# Solutions to Tutorial 9 – Views and Denormalisation

## Question 1

1. Views:
   1. Create View Manager3

AS Select \*

From Staff

Where branchNo = ‘B003’;

* 1. Create View Staff3

AS Select staffNo, name, position

From Manager3;

(Could also create from base table, but this way shows creation from another view.)

* 1. Create View StaffPropertyCount

AS Select staffNo, COUNT(\*)

From Property

Group By staffNo;

1. Note that ‘city’ is functionally dependent on ‘postcode’ in both Branch and Property tables. Hence, there is a transitive dependency and the tables are not in 3NF. If the full address of a branch is normally retrieved with its other details, this may be a useful denormalisation.

## Question 2

1. Add the non-key column p\_name from the Pirate table to the Pillage table. This eliminates the join from the required query. This is case 2 in our lecture. It can also be seen as similar to case 4; the Pillage table has been created to represent a ternary relationship between the other three entities. As a result it is on the many side of a many-to-one relationship with each of the other tables and many queries can be made faster by adding non-key columns into Pillage.
2. There are no one-to-one relationships, so case 1 does not apply. Every indirect relationship is many-to-many e.g. pirates have a many-to-many relationship to the boats they have used to pillage. So there is no use for case 3. Case 5 occurs when there is a one-to-many relationship with a known upper bound (3 in the lecture example) on the many part. There are no such cases in the database.