```sas
/* This code was used to remove the duplicates from the Dublinbikes API data
    and to calculate the activity or changes in the number of available bikes
    and stands in a given station over time. */

/* Connect to the CAS Server - to access the files in the public folder */
cas mySession2 host="localhost" port=5570 sessopts=(caslib=casuser timeout=18000);
caslib _all_ assign;

/* Increase system data limit to prevent restrictions */
options CASDATALIMIT="ALL";

/* Duplicating and sorting the API bikes data by address and then by the update time */
proc sort data=PUBLIC.ZZZ_BIKE_COMBINED_DATA out=work.sortDS equals;
    by address last_update;
run;

/* Removing duplicate rows from the dataset */
proc sort data = WORK.SORTDS out = work.SORTDS1 NODUPKEY;
    by _all_;
run;

/* Re-sorting the newly duplicate free API bike data */
proc sort data = WORK.SORTDS1 out = work.SORTDS2 equals;
    by address last_update;
run;

/* Going through the data for each station and creating new date changes column */
data WORK.SORTDS3;
    set WORK.SORTDS2;
    by address last_update;
    Date_Changes = lag(last_update);    /* Copying and moving the datemtime down 1 row */
    if first.address then do;           /* Set the 1st value for a new station to 0 */
        Date_Changes = 0;
    end;
    format Date_Changes datetime.;      /* Reformatting the new column to the SAS datetime format */
run;

/* Calculating time difference between updates per station */
data WORK.SORTDS4;
    set WORK.SORTDS3;
    by address last_update;
    start = last_update;            /* Creating copy of the update column */
    prev = Date_Changes;            /* Creating copy of the date changes column */
    Date_Difference = start - prev; /* Calculating difference in seconds between updates */
    if first.address then do;       /* Taking difference to be 0 at first appearnce of a station */
        Date_Difference = 0;
    end;
run;

/* Binary indicator for a gap over 1 hour in the update time */
data WORK.SORTDS5;
    set WORK.SORTDS4;
    by address last_update;
    Time_missed = 0;                    /* Create new column of zeros */
    if Date_Difference > 3600 then do;  /* Change 0 to 1 if over 1 hour gap between update times */
        Time_missed = 1;
    end;
run;

/* Caluculating the change in bikes and stands over time */
data WORK.SORTDS7;
    set WORK.SORTDS5;
    by address;
    /* Dif - gets the difference between the current value and previous value */
    Avail_Bike_Changes = dif(available_bikes);
    Avail_Bike_Stand_Changes = dif(available_bike_stands);
    if first.address or Time_missed = 1 then do;    /* Gap or 1st appearance of station value reset to 0 */
        Avail_Bike_Changes = 0;
        Avail_Bike_Stand_Changes = 0;
    end;
run;

/* Inserting new column as index of row per station */
data WORK.TDS5;
    set WORK.SORTDS7;
    by address last_update;
    if first.address then x=1;  /* Reset count to 1 for 1st value in each station */
    else x+1;                   /* Increase value by 1 for each row */
run;

/* Re-sort data by address and then by most recent update first */
proc sort data = WORK.TDS5;
```

```
    by address descending x;
run;

/* Creating new lagged columns */
data WORK.TDS6;
    set WORK.TDS5;
    by address descending x;
    l = lag(x);     /* Copy column and move it down 1 row */
    X_Avail_Bike_Changes = lag(Avail_Bike_Changes);
    X_Avail_Bike_Stand_Changes = lag(Avail_Bike_Stand_Changes);
    if first.address then do;   /* Set the 1st values for a new station to 0 */
        l = .;
        X_Avail_Bike_Changes = 0;
        X_Avail_Bike_Stand_Changes = 0;
    end;
run;

/* Re-sort data by address and then by first update */
proc sort data = WORK.TDS6;
    by address x;
run;

/* Calculate increase or decrease in bikes and stands per update */
data WORK.SORTDS8;
    set WORK.TDS6;
    /* Check if change positive and greater than 0 and save value to increase column */
    if X_Avail_Bike_Changes > 0 then Avail_Bikes_Increase = X_Avail_Bike_Changes;
    /* Otherwise set to 0 */
    else Avail_Bikes_Increase = 0;
    /* Check if change negative and less than 0 and save the absolute value to decrease column */
    if X_Avail_Bike_Changes < 0 then Avail_Bikes_Decrease = abs(X_Avail_Bike_Changes);
    /* Otherwise set to 0 */
    else Avail_Bikes_Decrease = 0;
    if X_Avail_Bike_Stand_Changes > 0 then Avail_Stands_Increase = X_Avail_Bike_Stand_Changes;
    else Avail_Stands_Increase = 0;
    if X_Avail_Bike_Stand_Changes < 0 then Avail_Stands_Decrease = abs(X_Avail_Bike_Stand_Changes);
    else Avail_Stands_Decrease = 0;
run;

/* Remove unnecessary columns */
data WORK.SORTDS9;
    set WORK.SORTDS8 (drop = Date_Changes Date_Difference 'start'n prev Time_missed x l Avail_Bike_Changes Avail_Bike_Stand_Changes);
run;

/* Renaming and promoting the data with added activity variables to the public server */
proc casutil;
    load data=WORK.SORTDS9 outcaslib="public" promote
    casout="ZZZ_Bike_Activity_Data";
run;
```