HDAP DQR Generation Process Documentation

2024-11-05

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

Initial Preparation

Code Description: DQR Script

```r
hdap = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing 
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/^hdap_cleaned_2024-09-
27.xlsx",sep.names = " ")
hdap_new_data_exclude = read.xlsx("//cdss/feed/Central 
Office/HHCRB/HHB/Housing Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning 
In Progress/hdap_new_data_exclude.xlsx",sep.names = " ")
hdap_duplicate_data_same_period_tbd = read.xlsx("//cdss/feed/Central 
Office/HHCRB/HHB/Housing Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning 
In Progress/hdap_duplicate_data_tbd.xlsx",sep.names = " ")
hdap_no_start_date_duplicate = read.xlsx("//cdss/feed/Central 
Office/HHCRB/HHB/Housing Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning 
in Progress/hdap_no_start_date_duplicate.xlsx",sep.names = " ")
if(nrow(hdap_no_start_date_duplicate)>0){
  hdap_no_start_date_duplicate = 
hdap_no_start_date_duplicate[(duplicated(hdap_no_start_date_duplicate[c("Coun
ty","Quarter Start","First Name","Last Name")])&
duplicated(hdap_no_start_date_duplicate[c("County","Quarter Start","First 
Name","Last Name")],fromlast=T)&
duplicated(hdap_no_start_date_duplicate[c("County","Quarter Start","First 
Name","Last Name")],fromlast=T)&
duplicated(hdap_no_start_date_duplicate[c("County","Quarter Start","First 
Name","Last Name")],fromlast=T)),]
}

#dropped participants
quarter = read.csv("//cdss/feed/Central Office/HHCRB/HHB/Housing 
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/R Scripts/Quarter_Period.csv")
hdap_noncumulative = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing 
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning In 
Progress/hdap_dropped.xlsx",sep.names = " ")
hdap_noncumulative = subset(hdap,as.Date(`Quarter Start`)<as.Date('2023-10-
01'))
hdap_noncumulative = 
merge(hdap_noncumulative,quarter[,c("Quarter_Start","Quarter")],by.x="Quarter
Start",by.y="Quarter_Start",all.x=T)
#remove possible typo list from the dropped record list
flag = c()
for(i in 1:nrow(hdap_noncumulative)){
  if((tolower(paste0(hdap_noncumulative$County[i],hdap_noncumulative$`First
Name`[i],hdap_noncumulative$`Date of Birth`[i])) %in%
tolower(paste0(hdap_new_data_exclude$County,hdap_new_data_exclude$`First
Name`,hdap_new_data_exclude$`Date of Birth`))) |
(tolower(paste0(hdap_noncumulative$County[i],hdap_noncumulative$`Date of
Birth`[i],hdap_noncumulative$`Last Name`[i])) %in%
tolower(paste0(hdap_new_data_exclude$County,hdap_new_data_exclude$`Date of
Birth`,hdap_new_data_exclude$`Last Name`))) |
(tolower(paste0(hdap_noncumulative$County[i],hdap_noncumulative$`First
Name`[i],hdap_noncumulative$`Last Name`[i])) %in%
tolower(paste0(hdap_new_data_exclude$County,hdap_new_data_exclude$`First
Name`,hdap_new_data_exclude$`Last Name`)))){
    flag = c(flag,i)
  }
}
if(length(flag)>0){
  hdap_noncumulative = hdap_noncumulative[-flag,]
}
```

- Importing hdap, hdap_new_data_exlude, hdap_noncumulative, and hdap_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 Quarter Start 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 hdap_noncumulative table, whatever participant data that are part of hdap_new_data_exclude table was removed as for these participants, we chose to keep the older data while flagging the newest ones.
- hdap_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 hdap_new_data_exclude. This table will be stored in "Key Information Discrepancies" sheet in the DQR.

<u>Code Description: Functions</u>

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:

```
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.

```
#data set of participant key info discrepant
hdap =
merge(hdap,quarter[which(!is.na(quarter$Quarter_Start)),2:3],by.x="Quarter
Start",by.y="Quarter_Start",all.x=T)
hdap = hdap[,c(2:4,7,5:6,8:grep("Tracking
18",names(hdap)),1,grep("End$",names(hdap)):ncol(hdap))]
hdap_participant_discrepant = unique(hdap_new_data_exclude[,c("County","First
Name","Last Name","Date of Birth","Project Start Date")])
names(hdap_participant_discrepant)[-1] =
paste0(names(hdap_participant_discrepant)[-1],"_new")
hdap_participant_discrepant =
unique(merge(hdap_participant_discrepant,hdap[,c(1:3,5)],by.x=c("County","Las
t Name_new","First Name_new"),by.y=c("County","Last Name","First
Name"),all.x=T))
hdap_participant_discrepant =
unique(merge(hdap_participant_discrepant,hdap[,c(1,grep("
Name$",names(hdap)),grep("Date of
Birth",names(hdap)),grep("Project",names(hdap)),grep("^Quarter$",names(hdap))
)],by.x=c("County","First Name_new","Date of Birth_new","Project Start
Date_new"),by.y=c("County","First Name","Date of Birth","Project Start
Date"),all.x=T))
hdap_participant_discrepant =
unique(merge(hdap_participant_discrepant,hdap[,c(1,grep("
Name$",names(hdap)),grep("Date of
Birth",names(hdap)),grep("Project",names(hdap)),grep("^Quarter$",names(hdap))
)],by.x=c("County","Last Name_new","Date of Birth_new","Project Start
Date_new"),by.y=c("County","Last Name","Date of Birth","Project Start
Date"),all.x=T))

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

- **hdap_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.

- **hdap_key_info_discrepant**: the final DQR table generated from hdap_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 hdap 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-ACL Responses

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

```r
##data auditing for each variable with drop down categories
#make sure you update Quarter Start below for the corresponding quarter
hdap_dqr = subset(hdap,`Quarter Start`=="2024-04-01")
unknown = paste0(8:9,"-",c("Client Does Not Know","Client Refused"))
na = "99-Data Not Collected"
mf = paste0(0:1,"-",c("Female","Male"))
race = c(paste0(1:6,"-",c("American Indian/Alaska
Native","Asian","Black/African American","Native Hawaiian/Other Pacific
Islander","White","Multiracial")),unknown)
eth = c(paste0(1:2,"-",c("Non-Hispanic/Non-
Latino","Hispanic/Latino")),unknown)
gender = c(mf,"2-Transgender Female","3-Transgender Male","4-Non-
Binary",unknown,"10-Another Gender Identity")
sex = c(mf,"2-Non-Binary","9-Decline to State")
so = c(paste0(0:4,"-
",c("Straight/Heterosexual","Gay/Lesbian","Bisexual","Queer","Another Sexual
Orientation")),unknown)
target = paste0(0:1,"-",c("No","Yes"))
vet = c(target,unknown) #same as disabled et al
ls = c(paste0(c(1:7,12:16,18:27),'-',c("Emergency Shelter","Transitional
Housing","Permanent Housing (except RRH) for Formerly Homeless
Persons","Psychiatric Hospital/Facility","Substance Abuse Treatment
Facility/Detox Center","Hospital/Residential Medical
Facility","Jail/Prison/Juvenile Facility","Living with Family","Living with
Friend","Hotel/Motel Paid without Emergency Shelter Voucher","Foster
Care/Group Home","Place Not Meant for Habitation","Safe Heaven","Rental with
VASH","Rental with Other Housing Subsidy (including RRH)","Owner with Housing
Subsidy","Rental without Housing Subsidy","Owner without Housing
Subsidy","Long-Term Care Facility/Nursing Home","Rental with GPD
TIP","Residential Project/Halfway House with No Homeless Criteria","Interim
Housing")),unknown)
previous_stay = c(paste0(c(2:5,10:11),"-",c("Between One week and One
Month","Between One Month and 90 Days","Between 90 days and 1 Year","1+
Year","One Night or Less","2-6 Nights")),unknown,na)
homeless_count = c(paste0(1:4,"-",c("Once","Twice","Three Times","4+
Times")),unknown,na)
homeless_length = c(unknown,na,"101-One Month",paste0(102:113,'-
',c(2:12,"12+")," Months"))
disabl_type = paste0(1:4,"-",c("SSI/SSP","SSDI","CAPI","Veteran's Benefits"))
denial_reason = paste0(0:11,"-",c("Loss of Contact","Capable to Re/Enter
Workforce","Insufficient Medical Evidence","Lack of Follow-Through with
Treatment Plan","Lack of Follow-Through with Application Process","Prior
Denial of Benefits","Did Not Meet Disability Criteria","Lack of Work
Credits","Not Disabled Prior to Last Insured","Excess
Resources","Unknown","Other"))
exit = c(paste0(c(1:7,10:25,28:32),"-",c("Emergency Shelter","Transitional
Housing","Permanent Housing (except RRH) for Formerly Homeless
Persons","Psychiatric Hospital/Facility","Substance Abuse Treatment
Facility/Detox Center","Hospital/Residential Medical
Facility","Jail/Prison/Juvenile Facility","Rental without Housing
Subsidy","Owner without Housing Subsidy","Living with Family
Temporarily","Living with Friend Temporarily","Hotel/Motel Paid without
Emergency Shelter Voucher","Foster Care/Group Home","Place Not Meant for
Habitation","Other","Safe Heaven","Rental with VASH","Rental with Other
Housing Subsidy","Owner with Housing Subsidy","Living with Family
Permanently","Living with Friend Permanently","Deceased","Long-Term Care
Facility/Nursing Home","Rental with GPD TIP","Residential Project/Halfway
House with No Homeless Criteria","No Exit Interview Completed","Rental with
RRH/Equivalent Subsidy","Retained Housing")),unknown)

#responses that are missing or off the drop-down options
hdap_drop_down_off = data.frame()
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$Race,race
,"Race")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$Ethnicity
,eth,"Ethnicity")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$`Gender
Identity`,gender,"Gender Identity")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$`Sex
Listed on Birth Certificate`,sex,"Sex Listed on Birth Certificate")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$`Sexual
Orientation`,so,"Sexual Orientation")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr[,grep("ta
rget pop",tolower(names(hdap)))],target,paste0("HDAP Target Population -
",c("GA/GR","CalWORKs","Diverted from Jail/Prison","Low Income
Veteran","Discharged from Intuition","Other Low/No Income")))
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr[,c(grep("
^veter",tolower(names(hdap))),grep("disabling",tolower(names(hdap))),grep("ch
ronically hom",tolower(names(hdap))))],vet,c("Veteran Status","Disabling
Condition","Chronically Homeless"))
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$`Living
Situation at Entry`,ls,"Living Situation at Entry")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr[,grep("ex
periencing homeless",tolower(names(hdap)))],target,c("At risk of experiencing
homelessness","Experiencing homelessness"))
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr,hdap_dqr$`Disabili
ty Benefit (A) - Type Applied For`,disabl_type,"Disability Benefit (A) - Type
Applied For")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr[which(!is.na(hdap_
dqr$`Disability Benefit (B) - Initial Application Submission
Date`)),],hdap_dqr$`Disability Benefit (B) - Type Applied
For`[which(!is.na(hdap_dqr$`Disability Benefit (B) - Initial Application
Submission Date`))],disabl_type,"Disability Benefit (B) - Type Applied For")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr[which(!is.na(hdap_
dqr$`Disability Benefit (A) - Final Denial Date`)),],hdap_dqr$`Disability
Benefit (A) - Reason for Final Denial`[which(!is.na(hdap_dqr$`Disability
Benefit (A) - Final Denial Date`))],denial_reason,"Disability Benefit (A) -
Reason for Final Denial")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr[which(!is.na(hdap_
dqr$`Disability Benefit (B) - Final Denial Date`)),],hdap_dqr$`Disability
Benefit (B) - Reason for Final Denial`[which(!is.na(hdap_dqr$`Disability
Benefit (B) - Final Denial Date`))],denial_reason,"Disability Benefit (B) -
Reason for Final Denial")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr[which(!is.na(hdap_
dqr$`Exit Date`)),],hdap_dqr$Destination[which(!is.na(hdap_dqr$`Exit
Date`))],exit,"Destination")
hdap_drop_down_off =
hdap_data_auditing_categorical(hdap_drop_down_off,hdap_dqr[which(!is.na(hdap_
dqr$`Exit Date`)),],hdap_dqr$`Exit Due to Loss of
```

```
Contact`[!is.na(hdap_dqr$`Exit Date`)],target,"Exit Due to Loss of Contact")
hdap_drop_down_off = unique(hdap_drop_down_off)
hdap_drop_down_off_empty =
unique(hdap_drop_down_off[is.na(hdap_drop_down_off$value),])
hdap_drop_down_off_invalid =
unique(hdap_drop_down_off[!is.na(hdap_drop_down_off$value),])
hdap_drop_down_off_no_start_date =
unique(hdap_drop_down_off[is.na(hdap_drop_down_off$`Project Start Date`),])
```

- **hdap_drop_down_off_missing**: subset list of hdap_drop_down_off, where the responses of the data elements with drop down options are missing while the projet start date of the case is not missing.
- **hdap_drop_down_off_invalid**: subset list of hdap_drop_down_off, where the responses of the data elements with drop down options are not aligned with ACL options while the projet start date of the case is not missing.
- **hdap_drop_down_off_no_start_date**: subset list of hdap_drop_down_off, where projet start dates are missing.

Missing or Partially Missing SSN's

```
#ssn empty or invalid
hdap_ssn_missing = hdap_dqr[is.na(hdap_dqr$`Social Security
Number`),c("County","First Name","Last Name","Date of Birth","Project Start
Date")]
hdap_ssn_invalid = hdap_dqr[which(!is.na(hdap_dqr$`Social Security Number`) &
!grepl("^[0-9]{9}$",hdap_dqr$`Social Security Number`)),c("County","Last
Name","First Name","Date of Birth","Project Start Date","Social Security
Number")]
```

Project Start Date with Other Variables

- Project Start Date earlier than Date of Birth

```
#people with negative ages
hdap_dqr$age = floor(elapsed_months(hdap_dqr$`Project Start
Date`,hdap_dqr$`Date of Birth`)/12)
hdap_negative_age = hdap_dqr[which(hdap_dqr$age<0),]
```

- Permanent Housing Move-In Date earlier than Project Start Date

```
#project start date later than permanent house move in date
flag = c()
for(i in which(!is.na(hdap_dqr$`Housing Move-In Date - Permanent Housing`) &
!is.na(hdap_dqr$`Project Start Date`))){
  if(hdap_dqr$`Project Start Date`[i]>hdap_dqr$`Housing Move-In Date -
Permanent Housing`[i]){
    flag = c(flag,i)
  }
}
hdap_start_paradox = hdap_dqr[flag,]
```

- Housing Stabilized/Retained Date earlier than Project Start Date

```
#housing stabilized/retained date earlier than project start date
flag = c()
for(i in which(!is.na(hdap_dqr$`Housing Stabilized/Retained Date`) &
!is.na(hdap_dqr$`Project Start Date`))){
  if(hdap_dqr$`Project Start Date`[i]>hdap_dqr$`Housing Stabilized/Retained
Date`[i]){
    flag = c(flag,i)
  }
}
hdap_start_paradox2 = hdap_dqr[flag,]
```

- Project Start Date later than Disability Benefit Approval Dates

```
#project start date later than approval dates
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Approval Date`) &
!is.na(hdap_dqr$`Project Start Date`))){
  if(hdap_dqr$`Project Start Date`[i]>hdap_dqr$`Disability Benefit (A) -
Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_start_paradox3_a = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Approval Date`) &
!is.na(hdap_dqr$`Project Start Date`))){
  if(hdap_dqr$`Project Start Date`[i]>hdap_dqr$`Disability Benefit (B) -
Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_start_paradox3_b = hdap_dqr[flag,]
```

Housing Date wih Other Variables

- Temporary Housing Move-In Date earlier than permanent Housing Move-In Date (not completely impossible, but not typical situation)

```
#Temporary housing move-in date earlier than permanent housing move-in date
flag = c()
for(i in which(!is.na(hdap_dqr$`Housing Move-In Date - Permanent Housing`) &
!is.na(hdap_dqr$`Housing Move-In Date - Temporary Housing`))){
  if(hdap_dqr$`Housing Move-In Date - Temporary Housing`[i]>hdap_dqr$`Housing
Move-In Date - Permanent Housing`[i]){
    flag = c(flag,i)
  }
}
hdap_housing_paradox = hdap_dqr[flag,]
```

- Rehousing Date later than Permanent Housing Date or Housing Stabilized/Retained Date

```
#rehousing date earlier than permanent housing date or housing stabilized
date
flag = c()
for(i in which(!is.na(hdap_dqr$`Rehousing Date`))){
  if(!is.na(hdap_dqr$`Housing Move-In Date - Permanent Housing`[i]) &
hdap_dqr$`Housing Move-In Date - Permanent Housing`[i]>=hdap_dqr$`Rehousing
Date`[i]){
    flag = c(flag,i)
  }
}
hdap_rehoused_paradox = unique(hdap_dqr[flag,])
hdap_rehoused_paradox2 = hdap_dqr[which(!is.na(hdap_dqr$`Rehousing Date`)  &
!is.na(hdap_dqr$`Housing Stabilized/Retained Date`)  & hdap_dqr$`Housing
Stabilized/Retained Date`>=hdap_dqr$`Rehousing Date`),]
```

At risk of experiencing homelessness and Other Variables

- At risk of experiencing homelessness not “Yes” while Housing Stabilized/Retained Date present

```
#those with housing stabilized/retained date should be "yes" for at-risk of
homelessness
hdap_stabilized_retained_at_risk = hdap_dqr[which(!is.na(hdap_dqr$`Housing
Stabilized/Retained Date`)&hdap_dqr$`At risk of experiencing
homelessness`!="Yes"),]
```

- At risk of homelessness and Experiencing Homelessness both “Yes” or “No”

```
#at risk of homelessness and experiencing homelessness both yes
hdap_homeless_status_paradox = hdap_dqr[which(hdap_dqr$`At risk of
experiencing homelessness`=="Yes" & hdap_dqr$`Experiencing
homelessness`=="Yes"),]
hdap_homeless_status_paradox2 = hdap_dqr[which(hdap_dqr$`At risk of
experiencing homelessness`=="No" & hdap_dqr$`Experiencing
homelessness`=="No"),]
```

Among Disability Benefit Dates

- Initial Submission Date later than Reconsideration Date

```
#initial submission date later than reconsideration date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Initial Application
Submission Date`) & !is.na(hdap_dqr$`Disability Benefit (A) - Reconsideration
Submission Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Initial Application Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Reconsideration Submission
Date`[i]){
    flag = c(flag,i)
  }
}
```

```r
hdap_disability_date_a_paradox1 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Initial Application
Submission Date`) & !is.na(hdap_dqr$`Disability Benefit (B) - Reconsideration
Submission Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Initial Application Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Reconsideration Submission
Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox1 = hdap_dqr[flag,]
```

- Reconsideration Date later than Subsequent Appeal Date

```r
#reconsideration date later than subsequent appeal date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Reconsideration
Submission Date`) & !is.na(hdap_dqr$`Disability Benefit (A) - Subsequent
Benefit Appeal Submission Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Reconsideration Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Subsequent Benefit Appeal
Submission Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_a_paradox2 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Reconsideration
Submission Date`) & !is.na(hdap_dqr$`Disability Benefit (B) - Subsequent
Benefit Appeal Submission Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Reconsideration Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Subsequent Benefit Appeal
Submission Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox2 = hdap_dqr[flag,]
```

- Subsequent Appeal Date later than Most Recent Appeal Date

```r
#subsequent appeal date later than most recent appeal date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Subsequent Benefit
Appeal Submission Date`) & !is.na(hdap_dqr$`Disability Benefit (A) - Most
Recent Appeal Submission Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Subsequent Benefit Appeal Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Most Recent Appeal Submission
Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_a_paradox3 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Subsequent Benefit
Appeal Submission Date`) & !is.na(hdap_dqr$`Disability Benefit (B) - Most
Recent Appeal Submission Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Subsequent Benefit Appeal Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Most Recent Appeal Submission
Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox3 = hdap_dqr[flag,]
```

- Most Recent Appeal Date later than Approval Date

```
#most recent appeal date later than approval date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (A) - Most Recent Appeal Submission
Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Most Recent Appeal Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_a_paradox4 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (B) - Most Recent Appeal Submission
Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Most Recent Appeal Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox4 = hdap_dqr[flag,]
```

- Subsequent Appeal Date later than Approval Date

```
#subsequent appeal date later than approval date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (A) - Subsequent Benefit Appeal
Submission Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Subsequent Benefit Appeal Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_a_paradox5 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (B) - Subsequent Benefit Appeal
Submission Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Subsequent Benefit Appeal Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox5 = hdap_dqr[flag,]
```

- Reconsideration Date later than Approval Date

```
#reconsideration date later than approval date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (A) - Reconsideration Submission
Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Reconsideration Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_a_paradox6 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (B) - Reconsideration Submission
Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Reconsideration Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox6 = hdap_dqr[flag,]
```

- Initial Submission Date later than Approval Date

```
#initial submission date later than approval date
flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (A) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (A) - Initial Application Submission
Date`))){
  if(hdap_dqr$`Disability Benefit (A) - Initial Application Submission
Date`[i]>=hdap_dqr$`Disability Benefit (A) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_a_paradox7 = hdap_dqr[flag,]

flag = c()
for(i in which(!is.na(hdap_dqr$`Disability Benefit (B) - Approval Date`) &
!is.na(hdap_dqr$`Disability Benefit (B) - Initial Application Submission
Date`))){
  if(hdap_dqr$`Disability Benefit (B) - Initial Application Submission
Date`[i]>=hdap_dqr$`Disability Benefit (B) - Approval Date`[i]){
    flag = c(flag,i)
  }
}
hdap_disability_date_b_paradox7 = hdap_dqr[flag,]
```

Final Denial Date with Other Variables

- Final Denial Date later than any of the disability benefit dates

```
flag = c()
for(i in 1:nrow(hdap_dqr)){
  if(any(!is.na(hdap_dqr[i,grep("(\\(a\\)) - [a-z, ]+ submission
date$",tolower(names(hdap)))])) & !is.na(hdap_dqr$`Disability Benefit (A) -
Final Denial Date`[i])){
    index = which(!is.na(hdap_dqr[i,grep("(\\(a\\)) - [a-z, ]+ submission
date$",tolower(names(hdap)))]))+36
    if(any(hdap_dqr[i,index]>=hdap_dqr$`Disability Benefit (A) - Final Denial
Date`[i])){
      flag = c(flag,i)
    }
  }
}
hdap_denial_a_paradox = hdap_dqr[flag,]

flag = c()
for(i in 1:nrow(hdap_dqr)){
  if(any(!is.na(hdap_dqr[i,grep("(\\(b\\)) - [a-z, ]+ submission
date$",tolower(names(hdap)))])) & !is.na(hdap_dqr$`Disability Benefit (B) -
Final Denial Date`[i])){
    index = which(!is.na(hdap_dqr[i,grep("(\\(b\\)) - [a-z, ]+ submission
date$",tolower(names(hdap)))]))+44
    if(any(hdap_dqr[i,index]>=hdap_dqr$`Disability Benefit (B) - Final Denial
Date`[i])){
      flag = c(flag,i)
    }
  }
}
hdap_denial_b_paradox = hdap_dqr[flag,]
```

- Final Denial Date missing while Reason for Final Denial exists

```
#denial date missing while the reason for denial present
hdap_denial_reason_a_paradox = hdap_dqr[which(is.na(hdap_dqr$`Disability
Benefit (A) - Final Denial Date`) & !is.na(hdap_dqr$`Disability Benefit (A) -
Reason for Final Denial`)),]
hdap_denial_reason_b_paradox = hdap_dqr[which(is.na(hdap_dqr$`Disability
Benefit (B) - Final Denial Date`) & !is.na(hdap_dqr$`Disability Benefit (B) -
Reason for Final Denial`)),]
```

- Reason for Final Denial says “Other”, while no further explanation exists.

```
#denial reason is "other" with no further explanation
hdap_denial_other_a_paradox = hdap_dqr[which(is.na(hdap_dqr$`Disability
Benefit (A) - Reason for Final Denial- Other`) & hdap_dqr$`Disability Benefit
(A) - Reason for Final Denial`=="Other"),]
hdap_denial_other_b_paradox = hdap_dqr[which(is.na(hdap_dqr$`Disability
Benefit (B) - Reason for Final Denial- Other`) & hdap_dqr$`Disability Benefit
(B) - Reason for Final Denial`=="Other"),]
```

Exit Date and Other Variables

- Exit Date earlier than Project Start Date

```
#project start date later than exit date
hdap_exit_paradox = hdap_dqr[which(hdap_dqr$`Project Start
Date`>=hdap_dqr$`Exit Date`),]
```

- Exit Date earlier than any of the Housing Dates (Move-Ins, Rehousing, Stabilized/Retained)

```
#exit date earlier than any of the housed dates (temporary, permanent,
rehoused)
flag = c()
for(i in which(!is.na(hdap_dqr$`Exit Date`))){
  if(any(!is.na(hdap_dqr[i,grep("housing.*date",tolower(names(hdap)))]))){
    index =
which(!is.na(hdap_dqr[i,grep("housing.*date",tolower(names(hdap)))]))+30
    if(any(hdap_dqr[i,index]>hdap_dqr$`Exit Date`[i])){
      flag = c(flag,i)
    }
  }
}
hdap_exit_paradox2 = hdap_dqr[flag,]
```

- Exit date missing when Destination present

```
#exit date missing when destination present
hdap_exit_paradox3 = hdap_dqr[which(is.na(hdap_dqr$`Exit Date`) &
!is.na(hdap_dqr$Destination)),]
```

- Exit Date later than the deadline of the most current HDAP report

```
#exit date later than the most current reporting period deadline (one month
after reporting period)
```

```
#make sure to change the date every quarter
hdap_exit_paradox4 = hdap_dqr[which(hdap_dqr$`Exit Date`>"2024-07-31"),]
```

Expenditure Info Missing

```
#check aggregate tab and see if there is expenditure info
hdap_county_without_expenditure = c()
# MAKE SURE to change the file_list, fy, data directory, and quarter
according to the proper fy and quarter
for(i in unique(hdap_dqr$County)){
  file_list = list.files("//cdss/feed/Central Office/HHCRB/HHB/Housing
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Grantee Reports/FY 23-24 Q4/")
  fy = "FY 2023-24"
  quarter_number = 4

  file_index = grep(tolower(i),tolower(file_list))
  if(length(file_index)==0){
    next
  }
  else if(length(file_index)>1){
    file_name =
file_list[grepl(tolower(i),tolower(file_list))&grepl("revised",tolower(file_l
ist))]
    if(grepl("xlsb",file_name)){
      library(readxlsb)
      data = read_xlsb(paste0("//Cdss/feed/Central Office/HHCRB/HHB/Housing
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Grantee Reports/FY 23-24
Q4/",file_name),sheet="Aggregate", skip = 2)
      names(data)[1:2] = gsub("\\."," ",names(data)[1:2])
    }
    else{
      data = read.xlsx(paste0("//Cdss/feed/Central Office/HHCRB/HHB/Housing
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Grantee Reports/FY 23-24
Q4/",file_name),sheet="Aggregate",check.names=F)
      names(data) = as.character(data[2,])
    }
    if(all(is.na(data[which(data$`Fiscal Year`==fy &
data$Quarter==quarter_number),3:ncol(data)]))){
      hdap_county_without_expenditure = c(hdap_county_without_expenditure,i)
    }
  }
  else{
    file_name = file_list[file_index]
    if(grepl("xlsb",file_name)){
      library(readxlsb)
      data = read_xlsb(paste0("//Cdss/feed/Central Office/HHCRB/HHB/Housing
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Grantee Reports/FY 23-24
Q4/",file_name),sheet="Aggregate", skip = 2)
      names(data)[1:2] = gsub("\\."," ",names(data)[1:2])
    }
    else{
      data = read.xlsx(paste0("//Cdss/feed/Central Office/HHCRB/HHB/Housing Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Grantee Reports/FY 23-24 Q4/",file_name),sheet="Aggregate",check.names=F)
      names(data) = as.character(data[2,])
    }
    if(all(is.na(data[which(data$`Fiscal Year`==fy & data$Quarter==quarter_number),3:ncol(data)]))){
      hdap_county_without_expenditure = c(hdap_county_without_expenditure,i)
    }
  }
}
```

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

```
#data quality report generation
hdap_error_table = data.frame()
hdap_error_table = error_table_generation(hdap_error_table,hdap_noncumulative[,c("County","First Name","Last Name","Project Start Date")],'Missing','Dropped Record',hdap_noncumulative$Quarter,paste0('Client was previously recorded in ',hdap_noncumulative$Quarter,' report and is missing in most recent report.'),'High')
if(nrow(hdap_negative_age)>0){
  hdap_error_table = error_table_generation(hdap_error_table,hdap_negative_age[,c("County","First Name","Last Name","Project Start Date")],"Invalid",'Date of Birth',as.character(hdap_negative_age$`Date of Birth`),"Date of Birth (Item 6) is later than Project Start Date (Item 29).",'High')
}
hdap_error_table = error_table_generation(hdap_error_table,hdap_start_paradox[,c("County","First Name","Last Name","Project Start Date")],"Invalid",'Housing Move-In Date - Permanent Housing',as.character(hdap_start_paradox$`Housing Move-In Date - Permanent Housing`),"Housing Move-In Date - Permanent Housing (Item 31) is earlier than Project Start Date (Item 29).",'High')
hdap_error_table = error_table_generation(hdap_error_table,hdap_start_paradox2[,c("County","First Name","Last Name","Project Start Date")],"Invalid",'Housing Stabilized/Retained Date',as.character(hdap_start_paradox2$`Housing Stabilized/Retained Date`),"Housing stabilized/retained date (Item 32) is earlier than Project Start Date (Item 29).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_start_paradox3_a[,c("County","Fi
rst Name","Last Name","Project Start Date")],"Invalid",'Disability Benefit
(A) - Approval Date',as.character(hdap_start_paradox3_a$`Disability Benefit
(A) - Approval Date`),"Disability Benefit (A) - Approval Date (Item 40) is
earlier than Project Start Date (Item 29).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_start_paradox3_b[,c("County","Fi
rst Name","Last Name","Project Start Date")],"Invalid",'Disability Benefit
(B) - Approval Date',as.character(hdap_start_paradox3_b$`Disability Benefit
(B) - Approval Date`),"Disability Benefit (B) - Approval Date (Item 48) is
earlier than Project Start Date (Item 29).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_housing_paradox[,c("County","Fir
st Name","Last Name","Project Start Date")],"Invalid",'Housing Move-In Date -
Temporary Housing',as.character(hdap_housing_paradox$`Housing Move-In Date -
Temporary Housing`),"Housing Move-In Date - Temporary Housing (Item 30) is
later than Housing Move-In Date - Permanent Housing (Item 31).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_stabilized_retained_at_risk[,c("
County","First Name","Last Name","Project Start Date")],"Invalid",'At risk of
experiencing homelessness',hdap_stabilized_retained_at_risk$`At risk of
experiencing homelessness`,"Housing Stabilized/Retained Date (Item 32)
exists, but the participant was not entered as 'Yes' for At risk of
homelessness (Item 22).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_rehoused_paradox[,c("County","Fi
rst Name","Last Name","Project Start Date")],"Invalid",'Rehousing
Date',as.character(hdap_rehoused_paradox$`Rehousing Date`),"Rehousing Date
(Item 33) is earlier than or equal to Housing Move-In Date - Permanent
Housing (Item 31).",'High')
if(nrow(hdap_rehoused_paradox2)>0){
  hdap_error_table =
error_table_generation(hdap_error_table,hdap_rehoused_paradox2[,c("County","F
irst Name","Last Name","Project Start Date")],"Invalid",'Rehousing
Date',as.character(hdap_rehoused_paradox2$`Rehousing Date`),"Rehousing Date
(Item 33) is earlier than or equal to Housing Stabilized/Retained Date (Item
32).",'High')
}
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox1[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Initial Application Submission
Date',as.character(hdap_disability_date_a_paradox1$`Disability Benefit (A) -
Initial Application Submission Date`),"Disability Benefit (A) - Initial
Application Submission Date (Item 36) is later than or equal to Disability
Benefit (A) - Reconsideration Submission Date (Item 37).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox1[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
```

```
Benefit (B) - Initial Application Submission
Date',as.character(hdap_disability_date_b_paradox1$`Disability Benefit (B) -
Initial Application Submission Date`),"Disability Benefit (B) - Initial
Application Submission Date (Item 44) is later than or equal to Disability
Benefit (B) - Reconsideration Submission Date (Item 45).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox7[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Initial Application Submission
Date',as.character(hdap_disability_date_a_paradox7$`Disability Benefit (A) -
Initial Application Submission Date`),"Disability Benefit (A) - Initial
Application Submission Date (Item 36) is later than or equal to Disability
Benefit (A) - Approval Date (Item 40).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox7[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (B) - Initial Application Submission
Date',as.character(hdap_disability_date_b_paradox7$`Disability Benefit (B) -
Initial Application Submission Date`),"Disability benefit (B) - Initial
Application Submission Date (Item 44) is later than or equal to Disability
Benefit (B) - Approval Date (Item 48).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox2[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Reconsideration Submission
Date',as.character(hdap_disability_date_a_paradox2$`Disability Benefit (A) -
Reconsideration Submission Date`),"Disability benefit (A) - Reconsideration
Submission Date (Item 37) is later than or equal to Disability Benefit (A) -
Subsequent Benefit Appeal Submission Date (Item 38).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox2[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (B) - Reconsideration Submission
Date',as.character(hdap_disability_date_b_paradox2$`Disability Benefit (B) -
Reconsideration Submission Date`),"Disability benefit (B) - Reconsideration
Submission Date (Item 45) is later than or equal to Disability Benefit (B) -
Subsequent Benefit Appeal Submission Date (Item 46).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox6[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Reconsideration Submission
Date',as.character(hdap_disability_date_a_paradox6$`Disability Benefit (A) -
Reconsideration Submission Date`),"Disability benefit (A) - Reconsideration
Submission Date (Item 37) is later than or equal to Disability Benefit (A) -
Approval Date (Item 40).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox6[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (B) - Reconsideration Submission
Date',as.character(hdap_disability_date_b_paradox6$`Disability Benefit (B) -
Reconsideration Submission Date`),"Disability benefit (B) - Reconsideration
Submission Date (Item 45) is later than or equal to Disability Benefit (B) -
Approval Date (Item 48).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox3[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Subsequent Benefit Appeal Submission
Date',as.character(hdap_disability_date_a_paradox3$`Disability Benefit (A) -
Subsequent Benefit Appeal Submission Date`),"Disability benefit (A) -
Subsequent Benefit Appeal Submission Date (Item 38) is later than or equal to
Disability Benefit (A) - Most Recent Appeal Submission Date (Item
39).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox3[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (B) - Subsequent Benefit Appeal Submission
Date',as.character(hdap_disability_date_b_paradox3$`Disability Benefit (B) -
Subsequent Benefit Appeal Submission Date`),"Disability Benefit (B) -
Subsequent Benefit Appeal Submission Date (Item 46) is later than or equal to
Disability Benefit (B) - Most Recent Appeal Submission Date (Item
47).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox5[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Subsequent Benefit Appeal Submission
Date',as.character(hdap_disability_date_a_paradox5$`Disability Benefit (A) -
Subsequent Benefit Appeal Submission Date`),"Disability Benefit (A) -
Subsequent Benefit Appeal Submission Date (Item 38) is later than or equal to
Disability Benefit (A) - Approval Date (Item 40).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox5[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (B) - Subsequent Benefit Appeal Submission
Date',as.character(hdap_disability_date_b_paradox5$`Disability Benefit (B) -
Subsequent Benefit Appeal Submission Date`),"Disability Benefit (B) -
Subsequent Benefit Appeal Submission Date (Item 46) is later than or equal to
Disability Benefit (B) - Approval Date (Item 48).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_a_paradox4[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (A) - Most Recent Appeal Submission
Date',as.character(hdap_disability_date_a_paradox4$`Disability Benefit (A) -
Most Recent Appeal Submission Date`),"Disability Benefit (A) - Most Recent
Appeal Submission Date (Item 39) is later than or equal to Disability Benefit
(A) - Approval Date (Item 40).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_disability_date_b_paradox4[,c("C
ounty","First Name","Last Name","Project Start Date")],"Invalid",'Disability
Benefit (B) - Most Recent Appeal Submission
Date',as.character(hdap_disability_date_b_paradox4$`Disability Benefit (B) -
Most Recent Appeal Submission Date`),"Disability Benefit (B) - Most Recent
Appeal Submission Date (Item 47) is later than or equal to Disability Benefit
(B) - Approval Date (Item 48).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_denial_a_paradox[,c("County","Fi
rst Name","Last Name","Project Start Date")],"Invalid",'Disability Benefit
(A) - Final Denial Date',as.character(hdap_denial_a_paradox$`Disability
Benefit (A) - Final Denial Date`),"Disability Benefit (A) - Final Denial Date
(Item 41) is earlier than or equal to any of the Disability Benefit (A)
Submission Dates (Items 36-39).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_denial_b_paradox[,c("County","Fi
rst Name","Last Name","Project Start Date")],"Invalid",'Disability Benefit
(B) - Final Denial Date',as.character(hdap_denial_b_paradox$`Disability
Benefit (B) - Final Denial Date`),"Disability Benefit (B) - Final Denial Date
(Item 49) is earlier than or equal to any of the Disability Benefit (B)
Submission Dates (Items 44-47).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_denial_reason_a_paradox[,c("Coun
ty","First Name","Last Name","Project Start Date")],"Missing",'Disability
Benefit (A) - Final Denial Date','',"Disability Benefit (A) - Final Denial
Date (Item 41) is missing when Disability Benefit (A) - Reason for Final
Denial (Item 42) exists.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_denial_reason_b_paradox[,c("Coun
ty","First Name","Last Name","Project Start Date")],"Missing",'Disability
Benefit (B) - Final Denial Date','',"Disability Benefit (B) - Final Denial
Date (Item 49) is missing when Disability Benefit (B) - Reason for Final
Denial (Item 50) exists.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_denial_other_a_paradox[,c("Count
y","First Name","Last Name","Project Start Date")],"Missing",'Disability
Benefit (A) - Reason for Final Denial - Other','',"Disability Benefit (A) -
Reason for Final Denial (Item 42) says 'Other', but further explanation is
missing.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_denial_other_b_paradox[,c("Count
y","First Name","Last Name","Project Start Date")],"Missing",'Disability
Benefit (B) - Reason for Final Denial - Other','',"Disability Benefit (B) -
Reason for Final Denial (Item 50) says 'Other', but further explanation is
missing.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_exit_paradox[,c("County","First
Name","Last Name","Project Start Date")],"Invalid",'Exit
Date',as.character(hdap_exit_paradox$`Exit Date`),"Exit Date (Item 52) is
earlier than Project Start Date (Item 29).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_exit_paradox2[,c("County","First
Name","Last Name","Project Start Date")],"Invalid",'Exit
Date',as.character(hdap_exit_paradox2$`Exit Date`),"Exit Date (Item 52) is
```

```
earlier than any housing date(s) (Items 30-33).",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_exit_paradox3[,c("County","First
Name","Last Name","Project Start Date")],"Missing",'Exit Date','',"Exit Date
(Item 52) is missing when Destination (Item 53) exists.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_exit_paradox4[,c("County","First
Name","Last Name","Project Start Date")],"Invalid",'Exit
Date',as.character(hdap_exit_paradox4$`Exit Date`),"Exit Date (Item 52) is
later than current reporting period.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_homeless_status_paradox[,c("Coun
ty","First Name","Last Name","Project Start Date")],"Invalid",'At risk of
experiencing homelessness & Experiencing homelessness',"Both 'Yes'","At risk
of experiencing homelessness (Item 22) and Experiencing homelessness (Item
23) cannot be both 'Yes' for any given participant.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_homeless_status_paradox2[,c("Cou
nty","First Name","Last Name","Project Start Date")],"Invalid",'At risk of
experiencing homelessness & Experiencing homelessness',"Both 'No'","At risk
of experiencing homelessness (Item 22) and Experiencing homelessness (Item
23) cannot be both 'No' for any given participant.",'High')
hdap_error_table =
error_table_generation(hdap_error_table,hdap_drop_down_off_no_start_date[,c("
County","First Name","Last Name","Project Start Date")],"Missing",'Project
Start Date','',"The response is missing and this is a required
element.","High")
hdap_error_table =
error_table_generation(hdap_error_table,hdap_drop_down_off_empty[,c("County",
"First Name","Last Name","Project Start
Date")],"Missing",hdap_drop_down_off_empty$discrepant_var,'',"The response is
missing and this is a required element.","High")
if(nrow(hdap_drop_down_off_invalid)>0){
  hdap_error_table =
error_table_generation(hdap_error_table,hdap_drop_down_off_invalid[,c("County
","First Name","Last Name","Project Start
Date")],"Invalid",hdap_drop_down_off_invalid$discrepant_var,hdap_drop_down_of
f_invalid$value,"This response is not listed as an option from ACIN.","High")
}
hdap_error_table =
error_table_generation(hdap_error_table,hdap_ssn_missing[,c("County","First
Name","Last Name","Project Start Date")],"Missing","Social Security
Number",'',"The response is missing and this is an important
element.","Medium")
hdap_error_table =
error_table_generation(hdap_error_table,hdap_ssn_invalid[,c("County","First
Name","Last Name","Project Start Date")],"Invalid","Social Security
Number",hdap_ssn_invalid$`Social Security Number`,"The response is not in the
proper SSN form.","Medium")
hdap_error_table =
```

```r
error_table_generation(hdap_error_table,hdap_duplicate_data_same_period_tbd[,
c("County","First Name","Last Name","Project Start Date")],'Duplicate
record','Quarter',hdap_duplicate_data_same_period_tbd$`Quarter Start`,'This
participant was found twice (or more than twice) in the report. The records
have the same Project Start Date (Item 29).','High')
if(nrow(hdap_no_start_date_duplicate)>0){
  hdap_error_table =
error_table_generation(hdap_error_table,hdap_no_start_date_duplicate[,c("Coun
ty","First Name","Last Name","Project Start Date")],'Duplicate
record','Quarter',hdap_no_start_date_duplicate$`Quarter Start`,'This
participant was found twice (or more than twice) in the report. The records
have the same Project Start Date (Item 29).','High')
}
hdap_county_without_expenditure = data.frame(hdap_county_without_expenditure)
names(hdap_county_without_expenditure) = "County"
hdap_county_without_expenditure$
hdap_error_table =
error_table_generation(hdap_error_table,hdap_county_without_expenditure)

hdap_error_table$`Additional Information/Instructions`[grep("^Participant
information",hdap_error_table$`Error Description`)] = "Please confirm the
participant's first name, last name, and dob to see if accurate."
hdap_error_table$`Additional Information/Instructions`[grep("^Client was
previously",hdap_error_table$`Error Description`)] = "Please enter this
record from the given report to the upcoming report."
hdap_error_table$`Additional Information/Instructions`[grep("date.*later
than",tolower(hdap_error_table$`Error Description`))] = "Please check both
dates before fixing either of the variables."
hdap_error_table$`Additional Information/Instructions`[grep("date.*earlier
than",tolower(hdap_error_table$`Error Description`))] = "Please check both
dates before fixing either of the variables."
hdap_error_table$`Additional Information/Instructions`[grep("^This response
is not listed as an option",hdap_error_table$`Error Description`)] = "Please
re-enter the proper option from ACIN."
hdap_error_table$`Additional Information/Instructions`[grep("proper
SSN",hdap_error_table$`Error Description`)] = "Please fill in the full SSN
with your best knowledge."
hdap_error_table$`Additional Information/Instructions`[grep("^This
participant was found twice",hdap_error_table$`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')."
hdap_error_table$`Additional Information/Instructions`[grep("cannot be
both",hdap_error_table$`Error Description`)] = "Please review both Item 22
and Item 23 for the participant and enter proper responses."
hdap_error_table$`Error Priority`[grepl("^social
sec",tolower(hdap_error_table$Item)) | grepl("^exit due
to",tolower(hdap_error_table$Item))]="Medium"
hdap_error_table = unique(hdap_error_table[order(hdap_error_table$County),])
```

```r
hdap_error_table_la = subset(hdap_error_table,grepl("Los A",County))
hdap_error_table = subset(hdap_error_table,!grepl("Los A",County))
```

- Just as hdap_key_info_discrepant table was generated, error_table_generation function was used to generate this table as well.
- hdap_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, Project 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
##Saving DQR
#Please update the proper file name as the quarter progresses
new_location = "//cdss/feed/Central Office/HHCRB/HHB/Housing 
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Data Quality Reports/Generated 
DQRs/FY 23-24 Q4/"
for(i in unique(c(hdap_error_table$County,"Los 
Angeles",hdap_participant_discrepant$County))){
  file = "//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/HDAP/HDAP 
Data/HDAP PII 21 Reports/Data Quality Reports/HDAP 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/HDAP/HDAP Data/HDAP PII 21 Reports/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),]
  errors_summary_dup = read.xlsx(file,sheet="Errors-Summary")[1,4:6]
  names(errors_summary_ch) = as.character(errors_summary_ch[1,])
  names(errors_summary_dup) = as.character(errors_summary_dup[1,])

  #subsetting the data based on the counties
  if(i=="Los Angeles"){
    data = hdap_error_table_la
    data_aggregate = sqldf("select Item,`Error Description`,count(Item) as 
num from data group by Item,`Error Description`")
    data_aggregate2 = sqldf("select County,Item,count(Item) as `# Records` 
from hdap_key_info_discrepant group by County,Item")
  }
  else{
    data = subset(hdap_error_table,County==i)
    data_aggregate = sqldf("select Item,`Error Description`,count(Item) as
num from data group by Item,`Error Description`")
    data_aggregate2 = sqldf("select County,Item,count(Item) as `# Records`
from hdap_key_info_discrepant group by County,Item")
  }

  #putting together aggregate table for DQR "Errors-Summary" sheet
  #filling in var names for invalid rows
  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)]
    }
  }

  for(r in 1:nrow(data_aggregate)){
    if(!grepl("^Dropped",data_aggregate$Item[r])
       & !grepl("^Quarter",data_aggregate$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]
    }
    if(grepl("^Dropped",data_aggregate$Item[r])){
      errors_summary_ch$`#
Errors`[grep("^Dropped",errors_summary_ch$Item)]=length(grep("^Dropped",data$
Item))
    }
  }

  style = createStyle(halign="left",valign="top",wrapText = T)
  removeWorksheet(wb,"Case History-Errors")
  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)
  writeData(wb,"Errors-
Summary",data_aggregate2[data_aggregate2$County==i,2:ncol(data_aggregate2)],s
tartCol=11,startRow=3,colNames = F)
  if(length(grep("Duplicate",data$`Error Type`))>0){
    errors_summary_dup =
rbind(errors_summary_dup,data[grep("Duplicate",data$`Error Type`),c("First
Name","Last Name","Project Start Date")])
    writeData(wb,"Errors-
Summary",errors_summary_dup[2:nrow(errors_summary_dup),],startCol=6,startRow=
3,colNames = F)
    removeWorksheet(wb,"Duplicates")
    addWorksheet(wb,"Duplicates")
    writeDataTable(wb,"Duplicates",data[grep("Duplicate",data$`Error
Type`),],tableStyle = "TableStyleMedium2")
    addStyle(wb,"Duplicates",style,2:(nrow(data[grep("Duplicate",data$`Error
Type`),])+1),1:ncol(data),gridExpand = T)
    addWorksheet(wb,"Case History-Errors")
    writeDataTable(wb,"Case History-Errors",data[-
grep("Duplicate",data$`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)
  setColWidths(wb,"Duplicates",8,32.7)
  setColWidths(wb,"Duplicates",9,48.5)
  setColWidths(wb,"Duplicates",11,12)

  removeWorksheet(wb,"Key Information Discrepancies")
  addWorksheet(wb,"Key Information Discrepancies")
  writeDataTable(wb,"Key Information
Discrepancies",hdap_key_info_discrepant[grep(i,hdap_key_info_discrepant$Count
y),],tableStyle = "TableStyleMedium2")
  addStyle(wb,"Key Information
Discrepancies",style,2:(nrow(hdap_key_info_discrepant[grep(i,hdap_key_info_di
screpant$County),])+1),1:ncol(hdap_key_info_discrepant[grep(i,hdap_key_info_d
iscrepant$County),]),gridExpand = T)
  if(length(grep(i,hdap_key_info_discrepant$County))>0){
    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)
  }
  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