package com.twitter.interaction\_graph.scio.common

import com.spotify.scio.coders.Coder

import com.spotify.scio.values.SCollection

import com.twitter.twadoop.user.gen.thriftscala.CombinedUser

import com.twitter.usersource.snapshot.flat.thriftscala.FlatUser

object UserUtil {

/\*\*

\* placeholder for the destId when representing vertex features with no dest (eg create tweet)

\* this will only be aggregated and saved in the vertex datasets but not the edge datasets

\*/

val DUMMY\_USER\_ID = -1L

def getValidUsers(users: SCollection[CombinedUser]): SCollection[Long] = {

users

.flatMap { u =>

for {

user <- u.user

if user.id != 0

safety <- user.safety

if !(safety.suspended || safety.deactivated || safety.restricted ||

safety.nsfwUser || safety.nsfwAdmin || safety.erased)

} yield {

user.id

}

}

}

def getValidFlatUsers(users: SCollection[FlatUser]): SCollection[Long] = {

users

.flatMap { u =>

for {

id <- u.id

if id != 0 && u.validUser.contains(true)

} yield {

id

}

}

}

def getInvalidUsers(users: SCollection[FlatUser]): SCollection[Long] = {

users

.flatMap { user =>

for {

valid <- user.validUser

if !valid

id <- user.id

} yield id

}

}

def filterUsersByIdMapping[T: Coder](

input: SCollection[T],

usersToBeFiltered: SCollection[Long],

userIdMapping: T => Long

): SCollection[T] = {

input

.withName("filter users by id")

.keyBy(userIdMapping(\_))

.leftOuterJoin[Long](usersToBeFiltered.map(x => (x, x)))

.collect {

// only return data if the key is not in the list of usersToBeFiltered

case (\_, (data, None)) => data

}

}

def filterUsersByMultipleIdMappings[T: Coder](

input: SCollection[T],

usersToBeFiltered: SCollection[Long],

userIdMappings: Seq[T => Long]

): SCollection[T] = {

userIdMappings.foldLeft(input)((data, mapping) =>

filterUsersByIdMapping(data, usersToBeFiltered, mapping))

}

}