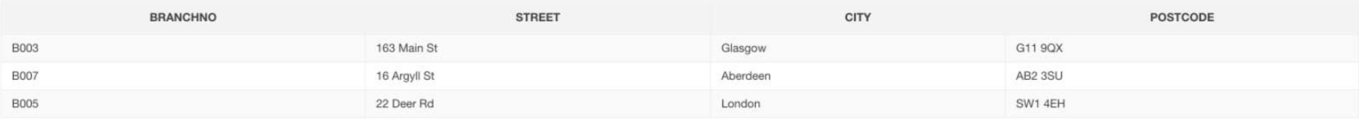
```
SELECT BranchNo, Street, City, Postcode FROM Branch JOIN Staff USING (BranchNo)
```

The result will display as below:



BRANCHNO	STREET	CITY	POSTCODE
B005	22 Deer Rd	London	SW1 4EH
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B003	163 Main St	Glasgow	G11 9QX
B003	163 Main St	Glasgow	G11 9QX

Notice that, a Branch with N staff will appear in the list N times (e.g. BranchNo B005, B003). We could use a DISTINCT keyword to get each branch to appear only once as shown below.



BRANCHNO	STREET	CITY	POSTCODE
B003	163 Main St	Glasgow	G11 9QX
B007	16 Argyll St	Aberdeen	AB2 3SU
B005	22 Deer Rd	London	SW1 4EH

Can write as Theta join using basic Selection and Cartesian product operations. Just replace the symbol of Semi-join.

$$R \ltimes_{F} S = \pi\, A(R \ltimes_{F} S)$$

<u>Query</u>:

Give a list of Branches with at least one staff.

<u>RA</u>

$$\mathbf{BRANCH} \ltimes \mathbf{STAFF} = \pi_{\text{BranchNo, Street, City, postcode}} (\text{Branch} \ltimes_{\text{BranchNo}} \text{Staff})$$

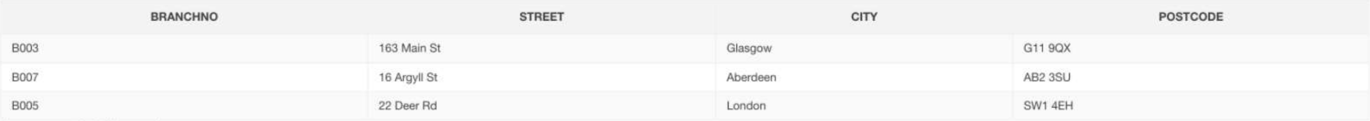
<u>SQL (with USING keyword)</u>

```
SELECT BranhNo, Street, City, Postcode FROM Branch JOIN Staff USING (BranchNo)
```

<u>SQL EXAMPLE 1 - (with EXIST keyword) (BRANCH ⋉ STAFF)</u>

```
SELECT *
FROM BRANCH
WHERE EXISTS
(SELECT *
FROM STAFF
WHERE BRANCH.BranchNo = STAFF.BranchNo)
```

<u>RESULT</u>



BRANCHNO	STREET	CITY	POSTCODE
B003	163 Main St	Glasgow	G11 9QX
B007	16 Argyll St	Aberdeen	AB2 3SU
B005	22 Deer Rd	London	SW1 4EH

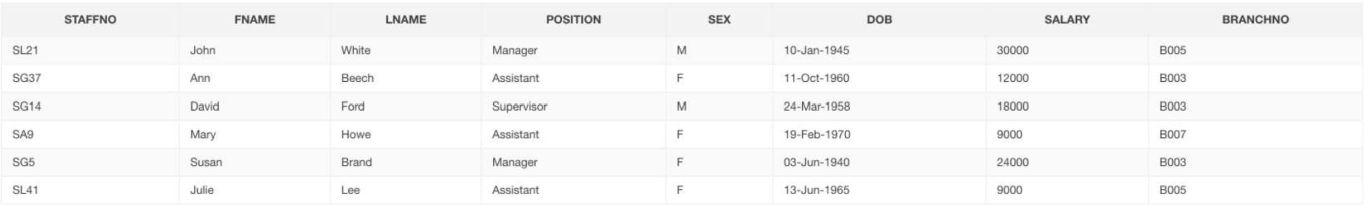
Semi-Join matches the rows of two relations and then shows the matching rows of the relation whose name is mentioned to the left side of ⋉ the Semi Join operator.

In this example, the table of Branch is located to the left side of the semi-join operator, so, we get the info about Branches.

<u>SQL EXAMPLE 2 - (with EXIST keyword) (STAFF ⋉ BRANCH)</u>

```
SELECT *
FROM STAFF
WHERE EXISTS
(SELECT *
FROM BRANCH
WHERE BRANCH.BranchNo = STAFF.BranchNo)
```

RESULT



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO
SL21	John	White	Manager	M	10-Jan-1945	30000	B005
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-1970	9000	B007
SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-1965	9000	B005

Query:

Give a complete detail of all staff who worked at the branch in 'Glasgow'

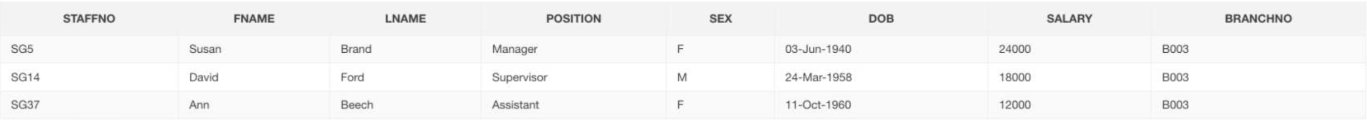
RA

$\mathbf{STAFF} \ltimes_{\text{Staff.BranchNo=Branch.BranchNo}} (\sigma_{\text{city='Glasgow'}} (\mathbf{BRANCH}))$

SQL

```
SELECT *
FROM STAFF
WHERE EXISTS
(SELECT *
FROM BRANCH
WHERE BRANCH.BranchNo = STAFF.BranchNo)
AND BRANCH.City = 'Glasgow'
```

RESULT



STAFFNO	FNAME	LNAME	POSITION	SEX	DOB	SALARY	BRANCHNO
SG5	Susan	Brand	Manager	F	03-Jun-1940	24000	B003
SG14	David	Ford	Supervisor	M	24-Mar-1958	18000	B003
SG37	Ann	Beech	Assistant	F	11-Oct-1960	12000	B003

Query:

Give a complete detail of branch where John White is worked

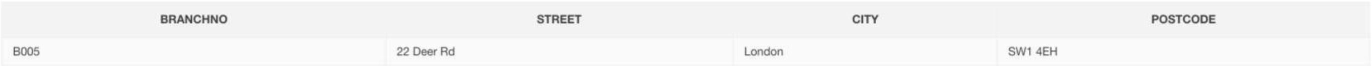
RA

$\mathbf{BRANCH} \ltimes_{\text{Branch.BranchNo=Staff.BranchNo}} (\sigma_{\text{FName='John' ^ LName = 'White'}} (\mathbf{STAFF}))$

SQL

```
SELECT *
FROM BRANCH
WHERE EXISTS
(SELECT *
FROM STAFF
WHERE BRANCH.BranchNo = STAFF.BranchNo
AND STAFF.FName = 'John'
AND STAFF.LName = 'White')
```

RESULT



BRANCHNO	STREET	CITY	POSTCODE
B005	22 Deer Rd	London	SW1 4EH