Week 5 & Week 6

*Normalisation*

* Anomalies:
  + Insertion anomaly: circular dependency (continually adding redundant information, i.e. roomCode -> roomNo, building, campus)
  + Deletion anomaly: losing unintended information (i.e. losing department info when removing an employee)
  + Modification anomaly: info changes in one tuple (i.e. update phone = dept info changes)

*Relational Algebra*

* Formal query language – unary or binary
* Selection (σ), Projection (Π), Cartesian product (×), Union (∪), Set Difference (-)
* Selection:
  + σ predicate(R) = Tuples of relation R that satisfy the condition (predicate)
    - i.e. σ salary < 10000 (Staff) = Select all staff with salary less than 10000
* Projection:
  + Πcol1, col2, … coln(R) = Attributes of relation R, extracting values from tuples of R, less duplicates.
    - i.e. ΠstaffNo, firstName, lastName, salary (Staff) = Produce a list of salaries for all staff, showing only staffNo, firstName, lastName and salary
* Cartesian product:
  + R x S = Concatenation of all tuples of R with all tuples of S
  + σ Client.clientNo = Viewing.clientNo

((∏clientNo,fName, lName(Client)) x (∏clientNo,propertyNo,comment(Viewing)))

* Union-compatibility: Union-compatible if the domain and degree of relations R & S are equal
  + Allows union (R ∪ S – all elements in R, S or both), intersection (R ∩ S – all elements in both only) and set difference (R – S – all elements in R but not S)
    - Union: Πcity(Branch) ∪ Πcity(PropertyForRent) = A list of all cities with branches and/or properties for rent
    - Set difference: Πcity(Branch) – Πcity(PropertyForRent) = A list of all cities with branches but no properties for rent
    - Intersection: Πcity(Branch) ∩ Πcity(PropertyForRent) = A list of all cities with both branches and properties for rent
    - Duplicates are avoided
* Join operations:
  + Theta join, equijoin, outer join, natural join
  + Theta join:
    - R |X|F S – Relation satisfying predicate F in Cartesian product R x S
      * Predicate is of form R.ai θ S.bi, where θ is a comparison operator
      * Also written as σF (R × S)
  + Equijoin:
    - Theta join where θ contains only equals
  + Natural join:
    - R |X|S – Equijoin of relations R x S over all common attributes z.
    - Common attributes have the same name and duplicates are removed
  + Outer joins: Display rows in the result that do not have matching values in the outer column
    - Left Outer Join: Result includes non-matching tuples in the left-hand relation
    - Right Outer Join: Result includes non-matching tuples in the right-hand relation
    - Full Outer Join: Result includes non-matching tuples on both sides
  + Division:
    - R ÷ S – Relation of a set of tuples from R that match with every attribute in S
    - T1 ← ΠC(R)
    - T2 ← ΠC((S x T1) – R)
    - T ← T1 – T2
  + Aggregate:
    - ℑAL(R)
    - Main functions: AVG(), SUM(), COUNT(), MIN(), MAX()
      * eg. how many properties cost more than $350 to rent?
        + ρR(myCount) ℑCOUNT propertyNo (σrent > 350(PropertyForRent))
    - GAℑAL(R) – group tuples of R by grouping attributes (GA), then applying the attribute function list (AL) to each to define a new relation
      * eg. find the number of staff and the total of their salaries in each branch.
        + ρR(Branch, Staff, Salary) BranchℑCOUNT(Staff), SUM(Salary)(Staff)