Lesson 3 Solutions

Stefanie Molin April 14, 2017

Let's do some practice problems to challenge your understanding.

1. Query for two dataframes: (1) all AS in your office along with their employee IDs and (2) the accounts in the US and the AS employee ID associated with them. Use dplyr filtering joins to (a) preview the results that will be lost from dataframe (1) if you do an inner join on both tables and (b) preview the results that will remain in dataframe (1) if you do an inner join. (c) inner join dataframes (1) and (2) and confirm your results.

library(dplyr)

```
# packages have been loaded along with QueryVertica()
# username/password have been predefined
# query for all AS in NY office
ny_as_query <- "
SELECT
    employee_id
    , full_name
FROM
WHERE
   cost_center_country = 'NY'
   AND job name =
GROUP BY
   employee id
    , full_name
nyAS <- QueryVertica(username, ny as query, password)
# query for US accounts
us_accounts_query <- "
SELECT
FROM
WHERE
   ranking = 'TIER 1'
   AND client country code = 'US'
GROUP BY
usAccounts <- QueryVertica(username, us_accounts_query, password)
```

```
# look at number of rows in each table
nrow(nyAS)
## [1] 16
nrow(usAccounts)
## [1] 867
# preview what will be lost from nyAS in inner join above (these AS don't have accounts)
asWithoutAccounts <- anti_join(nyAS, usAccounts,</pre>
                           by = c("employee_id" = "account_strategist_employee_id"))
nrow(asWithoutAccounts)
## [1] 1
head(asWithoutAccounts)
                       full name
     emplovee id
## 1
# preview what will be kept from nyAS in inner join above (all these AS have accounts)
asWithAccounts <- semi_join(nyAS, usAccounts,
                           by = c("employee_id" = "account_strategist_employee_id"))
nrow(asWithAccounts)
## [1] 15
head(asWithAccounts)
                               full_name
##
     employee_id
## 1
## 2
## 3
## 4
## 5
## 6
# inner join the two tables
nyAccounts <- inner_join(nyAS, usAccounts,</pre>
                           by = c("employee_id" = "account_strategist_employee_id"))
nrow(nyAccounts)
## [1] 149
head(nyAccounts)
##
     employee_id
                    full_name
                                         merchant_name
## 1
## 2
## 3
## 4
## 5
## 6
# how many employees are left after the join
length(unique(nyAccounts$employee_id))
```

[1] 15

2. Pull in the first names of every employee *currently* working at and (cost centers US, NY, IL, SF), and, in a second dataframe, the first names of every employee that currently works at but *not* in the US. Be sure to write a dynamic query, so that you only have to write one query! Use an rbind() to get the complete list of employee first names in a separate dataframe.

```
# dynamic query for first names
name_query <- "</pre>
SELECT
    first_name
FROM
WHERE
    cost_center_country %s
  AND job_status = 'LIVE'
usNames <- QueryVertica(username,
                         sprintf(name_query, " IN ('US', 'NY', 'IL', 'SF')"),
                         password)
notUSNames <- QueryVertica(username,</pre>
                                                  "NOT IN ('US', 'NY', 'IL', 'SF')"),
                            sprintf(name_query,
                            password)
# all employee first names
allNames <- rbind(usNames, notUSNames)</pre>
head(allNames)
```

```
## first_name

## 1

## 2

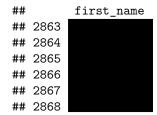
## 3

## 4

## 5

## 6
```

tail(allNames)



3. Using the two dataframes you queried for in (2) and set operations, (a) find all first names that are either in (cost centers US, NY, IL, SF) or any other office, but **not** in both; (b) count how many people have each name, and sort it from most common to least common and by name alphabetically. Then, (c) flag and return the top 10 most common along with their counts, and (d) find the first names of the employees that are the only one in the company with that name, and (e) compare this result to the result from (a). (Hint use setequal()).

```
OR elsewhere (a)
# names either in
noOverlap <- usNames %>%
  union(notUSNames) %>%
  setdiff(intersect(usNames, notUSNames))
head(noOverlap)
##
## 1
## 2
## 3
## 4
## 5
##
# count names and sort (b)
nameCount <- allNames %>%
  select(first_name) %>%
  group_by(first_name) %>%
  summarize(count = n()) %>%
  arrange(desc(count), first_name)
# top 10 most common (c)
head(nameCount, 10) %>% mutate(top_10 = TRUE)
## # A tibble: 10 × 3
##
      first_name count top_10
##
           <chr> <int>
                         <1g1>
## 1
                    28
                          TRUE
## 2
                    27
                          TRUE
## 3
                    27
                          TRUE
                    26
## 4
                          TRUE
                    26
## 5
                          TRUE
                    21
## 6
                          TRUE
## 7
                    17
                          TRUE
## 8
                     16
                          TRUE
                    16
## 9
                          TRUE
                    16
                          TRUE
# people with unique first names (d)
uniqueNames <- nameCount %>%
  filter(count == 1) %>%
  select(first_name)
# compare result from (d) to (a)
setequal(noOverlap, uniqueNames)
```

FALSE: Different number of rows