

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
  [redacted]
WHERE
  cost_center_country = 'NY'
  AND job_name = [redacted]
GROUP BY
  employee_id
  , full_name
"

nyAS <- QueryVertica(username, ny_as_query, password)

# query for US accounts
us_accounts_query <- "
SELECT
  [redacted]
FROM
  [redacted]
WHERE
  ranking = 'TIER 1'
  AND client_country_code = 'US'
GROUP BY
  [redacted]
"

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,
                               by = c("employee_id" = "account_strategist_employee_id"))
nrow(asWithoutAccounts)

## [1] 1

head(asWithoutAccounts)

##   employee_id      full_name
## 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)

##   employee_id      full_name
## 1 
## 2 
## 3 
## 4 
## 5 
## 6 

# inner join the two tables
nyAccounts <- inner_join(nyAS, usAccounts,
                          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 [REDACTED] (cost centers US, NY, IL, SF), and, in a second dataframe, the first names of every employee that currently works at [REDACTED] 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 <- "
SELECT
  first_name
FROM
  [REDACTED]
WHERE
  cost_center_country %s
  AND job_status = 'LIVE'
"

usNames <- QueryVertica(username,
                        sprintf(name_query, " IN ('US', 'NY', 'IL', 'SF')"),
                        password)
notUSNames <- QueryVertica(username,
                           sprintf(name_query, "NOT IN ('US', 'NY', 'IL', 'SF')"),
                           password)

# all employee first names
allNames <- rbind(usNames, notUSNames)
head(allNames)
```

```
##      first_name
## 1 [REDACTED]
## 2 [REDACTED]
## 3 [REDACTED]
## 4 [REDACTED]
## 5 [REDACTED]
## 6 [REDACTED]
```

```
tail(allNames)
```

```
##      first_name
## 2863 [REDACTED]
## 2864 [REDACTED]
## 2865 [REDACTED]
## 2866 [REDACTED]
## 2867 [REDACTED]
## 2868 [REDACTED]
```

3. Using the two dataframes you queried for in (2) and set operations, (a) find all first names that are either in [REDACTED] (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()`).

```
# names either in [REDACTED] OR elsewhere (a)
noOverlap <- usNames %>%
  union(notUSNames) %>%
  setdiff(intersect(usNames, notUSNames))
head(noOverlap)
```

```
##      first_name
## 1 [REDACTED]
## 2 [REDACTED]
## 3 [REDACTED]
## 4 [REDACTED]
## 5 [REDACTED]
## 6 [REDACTED]
```

```
# 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> <lgl>
## 1 [REDACTED]    28  TRUE
## 2 [REDACTED]    27  TRUE
## 3 [REDACTED]    27  TRUE
## 4 [REDACTED]    26  TRUE
## 5 [REDACTED]    26  TRUE
## 6 [REDACTED]    21  TRUE
## 7 [REDACTED]    17  TRUE
## 8 [REDACTED]    16  TRUE
## 9 [REDACTED]    16  TRUE
## 10 [REDACTED]    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
```