Relational Algebra

AY2019-20, Spring semester

COMP1041: Database and Interface

Week 4

Contents

- Basic operators in relational algebra.
- Cartesian product and Joins.
- Aggregations, division and grouping.

Relational Algebra

Operations of a relational database: Users should be able to store, retrieve and update data.

- We need a language to describe these operations.
- Theoretical/abstract language: relational algebra.
- Practical/concrete programming language: Structured Query Language (SQL)
- Relational algebra lays the theoretical foundation of (part of) SQL, through relational calculus.

Relational Algebra

- Principle of closure
  - Relations are closed under the algebra;
  - Just as: Numbers are closed under arithmetic operations.

  - Numbers after (nested) operations are still numbers.
  - Relations after (nested) relational algebra operations are still relations.

Relational Algebra

- Relational algebra works on one or more relations to define/derive another relation without affecting the original relations.
- Relational algebra is a family of algebras with a well-founded semantics used for modelling the data stored in relational databases, and defining queries on it. – Wikipedia

Relational Algebra

- Five fundamental operations
  - Selection (**σ**, 'sigma', unary)
  - Projection (**π**, 'pi', unary)
  - Union (**∪**, binary)
  - Cartesian Product (**×**, binary)
  - Set Difference (**-**, binary)
- Others can be expressed as combination of the five.
  - Join, Intersection, division, aggregation, grouping

Relational Algebra

- "Find all universities with > 20000 students"
  - Relational Algebra:
    - $\pi_{\text{uName}}(\sigma_{\text{Enrollment} > 20000}(\text{University}))$
  - SQL:
    - `SELECT uName FROM University WHERE University.Enrollment > 20000`

- Relational algebra is set-based, that means duplicated tuples in results are always eliminated.

Relational Algebra: Basic Operators

Selection: **σ** (Sigma)

- Usage: $\boldsymbol{\sigma}_{predicate}(R)$
- The Selection operation works on a single relation R and defines a relation that contains only those tuples of R that satisfy the specified condition (predicate).
- *List all staff with a salary greater than £10,000*
  - $\sigma_{\text{salary}>10000}(\text{staff})$
  - $\sigma_{\text{staff.salary}>10000}(\text{staff})$
  - Useful when a column name appears in two tables
- Predicate also supports logical operators ∧ (AND), ∨ (OR) and ~ (NOT).

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

$$\sigma_{salary>10000}(\text{Staff})$$

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
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Projection: π (Pi)

- Usage: $\boldsymbol{\pi}_{a_1,a_2,a_3\ldots,a_n}(R)$
- The Projection operation 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.

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

$$\pi_{\text{staffNo,fName,IName,salary}}(R)$$

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

$\pi_{\text{staffNo,fName,lName,salary}}(Staff)$

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

Union: ∪

- Usage: $R_1 \cup R_2$
- The union of two relations R_1 and R_2 defines a relation that contains all the tuples of R_1, or R_2, or both R_1 and R_2, duplicate tuples being eliminated. R_1 and R_2 must be union-compatible.

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

$$\pi_{\text{city}}(\text{Branch}) \cup \pi_{\text{city}}(\text{PropertyForRent})$$

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	postco
PA14	16 Holhead	Aberdeen	AB7 5SU
PL94	6 Argyll St	London	NW2
PG4	6 Lawrence St	Glasgow	G11 9Q
PG36	2 Manor Rd	Glasgow	G32 4Q
PG21	18 Dale Rd	Glasgow	G12
PG16	5 Novar Dr	Glasgow	G12 9A

$$\pi_{city}(Branch) \cup \pi_{city}(PropertyForRent)$$

city
London
Aberdeen
Glasgow
Bristol

Union Compatible

- UNION-compatible means that the numbers of attributes must be the same and their corresponding data types also match

- **union compatible:**
  ```
  A: (First_name (char), Last_name(char), Date_of_Birth(date))
  B: (FName(char),       LName(char),     DOB(date))
  ```
  Both table have 3 attributes and of same data type.

- **Not compatible**:
  ```
  A: (First_name (char), Last_name(char), Date_of_Birth(date))
  B: (FName(char),       LName(char),     PhoneNumber(number))
  ```

Set Difference: − , Minus

- Usage: $R - S$
- The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. R and S must be union-compatible.

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

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

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
PA14	16 Holhead	Aberdeen	AB7 5SU
PL94	6 Argyll St	London	NW2
PG4	6 Lawrence St	Glasgow	G11 9QX
PG36	2 Manor Rd	Glasgow	G32 4QX
PG21	18 Dale Rd	Glasgow	G12
PG16	5 Novar Dr	Glasgow	G12 9AX

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

city
Bristol

Intersection: ∩

- Usage: $R \cap S$
- The Intersection operation defines a relation consisting of the set of all tuples that are in both R and S. R and S must be union-compatible.
- Same as: $R - (R - S)$

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

$$\pi_{\text{city}}(\text{Branch}) \cap \pi_{\text{city}}(\text{PropertyForRent})$$

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	postcoc
PA14	16 Holhead	Aberdeen	AB7 5SU
PL94	6 Argyll St	London	NW2
PG4	6 Lawrence St	Glasgow	G11 9QX
PG36	2 Manor Rd	Glasgow	G32 4QX
PG21	18 Dale Rd	Glasgow	G12
PG16	5 Novar Dr	Glasgow	G12 9AX

$$\pi_{\text{city}}(\text{Branch}) \cap \pi_{\text{city}}(\text{PropertyForRent})$$

city
Aberdeen
London
Glasgow

Cartesian Product and Joins

Cartesian Product: ×

- Usage: $R \times S$
- The Cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

- List the names and comments of all clients who have viewed a property for rent.
- First step: build a relation of all possible clients (with their names) that might or might not have viewed a property

$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \times (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$

Client

clientNo	fName	lName	telNo	prefTy
CR76	John	Kay	0207-774-5632	Flat
CR56	Aline	Stewart	0141-848-1825	Flat
CR74	Mike	Ritchie	01475-392178	House
CR62	Mary	Tregear	01224-196720	Flat

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \times (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$

	client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
Client[0] + Viewing[0]	CR76	John	Kay	CR56	PA14	too small
Client[0] + Viewing[1]	CR76	John	Kay	CR76	PG4	too remote
Client[0] + Viewing[2]	CR76	John	Kay	CR56	PG4	
Client[0] + Viewing[3]	CR76	John	Kay	CR62	PA14	no dining room
Client[0] + Viewing[4]	CR76	John	Kay	CR56	PG36	
Client[1] + Viewing[0]	CR56	Aline	Stewart	CR56	PA14	too small
Client[1] + Viewing[1]	CR56	Aline	Stewart	CR76	PG4	too remote
Client[1] + Viewing[2]	CR56	Aline	Stewart	CR56	PG4	
Client[1] + Viewing[3]	CR56	Aline	Stewart	CR62	PA14	no dining room
Client[1] + Viewing[4]	CR56	Aline	Stewart	CR56	PG36	
Client[2] + Viewing[0]	CR74	Mike	Ritchie	CR56	PA14	too small
Client[2] + Viewing[1]	CR74	Mike	Ritchie	CR76	PG4	too remote
Client[2] + Viewing[2]	CR74	Mike	Ritchie	CR56	PG4	

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	

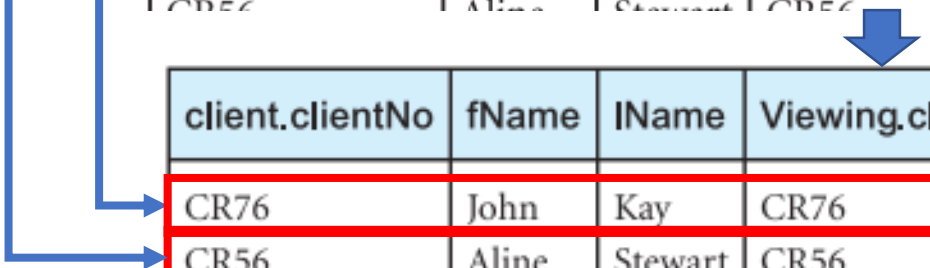
Cartesian Product

- Some client numbers in the result do not match, how do we remove these tuples?
- Here is the original expression and its generated table:

$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \times (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$

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

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



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

$$\sigma_{\text{Client.clientNo} = \text{Viewing.clientNo}}((\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \times (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing})))$$

(Use selection with a predicate)

Theta Join $\bowtie_F$

- Usage: $\mathrm{R} \bowtie_F \mathrm{S}$
- The Theta join defines a relation that contains tuples satisfying the predicate *F* from $R \times S$. The predicate *F* is of the form "$R.a_i \ \theta \ S.b_i$" where θ may be one of the comparison operators (<, ≤, >, ≥, =, ≠).
  - If F contains only equality (=), it is instead called **Equijoin**.
- The previous example can be simply represented by a **Theta Join**: $\mathrm{R} \bowtie_F S$
  - $\mathrm{R} \bowtie_F \mathrm{S} = \sigma_F (\mathrm{R} \times \mathrm{S})$

Theta Join $\bowtie_F$

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

$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \bowtie_{\text{Client.clientNo = Viewing.clientNo}} (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$

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

Same result as the previous one ;)

Natural Join ⋈

- Usage: R ⋈ S
- The Natural join is an Equijoin of the two relations R and S over all common attributes *x*.
- One occurrence of each common attribute is eliminated from the result. (see clientNo in the next slide)

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

$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \bowtie (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$

Client

clientNo	fName	lName	telNo	prefTy
CR76	John	Kay	0207-774-5632	Flat
CR56	Aline	Stewart	0141-848-1825	Flat
CR74	Mike	Ritchie	01475-392178	House
CR62	Mary	Tregear	01224-196720	Flat

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

$$(\Pi_{\text{clientNo, fName, lName}}(\text{Client})) \bowtie (\Pi_{\text{clientNo, propertyNo, comment}}(\text{Viewing}))$$

The extra clientNo is removed.

See theta join for comparison

clientNo	fName	lName	propertyNo	comment
CR76	John	Kay	PG4	too remote
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR56	Aline	Stewart	PG36	
CR62	Mary	Tregear	PA14	no dining room

Natural Join ⋈

- Natural Join works based on the attributes names and their types (domain)
  - If two tables have attributes with the same names and data types, then these attributes will be selected.

- Natural Join on multiple attributes:
  - All attributes from two tables must have the same value to be considered for concatenation.

Left Outer Join: ⟕

- Usage: R ⟕ S
- 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.
  - "Left (Outer numbers) right".
  - Like natural join, but use Nulls for missing values in table S.

- Produce a status report on property viewings.

$$(\Pi_{\text{propertyNo, street, city}}(\text{PropertyForRent})) \text{ ⟕ Viewing}$$

PropertyForRent

propertyNo	street	city	postcc
PA14	16 Holhead	Aberdeen	AB7 5S
PL94	6 Argyll St	London	NW2
PG4	6 Lawrence St	Glasgow	G11 9Q
PG36	2 Manor Rd	Glasgow	G32 4Q
PG21	18 Dale Rd	Glasgow	G12
PG16	5 Novar Dr	Glasgow	G12 9A

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

$$(\Pi_{\text{propertyNo, street, city}}(\text{PropertyForRent})) \bowtie \text{Viewing}$$

propertyNo	street	city	clientNo	viewDate	comment
PA14	16 Holhead	Aberdeen	CR56	24-May-04	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-04	no dining room
PL94	6 Argyll St	London	**null**	**null**	**null**
PG4	6 Lawrence St	Glasgow	CR76	20-Apr-04	too remote
PG4	6 Lawrence St	Glasgow	CR56	26-May-04	
PG36	2 Manor Rd	Glasgow	CR56	28-Apr-04	
PG21	18 Dale Rd	Glasgow	**null**	**null**	**null**
PG16	5 Novar Dr	Glasgow	**null**	**null**	**null**

PropertyForRent[0]

PA14	16 Holhead	Aberdeen	AB7 5S

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

Found two matches:

PA14	16 Holhead	Aberdeen	CR56	24-May-04	too small
PA14	16 Holhead	Aberdeen	CR62	14-May-04	no dining room

PropertyForRent[1]

PL94	6 Argyll St	London	NW2

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	too small
CR76	PG4	20-Apr-04	too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	no dining room
CR56	PG36	28-Apr-04	

No matches found, add Null

PL94	6 Argyll St	London	null	null	null

Left Outer Join: R ⟕ S

- Q: What if “Viewing” has more “PropertyNo” than in the “PropertyForRent”?
- A: Check the steps in the previous slide.
- A: Or check this Wikipedia page: can you work it out by yourself? (https://en.wikipedia.org/wiki/Relational_algebra#Outer_joins)

- Right Outer Join also exists.
  - Just the opposite of left outer Join.
  - Symbol is also reversed (⟖)
- Full Outer join (⟗) also exists.

Rename Operator: ρ, rho

- $\rho_s(E)$ or $\rho_{s(a_1,a_2,\ldots,a_n)}(E)$
- The Rename operation provides a new name S for the expression E, and optionally names the attributes as $a_1, a_2, \ldots, a_n$.
- Why rename operator?
  - E.g. Natural Join relies on the attribute name to work properly.
  - Some times the attributes of two different tables have different names, but they actually refer to the same kind of information.

- A: (First_name (char), Last_name(char), Date_of_Birth(date))
  B: (FName(char), LName(char), DOB(date))

Rename Operator: rho ρ

- $\rho_s(E)$ or $\rho_{s(a_1,a_2,\ldots,a_n)}(E)$
- $\rho_{uni}(University)$ changes "University" table to "uni"
- $\rho_{b(col1,col2,col3,col4)}(Branch)$ changes "Branch" table to "b" without changing its tuples.

col1	col2	col3	col4
Tuples	Stay	The	Same

Division, Aggregation, Grouping

Division: ÷

- Usage: $R \div S$
- Division is used when we wish to express queries with “all”:
  - “Which persons have a bank account at ALL the banks in the country?”
  - “Which students are registered on ALL the courses”

- Assume relation R is defined over the attribute set A and relation S is defined over the attribute set B such that B ⊆ A (B is a subset of A).

Example of relation A and relation B.

$\Pi_{clientNo,propertyNo}(Viewing)$

clientNo	propertyNo
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

A: (ClientNo, propertyNo)

$\Pi_{propertyNo}(\sigma_{rooms=3}(PropertyForRent))$

propertyNo
PG4
PG36

B: (propertyNo)

Identify all clients who have viewed all properties with three rooms:

$$(\Pi_{clientNo,\ propertyNo}(Viewing)) \div (\Pi_{propertyNo}(\sigma_{rooms\ =\ 3}(PropertyForRent)))$$

$\Pi_{clientNo,propertyNo}(Viewing)$

clientNo	propertyNo
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

$\Pi_{propertyNo}(\sigma_{rooms=3}(PropertyForRent))$

propertyNo
PG4
PG36

$$(\Pi_{clientNo,\ propertyNo}(Viewing)) \div (\Pi_{propertyNo}(\sigma_{rooms\ =\ 3}(PropertyForRent)))$$

RESULT

clientNo
CR56

Division: ÷

clientNo	propertyNo
CR56	PA14
CR76	PG4
CR56	PG4
CR62	PA14
CR56	PG36

propertyNo
PG4
PG36

Formal definition as in our text book:

- Assume relation R is defined over the attribute set A and relation S is defined over the attribute set B such that B ⊆ A (B is a subset of A).

- Let C = A − B, that is, C is the set of attributes of R that are not attributes of S (clientNo). The Division operation defines a relation over the attributes C that consists of the set of tuples from R that match the combination of every tuple (PG4, PG36) in S.

Aggregate:

- Usage: $_{AL}(R)$
- Applies the aggregate function list, AL, to the relation R to define a relation over the aggregate list. AL contains one or more (<aggregate_function>, <attribute>) pairs.

- Aggregate functions:
  - COUNT: returns the number of values in the associated attribute.
  - SUM: returns the sum of the values in the associated attribute.
  - AVG: returns the average of the values in the associated attribute.
  - MIN: returns the smallest value in the associated attribute.
  - MAX: returns the largest value in the associated attribute.

How many properties cost more than £350 per month to rent?

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

$$\rho_R(\text{myCount}) \quad \text{COUNT propertyNo}\,(\sigma_{\text{rent} > 350}(\text{PropertyForRent}))$$

myCount
5

Find the minimum, maximum, and average staff salary

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

ρ_R(myMin, myMax, myAverage) MIN salary, MAX salary, AVERAGE salary (Staff)

myMin	myMax	myAverage
9000	30000	17000

Grouping

- Usage: $_{GA\ AL}(R)$
- Groups the tuples of relation R by the grouping attributes, GA.
- And then applies the aggregate function list AL to define a new relation. AL contains one or more (<aggregate_function>, <attribute>) pairs.
- The resulting relation contains the grouping attributes, GA, along with the results of each of the aggregate functions.
- Simplified: find aggregation result for each different value in GA.

Grouping

- Find the number of staff working in each branch and the sum of their salaries.

$$\rho_R(\text{branchNo, myCount, mySum})\ {}_{\text{branchNo}}\ \mathbf{COUNT}\ \text{staffNo, SUM salary}\ (\text{Staff})$$

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

ρ_R(branchNo, myCount, mySum) branchNo COUNT staffNo, SUM salary (Staff)

branchNo	myCount	mySum
B003	3	54000
B005	2	39000
B007	1	9000