HSAPS DQR Generation Process Documentation

2024-10-29

DQR generation process begins with importing data sets generated from data cleaning process. Details regarding Home Safe data cleaning process can be found here as well: Home Safe Data Cleaning and Functions Documentation

Initial Preparation

Code Description: DQR Script

```
hsaps = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/^hsaps_cleaned_2024-08-30.xlsx",sep.names = " ")
hsaps =
merge(hsaps,quarter[which(!is.na(quarter$Report_Month)),c(1,3)],all.x=T)
hsaps = hsaps[,c(2:6,1,7:grep("Quarter",names(hsaps)))]

quarter = read.csv("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/R Scripts/Quarter_Period.csv")

hsaps_new_data_exclude = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/Cleaning In Progress/hsaps_new_data_exclude.xlsx",sep.names = " ")
hsaps_new_data_exclude = hsaps_new_data_exclude[-
grep("remove",tolower(hsaps_new_data_exclude$`Remove Case`)),]

hsaps_noncumulative = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/Cleaning In Progress/hsaps_dropped.xlsx",sep.names = " ")
hsaps_noncumulative =
merge(hsaps_noncumulative,quarter[,c("Report_Month","Quarter")],all.x=T)
#remove possible typo list from the dropped record list
flag = c()
for(i in 1:nrow(hsaps_noncumulative)){

if((tolower(paste0(hsaps_noncumulative$Reporting_Agency[i],hsaps_noncumulative$`First Name`[i],hsaps_noncumulative$`Date of Birth`[i])) %in%
tolower(paste0(hsaps_new_data_exclude$Reporting_Agency,hsaps_new_data_exclude$`First Name`,hsaps_new_data_exclude$`Date of Birth`))) |
(tolower(paste0(hsaps_noncumulative$Reporting_Agency[i],hsaps_noncumulative$`Date of Birth`[i],hsaps_noncumulative$`Last Name`[i])) %in%
tolower(paste0(hsaps_new_data_exclude$Reporting_Agency,hsaps_new_data_exclude$`Date of Birth`,hsaps_new_data_exclude$`Last Name`))) |
```

```r
(tolower(paste0(hsaps_noncumulative$Reporting_Agency[i],hsaps_noncumulative$`
First Name`[i],hsaps_noncumulative$`Last Name`[i])) %in%
tolower(paste0(hsaps_new_data_exclude$Reporting_Agency,hsaps_new_data_exclude
$`First Name`,hsaps_new_data_exclude$`Last Name`)))){
    flag = c(flag,i)
  }
}
if(length(flag)>0){
  hsaps_noncumulative = hsaps_noncumulative[-flag,]
}

hsaps_no_start_date_duplicate = read.xlsx("//cdss/feed/Central
Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports
to date/Cleaning In Progress/hsaps_no_start_date_duplicate.xlsx",sep.names =
" ")
hsaps_no_start_date_duplicate = hsaps_no_start_date_duplicate[-
grep("remove",tolower(hsaps_no_start_date_duplicate$`Remove Case`)),]
hsaps_no_start_date_duplicate =
hsaps_no_start_date_duplicate[(duplicated(hsaps_no_start_date_duplicate[c("Re
porting_Agency","Report_Month","First Name","Last Name")])
&
duplicated(hsaps_no_start_date_duplicate[c("Reporting_Agency","Report_Month",
"First Name","Last Name")],fromlast=T)
&
duplicated(hsaps_no_start_date_duplicate[c("Reporting_Agency","Report_Month",
"First Name","Last Name")],fromlast=T)
&
duplicated(hsaps_no_start_date_duplicate[c("Reporting_Agency","Report_Month",
"First Name","Last Name")],fromlast=T)),]
```

- Importing hsaps, hsaps_new_data_exlude, hsaps_noncumulative, and hsaps_no_start_date_duplicate; these tables were generated in the process of data cleaning and processing.
- **quarter**: the list of quarter labels paired up with Report_Month variable from Home Safe data and Quarter Start variable from HDAP data, to indicate the quarter where the data came from for specific data flags.
- For hsaps_noncumulative table, whatever participant data that are part of hsaps_new_data_exclude table was removed as for these participants, we chose to keep the older data while flagging the newest ones.
- For hsaps_new_data_exclude and hsaps_no_start_date_duplicate tables, all the rows with "Remove" responses in Remove Case variable were removed, so that it won't get flagged falsely.
- Additionally, hsaps_no_start_date_duplicate table was de-duplicated, so the DQR won't flag the same information twice.

Generating Key Information Discrepant List Table

Next, the code generates one of the DQR tables that contains the list of data from hsaps_new_data_exclude. This table will be stored in “Key Information Discrepancies” sheet in the DQR.

Code Description: Functions

The function, **error_table_generation**, was used to perform this task. It creates a data table (**dt**) fills in the values from function inputs (**error_type, item, item_value, error_desc, priority**), and then appends to the existing table (**error_dt**). The code is shown below:

```r
error_table_generation = 
function(error_dt,dt,error_type,item,item_value,error_desc,priority){
  dt$`Error Type`=error_type
  dt$Item = item
  dt$`Item Value`=item_value
  dt$`Error Description`=error_desc
  dt$`Additional Information/Instructions`=''
  dt$`Error Priority`=priority
  dt$`Fixed (Y/N)`=''
  dt$`Comments or questions from grantees`=''
  error_table = rbind(error_dt,dt)
  return(error_table)
}
```

Code Description: DQR Script

The relevant section of the code within the DQR generation script was shown below.

```r
#data set of participant key info discrepant
hsaps_participant_discrepant = 
unique(hsaps_new_data_exclude[,c("Reporting_Agency","First Name","Last 
Name","Date of Birth","Case Start Date")])
names(hsaps_participant_discrepant)[-1] = 
paste0(names(hsaps_participant_discrepant)[-1],"_new")
# hsaps_participant_discrepant = 
unique(merge(hsaps_participant_discrepant,hsaps[,c(1:3,5)],by.x=c("Reporting_
Agency","Last Name_new","First Name_new"),by.y=c("Reporting_Agency","Last 
Name","First Name"),all.x=T))
hsaps_participant_discrepant = 
unique(merge(hsaps_participant_discrepant,hsaps[,c(1:4,grep("Quarter",names(h
saps)))],by.x=c("Reporting_Agency","Last Name_new","First 
Name_new"),by.y=c("Reporting_Agency","Last Name","First Name"),all.x=T))
hsaps_participant_discrepant = 
unique(merge(hsaps_participant_discrepant,hsaps[,c(1:5,grep("Quarter",names(h
saps)))],by.x=c("Reporting_Agency","First Name_new","Date of Birth_new","Case 
Start Date_new"),by.y=c("Reporting_Agency","First Name","Date of Birth","Case 
Start Date"),all.x=T))
```

```
hsaps_participant_discrepant = 
unique(merge(hsaps_participant_discrepant,hsaps[,c(1:5,grep("Quarter",names(h
saps)))],by.x=c("Reporting_Agency","Last Name_new","Date of Birth_new","Case
Start Date_new"),by.y=c("Reporting_Agency","Last Name","Date of Birth","Case
Start Date"),all.x=T))

hsaps_key_info_discrepant = data.frame()
hsaps_key_info_discrepant = 
error_table_generation(hsaps_key_info_discrepant,hsaps_participant_discrepant
[which(hsaps_participant_discrepant$`First
Name`!=hsaps_participant_discrepant$`First
Name_new`|(is.na(hsaps_participant_discrepant$`First
Name_new`)&!is.na(hsaps_participant_discrepant$`First
Name`))),c(1,grep("First
Name_new",names(hsaps_participant_discrepant)),grep("Last
Name_new",names(hsaps_participant_discrepant)),grep("Date of
Birth_new",names(hsaps_participant_discrepant)),grep("Case Start
Date_new",names(hsaps_participant_discrepant)))],"Invalid","First
Name",hsaps_participant_discrepant$`First
Name`[which(hsaps_participant_discrepant$`First Name`!=
hsaps_participant_discrepant$`First
Name_new`|(is.na(hsaps_participant_discrepant$`Last
Name_new`)&!is.na(hsaps_participant_discrepant$`Last Name`)))],paste0("The
first name for this participant from the most recent report was different
from that from
",hsaps_participant_discrepant$Quarter[which(hsaps_participant_discrepant$`Fi
rst Name`!=hsaps_participant_discrepant$`First
Name_new`|(is.na(hsaps_participant_discrepant$`First
Name_new`)&!is.na(hsaps_participant_discrepant$`First Name`)))],"
report."),"High")
hsaps_key_info_discrepant = 
error_table_generation(hsaps_key_info_discrepant,hsaps_participant_discrepant
[which(hsaps_participant_discrepant$`Last
Name`!=hsaps_participant_discrepant$`Last
Name_new`|(is.na(hsaps_participant_discrepant$`Last
Name_new`)&!is.na(hsaps_participant_discrepant$`Last Name`))),c(1,grep("First
Name_new",names(hsaps_participant_discrepant)),grep("Last
Name_new",names(hsaps_participant_discrepant)),grep("Date of
Birth_new",names(hsaps_participant_discrepant)),grep("Case Start
Date_new",names(hsaps_participant_discrepant)))],"Invalid","Last
Name",hsaps_participant_discrepant$`Last
Name`[which(hsaps_participant_discrepant$`Last
Name`!=hsaps_participant_discrepant$`Last
Name_new`|(is.na(hsaps_participant_discrepant$`Last
Name_new`)&!is.na(hsaps_participant_discrepant$`Last Name`)))],paste0("The
last name for this participant from the most recent report was different from
that from
",hsaps_participant_discrepant$Quarter.y[which(hsaps_participant_discrepant$`
Last Name`!=hsaps_participant_discrepant$`Last
Name_new`|(is.na(hsaps_participant_discrepant$`Last
Name_new`)&!is.na(hsaps_participant_discrepant$`Last Name`)))],"
report."),"High")
hsaps_key_info_discrepant =
error_table_generation(hsaps_key_info_discrepant,hsaps_participant_discrepant
[which(hsaps_participant_discrepant$`Date of
Birth`!=hsaps_participant_discrepant$`Date of
Birth_new`|(is.na(hsaps_participant_discrepant$`Date of
Birth_new`)&!is.na(hsaps_participant_discrepant$`Date of
Birth`))),c(1,grep("First
Name_new",names(hsaps_participant_discrepant)),grep("Last
Name_new",names(hsaps_participant_discrepant)),grep("Date of
Birth_new",names(hsaps_participant_discrepant)),grep("Case Start
Date_new",names(hsaps_participant_discrepant)))],"Invalid","Date of
Birth",hsaps_participant_discrepant$`Date of
Birth`[which(hsaps_participant_discrepant$`Date of
Birth`!=hsaps_participant_discrepant$`Date of
Birth_new`|(is.na(hsaps_participant_discrepant$`Date of
Birth_new`)&!is.na(hsaps_participant_discrepant$`Date of
Birth`)))],paste0("The date of birth for this participant from the most
recent report was different from that from
",hsaps_participant_discrepant$Quarter.x[which(hsaps_participant_discrepant$`
Date of Birth`!=hsaps_participant_discrepant$`Date of
Birth_new`|(is.na(hsaps_participant_discrepant$`Date of
Birth_new`)&!is.na(hsaps_participant_discrepant$`Date of Birth`)))],"
report."),"High")
if(length(which(!(hsaps_participant_discrepant$`Case Start
Date`%in%hsaps$`Case Start Date`)))>0){
  hsaps_key_info_discrepant =
error_table_generation(hsaps_key_info_discrepant,hsaps_participant_discrepant
[which(!(hsaps_participant_discrepant$`Case Start Date`%in%hsaps$`Case Start
Date`)),c(1,grep("First
Name_new",names(hsaps_participant_discrepant)),grep("Last
Name_new",names(hsaps_participant_discrepant)),grep("Date of
Birth_new",names(hsaps_participant_discrepant)),grep("Case Start
Date_new",names(hsaps_participant_discrepant)))],"Invalid","Case Start
Date",hsaps_participant_discrepant$`Case Start
Date`[which(!(hsaps_participant_discrepant$`Case Start Date`%in%hsaps$`Case
Start Date`))],paste0("The case start date for this participant from the most
recent report was discrepant from that from the previous report."),"High")
}
hsaps_key_info_discrepant$`Additional
Information/Instructions`[grep("name",tolower(hsaps_key_info_discrepant$Item)
)] = "Please confirm that this change was intentional, and if it was a typo,
please fix it in the newest report. Please keep the naming convention
consistent as much as possible between the quarters."
hsaps_key_info_discrepant$`Additional Information/Instructions`[grep("date of
bir",tolower(hsaps_key_info_discrepant$Item))] = "Please confirm that this
change was accurate. if it was a mistake, please address it in the newest
report."
hsaps_key_info_discrepant$`Additional
```

```
Information/Instructions`[grep("case",tolower(hsaps_key_info_discrepant$Item)
)] = "Please confirm that this change was accurate. If it was a mistake,
please address it in the newest report."
```

- **hsaps_participant_discrepant**: list of merged data between participants' discrepant information from last quarter and the most current quarter. It shows which variable is discrepant for each row.
- **hsaps_key_info_discrepant**: the final DQR table generated from hsaps_participant_discrepant table and error_table_generation function

Data Validation Checks

The code then moves on with data validation checks and flagging missing or invalid responses.

Background/Decisions

We initially flagged data errors from all the rows of the cleaned hsaps data, which includes some data rows from reports in the previous quarters (as we flag out the newest rows with key information discrepancies). However, the grantees could not identify data errors of the data from the older reports (as what they only have is the newest data). Therefore, to resolve the inconvenience grantees experienced (letting us know and feeling confused why the DQR flagged issues when the newest data responses were not missing nor invalid), even though it was on their end that caused this inconvenience (ex. updating the participants' key information), we decided to relieve them by flagging the data issues from only the newest data they submitted.

Missing or Out-of-ACIN Responses

The code goes through all the data elements with drop down options, compares the entries with the responses listed from ACIN, and flags those that are missing or not compliant with options listed in ACIN.

```
##data auditing for each variable with drop down categories
#make sure you update Report_Month below for the corresponding quarter
hsaps_dqr = subset(hsaps,Report_Month=="2024-04-01")
#setting up categories
na = c("client doesn't know","client refused","data not collected")
gi = c("female","male","gender other than
female/male","transgender","questioning","unknown/not provided",na)
race1 = tolower(c("american indian/alaskan Native/Indigenous","Asian/asian
american","Asian","american indian/alaskan native","Black/African
American/African","Native Hawaiian/Pacific Islander","pacific islander/native
hawaiian","other","White",na,"Unknown/Not provided","black/african
american"))
ethn = c("non-
hispanic/latin(a)(o)(x)","hispanic/latin(a)(o)(x)","cuban","puerto
rican","mexican/chicano","other hispanic/latino","not
```

```
hispanic/latino","unknown/not provided",na)
ms = c("married","not married/living with
partner","divorced","separated","widowed","never married","data not
collected","unknown/not provided")
so =
c("straight/heterosexual","gay/lesbian","bisexual","questioning",na,"unknown/
not provided")
pl =
c("english","spanish","mandarin/cantonese","vietnamese","tagalog","korean","o
ther","data not collected")
veteran_cl3yrs_evics_discharge = c("yes","no",na,"unknown")
ls_entry = c("homeless","temporary housing","temporary- residential
program","permanent- residential program","rent leaseholder","owner","other
permanent housing","other","data not collected","with others rent","with
others no rent","homeless unsheltered","homeless sheltered","hotel no
rights","hotel with rights","owner lives alone","owner with others
rent","owner with others no rent","skilled nursing facility","residential
care facility","board and care facility")
abuse_neglect_prevaps_oldcols = c("yes","no","unknown")
rs = c("professional service provider","educator","financial service
provider","law enforcement","medical personnel","mental health
personnel","mental health","institutional employee","social
worker","unknown","other community professional","community
professional","clergy","self","family member","no relationship","anonymous")
ces = c("no","yes - before home safe","yes - after home safe","yes - prior to
home safe")
interv_type = c("no intervention","enhanced case management","mortgage
payment","rent back-pay","rent payment","housing navigation","temporary
housing","emergency shelter","security deposit","utilities","relocation
assistance/storage","home habitability","legal services","caregiver
services/respite care","other","deep cleaning/hoarding assistance","deep
cleaning","home repair","external housing navigation")
ls_exit_fu_ls = c(ls_entry,"deceased","unknown","not exited")
fu_method = c("unable to verify","hmis","aps system","verified - program
staff","verified - external staff")
fu_homeless = c("yes","no","client doesn't know","unknown","not applicable")

#applying the function
hsaps_drop_down_off = data.frame()
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Gen
der Identity"],gi,"Gender Identity")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Rac
e 1"],race1,"Race 1")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Eth
nicity"],ethn,"Ethnicity")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Cur
rent Marital Status"],ms,"Marital Status")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Sex
ual Orientation"],so,"Sexual Orientation")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Pre
ferred Language"],pl,"Preferred Language")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Vet
eran Status"],veteran_cl3yrs_evics_discharge,"Veteran Status")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Liv
ing Situation Upon Entry"],ls_entry,"Living Situation Upon Entry to Home
Safe")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Pre
vious Evictions or Foreclosures"],veteran_cl3yrs_evics_discharge,"Previous
Evictions or Foreclosures")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Cur
rent Eviction or Foreclosures"],veteran_cl3yrs_evics_discharge,"Current
Evictions or Foreclosures")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Dis
charge from Institution in the Last Six
Months"],veteran_cl3yrs_evics_discharge,"Discharge from Institution in the
Last Six Months")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Abu
se by Other - Financial"],abuse_neglect_prevaps_oldcols,"Abuse by Other -
Financial")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Abu
se by Other - Non-Financial"],abuse_neglect_prevaps_oldcols,"Abuse by Other -
Non-Financial")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Sel
f-Neglect"],abuse_neglect_prevaps_oldcols,"Self-Neglect")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Rep
orting Source"],rs,"Reporting Source")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Pre
vious APS Involvement"],abuse_neglect_prevaps_oldcols,"Previous APS
Involvement")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Cli
ent Referred to CES"],ces,"Client Referred to CES")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Int
ervention 1 - Type"],interv_type,"HSAPS Intervention 1 - Type")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Int
ervention 2 - Type"],interv_type,"HSAPS Intervention 2 - Type")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Int
ervention 3 - Type"],interv_type,"HSAPS Intervention 3 - Type")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Int
ervention 4 - Type"],interv_type,"HSAPS Intervention 4 - Type")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Int
ervention 5 - Type"],interv_type,"HSAPS Intervention 5 - Type")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr,hsaps_dqr[,"Int
ervention 6 - Type"],interv_type,"HSAPS Intervention 6 - Type")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[!is.na(hsaps_dq
r$`Living Situation at Exit`),],hsaps_dqr[!is.na(hsaps_dqr$`Living Situation
at Exit`),"Living Situation at Exit"],ls_exit_fu_ls,"Living Situation at
Exit")

##make sure to update the dates below as the time passes.
#for 6-month follow-up should be flagged after 6 months from the end of
current quarter. Similarly, for 12-month, for 12 months
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[which(hsaps_dqr
$`Case Closure Date`<="2023-12-31"),],hsaps_dqr[which(hsaps_dqr$`Case Closure
Date`<="2023-12-31"),"Six Month Follow-Up - Method"],fu_method,"Six Month
Follow-Up - Method")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[which(hsaps_dqr
$`Case Closure Date`<="2023-12-31"),],hsaps_dqr[which(hsaps_dqr$`Case Closure
Date`<="2023-12-31"),"Six Month Follow-Up - Living
Situation"],ls_exit_fu_ls,"Six Month Follow-Up - Living Situation")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[which(hsaps_dqr
$`Case Closure Date`<="2023-12-31"),],hsaps_dqr[which(hsaps_dqr$`Case Closure
Date`<="2023-12-31"),"Six Month Follow-Up - Homelessness"],fu_homeless,"Six
Month Follow-Up - Homelessness")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[which(hsaps_dqr
$`Case Closure Date`<="2023-06-30"),],hsaps_dqr[which(hsaps_dqr$`Case Closure
Date`<="2023-06-30"),"Twelve Month Follow-Up - Method"],fu_method,"Twelve
Month Follow-Up - Method")
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[which(hsaps_dqr
$`Case Closure Date`<="2023-06-30"),],hsaps_dqr[which(hsaps_dqr$`Case Closure
Date`<="2023-06-30"),"Twelve Month Follow-Up - Living
Situation"],ls_exit_fu_ls,"Twelve Month Follow-Up - Living Situation")
```

```
hsaps_drop_down_off =
hsaps_data_auditing_categorical(hsaps_drop_down_off,hsaps_dqr[which(hsaps_dqr
$`Case Closure Date`<="2023-06-30"),],hsaps_dqr[which(hsaps_dqr$`Case Closure
Date`<="2023-06-30"),"Twelve Month Follow-Up -
Homelessness"],fu_homeless,"Twelve Month Follow-Up - Homelessness")

hsaps_drop_down_off_empty =
unique(hsaps_drop_down_off[is.na(hsaps_drop_down_off$value)&!is.na(hsaps_drop
_down_off$`Case Start Date`),])
hsaps_drop_down_off_invalid =
unique(hsaps_drop_down_off[!is.na(hsaps_drop_down_off$value)&!is.na(hsaps_dro
p_down_off$`Case Start Date`),])
hsaps_drop_down_off_no_start_date =
unique(hsaps_drop_down_off[is.na(hsaps_drop_down_off$`Case Start Date`),])
```

- **hsaps_drop_down_off_missing**: subset list of hsaps_drop_down_off, where the responses of the data elements with drop down options are missing while the case start date of the case is not missing.
- **hsaps_drop_down_off_invalid**: subset list of hsaps_drop_down_off, where the responses of the data elements with drop down options are not aligned with ACIN options while the case start date of the case is not missing.
- **hsaps_drop_down_off_no_start_date**: subset list of hsaps_drop_down_off, where case start dates are missing.

Date of Birth later than Case Start Date

```
#people with negative ages
hsaps_negative_age = hsaps_dqr[which(hsaps_dqr$age<0),]
```

Case Closure Date & Living Situation at Exit

- The Case Closure Date exists, but Living Situation at Exit states “Not Exited”.

```
#case closure date exists but living situation at exit says "not exited"
hsaps_exit_paradox = hsaps_dqr[which(!is.na(hsaps_dqr$`Case Closure
Date`)&grepl("not exited",tolower(hsaps_dqr$`Living Situation at Exit`))),]
#living situation exit says other than "not exited" when case closure date is
blank
```

- The Case Closure Date is missing, but Living Situation at Exit states an exit location.

```
#living situation exit says other than "not exited" when case closure date is
blank
hsaps_exit_paradox2 = hsaps_dqr[which(is.na(hsaps_dqr$`Case Closure
Date`)&!is.na(hsaps_dqr$`Living Situation at Exit`)&!grepl("not
exited",tolower(hsaps_dqr$`Living Situation at Exit`))),]
```

Intervention

- Any of the Intervention Dates earlier than Case Start Date

```r
#intervention date before case start date -> bigger the date, the later it is
flag = c()
for(i in 1:nrow(hsaps_dqr)){
  if(any(hsaps_dqr$`Case Start Date`[i]>unlist(hsaps_dqr[i,grep("Intervention
[0-9]{1} - Date",names(hsaps))]),na.rm=T)){
    flag = c(flag,i)
  }
}
hsaps_start_paradox = hsaps_dqr[flag,]
```

- Case Closure Date earlier than any of the Intervention Dates

```r
#intervention date after case closure date
flag = c()
for(i in 1:nrow(hsaps_dqr)){
  if(any(hsaps_dqr$`Case Closure
Date`[i]<unlist(hsaps_dqr[i,grep("Intervention [0-9]{1} -
Date",names(hsaps))]),na.rm=T)){
    flag = c(flag,i)
  }
}
hsaps_close_paradox = hsaps_dqr[flag,]
```

- Intervention Type has a response, but the corresponding Intervention Date is missing.

```r
#intervention type has a response but the date is blank.
hsaps_interv_missing = hsaps_dqr[,c(1:4,grep("Case Start
Date",names(hsaps)),grep("Intervention",names(hsaps)))]
hsaps_interv_missing$var = ''
flag = c()
for(i in 1:nrow(hsaps_interv_missing)){
  for(j in grep("Intervention [0-9]{1} - Type",names(hsaps_interv_missing))){
    if(!is.na(hsaps_interv_missing[i,j])){
      if(hsaps_interv_missing[i,j]!="No
Intervention"&is.na(hsaps_interv_missing[i,(j+2)])){
        flag = c(flag,i)
        hsaps_interv_missing$var[i] = paste0(hsaps_interv_missing$var[i],',
',names(hsaps_interv_missing)[j+2])
      }
    }
  }
  flag = unique(flag)
}
hsaps_interv_date_missing = hsaps_interv_missing[flag,]
hsaps_interv_date_missing$var = gsub("^, ","",hsaps_interv_date_missing$var)
```

- Both Intervention Type and Intervention Date exist, but the corresponding Intervention Amount is missing.

```r
#Intervention Type or Date exist, but the amount is blank.
hsaps_interv_missing$var=''
flag = c()
```

```
for(i in 1:nrow(hsaps_interv_missing)){
  for(j in grep("Intervention [0-9]{1} - Type",names(hsaps_interv_missing))){
    if(!is.na(hsaps_interv_missing[i,j])){
      if((hsaps_interv_missing[i,j]!="No 
Intervention"|!is.na(hsaps_interv_missing[i,(j+2)]))&is.na(hsaps_interv_missi
ng[i,(j+3)])){
        flag = c(flag,i)
        hsaps_interv_missing$var[i] = paste0(hsaps_interv_missing$var[i],', 
',names(hsaps_interv_missing)[j+3])
      }
    }
  }
  flag = unique(flag)
}
hsaps_interv_amount_missing = hsaps_interv_missing[flag,]
hsaps_interv_amount_missing$var = gsub("^, 
","",hsaps_interv_amount_missing$var)
```

- Any of the Intervention Dates later than the reporting period

```
#intervention dates in the future
flag = c()
hsaps_interv_future = hsaps_dqr[,c(1:5,grep("Intervention",names(hsaps)))]
hsaps_interv_future$var = ''

##MAKE SURE TO CHANGE THE DATE BELOW WHENEVER NEW QUARTER COMES -> to the 
last day of submission (ex. Q3 ends in March 31st but the report deadline is 
Apr 30th)
for(i in 1:nrow(hsaps_interv_future)){
  for(j in grep("Intervention [0-9] - Date",names(hsaps_interv_future))){
    if(!is.na(hsaps_interv_future[i,j]) & hsaps_interv_future[i,j]>"2024-07-
31"){
      flag = c(flag,i)
      hsaps_interv_future$var[i] = paste0(hsaps_interv_future$var[i],', 
',names(hsaps_interv_future)[j])
    }
  }
}
hsaps_interv_future = hsaps_interv_future[flag,]
hsaps_interv_future$var = gsub("^, ","",hsaps_interv_future$var)
```

Consolidating Flags into DQR Tables

As the code flagged all the missing and invalid responses from the data, it now generates the table to list all those flags with descriptions of the case, the variables where the flags pertain to, the description of the flags, what grantees should do to fix the flags, and the priorities of the issues. This table is will be stored in “Case History-Errors” sheet of the DQR.

Code Description: DQR Script

```r
#putting together error report for grantees
hsaps_error_table = data.frame()
if(nrow(hsaps_negative_age)!=0){
  hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_negative_age[,c("Reporting_Age
ncy","First Name","Last Name","Case Start Date")],"Invalid",'Date of
Birth',as.character(hsaps_negative_age$`Date of Birth`),"Date of Birth is
later than Case Start Date.",'High')
}
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_noncumulative[,c("Reporting_Ag
ency","First Name","Last Name","Case Start Date")],'Missing','Dropped
Record',hsaps_noncumulative$Quarter,paste0('Client was previously recorded in
',hsaps_noncumulative$Quarter,' report and is no longer present in most
recent report.'),'High')
hsaps_duplicate_data_tbd = subset(read.xlsx("//cdss/feed/Central
Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports
to date/Cleaning In
Progress/hsaps_duplicate_data_same_period_sorted.xlsx",sheet="duplicates_tbd"
,sep.names = " "),!grepl("^Included",Reporting_Agency))
hsaps_duplicate_data_tbd =
merge(hsaps_duplicate_data_tbd,quarter[,c(1,3)],all.x=T)
#make sure to update the date for line 183 according to the newest quarter
hsaps_duplicate_data_tbd =
hsaps_duplicate_data_tbd[hsaps_duplicate_data_tbd$Report_Month=="2024-04-
01",]
hsaps_duplicate_data_tbd =
hsaps_duplicate_data_tbd[!(duplicated(hsaps_duplicate_data_tbd[c("First
Name","Last Name","Reporting_Agency","Case Start Date")])&
duplicated(hsaps_duplicate_data_tbd[c("First Name","Last
Name","Reporting_Agency","Case Start Date")],fromlast=T)&
duplicated(hsaps_duplicate_data_tbd[c("First Name","Last
Name","Reporting_Agency","Case Start Date")],fromlast=T)&
duplicated(hsaps_duplicate_data_tbd[c("First Name","Last
Name","Reporting_Agency","Case Start Date")],fromlast=T)),]
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_dqr[which(is.na(hsaps_dqr$`Cas
e Start Date`)),c("Reporting_Agency","First Name","Last Name","Case Start
Date")],'Missing','Home Safe Case Start Date','','The response is missing and
this is a required element.','High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_drop_down_off_invalid[,c("Repo
rting_Agency","First Name","Last Name","Case Start
Date")],'Invalid',hsaps_drop_down_off_invalid$discrepant_var,hsaps_drop_down_
off_invalid$value,'This response is not listed as an option from the drop-
down menu.','High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_drop_down_off_empty[,c("Report
ing_Agency","First Name","Last Name","Case Start
Date")],'Missing',hsaps_drop_down_off_empty$discrepant_var,'','The response
```

```
is missing and this is a required element.','High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_drop_down_off_no_start_date[wh
ich(is.na(hsaps_drop_down_off_no_start_date$value)),c("Reporting_Agency","Fir
st Name","Last Name","Case Start
Date")],'Missing',hsaps_drop_down_off_no_start_date$discrepant_var[which(is.n
a(hsaps_drop_down_off_no_start_date$value))],'','The response is missing and
this is a required element.','High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_drop_down_off_no_start_date[wh
ich(!is.na(hsaps_drop_down_off_no_start_date$value)),c("Reporting_Agency","Fi
rst Name","Last Name","Case Start
Date")],'Invalid',hsaps_drop_down_off_no_start_date$discrepant_var[which(!is.
na(hsaps_drop_down_off_no_start_date$value))],hsaps_drop_down_off_no_start_da
te$value[which(!is.na(hsaps_drop_down_off_no_start_date$value))],'This
response is not listed as an option from the drop-down menu.','High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_exit_paradox[,c("Reporting_Age
ncy","First Name","Last Name","Case Start Date")],"Invalid",'Home Safe Case
Closure Date',as.character(hsaps_exit_paradox$`Case Closure Date`),"Living
Situation at Exit states, 'Not Exited', when Case Closure Date
exists",'High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_exit_paradox2[,c("Reporting_Ag
ency","First Name","Last Name","Case Start Date")],"Missing",'Home Safe Case
Closure Date','',"Living Situation at Exit states a response when Case
Closure Date is blank.",'High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_start_paradox[,c("Reporting_Ag
ency","First Name","Last Name","Case Start Date")],"Invalid",'Home Safe Case
Start Date',as.character(hsaps_start_paradox$`Case Start Date`),"Case Start
Date is later than HSAPS Intervention Date(s).",'High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_close_paradox[,c("Reporting_Ag
ency","First Name","Last Name","Case Start Date")],"Invalid",'Home Safe Case
Closure Date',as.character(hsaps_close_paradox$`Case Closure Date`),"Case
Closure Date is earlier than HSAPS Intervention Date(s).",'High')
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_interv_date_missing[,c("Report
ing_Agency","First Name","Last Name","Case Start
Date")],"Missing",hsaps_interv_date_missing$var,'',"HSAPS Intervention Type
states a response, when Intervention Date is blank.","High")
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_interv_amount_missing[,c("Repo
rting_Agency","First Name","Last Name","Case Start
Date")],"Missing",hsaps_interv_amount_missing$var,'',"HSAPS Intervention Type
and/or Date state a response, when Intervention Amount is blank.","High")
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_interv_future[,c("Reporting_Ag
ency","First Name","Last Name","Case Start
Date")],"Invalid",hsaps_interv_future$var,'',"HSAPS Intervention Date(s) is
later than the current reporting period.","High")
hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_duplicate_data_tbd[,c("Reporti
ng_Agency","First Name","Last Name","Case Start Date")],'Duplicate
record','Quarter',hsaps_duplicate_data_tbd$Quarter,'This participant was
found twice (or more than twice) in the report. The records have the same
case start date.','High')
hsaps_no_start_date_duplicate =
merge(hsaps_no_start_date_duplicate,quarter[,c(1,3)],all.x=T)
if(nrow(hsaps_no_start_date_duplicate)>0){
  hsaps_error_table =
error_table_generation(hsaps_error_table,hsaps_no_start_date_duplicate[,c("Re
porting_Agency","First Name","Last Name","Case Start Date")],'Duplicate
record','Quarter',hsaps_no_start_date_duplicate$Quarter,'This participant was
found twice (or more than twice) in the report. The records have the same
case start date.','High')
}
hsaps_error_table =
unique(hsaps_error_table[order(hsaps_error_table$Reporting_Agency),])

hsaps_error_table$`Additional Information/Instructions`[grep("^Date of Birth
is la",hsaps_error_table$`Error Description`)] = "Please check both Date of
Birth and Case Start Date before fixing either of the variable."
hsaps_error_table$`Additional Information/Instructions`[grep("^Client was
previously",hsaps_error_table$`Error Description`)] = "Please enter this
record from the given report to the upcoming report."
hsaps_error_table$`Additional Information/Instructions`[grep("^Case Start
Date is later",hsaps_error_table$`Error Description`)] = "Please check both
Case Start Date and HSAPS Intervention Date(s) to fix the corresponding
element(s)."
hsaps_error_table$`Additional Information/Instructions`[grep("^Case Closure
Date is earlier",hsaps_error_table$`Error Description`)] = "Please check both
Case Closure Date and HSAPS Intervention Date(s) to fix the corresponding
element(s)."
hsaps_error_table$`Additional Information/Instructions`[grep("^HSAPS
Intervention Type and/or",hsaps_error_table$`Error Description`)] = "Please
enter the invervention amount to the corresponding element (showin in 'Item'
column)."
hsaps_error_table$`Additional Information/Instructions`[grep("^HSAPS
Intervention Type states",hsaps_error_table$`Error Description`)] = "Please
enter the intervention date(s) to the corresponding element (showin in 'Item'
column)."
hsaps_error_table$`Additional Information/Instructions`[grep("^HSAPS
Intervention Date\\(s\\)",hsaps_error_table$`Error Description`)] = "Please
do not list intervention(s) until it is provided to the participant."
hsaps_error_table$`Additional Information/Instructions`[grep("^Living
Situation at Exit states a res",hsaps_error_table$`Error Description`)] =
"Please enter the Case Closure Date, or change the Living Situation at Exit
to 'Not Exited'."
```

```
hsaps_error_table$`Additional Information/Instructions`[grep("^Living
Situation at Exit states, 'Not",hsaps_error_table$`Error Description`)] =
"Please fix Living Situation at Exit to a proper response, or delete the Case
Closure Date."
hsaps_error_table$`Additional Information/Instructions`[grep("^This
participant was found twice",hsaps_error_table$`Error Description`)] =
"Verify that the identified duplicate is an actual duplicate. If it is,
select the duplicate record to remove among the pair by removing the record
within the report submitted in the quarter indicated in column G. Comment
'Duplicate record removed' in the column L ('Comments or questions from
grantee')."
hsaps_error_table$`Additional Information/Instructions`[grep("^This response
is not listed as an option",hsaps_error_table$`Error Description`)] = "Please
re-enter the proper option from the drop-down menu."
hsaps_error_table$`Additional Information/Instructions`[grepl("^The response
is missing",hsaps_error_table$`Error Description`)& grepl("^HSAPS
Interv",hsaps_error_table$Item)] = "If there was no intervention given,
please use the drop-down menu, 'No Intervention'."
hsaps_error_table$Item = gsub("^Intervention","HSAPS
Intervention",hsaps_error_table$Item)
hsaps_error_table$Item = gsub(", Intervention",", HSAPS
Intervention",hsaps_error_table$Item)
hsaps_error_table$`Error Priority`[grepl("up -
method",tolower(hsaps_error_table$Item))|grepl("^abuse",tolower(hsaps_error_t
able$Item))|grepl("self-
neglect",tolower(hsaps_error_table$Item))|grepl("eviction",tolower(hsaps_erro
r_table$Item))|grepl("^discharge",tolower(hsaps_error_table$Item))|grepl("rep
orting source",tolower(hsaps_error_table$Item))|grepl("previous
aps",tolower(hsaps_error_table$Item))]="Medium"
hsaps_error_table$`Error Priority`[grepl("^preferred
",tolower(hsaps_error_table$Item))]="Low"
hsaps_error_table =
unique(hsaps_error_table[order(hsaps_error_table$Reporting_Agency),])
```

- Just as hsaps_key_info_discrepant table was generated, error_table_generation function was used to generate this table as well.
- hsaps_duplicate_data_tbd table was imported to be included in the DQR. The data was de-duplicated based on the key variables of the case (county, first & last names, date of birth, case start date). This list, unlike others, was stored in “Duplicates” sheet.
- After the table was generated, specific instruction on how to resolve the issues were coded.

DQR Generation & Formatting

Lastly, the code goes through each county, automatically updating DQR tables and exporting them with the proper naming convention into the corresponding folder.

Code Description: DQR Script

```r
##DQR Generation
#Please update the proper folder name as the quarter progresses
new_location = "//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home
Safe/Data and Evaluation/Data Reports to date/Data Quality Reports/Generated
DQRs/FY 23-24 Q4"
for(i in unique(hsaps_error_table$Reporting_Agency)){
  file = "//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home
Safe/Data and Evaluation/Data Reports to date/Data Quality Reports/Home Safe
Data Quality Report Template.xlsx"
  #make sure to change the file name according to the newest quarter
  file.copy(file,file.path(new_location,paste0(str_to_title(i),"-FY 23-24 Q4
Data Quality Report.xlsx")),overwrite = T)
  new_file = paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing
Programs/Home Safe/Data and Evaluation/Data Reports to date/Data Quality
Reports/Generated DQRs/FY 23-24 Q4/",str_to_title(i),"-FY 23-24 Q4 Data
Quality Report.xlsx")
  wb = loadWorkbook(new_file)
  errors_summary_ch = read.xlsx(file,sheet="Errors-Summary")[,1:3]
  errors_summary_ch = errors_summary_ch[-nrow(errors_summary_ch),]
  names(errors_summary_ch) = as.character(errors_summary_ch[1,])

  data = subset(hsaps_error_table,Reporting_Agency==i)
  data_aggregate = sqldf("select Item,`Error Description`,count(Item) as num
from data group by Item,`Error Description`")
  data_aggregate2 = sqldf("select Reporting_Agency,Item,count(Item) as `#
Records` from hsaps_key_info_discrepant group by Reporting_Agency,Item")

  #putting together aggregate table for DQR "Errors-Summary" sheet
  #filling in var names for invalid data flag rows in errors_summary_ch table
  for(r in 1:nrow(errors_summary_ch)){
    if(is.na(errors_summary_ch$Item[r])){
      errors_summary_ch$Item[r] = errors_summary_ch$Item[(r-1)]
    }
  }

  if(any(grepl("^Dropped",data_aggregate$Item))){
    errors_summary_ch$`#
Errors`[grep("^Dropped",errors_summary_ch$Item)]=length(grep("^Dropped",data$
Item))
  }
  if(any(grepl("Amount",data_aggregate$Item))){
    errors_summary_ch$`#
Errors`[which(grepl("intervention",tolower(errors_summary_ch$Item))&
grepl("amount",tolower(errors_summary_ch$Item)))] =
length(grep("amount",tolower(data$Item)))
  }
  for(r in 1:nrow(data_aggregate)){
    data_aggregate$Item[r] = gsub("Follow-Up","Follow
Up",data_aggregate$Item[r])
```

```r
    data_aggregate$Item[r] = gsub("^Client ","",data_aggregate$Item[r])
    if(grepl("Intervention",data_aggregate$Item[r])){
      if(grepl("Date",data_aggregate$Item[r])){
        if(all(errors_summary_ch$`#
Errors`[which(grepl("date",tolower(errors_summary_ch$Item))&
grepl("intervention",tolower(errors_summary_ch$Item)))]!=0)){
          next
        }
        else if(grepl("later than",data_aggregate$`Error Description`[r])){
          errors_summary_ch$`# Errors`[grep("^hsaps intervention
date\\(s\\)",tolower(errors_summary_ch$`Error Description`))] =
length(grep("^hsaps intervention date\\(s\\)",tolower(data$`Error
Description`)))
        }
        else{
          errors_summary_ch$`# Errors`[grep("^intervention type states
a",tolower(errors_summary_ch$`Error Description`))] =
length(grep("intervention type states a",tolower(data$`Error Description`)))
        }
      }
      else if(grepl("Type",data_aggregate$Item[r])){
        if(all(errors_summary_ch$`#
Errors`[grep("type",tolower(errors_summary_ch$Item))]!=0)){
          next
        }
        else if(grepl("missing",data_aggregate$`Error Description`[r])){
          errors_summary_ch$`#
Errors`[which(grepl("type",tolower(errors_summary_ch$Item))&
grepl("missing",tolower(errors_summary_ch$`Error Description`)))] =
length(which(grepl("type",tolower(data$Item)) &
grepl("missing",tolower(data$`Error Description`))))
        }
        else{
          errors_summary_ch$`#
Errors`[which(grepl("type",tolower(errors_summary_ch$Item))& grepl("not
listed",tolower(errors_summary_ch$`Error Description`)))] =
length(which(grepl("type",tolower(data$Item)) & grepl("not
listed",tolower(data$`Error Description`))))
        }
      }
    }
    else if(!grepl("^Quarter",data_aggregate$Item[r])){
    errors_summary_ch$`#
Errors`[which(errors_summary_ch$Item==data_aggregate$Item[r] &
errors_summary_ch$`Error Description`==data_aggregate$`Error
Description`[r])] = data_aggregate$num[r]
    }
  }

  style = createStyle(halign="left",valign="top",wrapText = T)
  writeData(wb,"Errors-
Summary",as.numeric(errors_summary_ch[2:nrow(errors_summary_ch),ncol(errors_s
ummary_ch)]),startCol=3,startRow=3,colNames = F)
  if(length(grep(i,data_aggregate2$Reporting_Agency))>0){
    writeData(wb,"Errors-
Summary",data_aggregate2[data_aggregate2$Reporting_Agency==i,2:ncol(data_aggr
egate2)],startCol=11,startRow=3,colNames = F)
  }

  removeWorksheet(wb,"Duplicates")
  addWorksheet(wb,"Duplicates")
  writeDataTable(wb,"Duplicates",data[grep("Duplicate",data$`Error
Type`),],tableStyle = "TableStyleMedium2")
  removeWorksheet(wb,"Case History-Errors")
  if(length(grep("Duplicate",data$`Error Type`))>0){
    errors_summary_dup = read.xlsx(file,sheet="Errors-Summary")[1,4:6]
    names(errors_summary_dup) = as.character(errors_summary_dup[1,])
    errors_summary_dup =
rbind(errors_summary_dup,data[grep("Duplicate",data$`Error Type`),c("First
Name","Last Name","Case Start Date")])
    writeData(wb,"Errors-
Summary",errors_summary_dup[2:nrow(errors_summary_dup),],startCol=6,startRow=
3,colNames = F)
    addStyle(wb,"Duplicates",style,2:(nrow(data[grep("Duplicate",data$`Error
Type`),])+1),1:ncol(data),gridExpand = T)
    setColWidths(wb,"Duplicates",8,32.7)
    setColWidths(wb,"Duplicates",9,48.5)
    setColWidths(wb,"Duplicates",11,12)
    addWorksheet(wb,"Case History-Errors")
    writeDataTable(wb,"Case History-Errors",data[-
grep("Duplicate",data$`Error Type`),],tableStyle = "TableStyleMedium2")
  }
  else{
    addWorksheet(wb,"Case History-Errors")
    writeDataTable(wb,"Case History-Errors",data,tableStyle =
"TableStyleMedium2")
  }
  addStyle(wb,"Case History-
Errors",style,2:(nrow(data)+1),1:ncol(data),gridExpand = T)
  setColWidths(wb,"Case History-Errors",6,23.5)
  setColWidths(wb,"Case History-Errors",8,29.5)
  setColWidths(wb,"Case History-Errors",9,30.5)
  setColWidths(wb,"Case History-Errors",11,12)
  setColWidths(wb,"Case History-Errors",12,35)

  removeWorksheet(wb,"Key Information Discrepancies")
  addWorksheet(wb,"Key Information Discrepancies")
  writeDataTable(wb,"Key Information
Discrepancies",hsaps_key_info_discrepant[grep(i,hsaps_key_info_discrepant$Rep
orting_Agency),],tableStyle = "TableStyleMedium2")
```

```
  if(length(grep(i,hsaps_key_info_discrepant$Reporting_Agency))>0){
    addStyle(wb,"Key Information
Discrepancies",style,2:(nrow(hsaps_key_info_discrepant[grep(i,hsaps_key_info_
discrepant$Reporting_Agency),])+1),1:ncol(hsaps_key_info_discrepant[grep(i,hs
aps_key_info_discrepant$Reporting_Agency),]),gridExpand = T)
    setColWidths(wb,"Key Information Discrepancies",7,15)
    setColWidths(wb,"Key Information Discrepancies",8,15)
    setColWidths(wb,"Key Information Discrepancies",9,30)
    setColWidths(wb,"Key Information Discrepancies",10,35)
    setColWidths(wb,"Key Information Discrepancies",13,35)
  }
  saveWorkbook(wb,new_file,overwrite=T)
}
```

- Creating a new folder location for new DQR's
- Copying the DQR template to the new folder and changing its name as "[County Name]-FY 23-24 Q4 Data Quality Report".
- Importing tables from "Errors-Summary" sheet, populating the aggregate numbers into the tables, and exporting the table back to the sheet
- Exporting generated DQR tables to "Duplicates", "Case History-Errors", "Key Information Discrepancies" sheets
    - After being exported, the code applies a specific table style to each table and adjusts widths of selected columns.
- Saving all the changes made to the new workbook