RELATIONAL DATA MODEL

2.2 OPERATORS OF RELATIONAL ALGEBRA

Mdm. ZATI HANANI ZAINAL (JTMK POLIMAS)

LEARNING OUTCOMES

2.1 Relational Databases

2.2 Operators of relational algebra

2.2.1 Describe the relational algebra.

2.2.2 Identify the fundamentals operators used to retrieve information from a relational database:

a) Restrict (select)
b) Project
c) Join (outer, inner)
d) Cross Product

2.2.3 Describe the purpose and input of each of the operators and expression.

2.2.4 Write the expressions by using the operators based on relational tables given.

2.2.5 Illustrate the expression output.

2.2.6 Define the traditional set of operators for relational tables:

a) Union
b) Intersection
c) Difference

2.2.7 Define union compatibility.

2.2.8 Illustrate the union, intersect, and difference set operators based on tables given.

WHAT is

RELATION ALGEBRA

❑ A procedural query language used to query the database tables to access data in different ways.

Define the ways in which relations (tables) can be operated to manipulate their data and used as the basis of SQL for relational databases and illustrates the basic operations required of any DML.

❑ The formal description of **how a relational database operates**.

❑ Relational Algebra is a relation-at-a-time (or set) language where all tuples are controlled in one statement without the use of a loop. It works on the whole table at once, so we do not have to use loops to iterate over all the rows (tuples) of data one by one.

The operators take one or more relations as inputs and give a new relation as a result.

RELATIONAL VS SQL

RELATION ALGEBRA

❑The mathematics which underpin **SQL operations**

❑Operators in relational algebra are not necessarily the same as SQL operators, even if they have the same name.

For example, the SELECT statement exists in SQL, and also exists in relational algebra. These two uses of SELECT are <u>not the same</u>.

Relation algebra

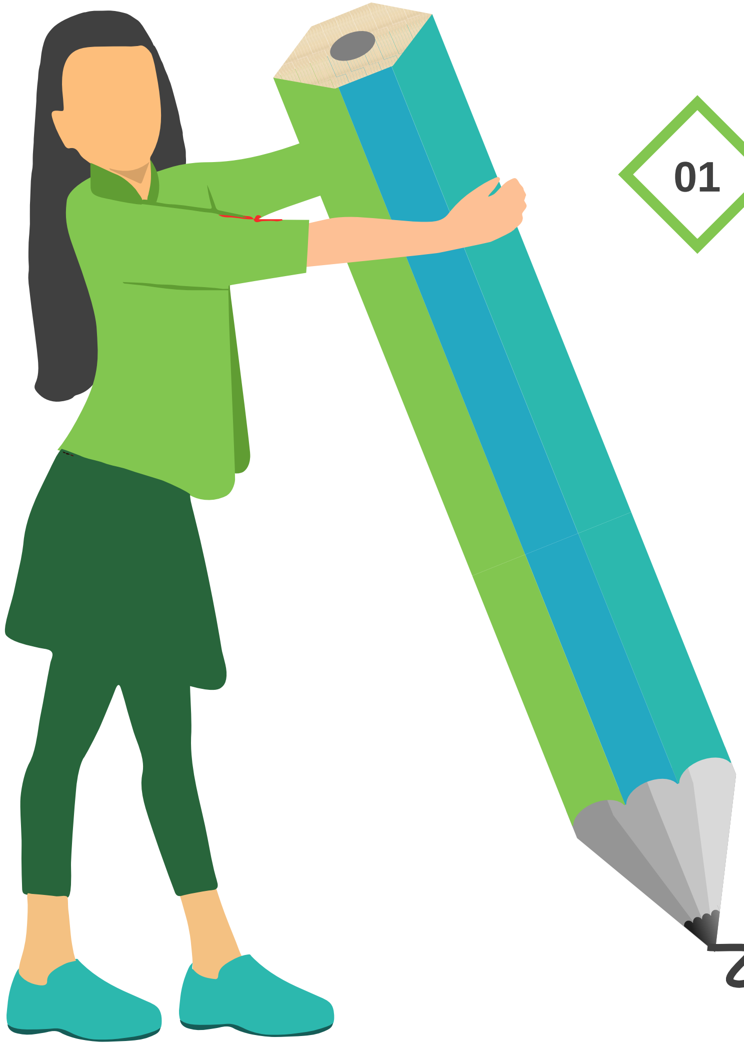
σsubject = “database” (teacher)

SQL statement

```
SELECT *
FROM teacher
WHERE subject = “database”
```

OPERATORS

RELATION ALGEBRA



01

Fundamental Operators

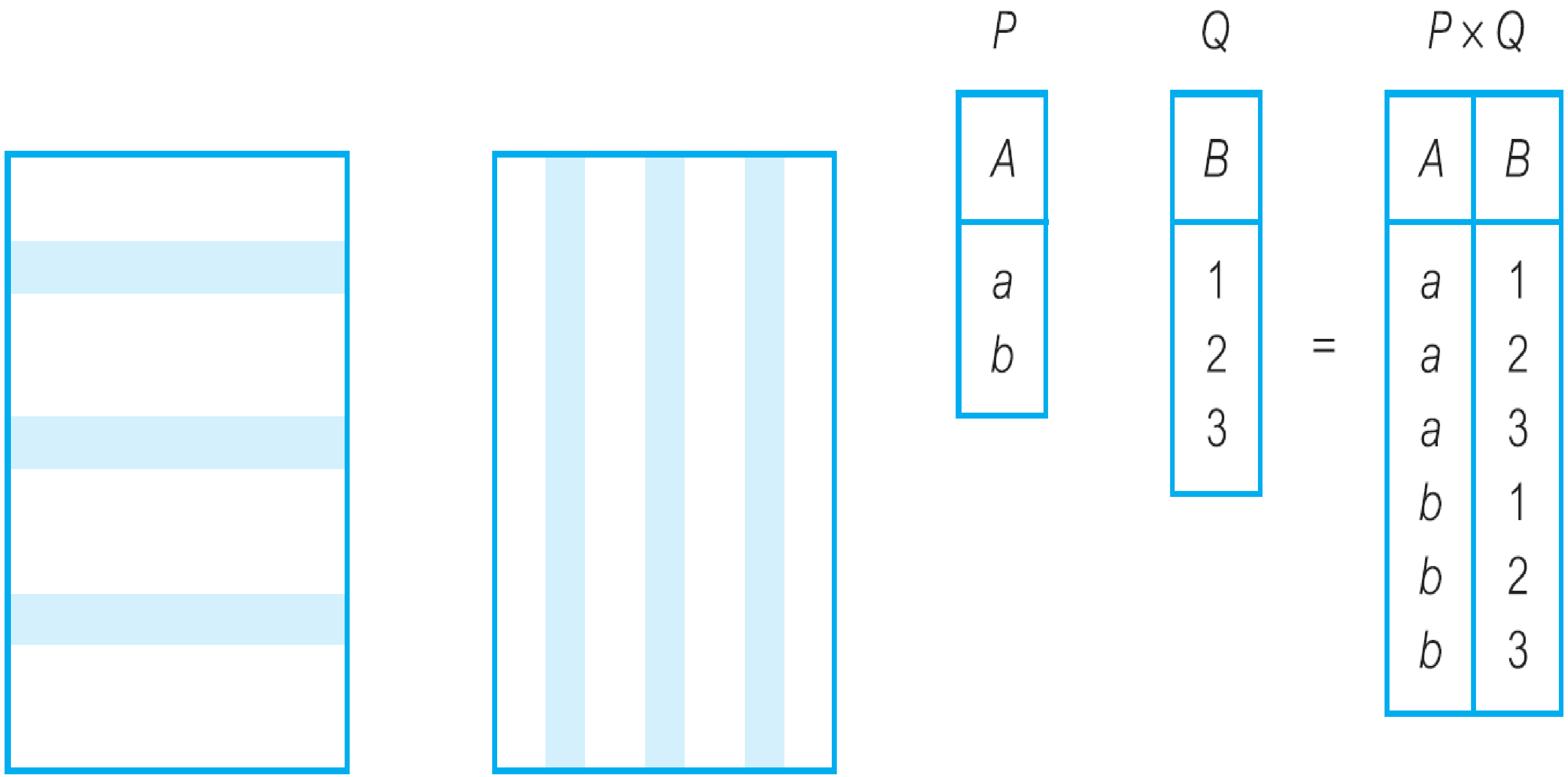
- Select (Restrict)
- Project
- Join (Outer, Inner)
- Cross Product

02

Traditional Set

- Union
- Intersect
- Difference

ILLUSTRATION ROA FUNCTION



(a) Selection

(b) Projection

P

A
a
b

Q

B
1
2
3

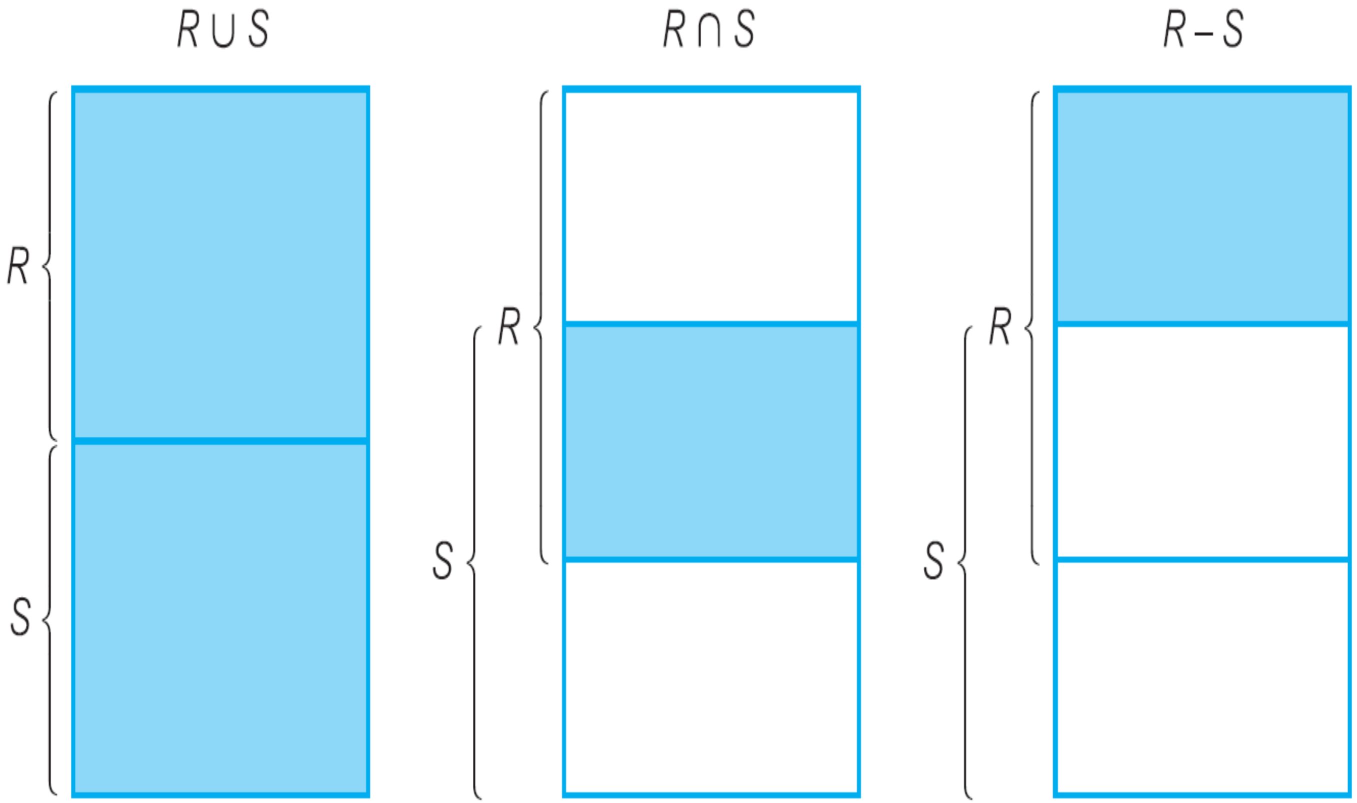
=

P × *Q*

A	*B*
a	1
a	2
a	3
b	1
b	2
b	3

(c) Cartesian product

ILLUSTRATION ROA FUNCTION



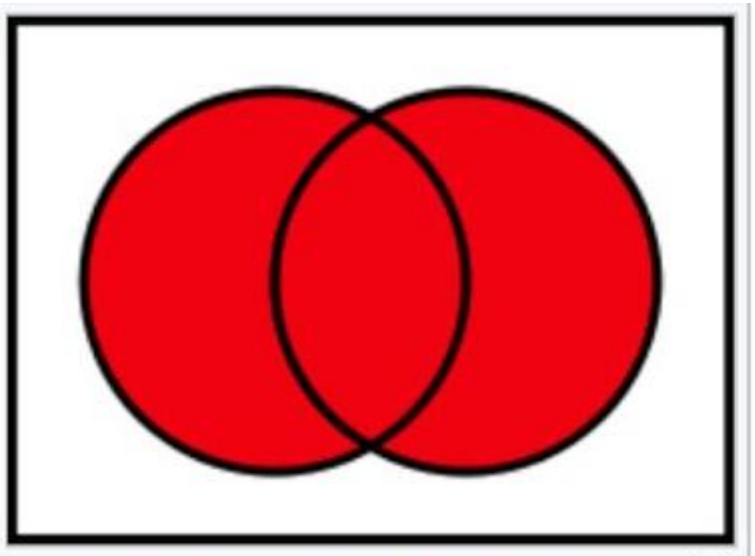
(d) Union

(e) Intersection

(f) Set difference

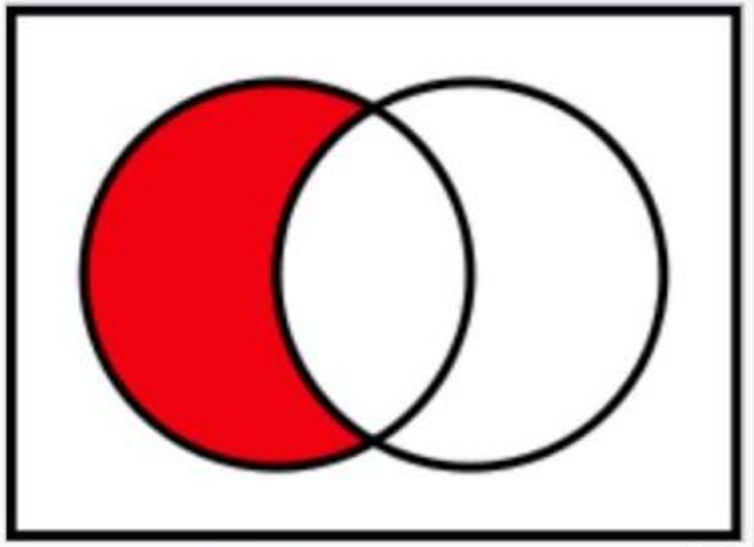
ILLUSTRATION ROA FUNCTION

UNION



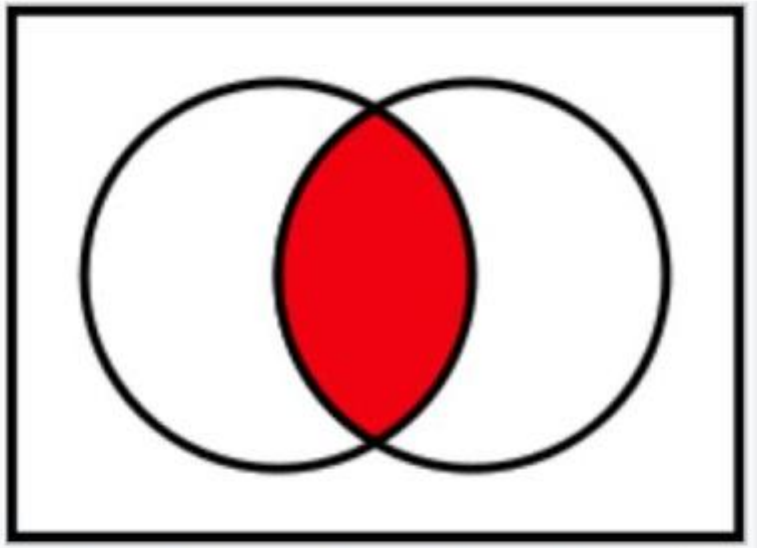
The **union** of *A* and *B*, denoted *A* ∪ *B*

DIFFERENCE



The **difference** of *B* in *A*

INTERSECT

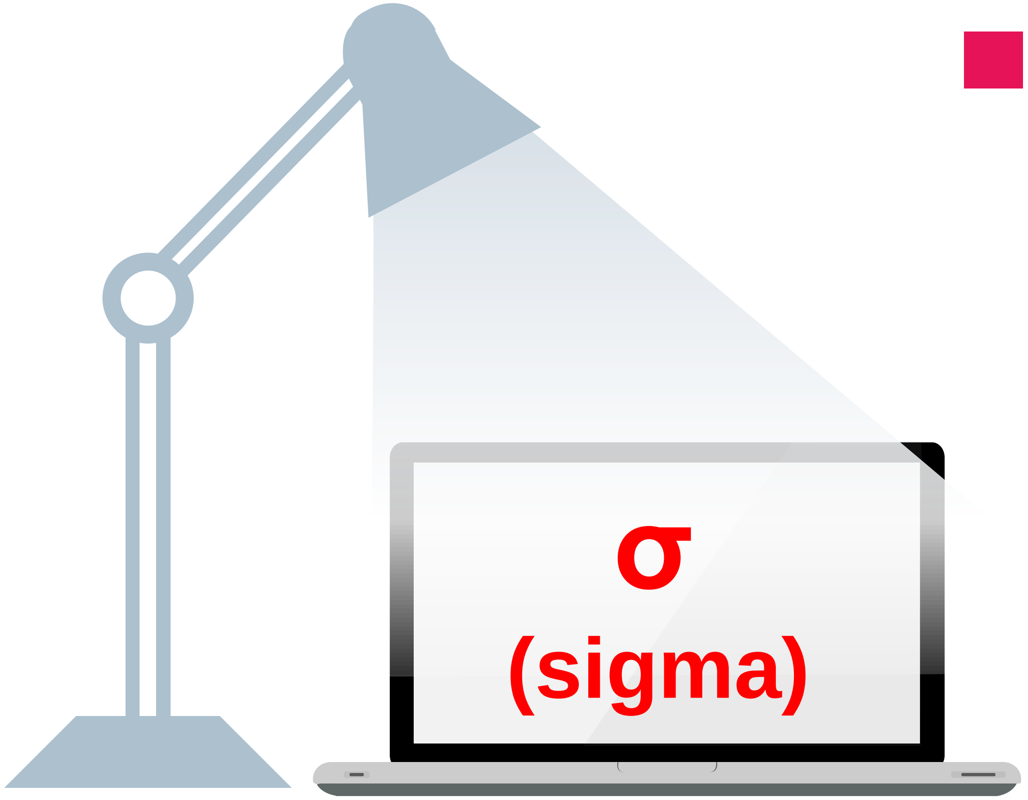


The **intersection** of *A* and *B*, denoted *A* ∩ *B*.

SELECT OPERATION

- This is used to fetch rows(tuples) from table(relation) which satisfies a given condition.

- $\sigma_{\text{condition}}$**(R)**: This selection operation functions on a single relation R and describes a relation which contains only those tuples of R that satisfy the specified condition (predicate).



Symbol :

σ (sigma) used to denote the SELECT operator

(R) = relation name/table name

Syntax :

$\sigma_{\text{<selection condition >}}(R)$

$\sigma_{\text{predicate}}(R)$

SELECT OPERATION

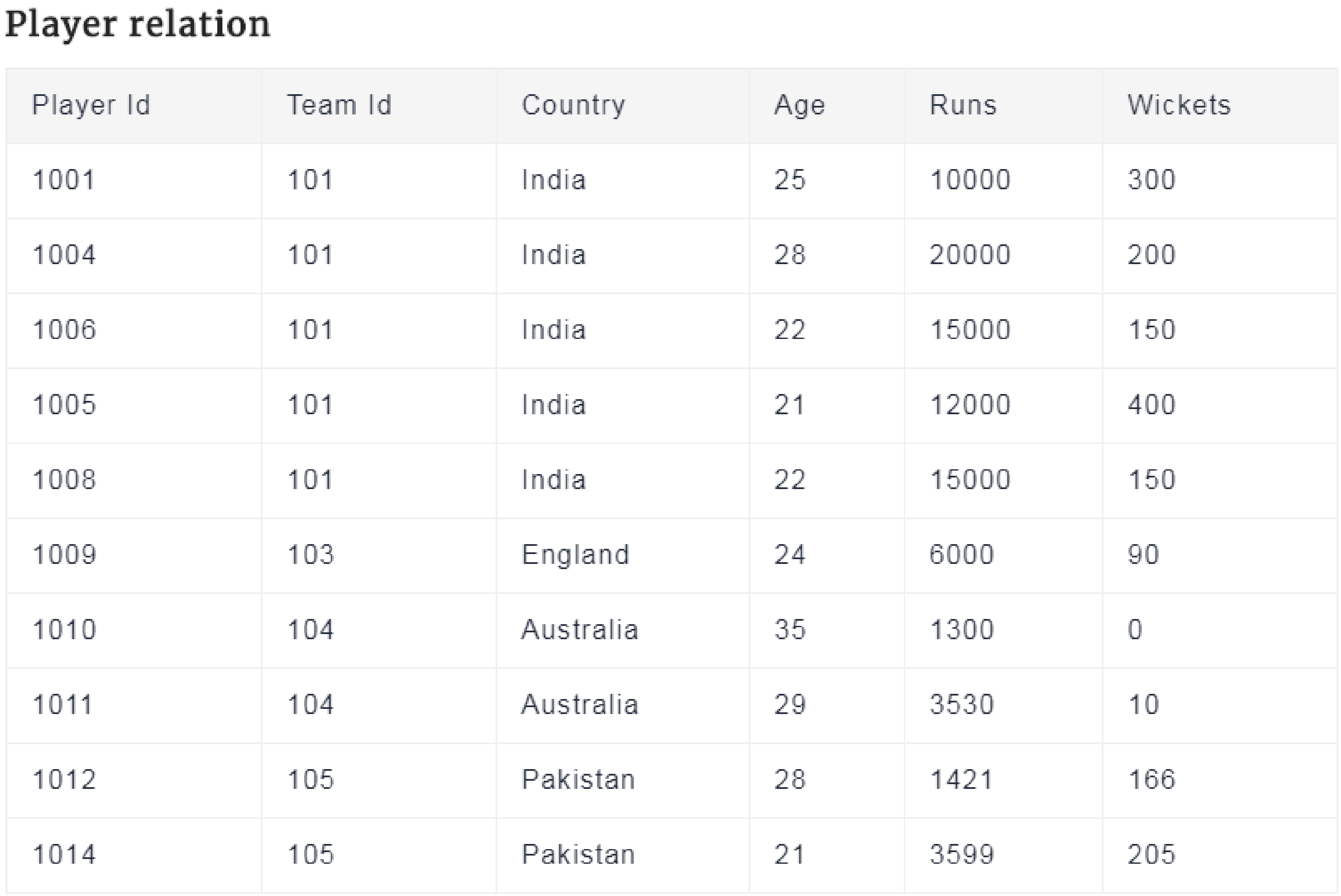
Normally combine with ***Comparison Operator***

=, <, >, <=, >=, =

May combine with ***Boolean Operator***

- **AND**
- **OR**
- **NOT**

SELECT OPERATION



Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

Question A. Find all tuples from player relation for which country is India.

Question B. Select all the tuples for which runs are greater than or equal to 15000.

Question C. Select all the players whose runs are greater than or equal to 6000 and age is less than 25.

SELECT OPERATION

$\sigma_{\text{subject = "database"}}\text{(BookStore)}$

Output − Selects tuples from book store where subject is database.

$\sigma_{\text{subject = "database" AND price = 125}}\text{(BookStore)}$

Output − Selects tuples from book store where subject is database and book price is RM125.

$\sigma_{\text{subject = “IT Security" AND price = 250 OR year > 2012}}\text{(BookStore)}$

Output − Selects tuples from book store where subject is IT Security and price is RM250 or those books published after 2012.

EXERCISE 1:
SELECT OPERATION

Table : EMPLOYEE

empNum	empFName	empLName	age	gender	empAdd	position	salary	deptNo

1 To select the **EMPLOYEE** tuples whose department number is 4

2 Select salary is greater than RM30,000

EXERCISE 1: SELECT OPERATION

Table : EMPLOYEE

empNum	empFName	empLName	age	gender	empAdd	position	salary	deptNo

1 To select the **EMPLOYEE** tuples whose department number is 4

σ deptNo = 4 (EMPLOYEE)

2 Select salary is greater than $30,000

σ salary > 30000 (EMPLOYEE)

EXERCISE 1:
SELECT OPERATION

Table : EMPLOYEE

empNum	empFName	empLName	age	gender	empAdd	position	salary	deptNo

3 Select the tuples for all employees who either work in department 4 and make over RM2500 per year, or work in department 5 and make over RM3000.

$$\sigma_{\text{(deptNo=4 AND Salary>2500) OR (deptNo=5 AND Salary>3000)}}\text{(EMPLOYEE)}$$

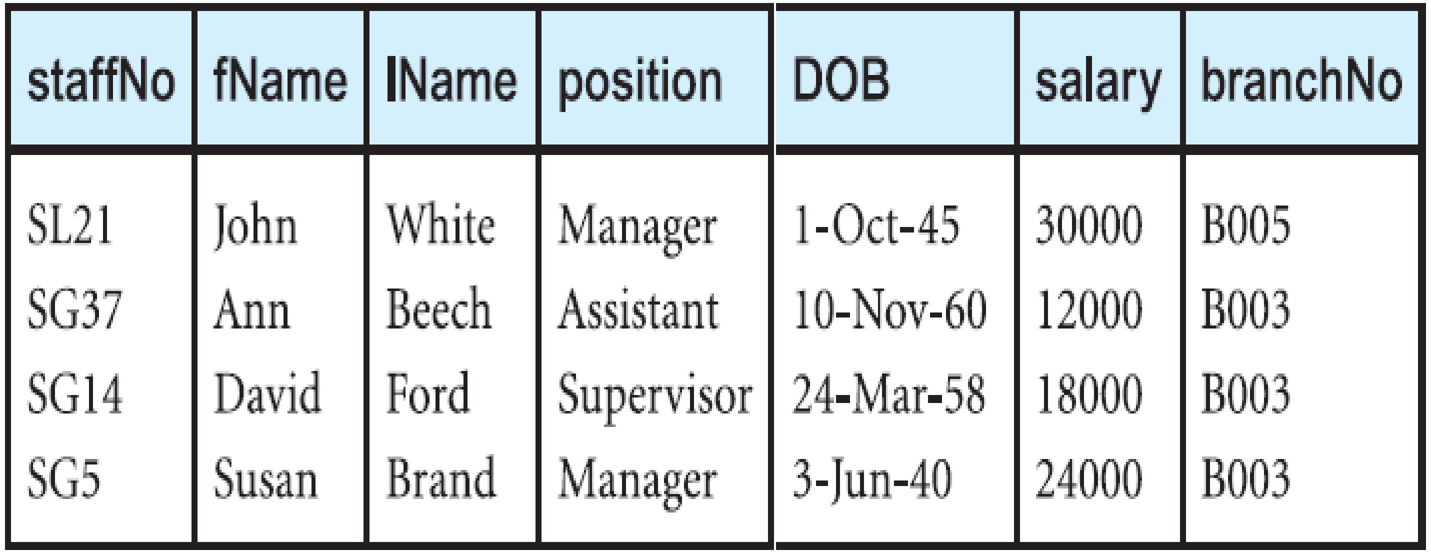
EXERCISE 1: SELECT OPERATION

$\text{EMP\_RESULT} \leftarrow \sigma_{(\text{deptNo=4 AND Salary>2500}) \text{ OR } (\text{deptNo AND Salary>3000})} \text{(EMPLOYEE)}$

Table : EMP_RESULT

empNum	empFName	empLName	age	gender	empAdd	position	salary	deptNo
S001	Mary	Alice	25	F	Skudai	Admin	4000	5
S009	John	Navine	31	M	Kulai	Manager	4300	4
S022	Catrine	Carol	22	F	Kulai	Admin	3800	5

List all staff with a salary greater than RM10000



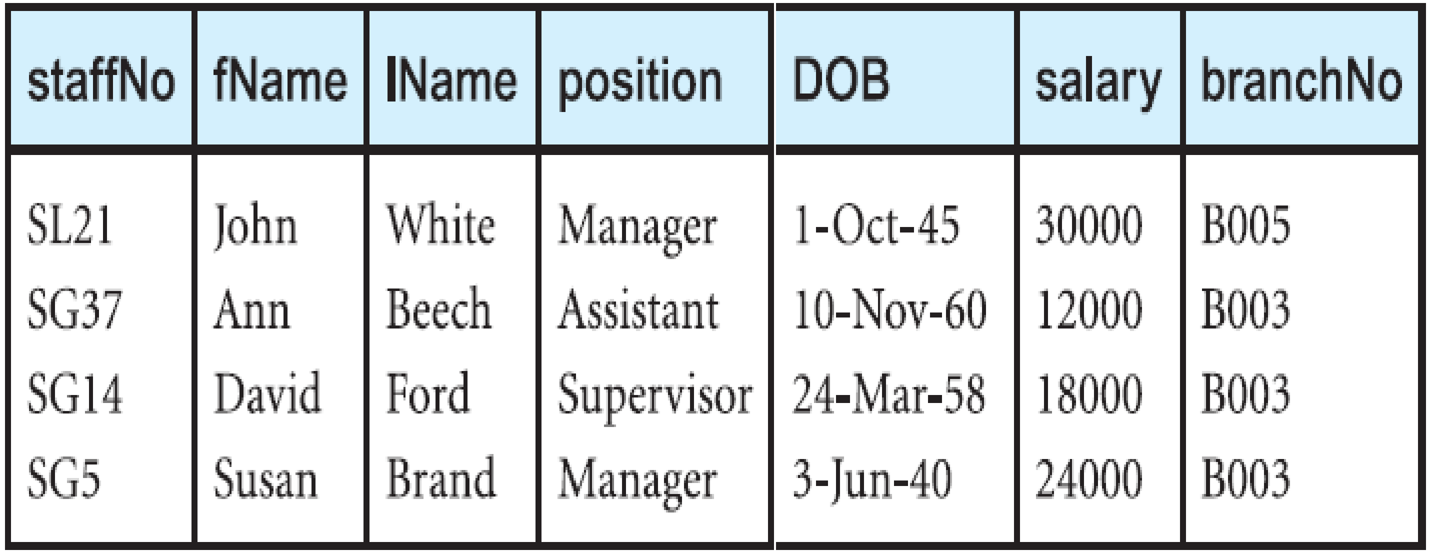
staffNo	fName	lName	position	DOB	salary	branchNo
SL21	John	White	Manager	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	24-Mar-58	18000	B003
SG5	Susan	Brand	Manager	3-Jun-40	24000	B003

Table : Staff

ACTIVITY

Select the record of employee position is Manager and he works at Branch Number B005.

ACTIVITY



staffNo	fName	lName	position	DOB	salary	branchNo
SL21	John	White	Manager	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	24-Mar-58	18000	B003
SG5	Susan	Brand	Manager	3-Jun-40	24000	B003

Table : Staff

Projection Operation (Л)

- Project operation ***selects certain columns*** from the table and ***discards the other columns***.

> *Project operation is used to project only a certain set of attributes of a relation. It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns*

- The PROJECT creates a vertical partitioning – works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.

Symbol :

∏(pi)

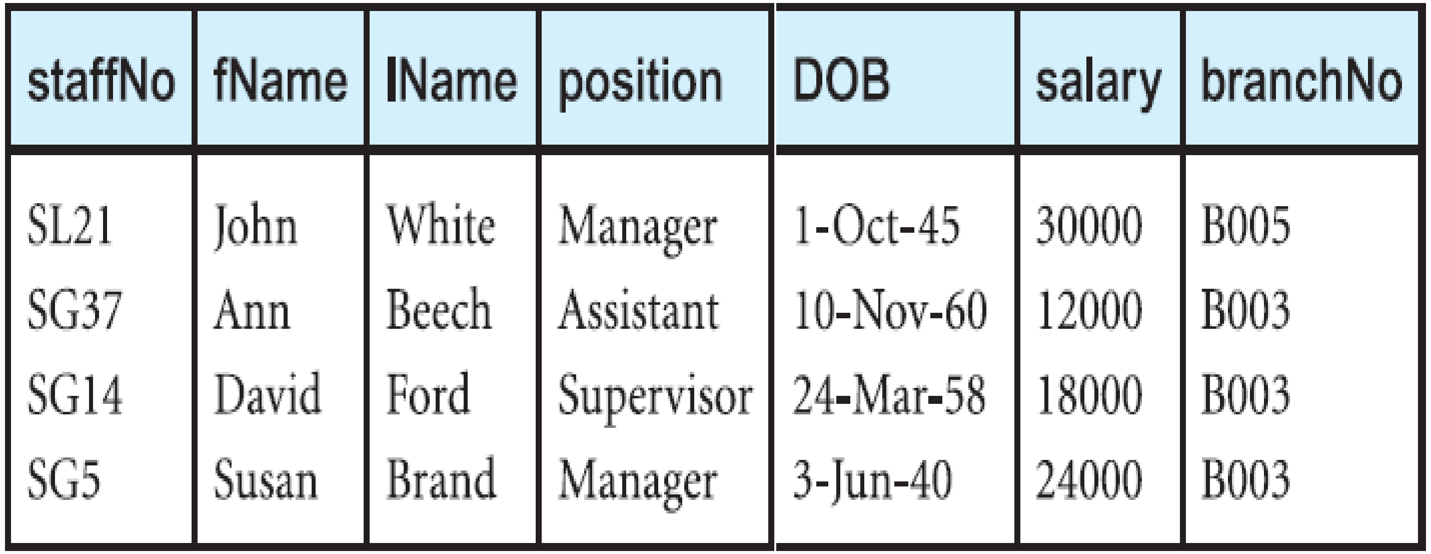
Syntax :

∏ <attribute list> (R)

Exercise 1:
Projection Operation (Л)

Produce a list of salaries for all staff, showing only the staffNo, fName, lName, and salary details.

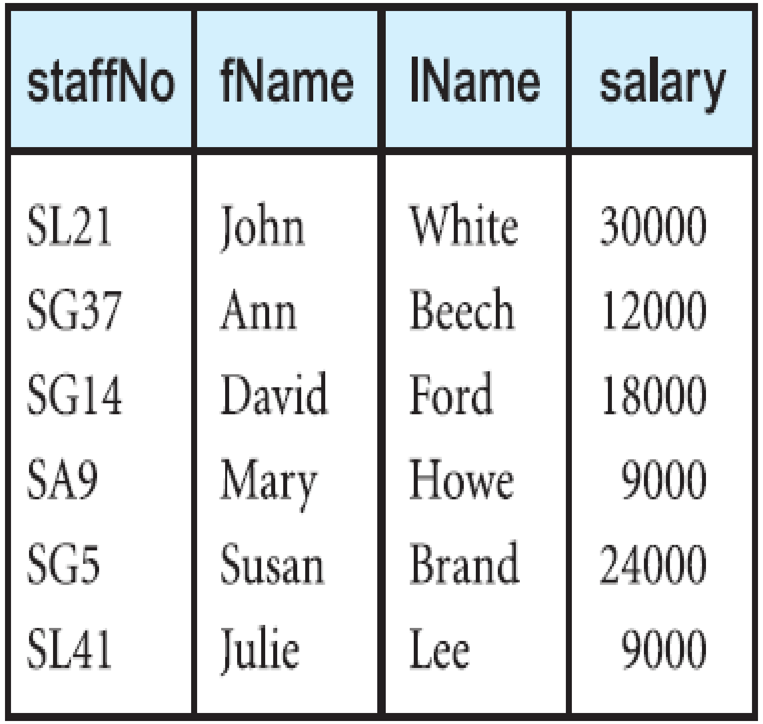
Table : Staff



staffNo	fName	lName	position	DOB	salary	branchNo
SL21	John	White	Manager	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	24-Mar-58	18000	B003
SG5	Susan	Brand	Manager	3-Jun-40	24000	B003

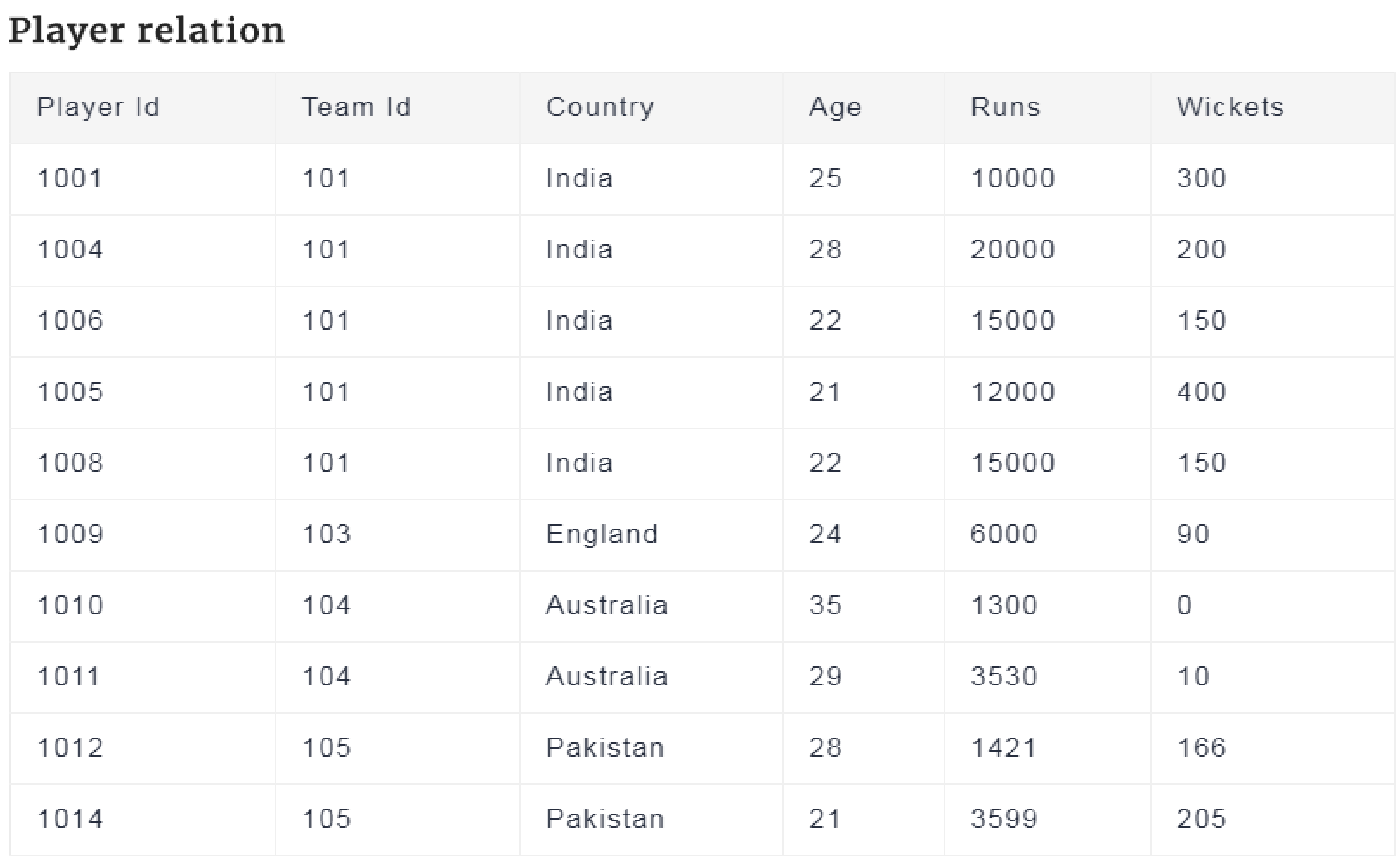
Answer Exercise 1:
Projection Operation (Л)

R1← Л staffNo, fName, lName, salary **(STAFF)**



staffNo	fName	lName	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

PROJECT OPERATION



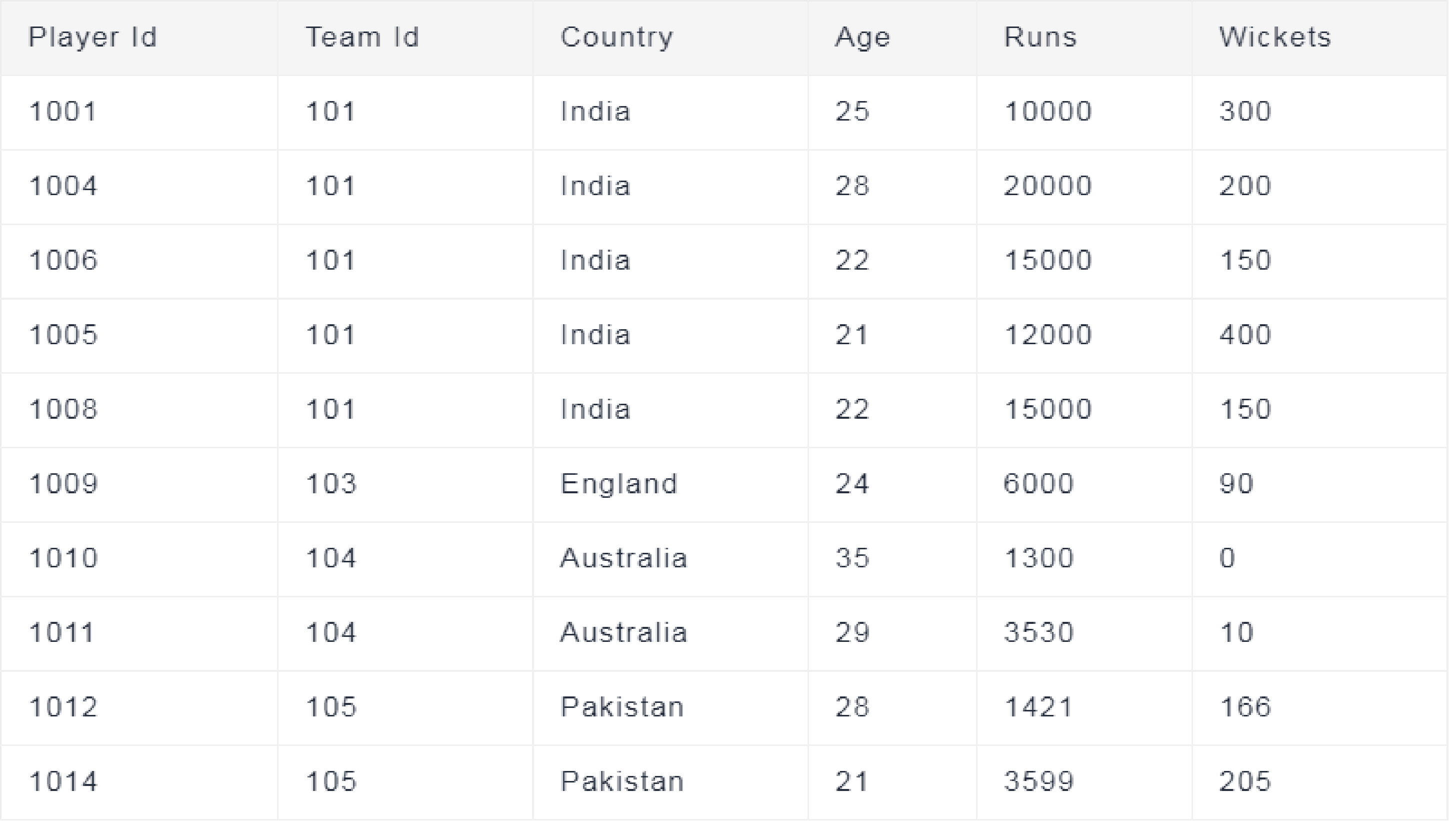
Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

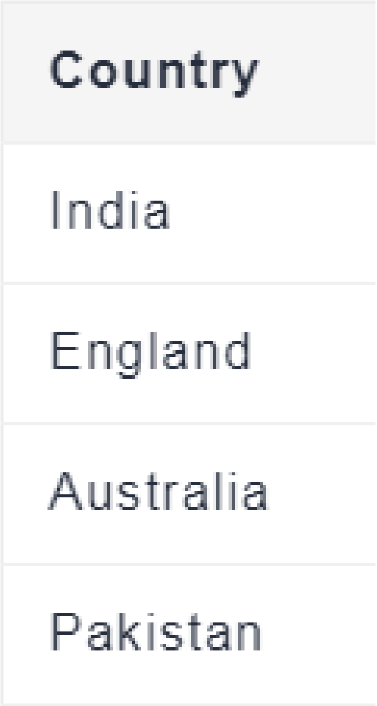
Question A. List all the countries in Player relation.

PROJECT OPERATION

Player relation



Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

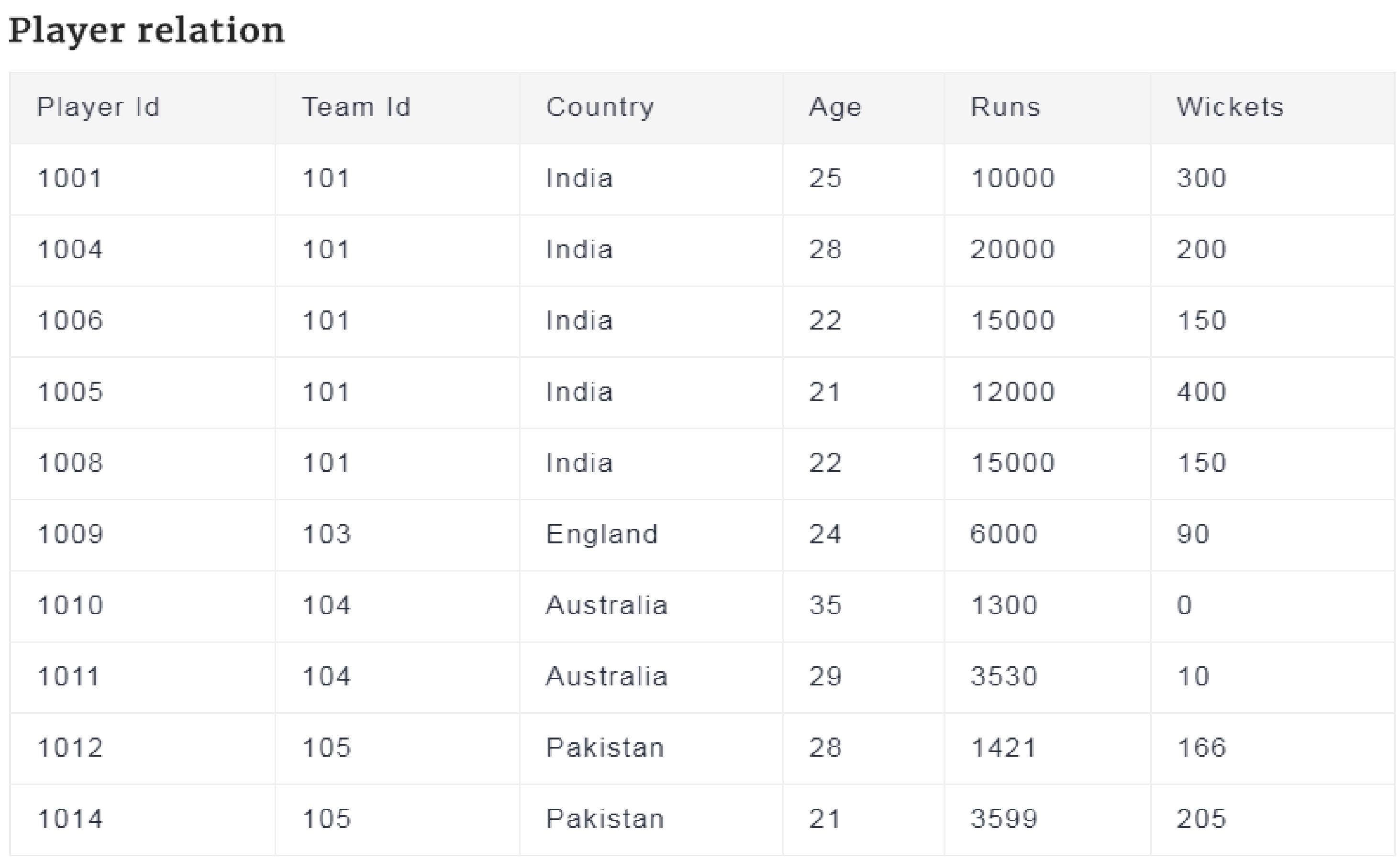


Country
India
England
Australia
Pakistan

Question A. List all the countries in Player relation.

$\Pi_{\text{Country}}(\text{PLAYER})$

PROJECT OPERATION

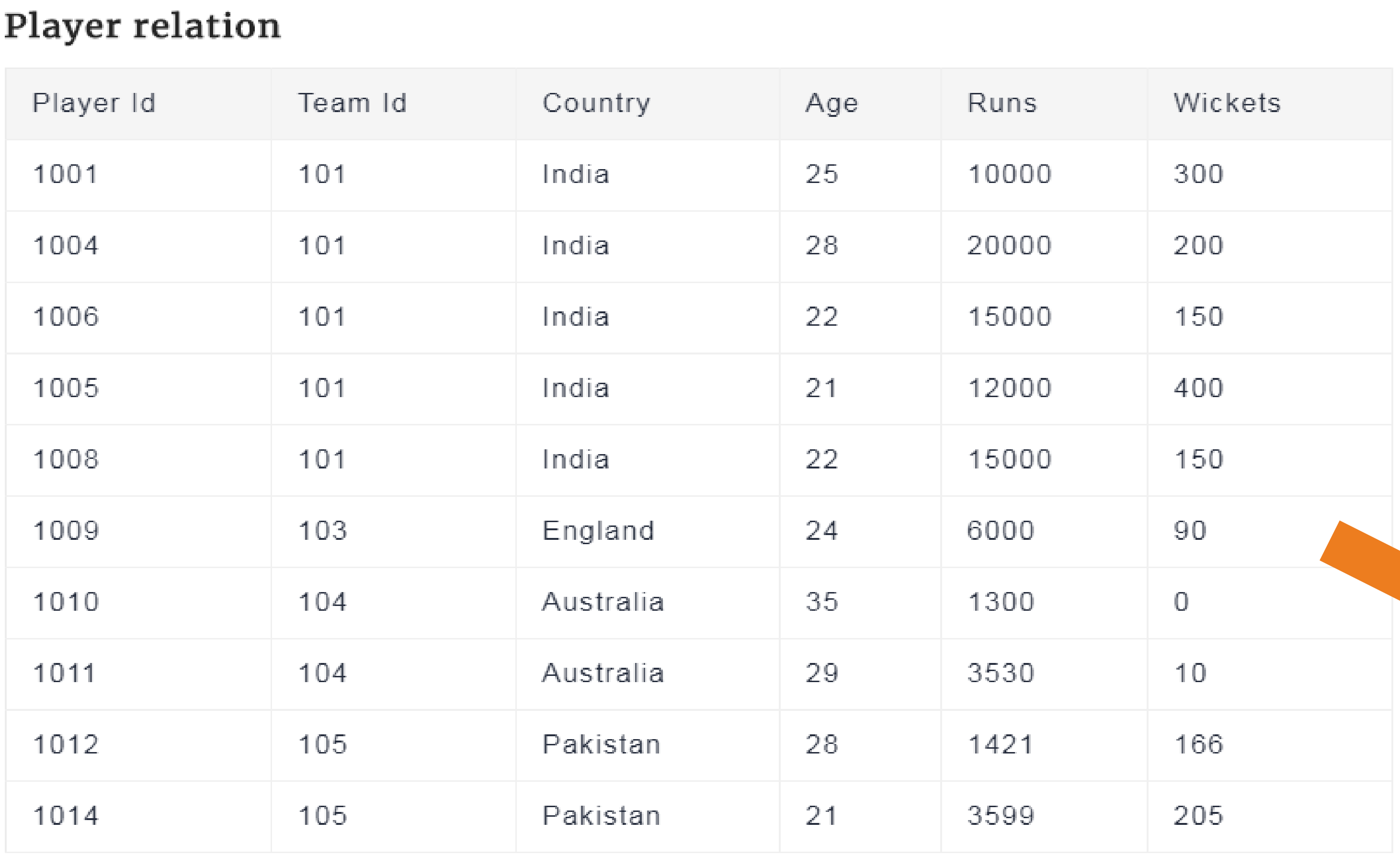


Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

Question B. List all the team ids and countries in Player Relation

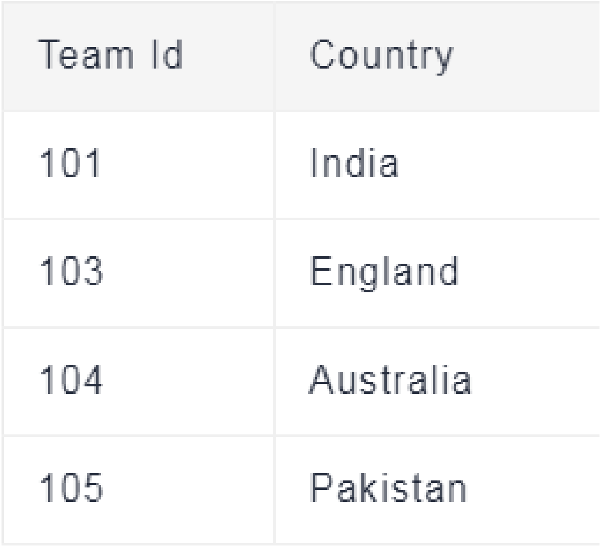
PROJECT OPERATION



Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

Π Team_Id, Country (PLAYER)



Team Id	Country
101	India
103	England
104	Australia
105	Pakistan

Question B. List all the team ids and countries in Player Relation

Projection (Л) & SELECT (σ)

We can combine **project with select operators** when we want to select certain *column* that specify specific *conditions.*

condition

Example Q:

*Select **student name** and address for student who **live in Jitra**.*

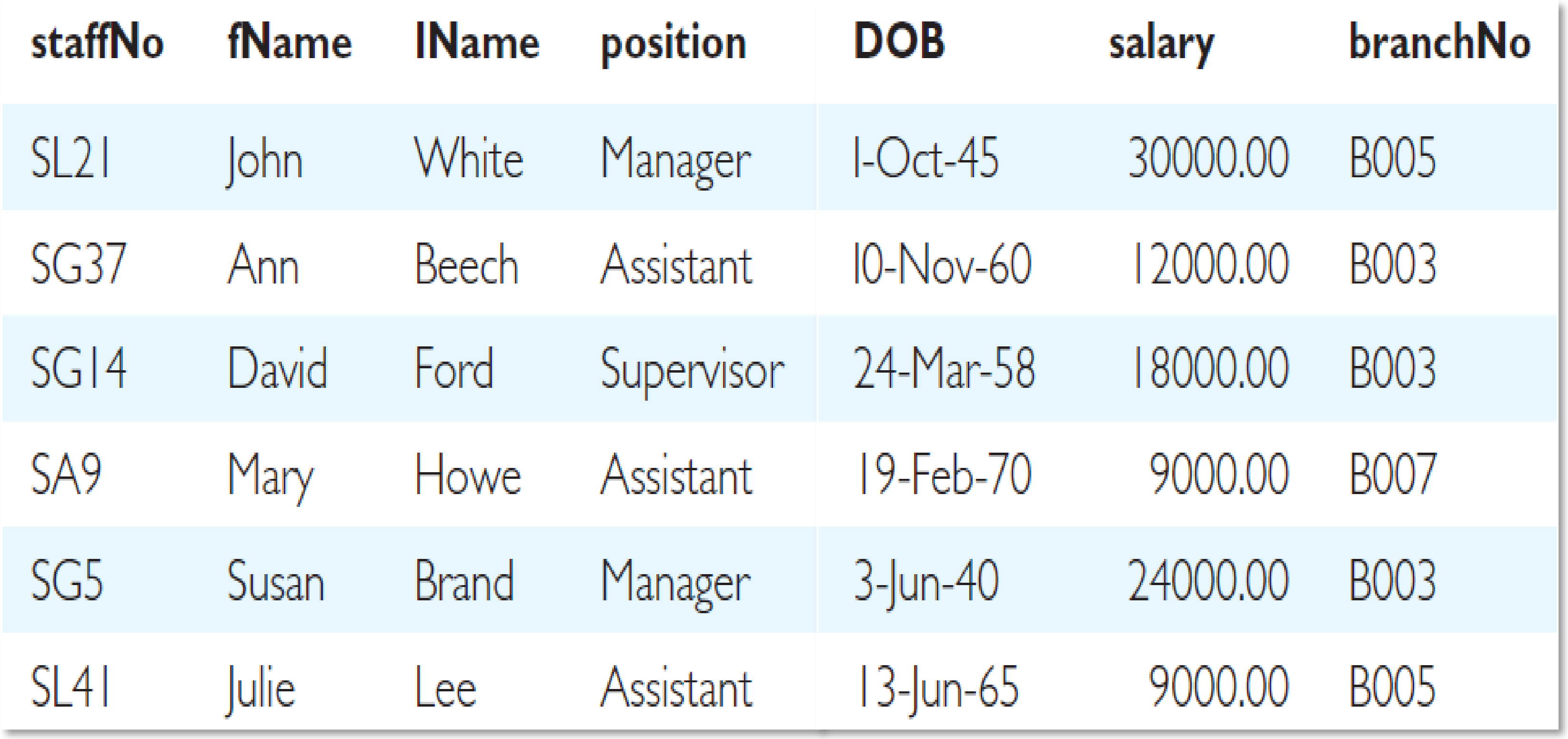
Attribute name

1st step : **SELECT** from temporary table who live in Jitra

2nd step : Select : List name and address → temporary table

Example 1 : Combination Л & σ

Retrieve the first name, last name, salary and branch number of all employees who work in branch B003.



staffNo	fName	lName	position	DOB	salary	branchNo
SL21	John	White	Manager	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	13-Jun-65	9000.00	B005

The sequence is :

Retrieve the first name, last name, salary and branch number of all employees who work in branch B003.

step 1

$R1 \leftarrow \sigma_{branchNo=B003}$ **(STAFF)**

step 2

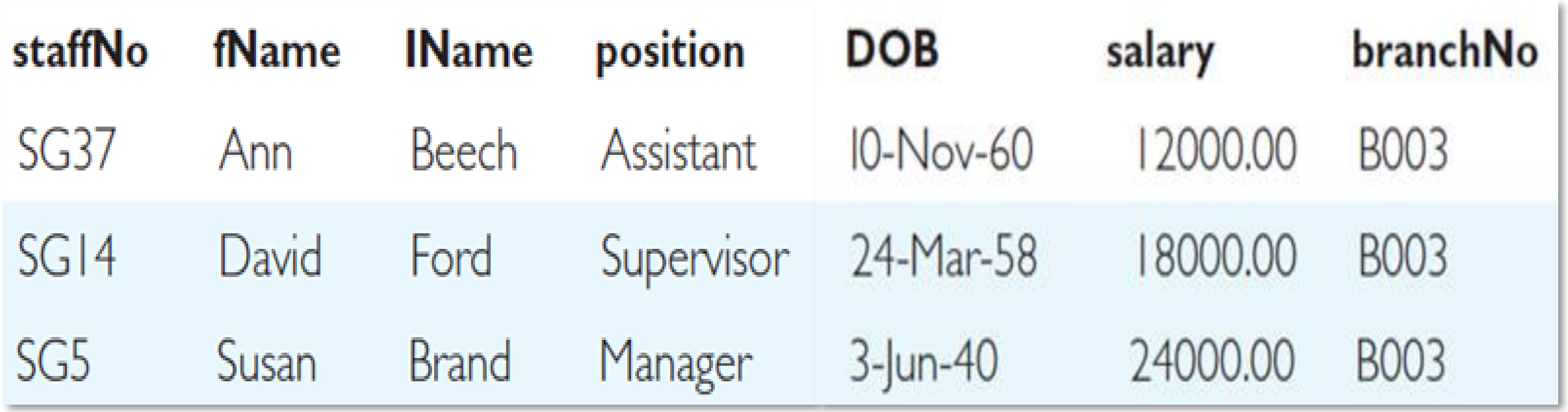
$R2 \leftarrow Л_{Fname,Lname,Salary,branchNo}$ **(R1)**

The sequence is :

Retrieve the first name, last name, salary and branch number of all employees who work in branch B003.

$$R1 \leftarrow \sigma_{branchNo=B003}(STAFF)$$

step 1

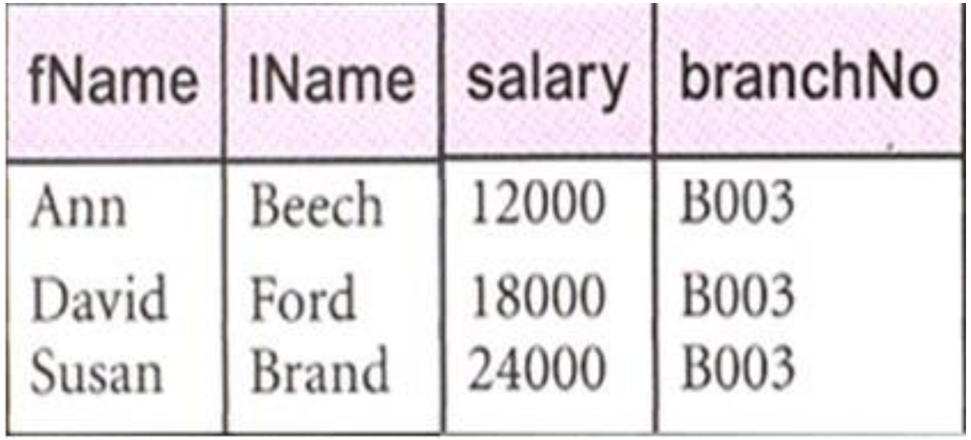


staffNo	fName	lName	position	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	3-Jun-40	24000.00	B003

The sequence is :

Retrieve the first name, last name, salary and branch number of all employees who work in branch B003.

R2← Л fName, lName, salary, branchNo **(R1)**



fName	lName	salary	branchNo
Ann	Beech	12000	B003
David	Ford	18000	B003
Susan	Brand	24000	B003

Example 1 : Combination Л & σ

R1← σ branchNo=B003 (STAFF)
R2 ← Л fName, lName, salary, branchNo (R1)

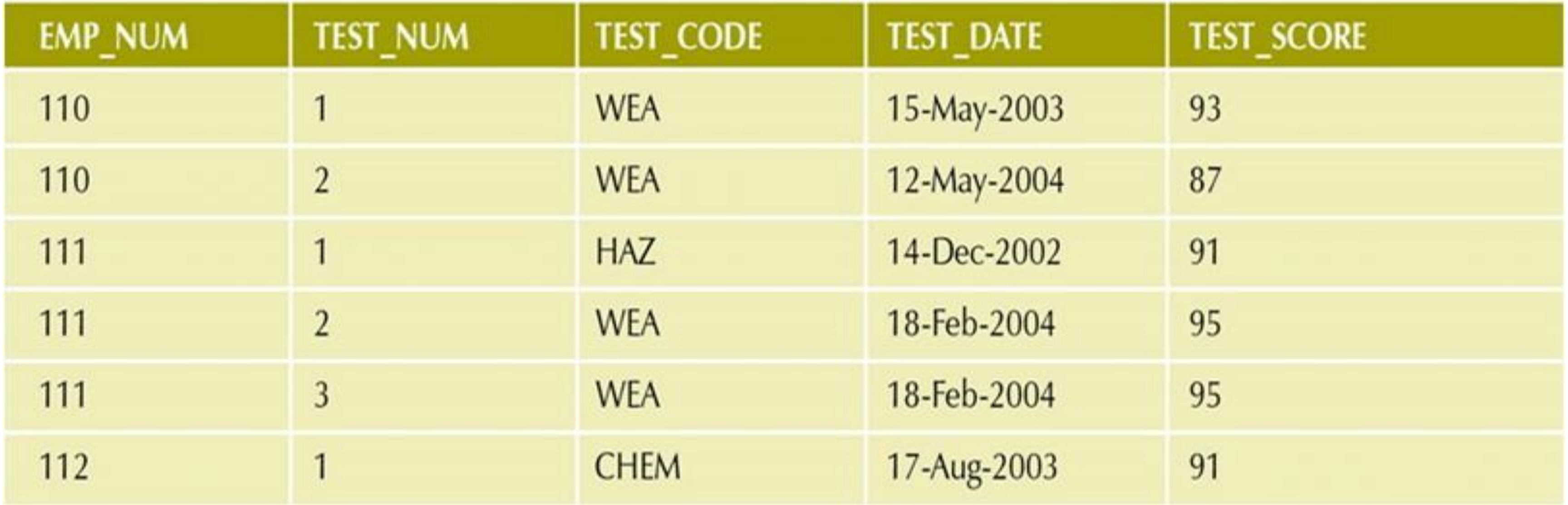
You can also ***combine*** the operation as below:

ЛfName, lName, salary, branchNo
((σbranchNo=B003(STAFF))

Example 2 : Combination Л & σ

Using relational algebra

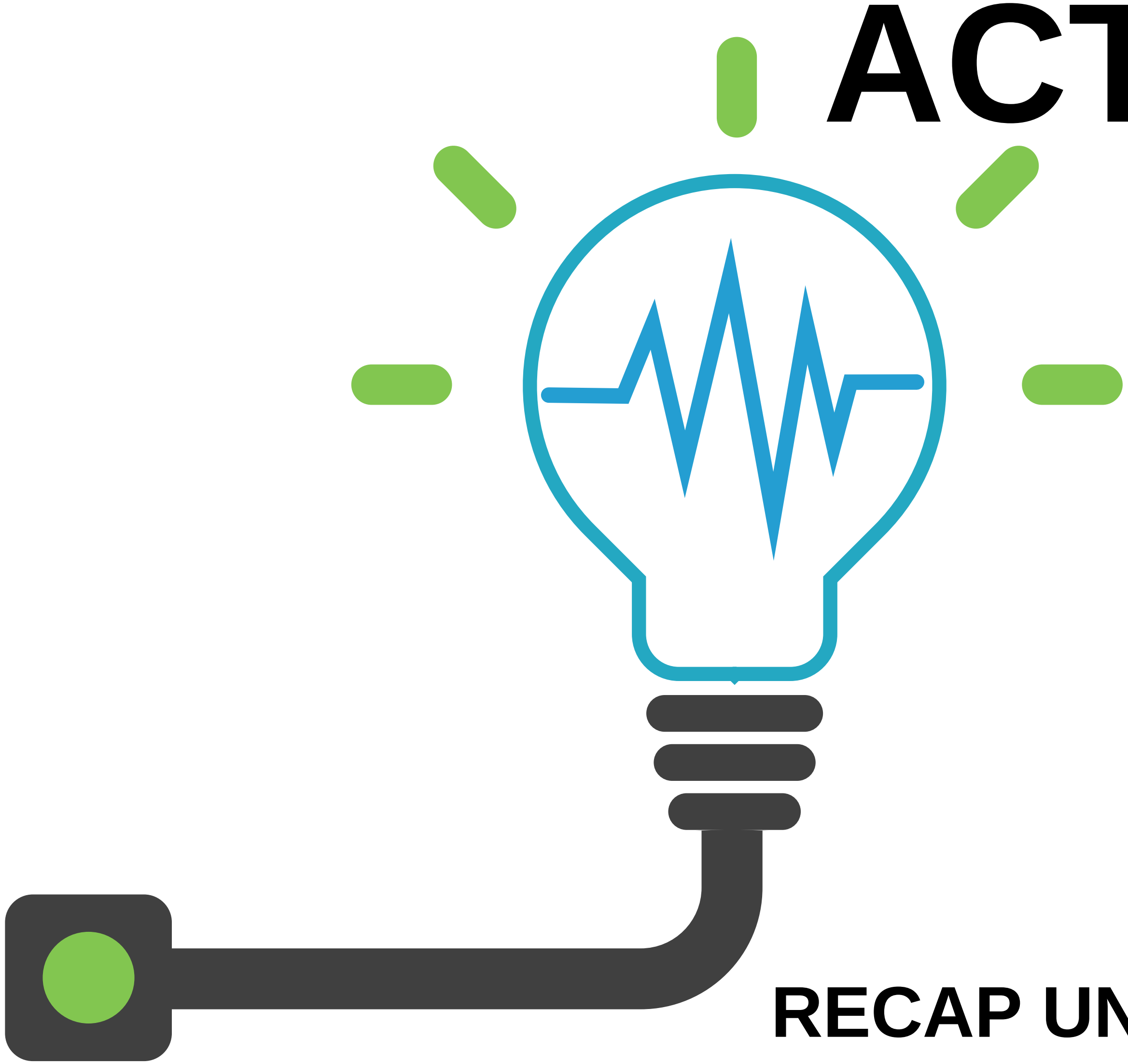
1. Select Employee whose test score is greater than 91.
2. List all the employee number, test code and score for test record.
3. List all the test number, test code for employee number 110.



EMP_NUM	TEST_NUM	TEST_CODE	TEST_DATE	TEST_SCORE
110	1	WEA	15-May-2003	93
110	2	WEA	12-May-2004	87
111	1	HAZ	14-Dec-2002	91
111	2	WEA	18-Feb-2004	95
111	3	WEA	18-Feb-2004	95
112	1	CHEM	17-Aug-2003	91

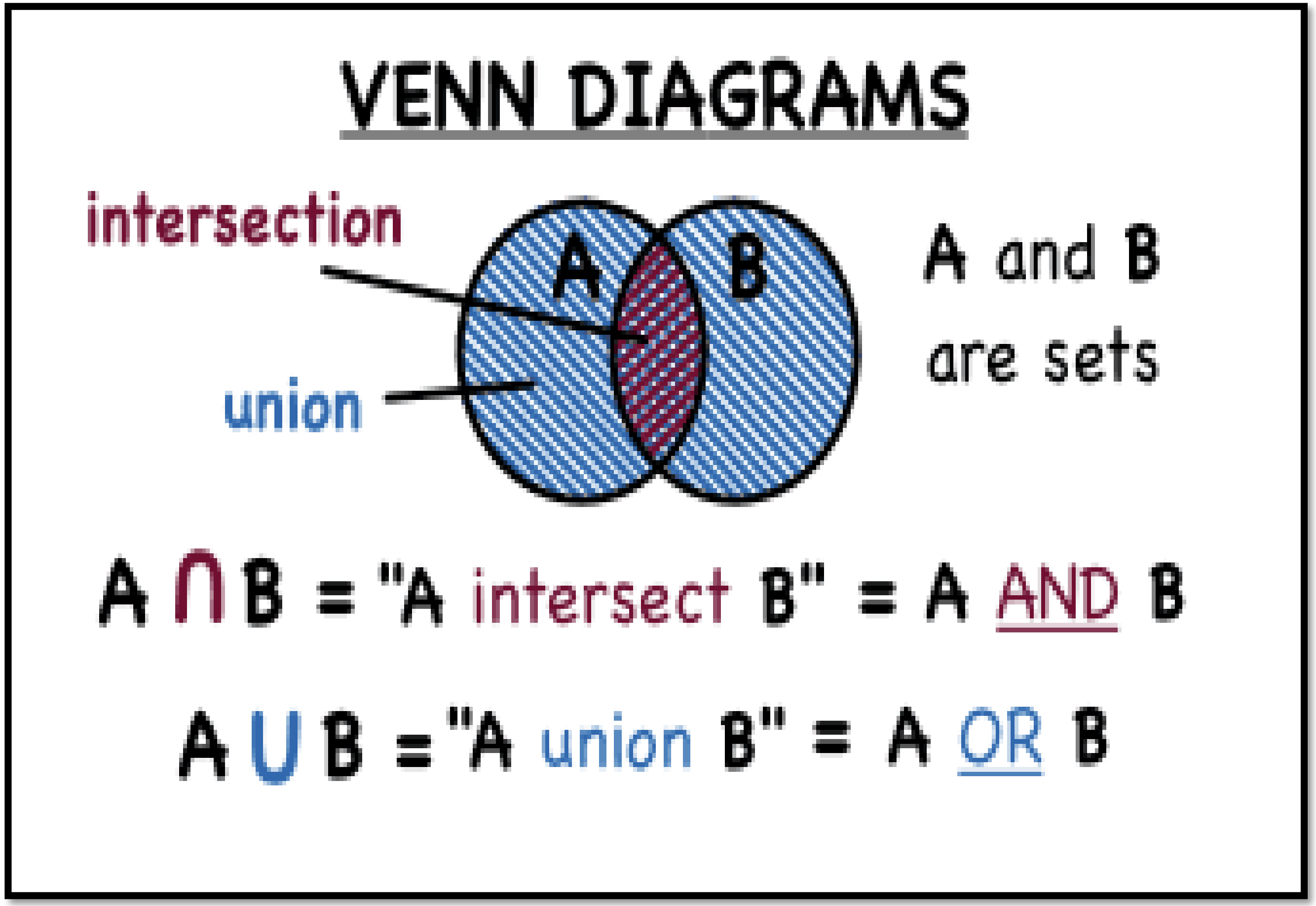
Table : Employee

WORKBOOK ACTIVITY



RECAP UNDERSTANDING

TRADITIONAL SET OF OPERATOR



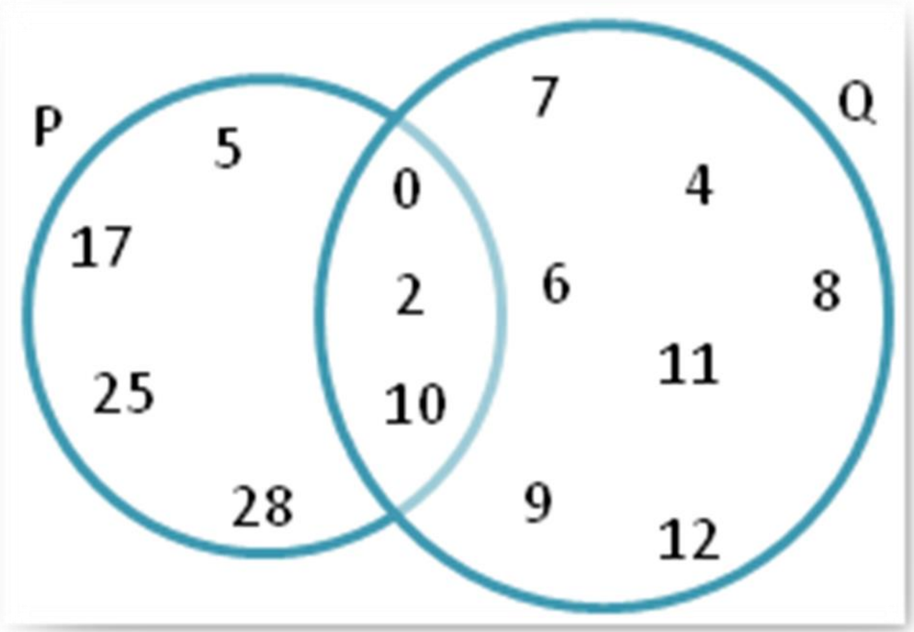
VENN DIAGRAMS

A and B are sets

$A \cap B$ = "A intersect B" = A AND B

$A \cup B$ = "A union B" = A OR B

ACTIVITY

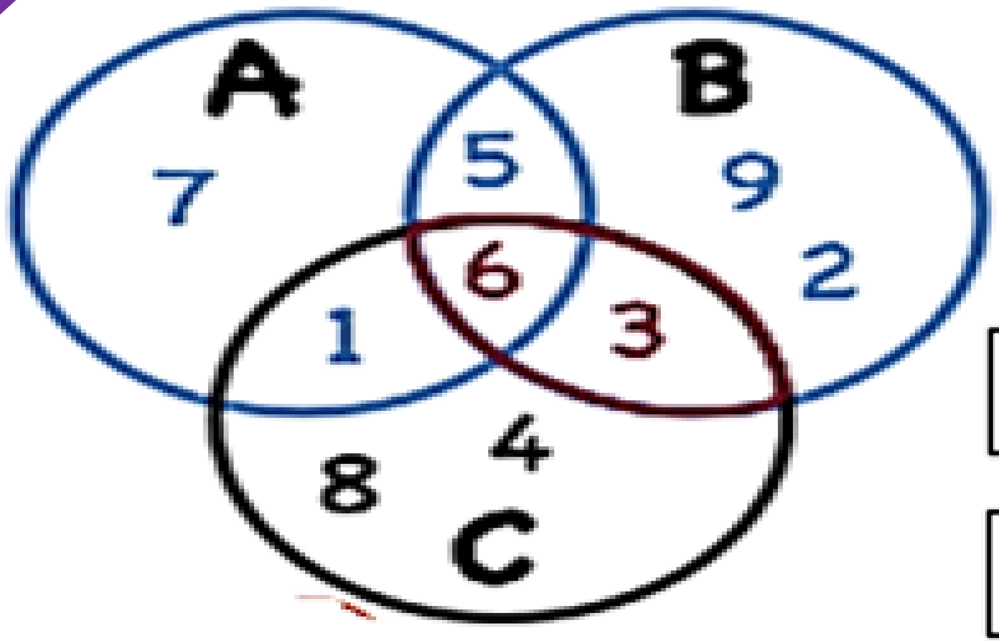


$P \cap Q$

$P - Q$

$Q \cup P$

ACTIVITY



$A \cup B$	$A - B$
$A \cap C$	$B - C$
$B \cap C$	$A - C$

ACTIVITY

$A = \{$ ⬟ , ◆ , 🟩 , ▯ $\}$

$B = \{$ ★ , 🟩 , ▲ , ⬟ $\}$

$\boxed{A \cap B}$ $\boxed{A - B}$

$\boxed{B \cup A}$

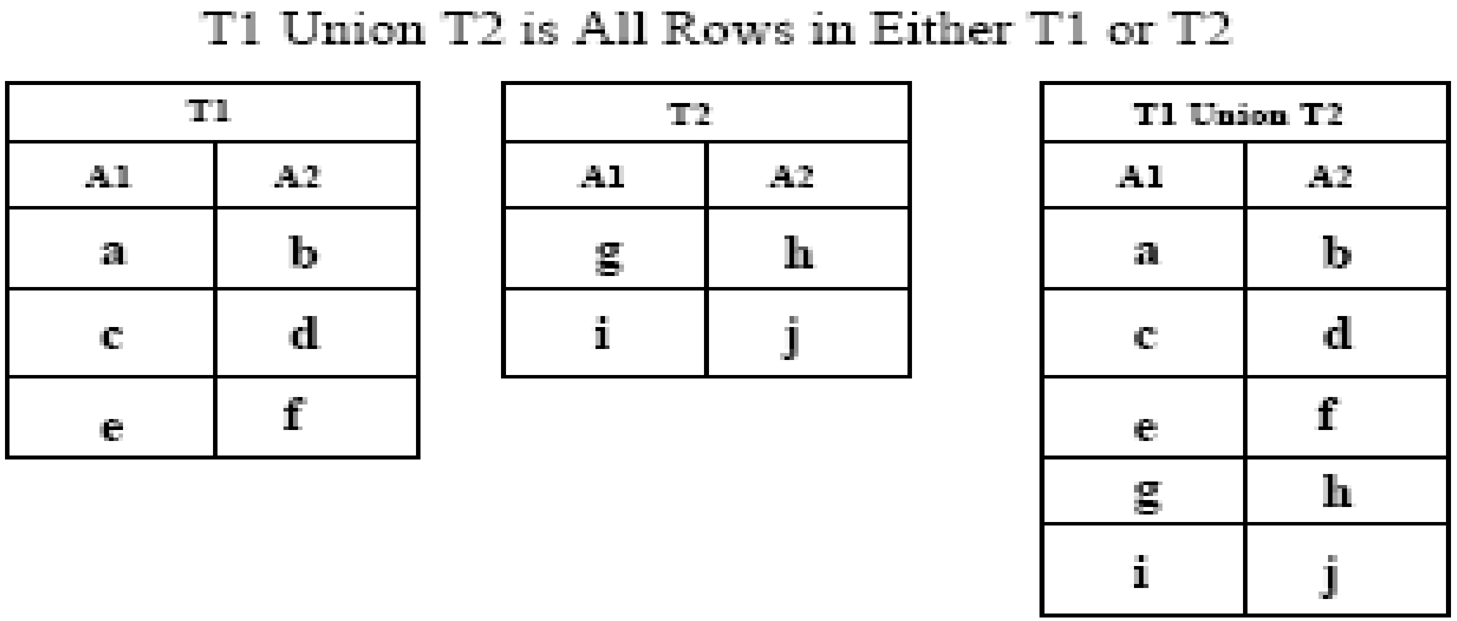
UNION ∪

- UNION is used to combine the results of two or more SELECT statements
- Relation that includes all tuples that are either in R or in S or in both R and S.
- Table must have the **same number of attribute**.
- Duplicate tuples are eliminated.

Symbol : ∪

Syntax : **Л <attribute list> (R) ∪ Л <attribute list> (R)**

Union Operator (∪)

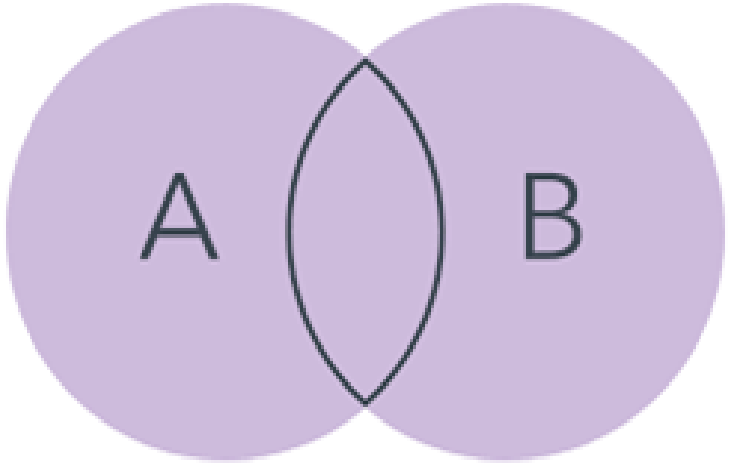


T1 Union T2 is All Rows in Either T1 or T2

T1	
A1	A2
a	b
c	d
e	f

T2	
A1	A2
g	h
i	j

T1 Union T2	
A1	A2
a	b
c	d
e	f
g	h
i	j



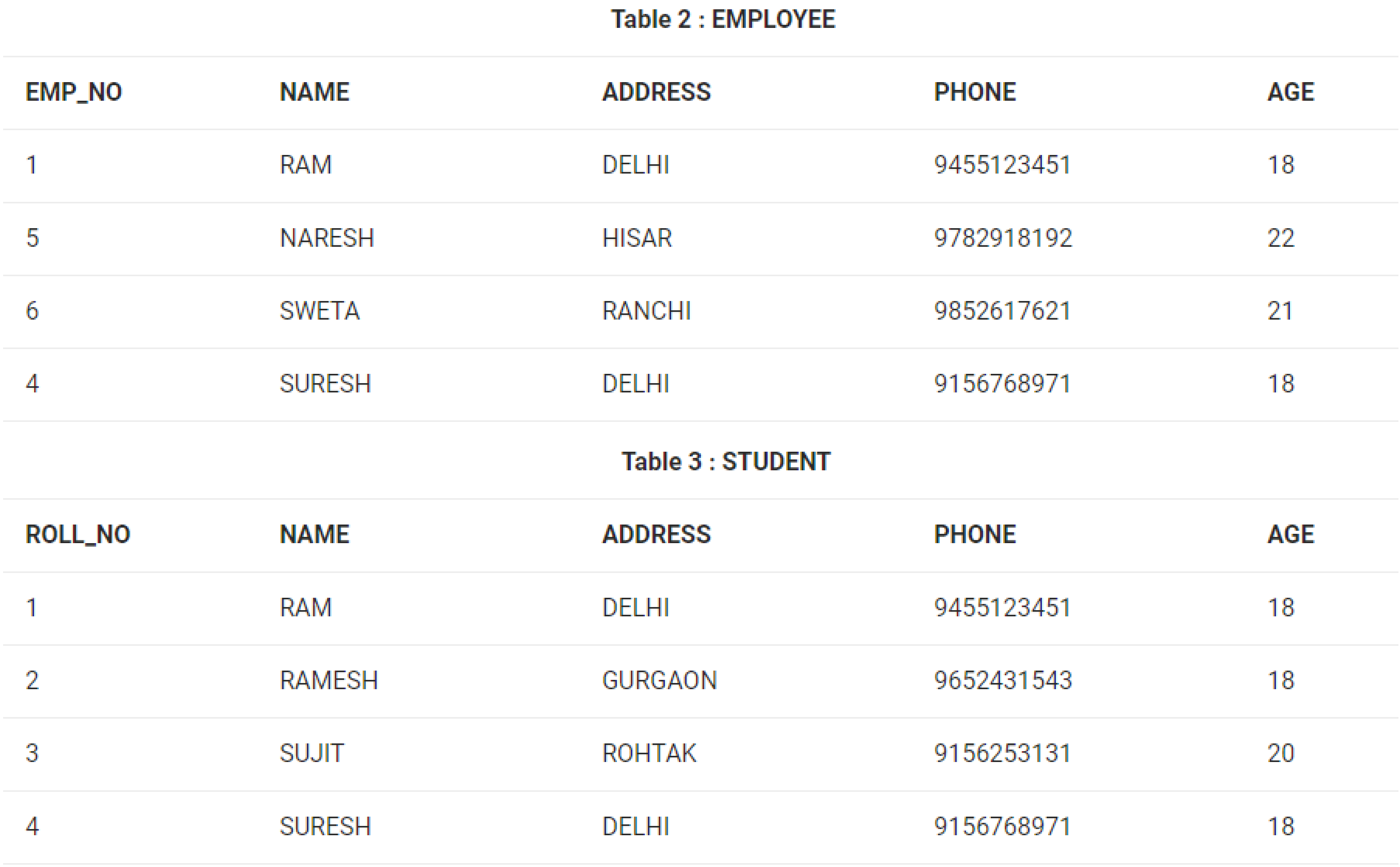


Table 2 : EMPLOYEE

EMP_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21
4	SURESH	DELHI	9156768971	18

Table 3 : STUDENT

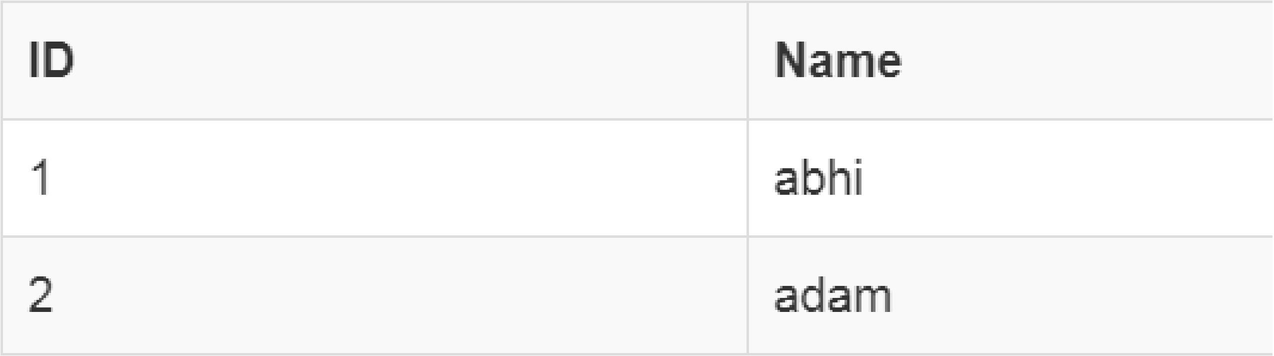
ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

EMPLOYEE ∪ STUDENT

Union Operator

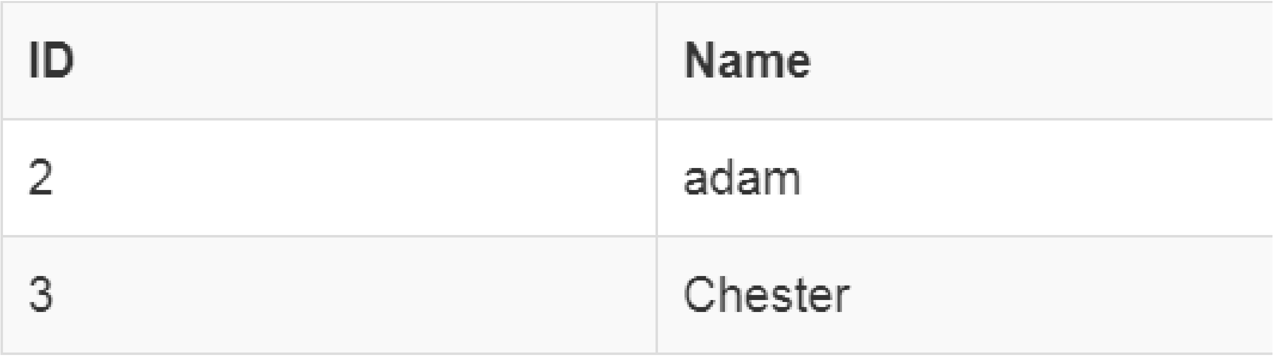
Union Operator (A∪B)

A



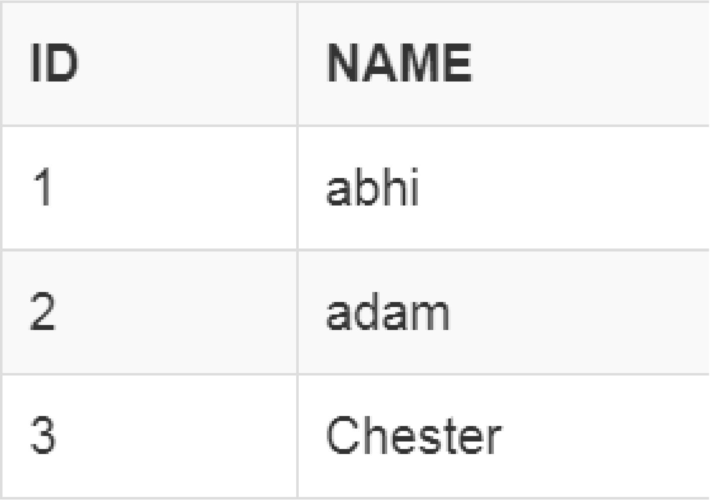
ID	Name
1	abhi
2	adam

B



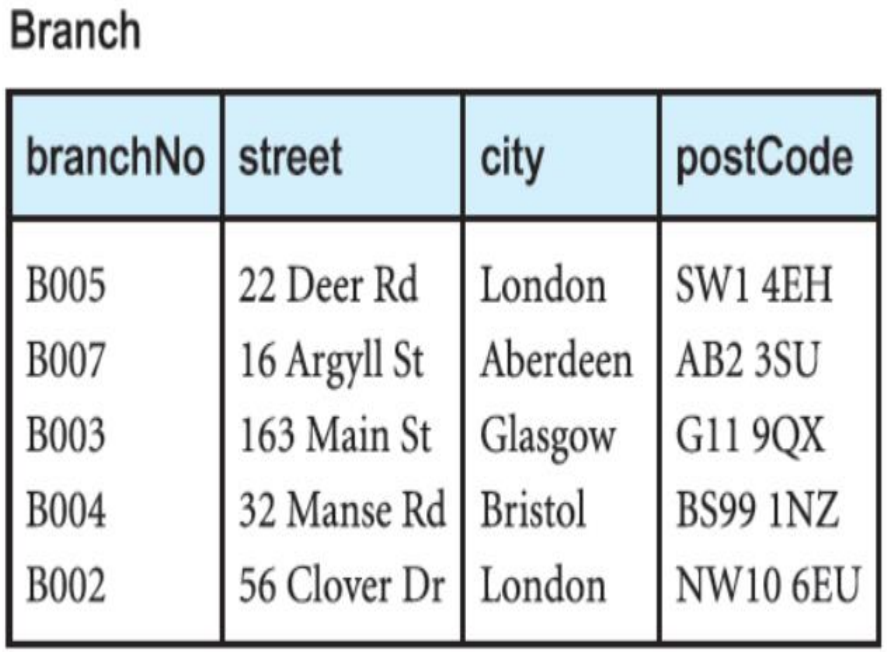
ID	Name
2	adam
3	Chester

A∪B



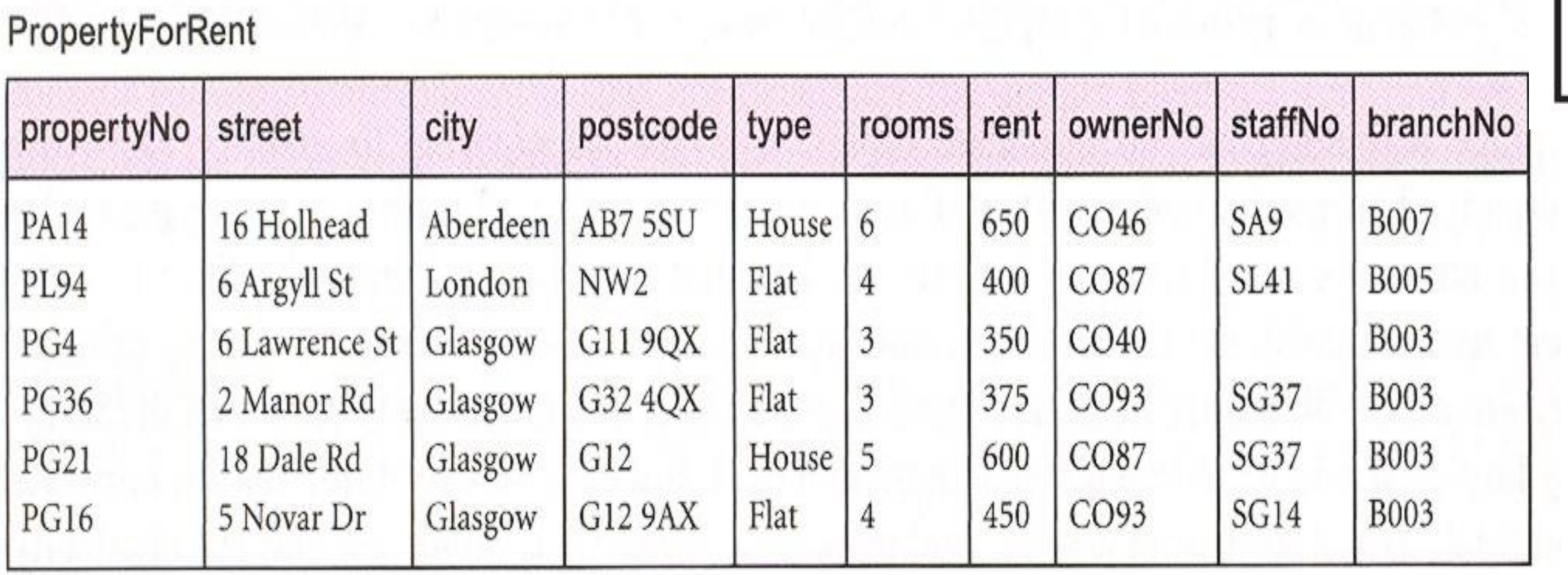
ID	NAME
1	abhi
2	adam
3	Chester

Example 1 :
Table Branch & PropertyForRent



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



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

Union Operator (∪)

UNION OPERATOR (∪)
Table Branch & PropertyForRent

List all cities where there is in branch office or in property for rent.

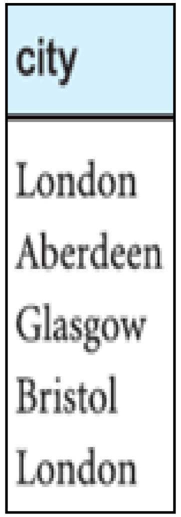
@

List all cities where there is either a branch office or a property for rent.

$R1 \leftarrow \prod \text{city (Branch)}$

$R2 \leftarrow \prod \text{city (PropertyForRent)}$

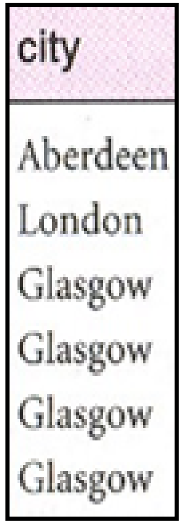
$$\mathbf{R3 \leftarrow R1 \cup R2}$$



city
London
Aberdeen
Glasgow
Bristol
London

R1

∪



city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

R2

Union Operator (∪)

EXAMPLE 1: UNION OPERATOR (∪)

List all cities where there is in branch office or in property for rent.

@

List all cities where there is either a branch office or a property for rent.

$R1 \leftarrow \prod \text{city } (\text{Branch})$

$R2 \leftarrow \prod \text{city } (\text{PropertyForRent})$

$\mathbf{R3 \leftarrow R1 \cup R2}$

$\mathbf{R3 \leftarrow \prod \text{city } (\text{Branch}) \cup \prod \text{city } (\text{PropertyForRent})}$

city
London
Aberdeen
Glasgow
Bristol

INTERSECT OPERATOR (∩)

Only those tuples that appear in both of the named relation are given as an output result.

used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements

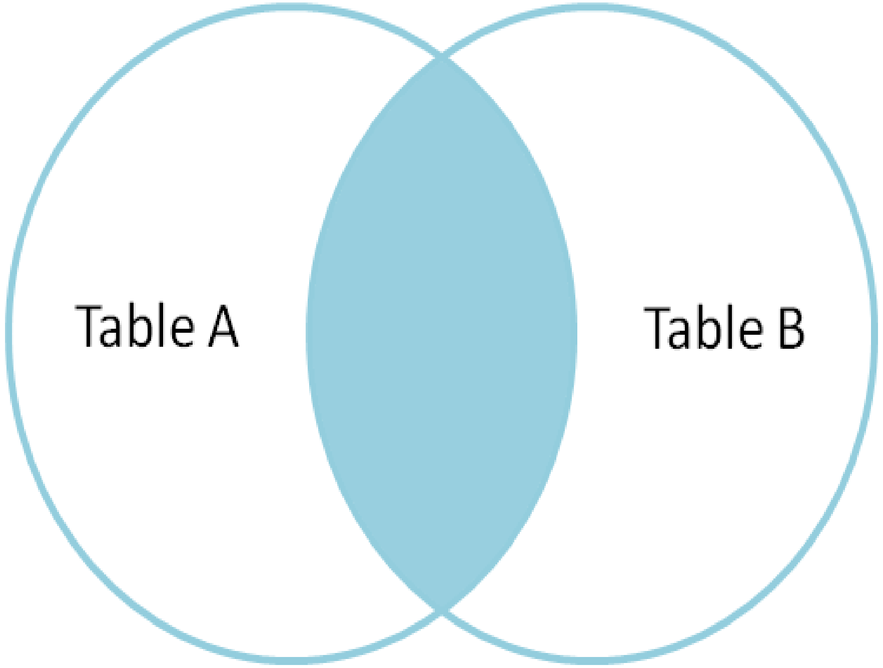
The two operand must be type compatible.

Symbol : ∩

R ∩ S

Syntax: **Л <attributelist>(R) ∩ Л <attributelist>(R)**

INTERSECT OPERATOR (∩)



R

A	1
B	2
D	3
F	4
E	5

S

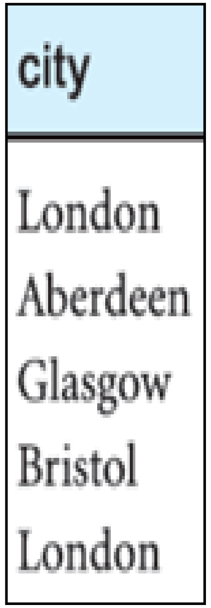
A	1
C	2
D	3
E	4

R |NTERSECTION S

A	1
D	3

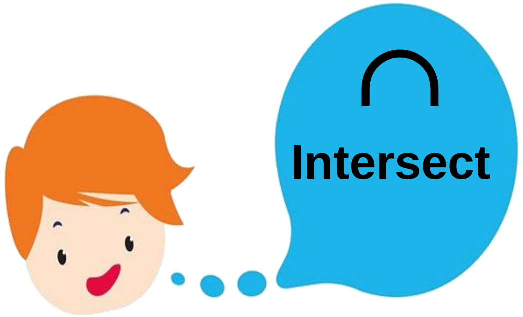
EX 2 : INTERSECT OPERATOR (∩)

List all cities where there is both a branch office and at least one property for rent.



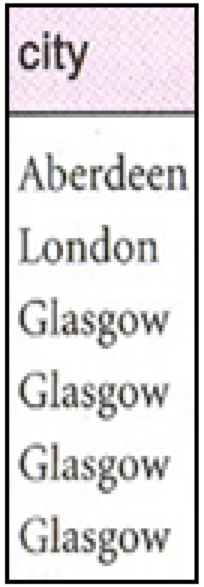
city
London
Aberdeen
Glasgow
Bristol
London

R1

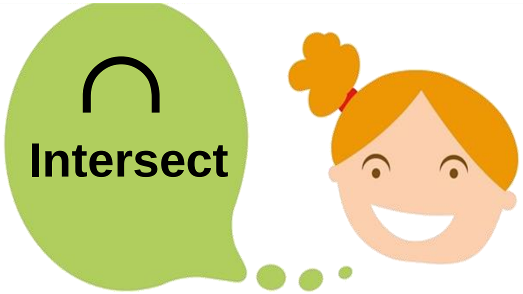


$$\Pi_{city}(Branch) \cap \Pi_{city}(PropertyForRent)$$

R2



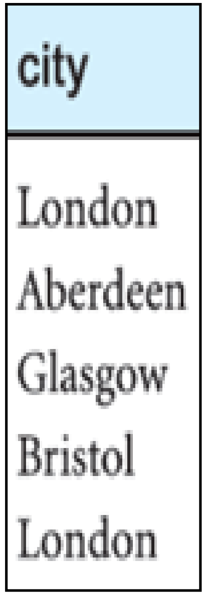
city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow



ANS EX 2 : INTERSECT OPERATOR (∩)

List all cities where there is both a branch office and at least one property for rent.

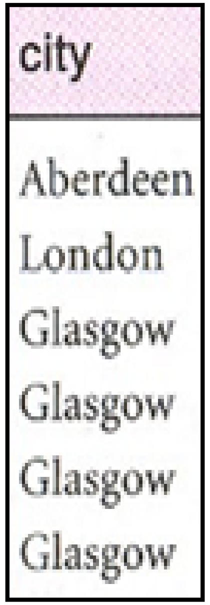
$$\Pi_{city}(\textbf{Branch}) \cap \Pi_{city}(\textbf{PropertyForRent})$$



city
London
Aberdeen
Glasgow
Bristol
London

R1

∩



city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

R2

=

city
Aberdeen
London
Glasgow

DIFFERENCE OPERATOR (–)

The set difference operators takes the two sets and returns the values that are in the first set but not the second set.

The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. Create a new relation.

The two operand must be type compatible.

Symbol : **– (minus)**

Syntax: $\text{Л}_{\text{<attributelist>}}(R) - \text{Л}_{\text{<attributelist>}}(R)$

DIFFERENCE OPERATOR (–)

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R – S

=

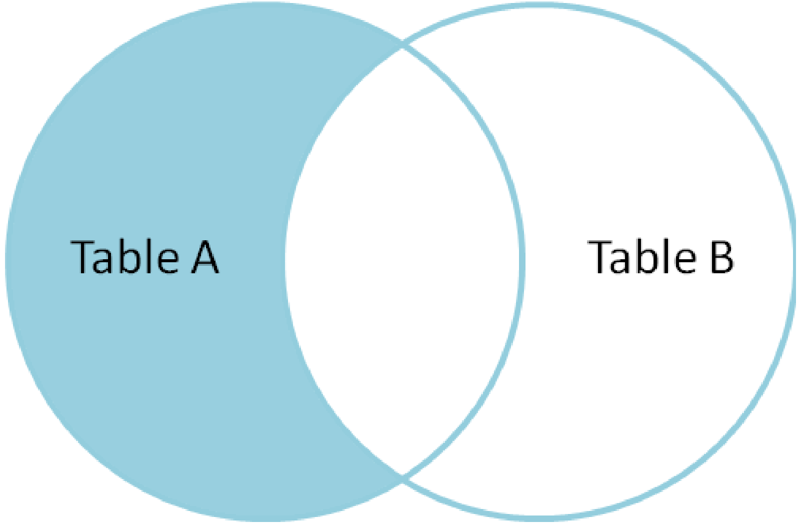
S – R

R DIFFERENCE S

B	2
F	4
E	5

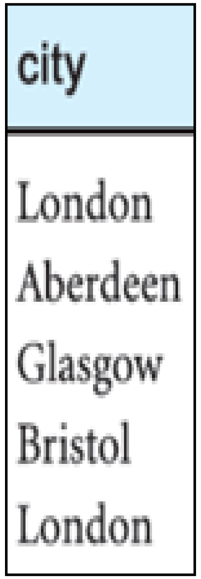
S DIFFERENCE R

C	2
E	4



EX 3 : DIFFERENCE OPERATOR (–)

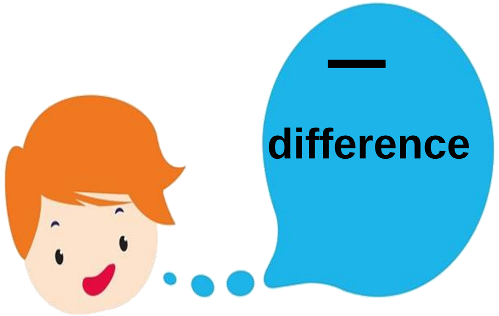
List all cities where there is a branch office but no properties for rent.

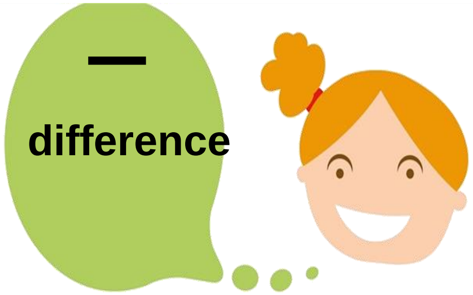


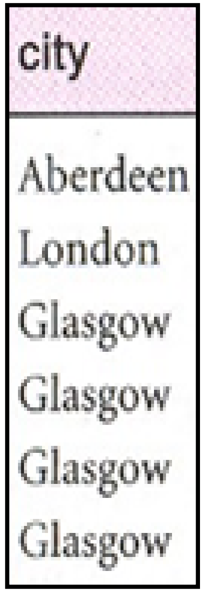
city
London
Aberdeen
Glasgow
Bristol
London

R1

$Л_{city}$(Branch) – $Л_{city}$(PropertyForRent)







city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

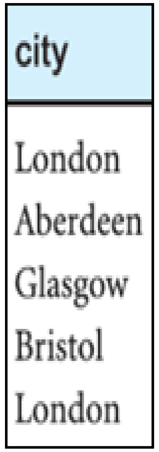
R2

EX 3 : DIFFERENCE OPERATOR (–)

List all cities where there is a branch office but no properties for rent.

$$\text{Л}_{city}(\text{Branch}) - \text{Л}_{city}(\text{PropertyForRent})$$

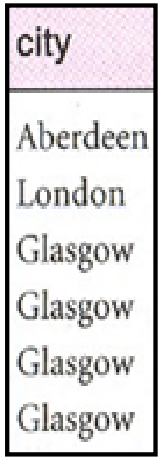
R1 – R2



city
London
Aberdeen
Glasgow
Bristol
London

R1

–



city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

R2

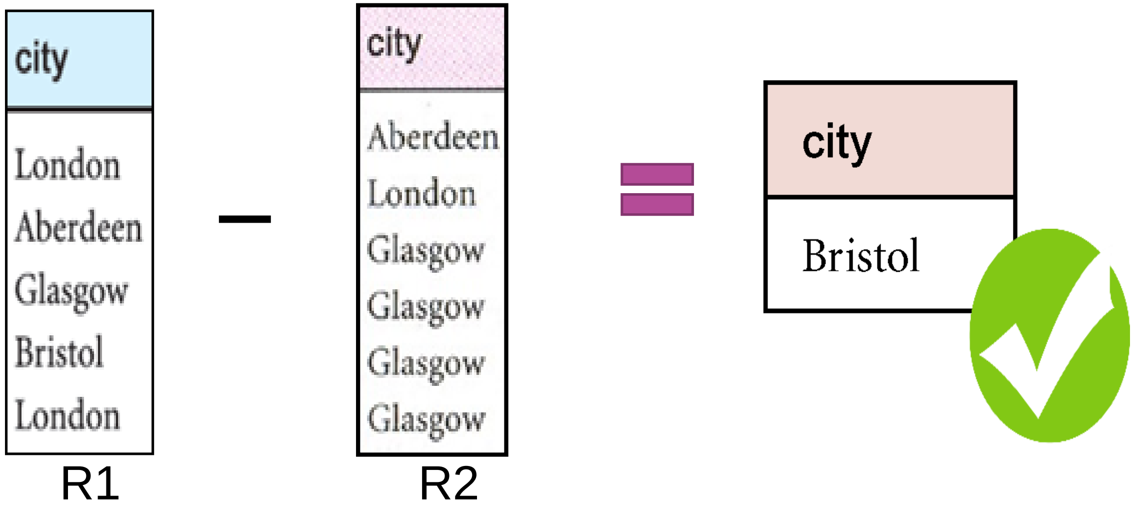
=

EX 3 : DIFFERENCE OPERATOR (–)

List all cities where there is a branch office but no properties for rent.

$$\Pi_{city}(\text{Branch}) - \Pi_{city}(\text{PropertyForRent})$$

R1 – R2



city
London
Aberdeen
Glasgow
Bristol
London

R1

–

city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

R2

=

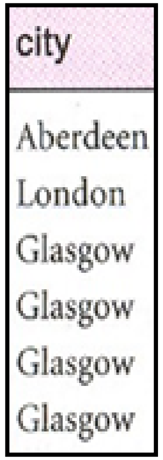
city
Bristol

EX 3 : DIFFERENCE OPERATOR (–)

List all cities where there is a branch office but no properties for rent.

$$\text{Л}_{city}(\text{PropertyForRent}) - \text{Л}_{city}(\text{Branch})$$

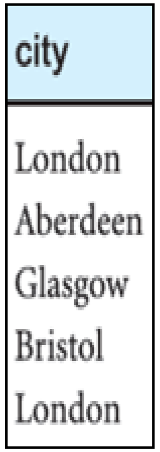
R2 – R1



city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

R2

–



city
London
Aberdeen
Glasgow
Bristol
London

R1

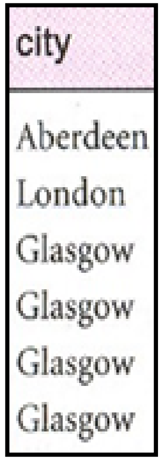
=

EX 3 : DIFFERENCE OPERATOR (–)

List all cities where there is a branch office but no properties for rent.

$$\text{Л}_{\text{city}}(\text{PropertyForRent}) - \text{Л}_{\text{city}}(\text{Branch})$$

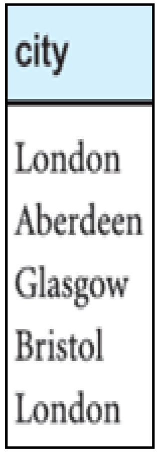
R2 – R1



city
Aberdeen
London
Glasgow
Glasgow
Glasgow
Glasgow

R2

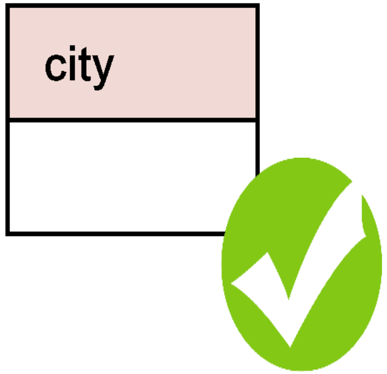
–



city
London
Aberdeen
Glasgow
Bristol
London

R1

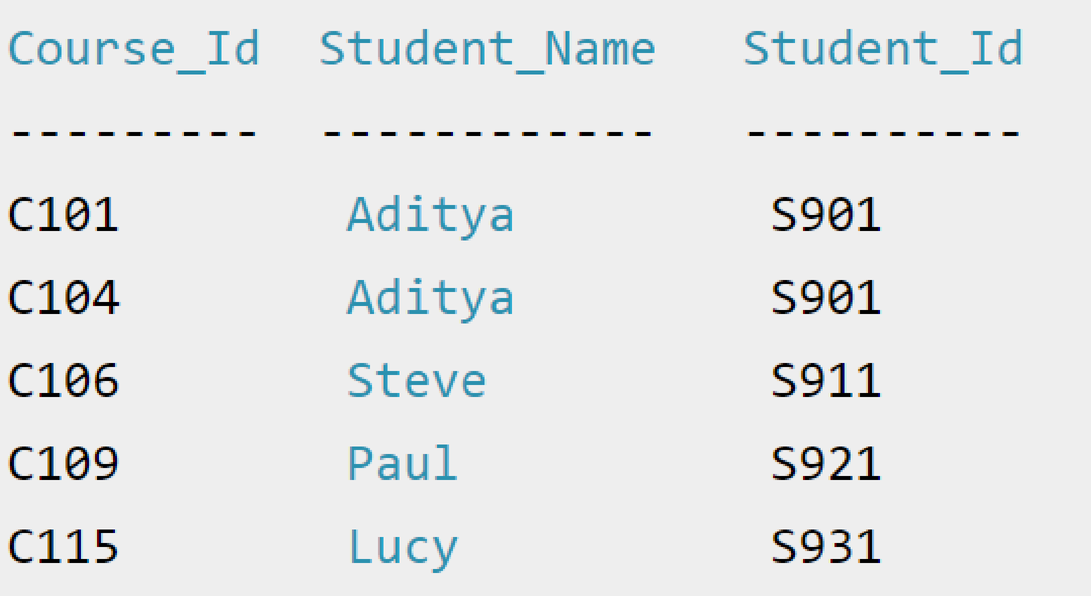
=



city

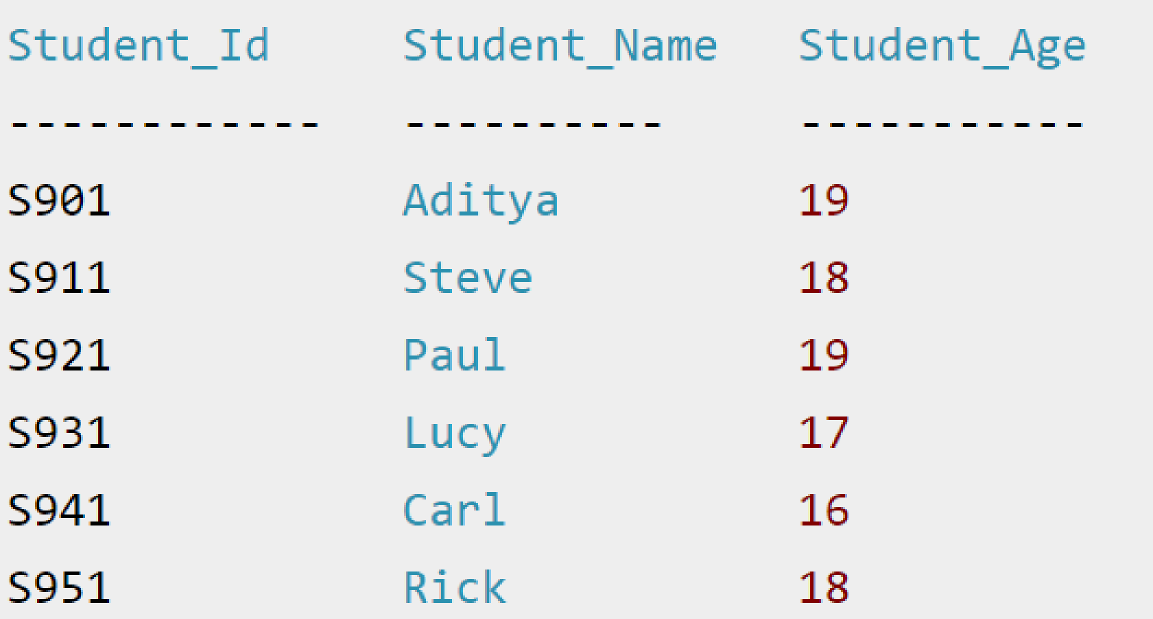
EX 4 : DIFFERENCE OPERATOR (–)

Table 1: COURSE



```
Course_Id  Student_Name   Student_Id
---------  ------------   ----------
C101        Aditya         S901
C104        Aditya         S901
C106        Steve          S911
C109        Paul           S921
C115        Lucy           S931
```

Table 2: STUDENT



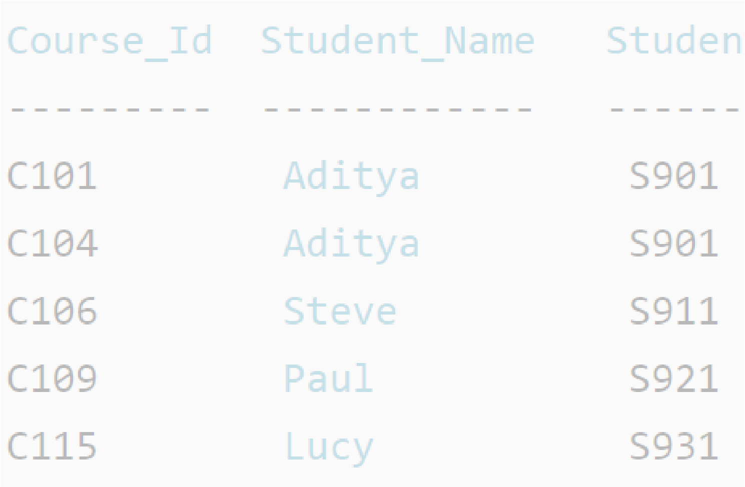
```
Student_Id    Student_Name  Student_Age
------------  ----------    -----------
S901          Aditya        19
S911          Steve         18
S921          Paul          19
S931          Lucy          17
S941          Carl          16
S951          Rick          18
```

QUESTION A.

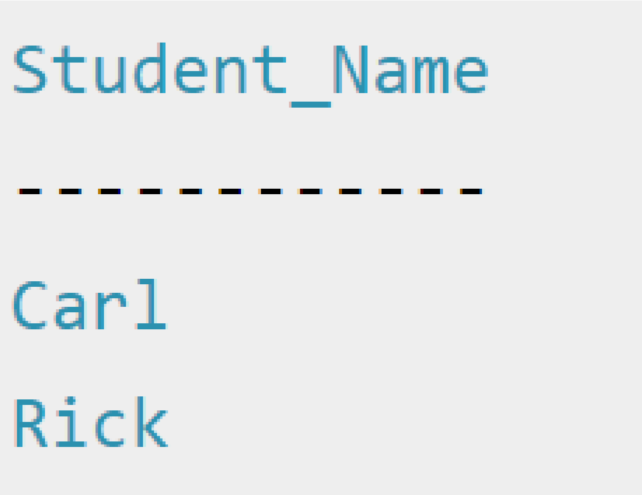
To select those student names that are present in STUDENT table but not present in COURSE table.

EX 4 : DIFFERENCE OPERATOR (–)

Table 1: COURSE

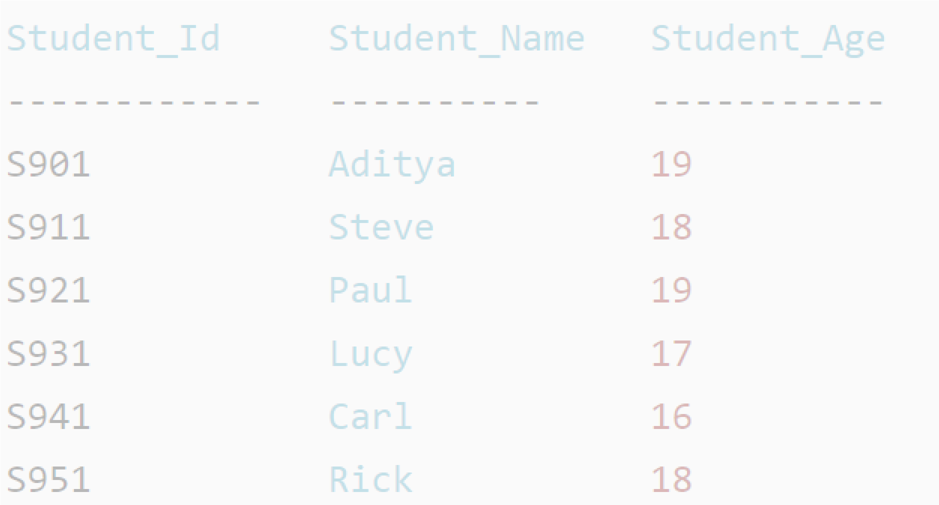


```
Course_Id   Student_Name    Studen
---------   ------------    ------
C101         Aditya          S901
C104         Aditya          S901
C106         Steve           S911
C109         Paul            S921
C115         Lucy            S931
```



```
Student_Name
-------------
Carl
Rick
```

Table 2: STUDENT



```
Student_Id    Student_Name   Student_Age
-----------   ----------     -----------
S901          Aditya         19
S911          Steve          18
S921          Paul           19
S931          Lucy           17
S941          Carl           16
S951          Rick           18
```

Π Student_Name (STUDENT) - Π Student_Name (COURSE)

CROSS PRODUCT (X)

This is used to combine data from two different relations(tables) into one and fetch data from the combined relation.

attributes in both relations being combined

The **CARTESIAN PRODUCT** operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

CROSS PRODUCT (X)

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- Concatenation of every tuple of relation R with every tuple of relation S.

Symbol : Denoted by X

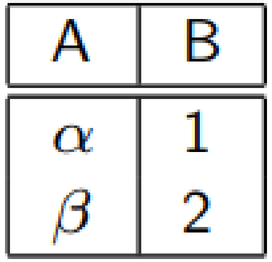
R X S → every row of Relation1, each row of Relation2 is concatenate.

Syntax :

(Л <attributelist>(R)) X (Л <attributlist> (R))

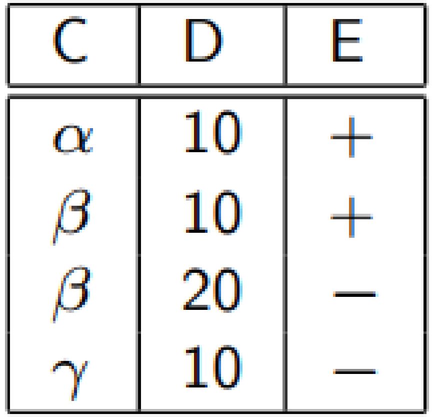
EXAMPLE : CROSS PRODUCT (X)

r



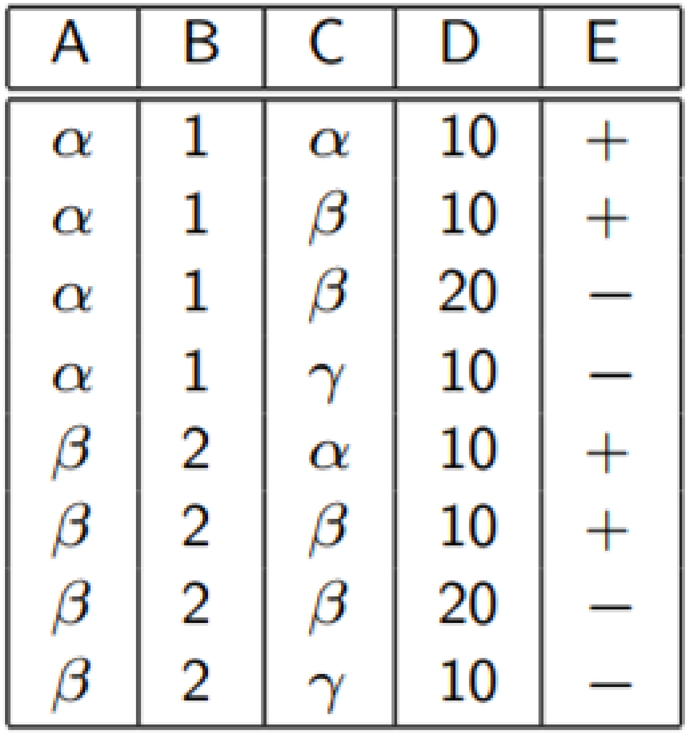
A	B
α	1
β	2

s



C	D	E
α	10	+
β	10	+
β	20	−
γ	10	−

r x s



A	B	C	D	E
α	1	α	10	+
α	1	β	10	+
α	1	β	20	−
α	1	γ	10	−
β	2	α	10	+
β	2	β	10	+
β	2	β	20	−
β	2	γ	10	−

CROSS PRODUCT (X)

List the names and comments of all clients who have viewed a property for rent.

TABLE: CLIENT

ClientNo	fName	lname
CR76	John	Kay
CR56	Aline	Stewart
CR74	Mike	Ritchie
CR62	Mary	Tregear

TABLE: VIEWING

ClientNo	propertyNo	comment
CR56	PA14	Too small
CR76	PG4	Too remote
CR56	PG4	
CR62	PA14	No dining room
CR56	PG36	

CROSS PRODUCT (X)

List the names and comments of all clients who have viewed a property for rent.

R1← ЛclientNo, fName, lName(Client)

R2← ЛclientNo,propertyNo, comment (Viewing)

R3 ← R1 X R2

R1←ЛclientNo, fName, lName(Client)) X
((ЛclientNo,propertyNo, comment (Viewing))

TABLE: CLIENT

ClientNo	fName	lname
CR76	John	Kay
CR56	Aline	Stewart
CR74	Mike	Ritchie
CR62	Mary	Tregear

TABLE: VIEWING

ClientNo	propertyNo	comment
CR56	PA14	Too small
CR76	PG4	Too remote
CR56	PG4	
CR62	PA14	No dining room
CR56	PG36	

CLIENT **X** VIEWING

client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR56	PA14	too small
CR76	John	Kay	CR76	PG4	too remote
CR76	John	Kay	CR56	PG4	
CR76	John	Kay	CR62	PA14	no dining room
CR76	John	Kay	CR56	PG36	
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR76	PG4	too remote
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR62	PA14	no dining room
CR56	Aline	Stewart	CR56	PG36	
CR74	Mike	Ritchie	CR56	PA14	too small
CR74	Mike	Ritchie	CR76	PG4	too remote
CR74	Mike	Ritchie	CR56	PG4	
CR74	Mike	Ritchie	CR62	PA14	no dining room
CR74	Mike	Ritchie	CR56	PG36	
CR62	Mary	Tregear	CR56	PA14	too small
CR62	Mary	Tregear	CR76	PG4	too remote
CR62	Mary	Tregear	CR56	PG4	
CR62	Mary	Tregear	CR62	PA14	no dining room
CR62	Mary	Tregear	CR56	PG36	

JOIN Operator (⋈)

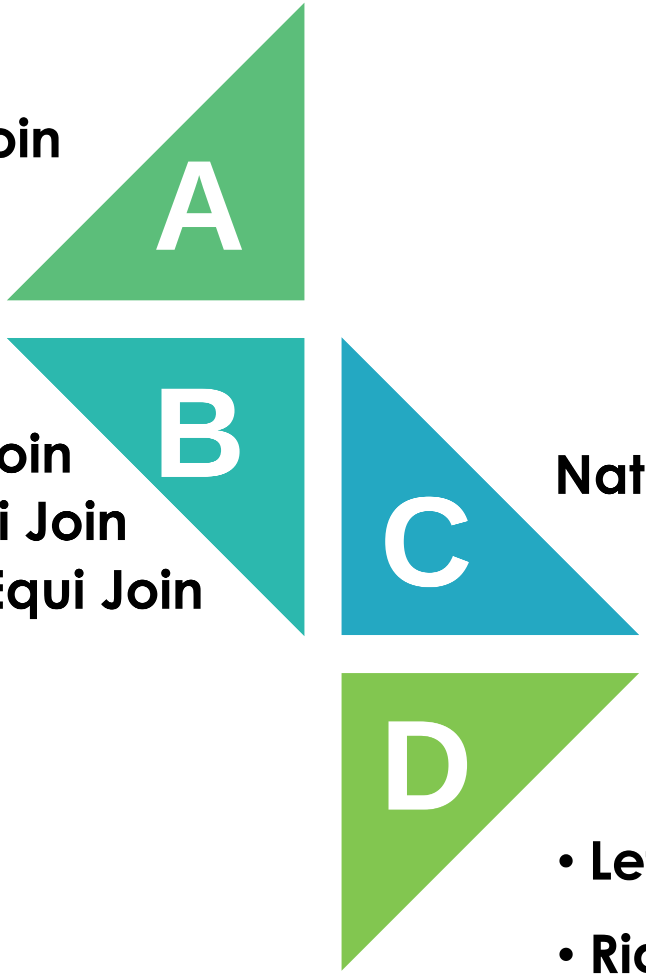
- *Two relations R and S over all common attributes.*
- *One occurrence of each common attribute is eliminated from the result.*

Symbol : ⋈

Syntax :

(Л <attributelist>(R)) ⋈ (Л <attributlist> (R))

TYPES OF JOIN



Inner Join

Theta Join

Semi Join

Equi Join

Natural Join

Outer Join

- Left Outer Join
- Right Outer Join

Inner JOIN

- An INNER JOIN produces a result table containing composite rows created by **combining rows from two tables** where some join condition evaluates to true.
- Rows that **DO NOT SATISFY** the join condition will ***NOT*** *appear* in the result table of an inner join.
- The inner join is the default join type, therefore the keyword INNER is optional and may be omitted.

EX1 : Natural JOIN (Inner Join)

TABLE: CLIENT

ClientNo	fName	lname
CR76	John	Kay
CR56	Aline	Stewart
CR74	Mike	Ritchie
CR62	Mary	Tregear

TABLE: VIEWING

ClientNo	propertyNo	comment
CR56	PA14	Too small
CR76	PG4	Too remote
CR56	PG4	
CR62	PA14	No dining room
CR56	PG36	

R1←(ЛclientNo, fName, lName(Client)) ⋈
((ЛclientNo,propertyNo, comment (Viewing))

EX1 : Natural JOIN (Inner Join)

Client.ClientNo	fname	lname	Viewing.ClientNo	propertyNo	comment
CR76	John	Kay	CR76	PG4	Too remote
CR56	Aline	Stewart	CR56	PA14	Too small
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR56	PG36	
CR62	Mary	Tregear	CR62	PA14	No dining room

R1←(ЛclientNo, fName, lName(Client)) ⋈

((ЛclientNo,propertyNo, comment (Viewing))

EX2 : Natural JOIN (Inner Join)

Students

studNo	name	course
100	Yuzlaan	PH
200	Raimi	CM
300	Arsyad	CM

Courses

courseCode	name
PH	Pharmacy
CM	Computing

$$R1 = \text{Students} \bowtie (\text{course} = \text{courseCode})\,\text{Courses}$$

$$R2 = \pi_{<\textit{studNo, Students.name, courseCode, Courses.name}>} R1$$

studNo	Students.name	course	Courses.name
100	Yuzlaan	PH	Pharmacy
200	Raimi	CM	Computing
300	Arsyad	CM	Computing

Left Outer Join (⟕)

- The (left) Outer join is a join in which tuples from R that do not have matching values in the common attributes of S are also included in the result relation.
- **Missing values** in the second relation are set to ***NULL***.
- Symbol : ⟕

The left outer join operation keeps every tuple in the first or left relation R in R and S; If no matching tuple is found in S, then the attributes of S in the join result are filled with null values.

LET'S GET STARTED

activity

EX1: Left Outer Join (⟕)

Students

studNo	name	course
100	Yuzlaan	PH
200	Raimi	CM
300	Arsyad	CM
400	Auni	EN

⟕

Courses

course	name
PH	Pharmacy
CM	Computing
CH	Chemistry

Students ⟕ Courses

Output EX1 : Left Outer Join (⟕)

Students_Courses

studNo	name	Students.course	Courses.course	name
100	Yuzlaan	PH	PH	Pharmacy
200	Raimi	CM	CM	Computing
300	Arsyad	CM	CM	Computing
400	Auni	EN	**NULL**	**NULL**

Students ⟕ Courses

EX2: Right Outer Join (⋉)

Students

studNo	name	course
100	Yuzlaan	PH
200	Raimi	CM
300	Arsyad	CM
400	Auni	EN

⋉

Courses

course	name
PH	Pharmacy
CM	Computing
CH	Chemistry

Students ⋉ Courses

Output EX2: Right Outer Join (⋉)

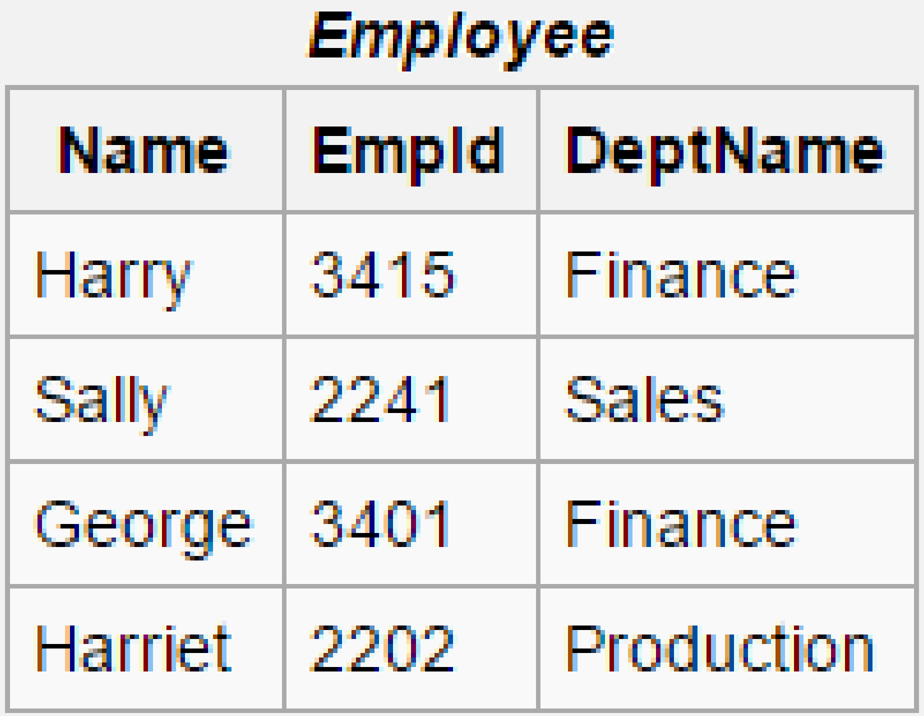
Students_Courses

studNo	name	Students.course	Courses.course	name
100	Yuzlaan	PH	PH	Pharmacy
200	Raimi	CM	CM	Computing
300	Arsyad	CM	CM	Computing
NULL	**NULL**	**NULL**	CH	Chemistry

Students ⋉ Courses

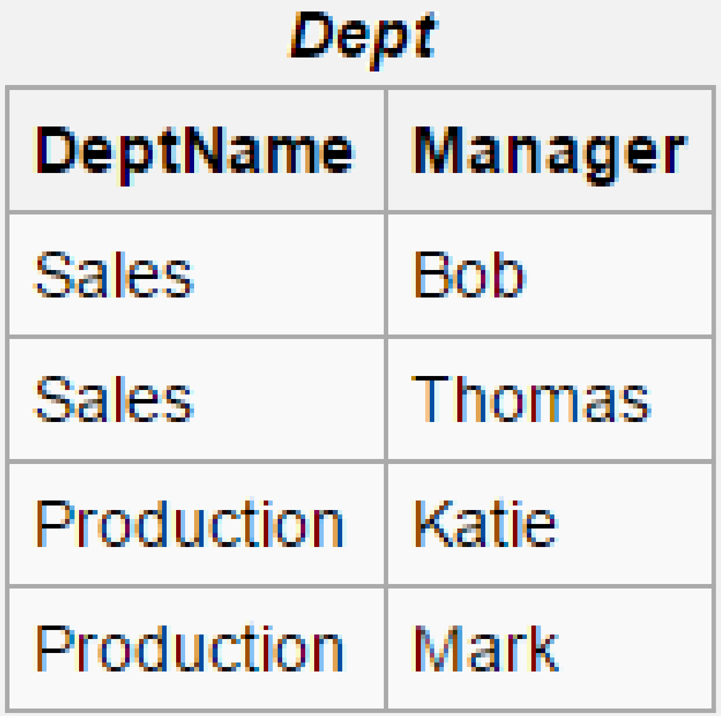
Outer Join

EX3: Left OJ (⋊) & Right OJ (⋉)



Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Production



Dept

DeptName	Manager
Sales	Bob
Sales	Thomas
Production	Katie
Production	Mark

$$\text{Employee} \rtimes \text{Dept}$$

Outer Join

EX4: EMPLOYEE⋊Parking

Employee

Emp_id	Emp_name	Emp_office
1001	Bob	10
1002	Alice	11
1003	Sandy	10
1004	Larry	11
1005	Susan	11

Parking

Emp_id	Parking_lot	Parking_space
1001	A	6
1002	A	14
1003	B	17
1004	B	6
1005	A	12

Outer Join

EX4: EMPLOYEE⋉Parking

Employee

Emp_id	Emp_name	Emp_office
1001	Bob	10
1002	Alice	11
1003	Sandy	10
1004	Larry	11
1005	Susan	11

Parking

Emp_id	Parking_lot	Parking_space
1001	A	6
1002	A	14
1003	B	17
1004	B	6
1005	A	12