

Widening Access premium – extraction of data from LLWR records

```
*****
* Algorithm for institutions to use with PROGRAMME dataset merged with ACTIVITY dataset *
* Example for use with 2008/09 LLWR data for 2010/11 funding *
*****
```

```
*****
* Please also refer to guidance given in circular Higher Education Data Requirements *
* 2008/09 (W09/12HE) *
*****
```

```
*****
* 1. Extract eligible students *
* (variables taken from LLWR tables are in upper case) *
*****
```

```
*****
* Notes *
* ne means not equals *
* proc means procedure *
* =: means begins with *
*****
```

```
data popn;
set llwr;
```

```
where LA11='2' and
      LP17 in ('51','52','53','55','56','57','58','59');
```

```
*****
* 2. Include students active between 1 August 2008 and 31 July 2009 *
*****
```

```
if LA09<='31Jul2009'd and
   (LA30>='1Aug2008'd or LA30='');
```

```
*****
* Merge the student data file selected above with a file containing students' postcodes *
* The postcode file (postmap) contains two fields entitled low_affluence_ind and *
* community_first_ind. If a student's postcode is located within a low affluence area *
* or a community first area the value of the corresponding field will be 1. *
* If students have more than one instance with differing postcodes then *
* preference is given to the postcode which maps to the low affluence area *
*****
```

```
data widacc;
merge popn postmap(keep=LA02 LA03 LA low_affluence_ind community_first_ind);
by LA02 LA03;
```

```
*****
* data is sorted by a field which indicates whether a student has a low affluence postcode, *
* if a student has more than one instance, the instance with a low affluence postcode will *
* be listed before an instance without a low affluence postcode *
*****
```

```
*****
* nodupkey means delete any duplicates having the same key values of *
* learner identifier LA02 and provider identifier LA03 *
* the record that appears first will be kept *
*****
```

```
proc sort nodupkey;
by LA02 LA03;
```

```
*****
* 3. Identify students eligible for low affluence or community first element *
* using low_affluence_ind and community_first_ind *
*****
```

```
if low_affluence_ind=1 then wa=1;
else wa=0;
```

```
if community_first_ind=1 then cf=1;
else cf=0;
```

```
*****
* 4. Identify students eligible for non-traditional highest qualification on entry element *
* using type LP21 and level LP22 of highest qualification on entry. *
*****
```

```
if LP21 in ('20','31','32','35','40','50','71','72','98') and
LP22 in ('E','0','1','2','3') then non_trad=1 ;
```

```
else if LP21 in ('10','30') and
LP22 in ('E','0','1','2') then non_trad=1 ;
```

```
*****
** indicator to count number of students who are eligible for premium *
** i.e. all students who meet initial criteria before low affluence, *
** community first and non-traditional highest qualification on entry *
** elements are applied *
*****
```

```
total=1;
```

```
*****
* 5. Summarise by provider to get total number of eligible students *
* and numbers allocated each element of the premium *
*****
```

```
proc summary;
by LA03;
var total wa cf non_trad;
output out=outwa sum= total wa cf non_trad;
```