Database Systems

Tutorial Week 5

Objectives

I. Relational algebra (RA) review
II. Relational algebra and SQL statements

Relational Algebra

- Theory behind SQL
- Gives a plan for evaluating a query
- Exploiting “equivalencies” of relational operators can lead to faster queries
  - E.g. joins can be expensive, but we can “push” selections and projections ahead of joins so you’re joining on a smaller table
- Input to an operation?
  - Table(s) of a relation with rows and columns
- Output?
  - A table of a relation with rows and columns

Fundamental Operations

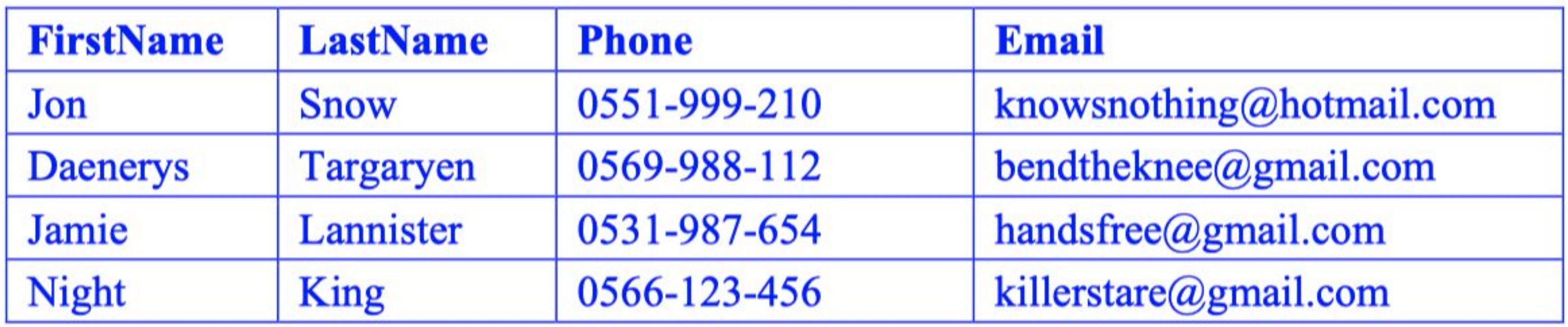
- 5 basic operators of Relational Algebra
- Can form other compound operations
- What are they?
    - Removal operators:
        - Projection (π)
        - Selection (σ)
    - Set operators:
        - Union (∪)
        - Set difference (–)
        - Cross product (×)

Removal Operators

- Remove components from a table of a relation
- Projection (π)
  - Removes columns
  - $\pi_{A1,\ A2,\ \ldots,\ An}(R)$ where R = relation and A1, A2, …, A*n* are attributes that are “projected”
  - Creates a new relation with a subset of attributes
  - All the tuples are included in the new relation, but only the attributes A1, A2, …, A*n* are kept
  - SQL: read π as `SELECT`
- Selection (σ)
  - Removes rows
  - $\sigma_C(R)$ where R = relation and C = condition used to filter rows
  - Creates a new relation with rows where C is true
  - SQL: read σ as `WHERE`

Example Time

Consider the following Person table:

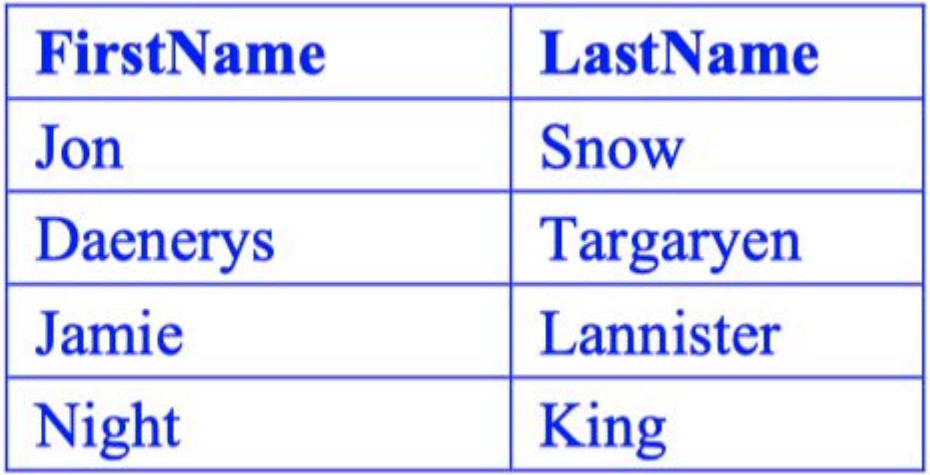


FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Daenerys	Targaryen	0569-988-112	bendtheknee@gmail.com
Jamie	Lannister	0531-987-654	handsfree@gmail.com
Night	King	0566-123-456	killerstare@gmail.com

What will $\pi_{(\text{FirstName, LastName})}$(Person) result in?

Example Time

What will $\pi_{(\text{FirstName, LastName})}$(Person) result in?

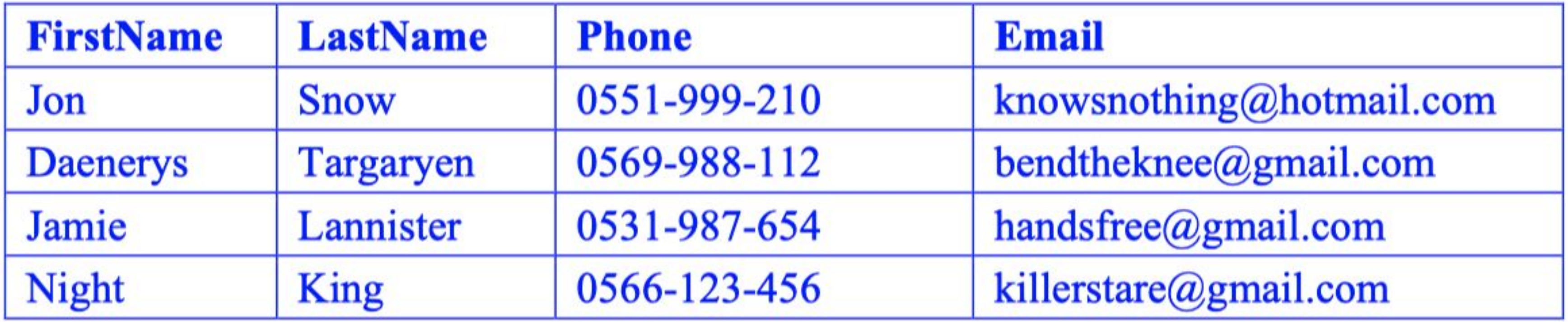


FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

Example Time

For the same original Person table, what will $\sigma_{\text{FirstName = 'Jon'} \lor \text{LastName = 'King'}}$(Person) result in?

And: $\land$
Or: $\lor$

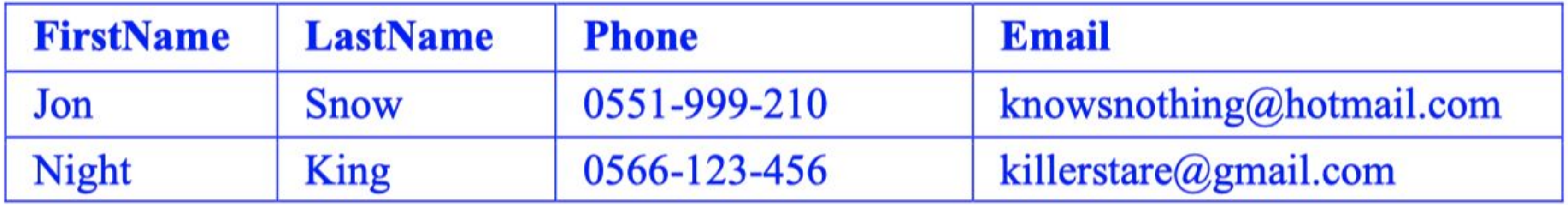


FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Daenerys	Targaryen	0569-988-112	bendtheknee@gmail.com
Jamie	Lannister	0531-987-654	handsfree@gmail.com
Night	King	0566-123-456	killerstare@gmail.com

Example Time

For the same original Person table, what will $\sigma_{\text{FirstName = 'Jon'} \lor \text{LastName = 'King'}}(\text{Person})$ result in?

And: $\land$
Or: $\lor$

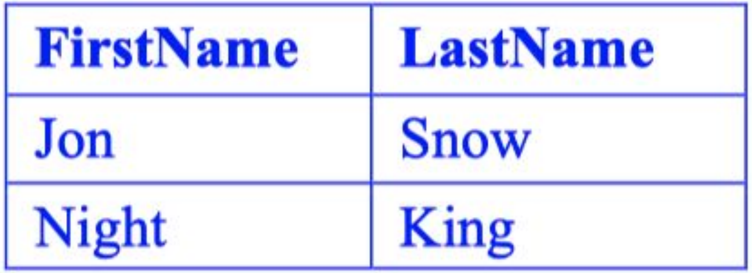


FirstName	LastName	Phone	Email
Jon	Snow	0551-999-210	knowsnothing@hotmail.com
Night	King	0566-123-456	killerstare@gmail.com

Example Time

Can combine the two operations in one expression:

$$\pi_{\text{FirstName, LastName}}(\sigma_{\text{FirstName= 'Jon' } \lor \text{ LastName = 'King'}}(\text{Person}))$$



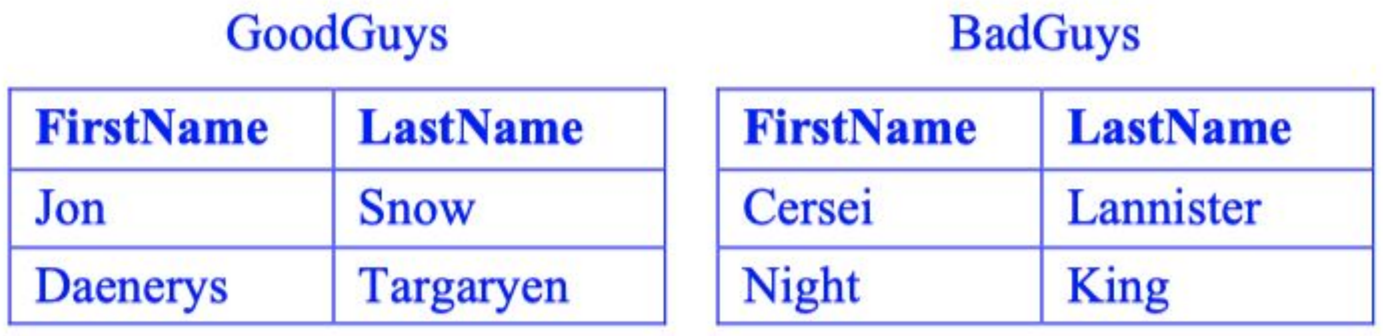
FirstName	LastName
Jon	Snow
Night	King

Set Operators

- Operate on 2 relations
- Union and Difference operations take two input relations, which must be **union-compatible**:
  - They must have the same attribute names in the same order
  - Corresponding attributes must have the same datatype
- Union (∪)
  - R ∪ S where R and S are relations
  - Result: every row which is either in R or S
- Set difference (–)
  - R – S where R and S are relations
  - Result: every row which is in R but <u>not</u> in S
  - Set-difference is **not** symmetrical: R – S ≠ S – R!!!!!!

Example Time

Consider the following tables, GoodGuys and BadGuys:



GoodGuys

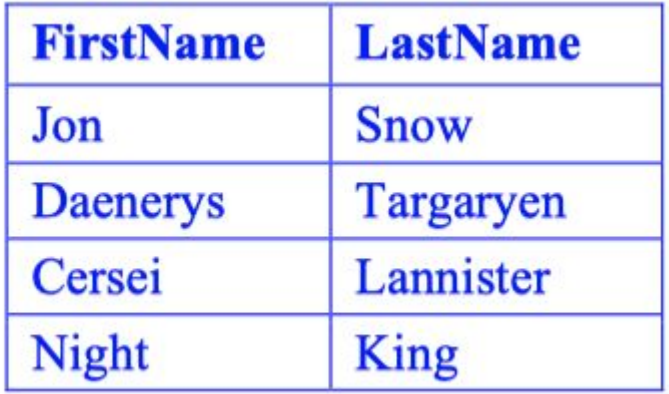
FirstName	LastName
Jon	Snow
Daenerys	Targaryen

BadGuys

FirstName	LastName
Cersei	Lannister
Night	King

What will GoodGuys ∪ BadGuys result in?

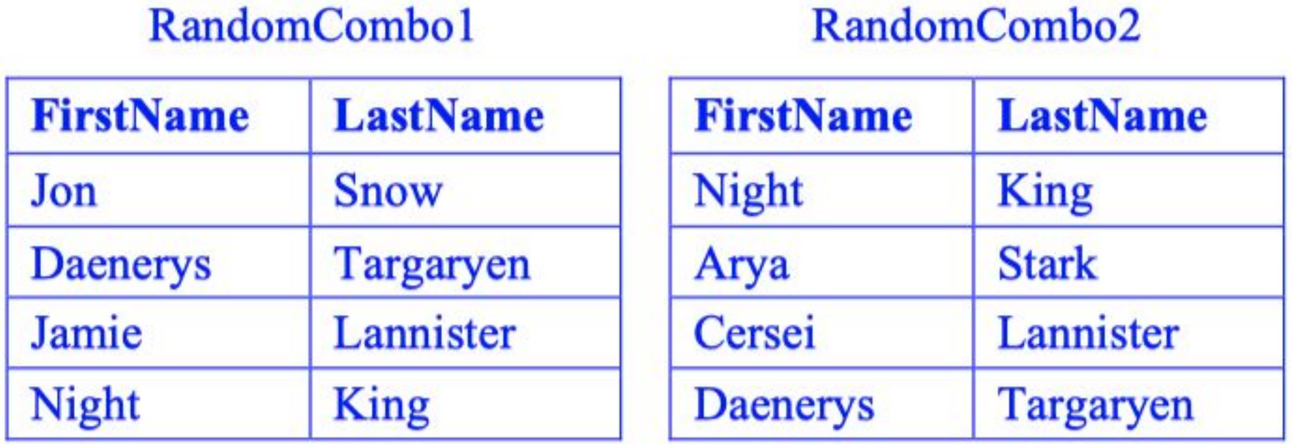
Example Time



FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Cersei	Lannister
Night	King

Example Time

Consider the following tables, RandomCombo1 and RandomCombo2:



RandomCombo1

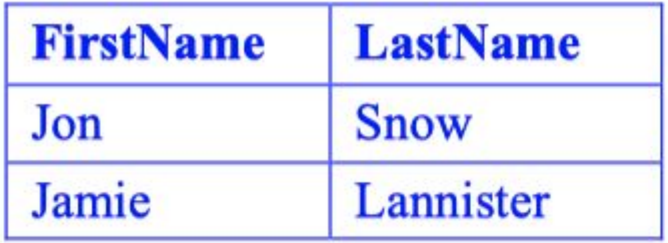
FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

RandomCombo2

FirstName	LastName
Night	King
Arya	Stark
Cersei	Lannister
Daenerys	Targaryen

What will RandomCombo1 – RandomCombo2 result in?

Example Time



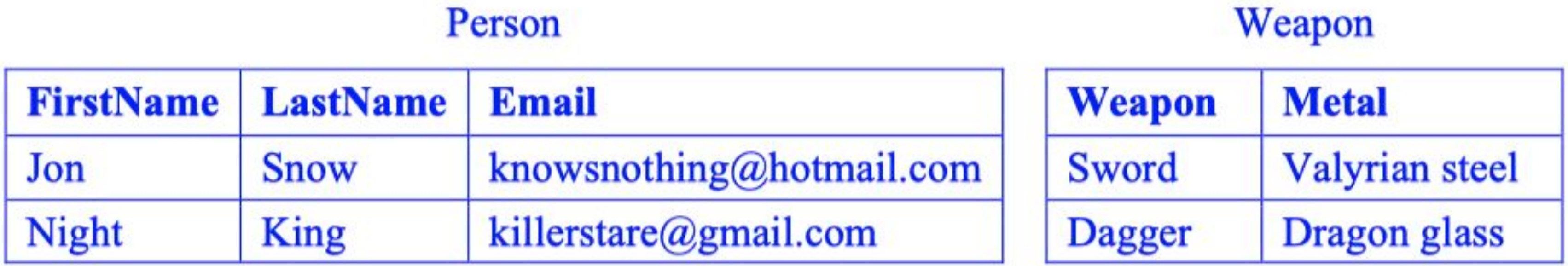
FirstName	LastName
Jon	Snow
Jamie	Lannister

Set Operators

- Cross product (×)
  - R × S where R and S are relations
  - Each row of R pairs with each row of S
  - Resulting schema has all attributes from both relations
  - If some attributes have the same name, rename them by using the renaming operator (will cover later in the tute)

Example Time

Consider the following tables, Person and Weapon:



Person

FirstName	LastName	Email
Jon	Snow	knowsnothing@hotmail.com
Night	King	killerstare@gmail.com

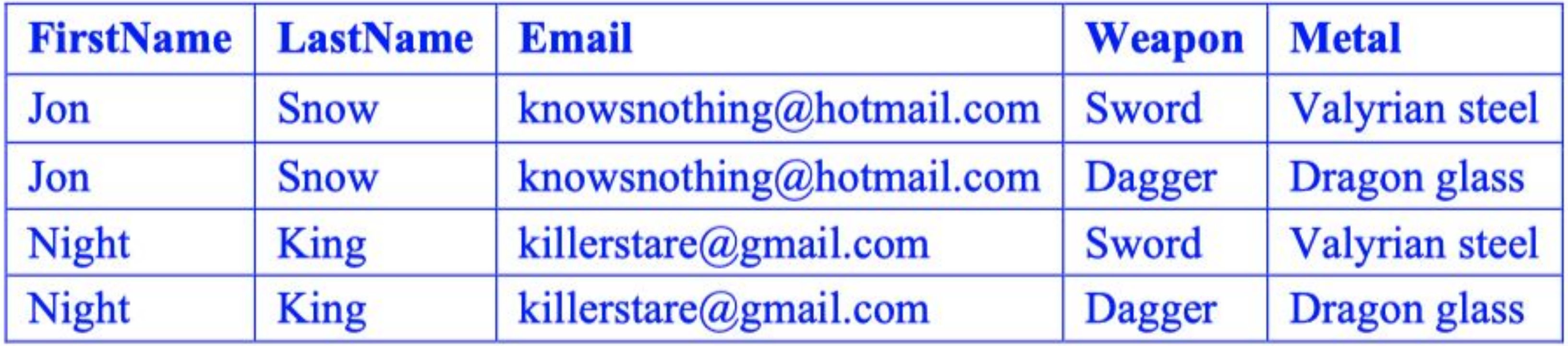
Weapon

Weapon	Metal
Sword	Valyrian steel
Dagger	Dragon glass

What will Person × Weapon result in?

Example Time

Each row of Person pairs with each row of Weapon



FirstName	LastName	Email	Weapon	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Valyrian steel
Jon	Snow	knowsnothing@hotmail.com	Dagger	Dragon glass
Night	King	killerstare@gmail.com	Sword	Valyrian steel
Night	King	killerstare@gmail.com	Dagger	Dragon glass

Compound Operators

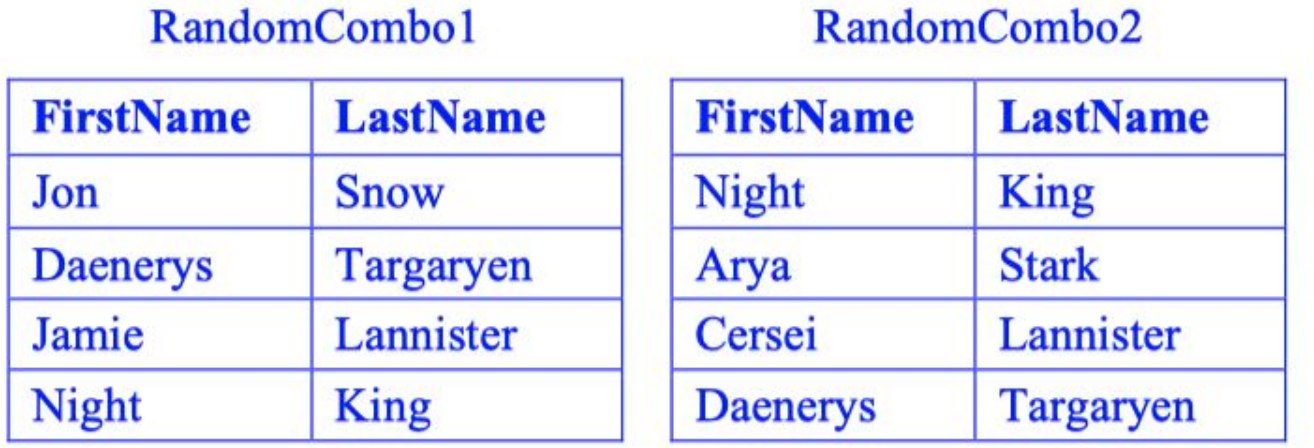
- Useful shorthand
- Can be expressed using basic operators
- What are they?
  - Intersection (∩)
  - Natural join (⋈)
  - Condition join / theta join / inner join ($\bowtie_C$)

Compound Operators

- Intersection (∩)
  - Also a set operator (but not a basic one lol)
  - Takes two input relations
  - Result: a relation containing all tuples which are present in *both* relations
  - Can be expressed using set differences, so the two input relations need to be union-compatible
  - $R \cap S = R - (R - S)$

Example Time

Consider the following tables, RandomCombo1 and RandomCombo2:



RandomCombo1

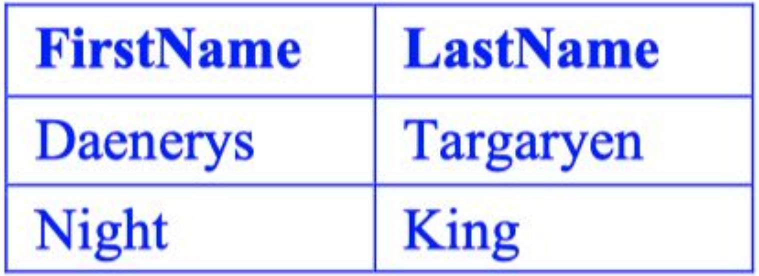
FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

RandomCombo2

FirstName	LastName
Night	King
Arya	Stark
Cersei	Lannister
Daenerys	Targaryen

What will RandomCombo1 ∩ RandomCombo2 result in?

Example Time

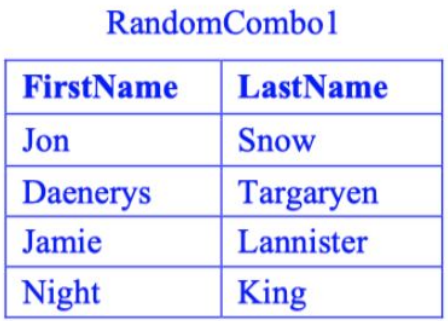


FirstName	LastName
Daenerys	Targaryen
Night	King

Example Time

Also, RandomCombo1 ∩ RandomCombo2

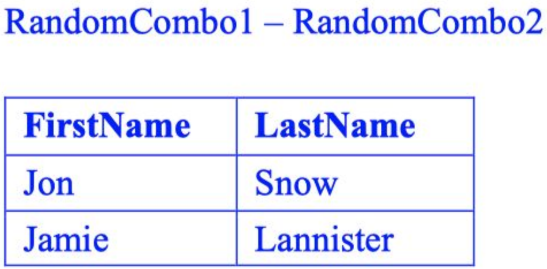
= RandomCombo1 – (RandomCombo1 – RandomCombo2)



RandomCombo1

FirstName	LastName
Jon	Snow
Daenerys	Targaryen
Jamie	Lannister
Night	King

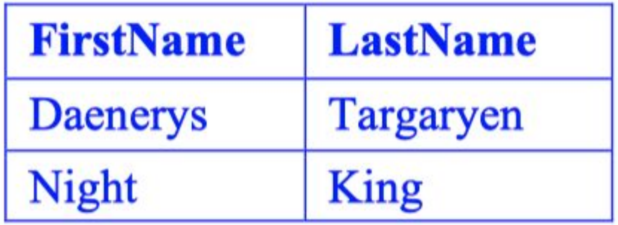
–



RandomCombo1 – RandomCombo2

FirstName	LastName
Jon	Snow
Jamie	Lannister

=



FirstName	LastName
Daenerys	Targaryen
Night	King

Compound Operators

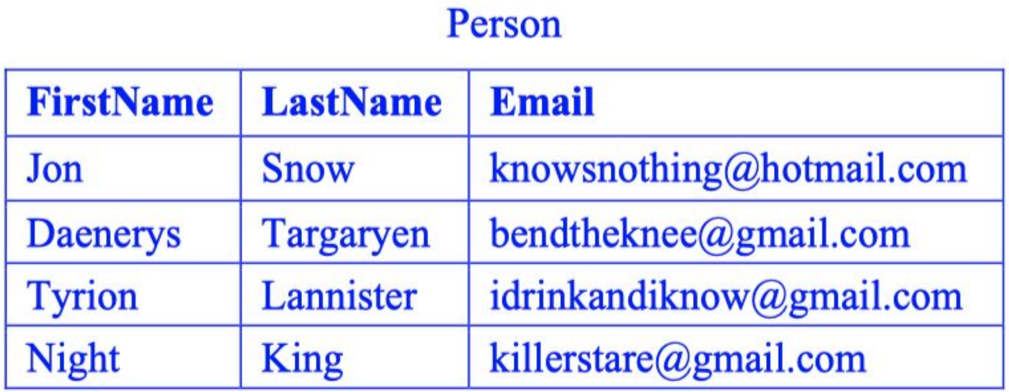
- Joins
  - Compound operators involving cross product, selection and (sometimes) projection
- Natural join (⋈)
  - Often just called “join”
  - Takes two input relations e.g. R and S
  - Result: a new relation, pairing each tuple from R and S where the common attributes are equal
  - Can be broken down:
    - Compute R × S
    - Select rows where attributes that appear in both relations have equal values
    - Project all unique attributes and one copy of the common ones
  - If there are no attributes which have the same name, there’s nothing to select
    - If this is the case, what will

      ```
      SELECT * FROM Relation1 NATURAL JOIN Relation2
      ```

      result in?
    - The cross product! ;)

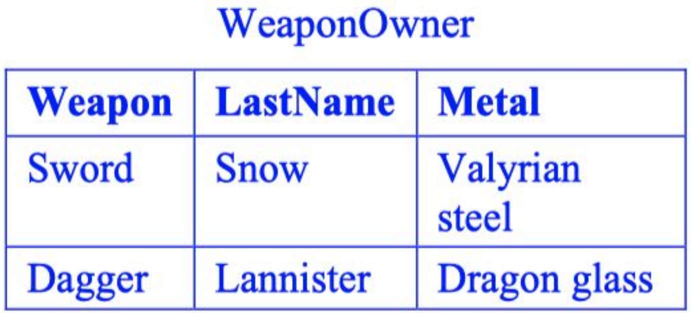
Example Time

Consider the following Person and WeaponOwner tables:



Person

FirstName	LastName	Email
Jon	Snow	knowsnothing@hotmail.com
Daenerys	Targaryen	bendtheknee@gmail.com
Tyrion	Lannister	idrinkandiknow@gmail.com
Night	King	killerstare@gmail.com

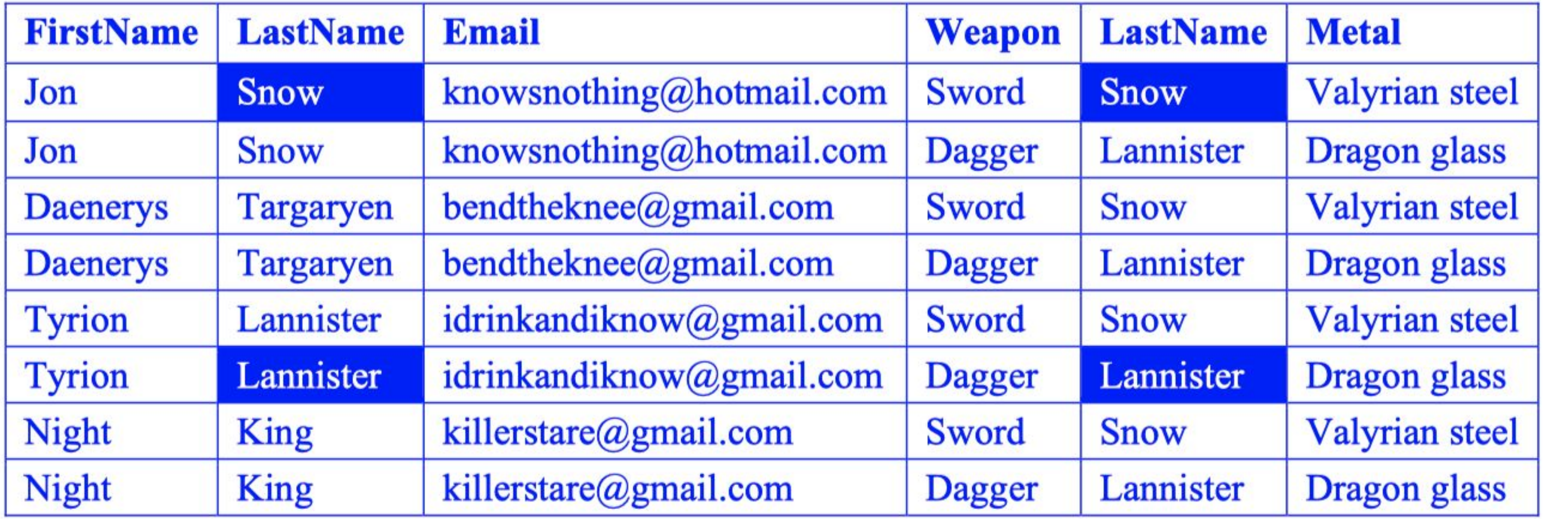


WeaponOwner

Weapon	LastName	Metal
Sword	Snow	Valyrian steel
Dagger	Lannister	Dragon glass

Example Time

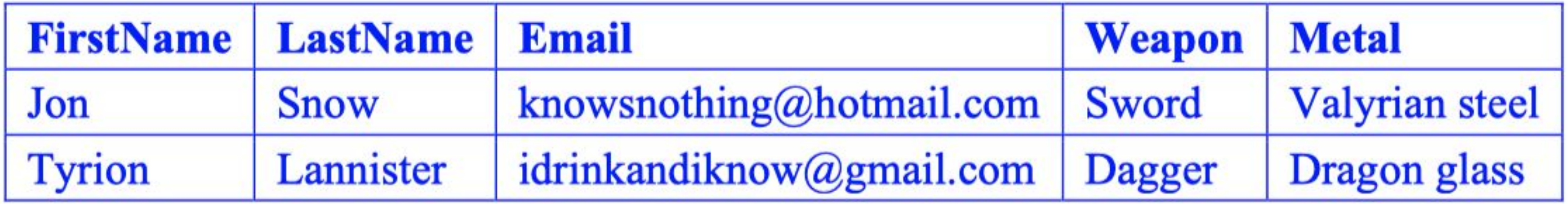
Person × WeaponOwner (intermediate result)



FirstName	LastName	Email	Weapon	LastName	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Snow	Valyrian steel
Jon	Snow	knowsnothing@hotmail.com	Dagger	Lannister	Dragon glass
Daenerys	Targaryen	bendtheknee@gmail.com	Sword	Snow	Valyrian steel
Daenerys	Targaryen	bendtheknee@gmail.com	Dagger	Lannister	Dragon glass
Tyrion	Lannister	idrinkandiknow@gmail.com	Sword	Snow	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Lannister	Dragon glass
Night	King	killerstare@gmail.com	Sword	Snow	Valyrian steel
Night	King	killerstare@gmail.com	Dagger	Lannister	Dragon glass

Example Time

Person $\bowtie$ WeaponOwner (natural join)



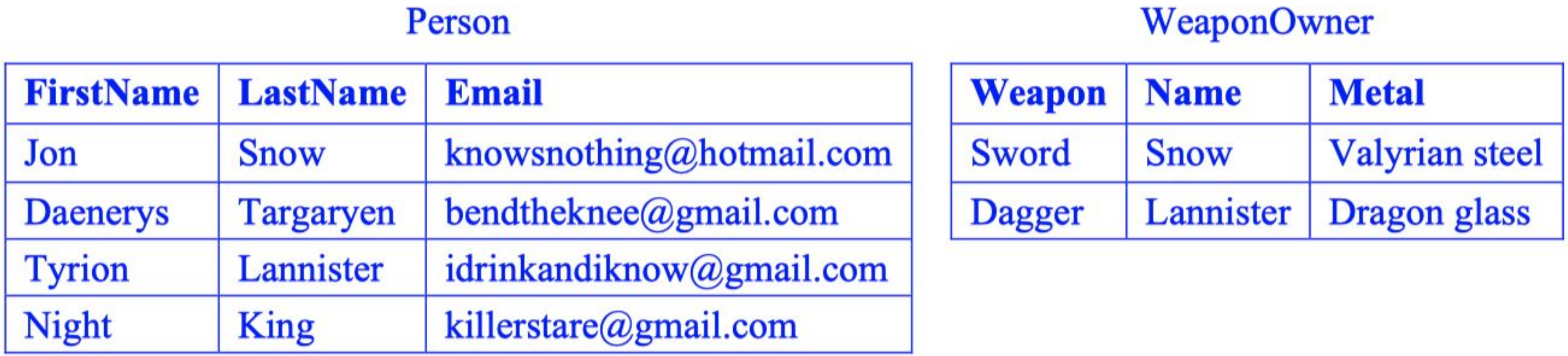
FirstName	LastName	Email	Weapon	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Dragon glass

Compound Operators

- Condition join / theta join / inner join ($\bowtie_C$)
  - Takes two input relations e.g. R and S
  - $R \bowtie_C S$ joins rows from relations R and S such that the Boolean condition C is true
  - Most commonly, C is of the type A = B (“equi-join”)
  - Can be expressed using basic operators
    - $R \bowtie_C S = \sigma_C(R \times S)$
  - Condition C often refers to equality of attributes e.g. $R \bowtie_{r.rid\ =\ s.sid} S$

Example Time

Consider the following tables, Person and WeaponOwner:



Person

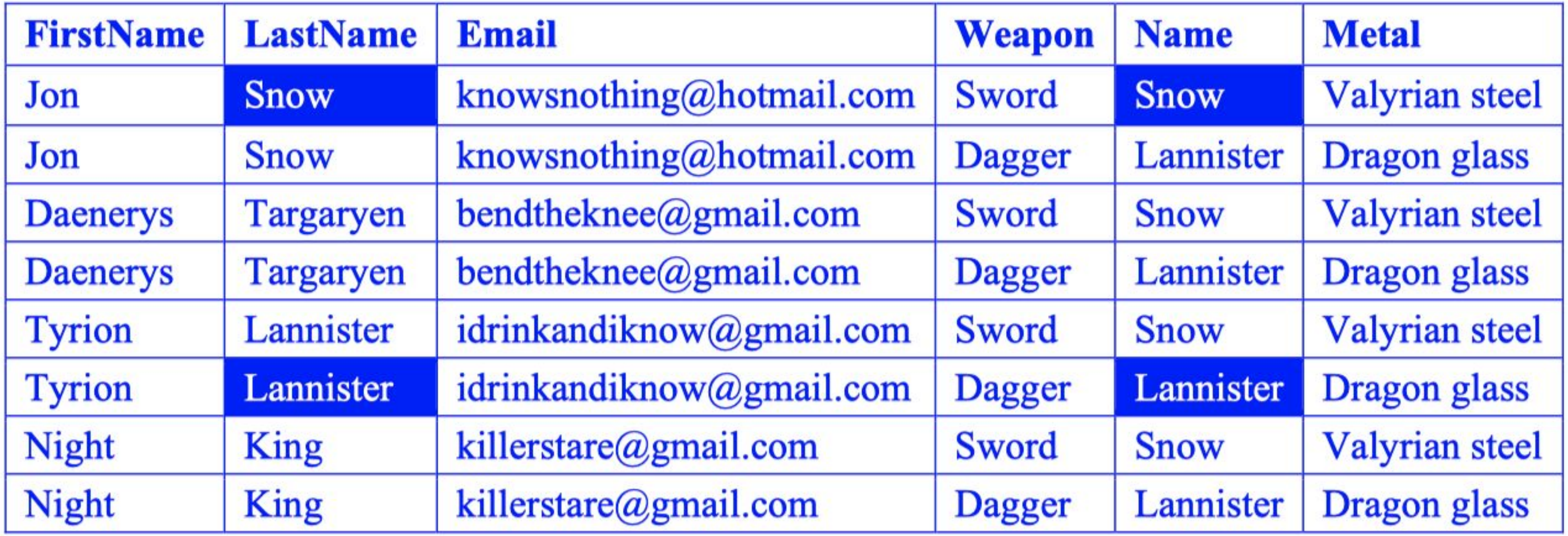
FirstName	LastName	Email
Jon	Snow	knowsnothing@hotmail.com
Daenerys	Targaryen	bendtheknee@gmail.com
Tyrion	Lannister	idrinkandiknow@gmail.com
Night	King	killerstare@gmail.com

WeaponOwner

Weapon	Name	Metal
Sword	Snow	Valyrian steel
Dagger	Lannister	Dragon glass

Example Time

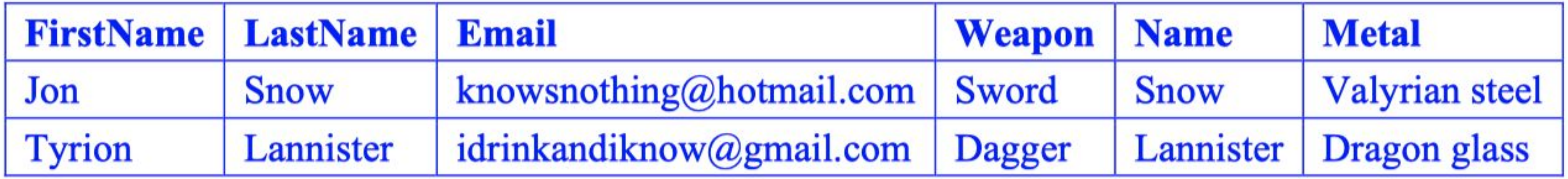
Person × Weapon (intermediate result)



FirstName	LastName	Email	Weapon	Name	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Snow	Valyrian steel
Jon	Snow	knowsnothing@hotmail.com	Dagger	Lannister	Dragon glass
Daenerys	Targaryen	bendtheknee@gmail.com	Sword	Snow	Valyrian steel
Daenerys	Targaryen	bendtheknee@gmail.com	Dagger	Lannister	Dragon glass
Tyrion	Lannister	idrinkandiknow@gmail.com	Sword	Snow	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Lannister	Dragon glass
Night	King	killerstare@gmail.com	Sword	Snow	Valyrian steel
Night	King	killerstare@gmail.com	Dagger	Lannister	Dragon glass

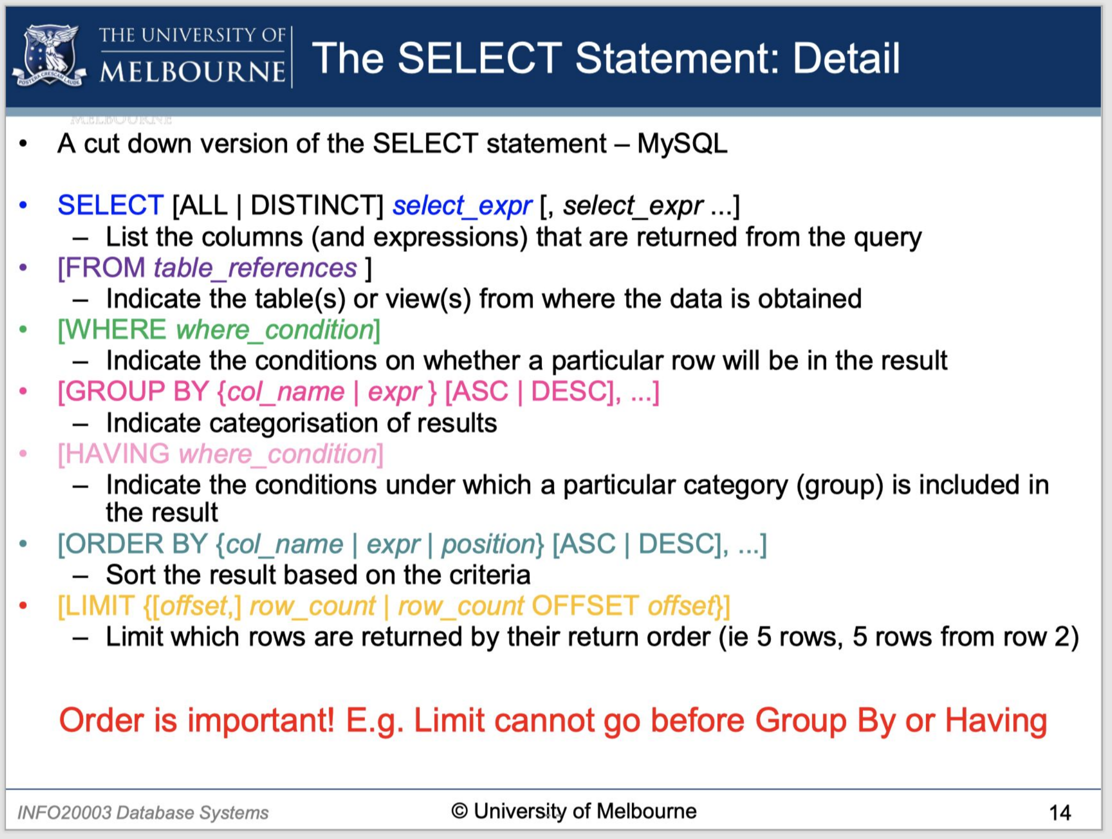
Example Time

$\text{Person} \bowtie_{\text{LastName = Name}} \text{Weapon}$



FirstName	LastName	Email	Weapon	Name	Metal
Jon	Snow	knowsnothing@hotmail.com	Sword	Snow	Valyrian steel
Tyrion	Lannister	idrinkandiknow@gmail.com	Dagger	Lannister	Dragon glass

Relational Algebra and SQL Statements



The SELECT Statement: Detail

- A cut down version of the SELECT statement – MySQL

- SELECT [ALL | DISTINCT] *select_expr* [, *select_expr* ...]
  - List the columns (and expressions) that are returned from the query
- [FROM *table_references*]
  - Indicate the table(s) or view(s) from where the data is obtained
- [WHERE *where_condition*]
  - Indicate the conditions on whether a particular row will be in the result
- [GROUP BY {*col_name* | *expr* } [ASC | DESC], ...]
  - Indicate categorisation of results
- [HAVING *where_condition*]
  - Indicate the conditions under which a particular category (group) is included in the result
- [ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...]
  - Sort the result based on the criteria
- [LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]
  - Limit which rows are returned by their return order (ie 5 rows, 5 rows from row 2)

Order is important! E.g. Limit cannot go before Group By or Having

Relational Algebra and SQL Statements

Solutions [Link to Jupyter Notebook]

Week 5 Lab

- Canvas → Modules → Week 5 → Lab → L05 SQL 1 (PDF)
- Objectives:
  - Install the lab schemas, tables and data
  - Learn SQL (Structured Query Language) SELECT syntax
  - Practise writing SQL queries
  - Join tables using natural and inner joins
- Breakout rooms, “ask for help” button if you need help or have any questions