Get Cases with Revascularization procedure codes

```
library(tidyverse)
library(bigrquery)

# This query represents dataset "CAD_revascularization" for domain
"procedure" and was generated for All of Us Controlled Tier Dataset v7
dataset_47740110_procedure_sql <- paste("
    SELECT
        procedure.person_id,
        procedure.procedure_concept_id,
        p_standard_concept.concept_name as standard_concept_name,
        p_standard_concept.concept_code as standard_concept_code,
        p_standard_concept.vocabulary_id as standard_vocabulary
    FROM
        ( SELECT
            *
        FROM
            `procedure_occurrence` procedure
        WHERE
            (
                procedure_concept_id IN (SELECT
                    DISTINCT c.concept_id
                FROM
                    `cb_criteria` c
                JOIN
                    (SELECT
                        CAST(cr.id as string) AS id
                    FROM
                        `cb_criteria` cr
                    WHERE
                        concept_id IN (4034857, 4217445, 4219321,
4284104, 43533186, 36969009,
                        43533187, 43533223, 43533242, 43533354,
41339005, 14201006, 276861004)
                        AND full_text LIKE '%_rank1]%'      ) a
                        ON (c.path LIKE CONCAT('%.', a.id, '.%')
                        OR c.path LIKE CONCAT('%.', a.id)
                        OR c.path LIKE CONCAT(a.id, '.%')
                        OR c.path = a.id)
                WHERE
                    is_standard = 1
                    AND is_selectable = 1)
            )
            AND (
```

```
                procedure.PERSON_ID IN (SELECT
                    distinct person_id
                FROM
                    `cb_search_person` cb_search_person
                WHERE
                    cb_search_person.person_id IN (SELECT
                        person_id
                    FROM
                        `cb_search_person` p
                    WHERE
                        has_whole_genome_variant = 1 )
                    AND cb_search_person.person_id IN (SELECT
                        person_id
                    FROM
                        `cb_search_person` p
                    WHERE
                        has_ehr_data = 1 ) )
            )) procedure
    LEFT JOIN
        `concept` p_standard_concept
            ON procedure.procedure_concept_id =
p_standard_concept.concept_id", sep="")

# Formulate a Cloud Storage destination path for the data exported
from BigQuery.
# NOTE: By default data exported multiple times on the same day will
overwrite older copies.
#       But data exported on a different days will write to a new
location so that historical
#       copies can be kept as the dataset definition is changed.
procedure_47740110_path <- file.path(
  Sys.getenv("WORKSPACE_BUCKET"),
  "bq_exports",
  Sys.getenv("OWNER_EMAIL"),
  strftime(lubridate::now(), "%Y%m%d"),  # Comment out this line if
you want the export to always overwrite.
  "procedure_47740110",
  "procedure_47740110_*.csv")
message(str_glue('The data will be written to
{procedure_47740110_path}. Use this path when reading ',
                 'the data into your notebooks in the future.'))

# Perform the query and export the dataset to Cloud Storage as CSV
files.
# NOTE: You only need to run `bq_table_save` once. After that, you can
#       just read data from the CSVs in Cloud Storage.
bq_table_save(
  bq_dataset_query(Sys.getenv("WORKSPACE_CDR"),
dataset_47740110_procedure_sql, billing =
```

```
Sys.getenv("GOOGLE_PROJECT")),
  procedure_47740110_path,
  destination_format = "CSV")


# Read the data directly from Cloud Storage into memory.
# NOTE: Alternatively you can `gsutil -m cp {procedure_47740110_path}`
to copy these files
#       to the Jupyter disk.
read_bq_export_from_workspace_bucket <- function(export_path) {
  col_types <- cols(standard_concept_name = col_character(),
standard_concept_code = col_character(), standard_vocabulary =
col_character())
  bind_rows(
    map(system2('gsutil', args = c('ls', export_path), stdout = TRUE,
stderr = TRUE),
        function(csv) {
          message(str_glue('Loading {csv}.'))
          chunk <- read_csv(pipe(str_glue('gsutil cat {csv}')),
col_types = col_types, show_col_types = FALSE)
          if (is.null(col_types)) {
            col_types <- spec(chunk)
          }
          chunk
        }))
}
procedure_df <-
read_bq_export_from_workspace_bucket(procedure_47740110_path)

dim(procedure_df)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.2     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.4
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors
The data will be written to gs://fc-secure-672eeb92-4859-4ed9-9f59-
d4349f3534a0/bq_exports/micah_hysong@researchallofus.org/20250825/
procedure_47740110/procedure_47740110_*.csv. Use this path when
reading the data into your notebooks in the future.

Loading
```

```
gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/bq_exports/micah_hysong@researchallofus.org/20250825/procedure_47740110/procedure_47740110_000000000000.csv.


[1] 2528    5

unique(procedure_df$standard_concept_name)

 [1] "Myocardial revascularization"

 [2] "Aortocoronary bypass of two coronary arteries"

 [3] "Aortocoronary artery bypass graft"

 [4] "Percutaneous transluminal revascularization of chronic total
occlusion, coronary artery, coronary artery branch, or coronary artery
bypass graft, any combination of drug-eluting intracoronary stent,
atherectomy and angioplasty; each additional coronary..."
 [5] "Percutaneous transluminal revascularization of or through
coronary artery bypass graft (internal mammary, free arterial,
venous), any combination of drug-eluting intracoronary stent,
atherectomy and angioplasty, including distal protection when
perform..."
 [6] "Aortocoronary bypass of three coronary arteries"

 [7] "Transmyocardial revascularization by laser technique"

 [8] "Aortocoronary bypass of four or more coronary arteries"

 [9] "Aortocoronary bypass of one coronary artery"

[10] "Percutaneous transluminal revascularization of chronic total
occlusion, coronary artery, coronary artery branch, or coronary artery
bypass graft, any combination of drug-eluting intracoronary stent,
atherectomy and angioplasty; single vessel"
[11] "Anastomosis of internal mammary artery to coronary artery,
double vessel"

[12] "Percutaneous transluminal revascularization of acute
total/subtotal occlusion during acute myocardial infarction, coronary
artery or coronary artery bypass graft, any combination of drug-
eluting intracoronary stent, atherectomy and angioplasty, includi..."

case_definition1<-unique(procedure_df$person_id)
length(case_definition1)

[1] 704
```

Case Definition 2: hard diagnostic codes on 2 separate instances in the EHR

```
library(tidyverse)
library(bigrquery)

# This query represents dataset "CAD_conditions" for domain
"condition" and was generated for All of Us Controlled Tier Dataset v7
dataset_32581449_condition_sql <- paste("
    SELECT
        c_occurrence.person_id,
        c_occurrence.condition_concept_id,
        c_standard_concept.concept_name as standard_concept_name,
        c_standard_concept.concept_code as standard_concept_code,
        c_standard_concept.vocabulary_id as standard_vocabulary,
        c_occurrence.condition_start_datetime
    FROM
        ( SELECT
            *
        FROM
            `condition_occurrence` c_occurrence
        WHERE
            (
                condition_concept_id IN (SELECT
                    DISTINCT c.concept_id
                FROM
                    `cb_criteria` c
                JOIN
                    (SELECT
                        CAST(cr.id as string) AS id
                    FROM
                        `cb_criteria` cr
                    WHERE
                        concept_id IN (312327, 314666, 319038,
37311078, 4108220, 4108679, 4108680, 4119462, 4119606, 4119953,
4121477, 4124687, 4158567, 4161462, 4178622, 4198141, 4322145,
4329847, 4353828, 438172, 46269996)
                        AND full_text LIKE '%_rank1]%'      ) a
                        ON (c.path LIKE CONCAT('%.', a.id, '.%')
                        OR c.path LIKE CONCAT('%.', a.id)
                        OR c.path LIKE CONCAT(a.id, '.%')
                        OR c.path = a.id)
                WHERE
                    is_standard = 1
                    AND is_selectable = 1)
            )
            AND (
                c_occurrence.PERSON_ID IN (SELECT
```

```
                    distinct person_id
                FROM
                    `cb_search_person` cb_search_person
                WHERE
                    cb_search_person.person_id IN (SELECT
                        person_id
                    FROM
                        `cb_search_person` p
                    WHERE
                        has_whole_genome_variant = 1 )
                    AND cb_search_person.person_id IN (SELECT
                        person_id
                    FROM
                        `cb_search_person` p
                    WHERE
                        has_ehr_data = 1 ) )
            )) c_occurrence
    LEFT JOIN
        `concept` c_standard_concept
            ON c_occurrence.condition_concept_id =
c_standard_concept.concept_id", sep="")

# Formulate a Cloud Storage destination path for the data exported
from BigQuery.
# NOTE: By default data exported multiple times on the same day will
overwrite older copies.
#       But data exported on a different days will write to a new
location so that historical
#       copies can be kept as the dataset definition is changed.
condition_32581449_path <- file.path(
  Sys.getenv("WORKSPACE_BUCKET"),
  "bq_exports",
  Sys.getenv("OWNER_EMAIL"),
  strftime(lubridate::now(), "%Y%m%d"),  # Comment out this line if
you want the export to always overwrite.
  "condition_32581449",
  "condition_32581449_*.csv")
message(str_glue('The data will be written to
{condition_32581449_path}. Use this path when reading ',
                 'the data into your notebooks in the future.'))

# Perform the query and export the dataset to Cloud Storage as CSV
files.
# NOTE: You only need to run `bq_table_save` once. After that, you can
#       just read data from the CSVs in Cloud Storage.
bq_table_save(
  bq_dataset_query(Sys.getenv("WORKSPACE_CDR"),
dataset_32581449_condition_sql, billing =
Sys.getenv("GOOGLE_PROJECT")),
```

```
  condition_32581449_path,
  destination_format = "CSV")


# Read the data directly from Cloud Storage into memory.
# NOTE: Alternatively you can `gsutil -m cp {condition_32581449_path}`
to copy these files
#       to the Jupyter disk.
read_bq_export_from_workspace_bucket <- function(export_path) {
  col_types <- cols(standard_concept_name = col_character(),
standard_concept_code = col_character(), standard_vocabulary =
col_character())
  bind_rows(
    map(system2('gsutil', args = c('ls', export_path), stdout = TRUE,
stderr = TRUE),
        function(csv) {
          message(str_glue('Loading {csv}.'))
          chunk <- read_csv(pipe(str_glue('gsutil cat {csv}')),
col_types = col_types, show_col_types = FALSE)
          if (is.null(col_types)) {
            col_types <- spec(chunk)
          }
          chunk
        }))
}
condition_df <-
read_bq_export_from_workspace_bucket(condition_32581449_path)

dim(condition_df)

names(condition_df)<-c("person_id", "concept_id",
"standard_concept_name", "standard_concept_code",
"standard_vocabulary", "datetime")
```

```
The data will be written to gs://fc-secure-672eeb92-4859-4ed9-9f59-
d4349f3534a0/bq_exports/micah_hysong@researchallofus.org/20250825/
condition_32581449/condition_32581449_*.csv. Use this path when
reading the data into your notebooks in the future.

Loading
gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/bq_exports/micah_h
ysong@researchallofus.org/20250825/condition_32581449/
condition_32581449_000000000000.csv.


[1] 1273809       6
```

```
library(tidyverse)
library(bigrquery)

# This query represents dataset "CAD_observations" for domain
```

```
"observation" and was generated for All of Us Controlled Tier Dataset
v7
dataset_00963032_observation_sql <- paste("
    SELECT
        observation.person_id,
        observation.observation_concept_id,
        o_standard_concept.concept_name as standard_concept_name,
        o_standard_concept.concept_code as standard_concept_code,
        o_standard_concept.vocabulary_id as standard_vocabulary,
        observation.observation_datetime
    FROM
        ( SELECT
            *
        FROM
            `observation` observation
        WHERE
            (
                observation_concept_id IN (4163883, 4177223, 4179076,
4180760, 4181951, 43022009, 4324192, 4324758, 44782697, 44782699,
44782701, 44782708, 46269964)
            )
            AND (
                observation.PERSON_ID IN (SELECT
                    distinct person_id
                FROM
                    `cb_search_person` cb_search_person
                WHERE
                    cb_search_person.person_id IN (SELECT
                        person_id
                    FROM
                        `cb_search_person` p
                    WHERE
                        has_whole_genome_variant = 1 )
                    AND cb_search_person.person_id IN (SELECT
                        person_id
                    FROM
                        `cb_search_person` p
                    WHERE
                        has_ehr_data = 1 ) )
            )) observation
    LEFT JOIN
        `concept` o_standard_concept
            ON observation.observation_concept_id =
o_standard_concept.concept_id", sep="")

# Formulate a Cloud Storage destination path for the data exported
from BigQuery.
# NOTE: By default data exported multiple times on the same day will
overwrite older copies.
```

```
#       But data exported on a different days will write to a new
location so that historical
#       copies can be kept as the dataset definition is changed.
observation_00963032_path <- file.path(
  Sys.getenv("WORKSPACE_BUCKET"),
  "bq_exports",
  Sys.getenv("OWNER_EMAIL"),
  strftime(lubridate::now(), "%Y%m%d"),  # Comment out this line if
you want the export to always overwrite.
  "observation_00963032",
  "observation_00963032_*.csv")
message(str_glue('The data will be written to
{observation_00963032_path}. Use this path when reading ',
                 'the data into your notebooks in the future.'))

# Perform the query and export the dataset to Cloud Storage as CSV
files.
# NOTE: You only need to run `bq_table_save` once. After that, you can
#       just read data from the CSVs in Cloud Storage.
bq_table_save(
  bq_dataset_query(Sys.getenv("WORKSPACE_CDR"),
dataset_00963032_observation_sql, billing =
Sys.getenv("GOOGLE_PROJECT")),
  observation_00963032_path,
  destination_format = "CSV")


# Read the data directly from Cloud Storage into memory.
# NOTE: Alternatively you can `gsutil -m cp
{observation_00963032_path}` to copy these files
#       to the Jupyter disk.
read_bq_export_from_workspace_bucket <- function(export_path) {
  col_types <- cols(standard_concept_name = col_character(),
standard_concept_code = col_character(), standard_vocabulary =
col_character())
  bind_rows(
    map(system2('gsutil', args = c('ls', export_path), stdout = TRUE,
stderr = TRUE),
        function(csv) {
          message(str_glue('Loading {csv}.'))
          chunk <- read_csv(pipe(str_glue('gsutil cat {csv}')),
col_types = col_types, show_col_types = FALSE)
          if (is.null(col_types)) {
            col_types <- spec(chunk)
          }
          chunk
        }))
}
observation_df <-
read_bq_export_from_workspace_bucket(observation_00963032_path)
```

```r
dim(observation_df)

names(observation_df)<-c("person_id", "concept_id",
"standard_concept_name", "standard_concept_code",
"standard_vocabulary", "datetime")
```

```
The data will be written to gs://fc-secure-672eeb92-4859-4ed9-9f59-
d4349f3534a0/bq_exports/micah_hysong@researchallofus.org/20250825/
observation_00963032/observation_00963032_*.csv. Use this path when
reading the data into your notebooks in the future.

Loading
gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/bq_exports/micah_h
ysong@researchallofus.org/20250825/observation_00963032/
observation_00963032_000000000000.csv.


[1] 27  6
```

```r
ehr_codes<-rbind(condition_df, observation_df)

unique(ehr_codes$standard_concept_name)
```

```
  [1] "Lipid-rich atherosclerosis of coronary artery"

  [2] "Atherosclerosis of autologous coronary artery bypass graft"

  [3] "Acute myocardial infarction of anterior wall"

  [4] "Coronary thrombosis not resulting in myocardial infarction"

  [5] "Recurrent coronary arteriosclerosis after percutaneous
transluminal coronary angioplasty"
  [6] "Acute myocardial infarction of anterolateral wall"

  [7] "Coronary occlusion"

  [8] "Coronary artery bypass graft occlusion"

  [9] "Arteriosclerosis of autologous vein coronary artery bypass
graft"
 [10] "Post-infarction ventricular septal defect"

 [11] "Acute ST segment elevation myocardial infarction due to right
coronary artery occlusion"
 [12] "Delayed postmyocardial infarction pericarditis"

 [13] "Coronary artery spasm"

 [14] "Triple vessel disease of the heart"
```

```
 [15] "Subsequent non-ST segment elevation myocardial infarction"

 [16] "Post-infarction mural thrombus"

 [17] "Old inferior myocardial infarction"

 [18] "Coronary artery stenosis"

 [19] "Coronary artery atheroma"

 [20] "Angina associated with type 2 diabetes mellitus"

 [21] "Acute myocardial infarction due to left coronary artery
occlusion"
 [22] "Calcific coronary arteriosclerosis"

 [23] "Acute myocardial infarction of inferoposterior wall"

 [24] "Subsequent ST segment elevation myocardial infarction"

 [25] "Left main coronary artery disease"

 [26] "Coronary graft stenosis"

 [27] "Anomalous communication of coronary artery"

 [28] "Acute myocardial infarction of inferior wall"

 [29] "Non-Q wave myocardial infarction"

 [30] "Subsequent myocardial infarction of inferior wall"

 [31] "Subsequent myocardial infarction of anterior wall"

 [32] "Post-infarction pericarditis"

 [33] "Myocardial infarction"

 [34] "True posterior myocardial infarction"

 [35] "Acute ST segment elevation myocardial infarction involving left
anterior descending coronary artery"
 [36] "Calcification of coronary artery"

 [37] "Acute ST segment elevation myocardial infarction due to left
coronary artery occlusion"
 [38] "Acute anterior ST segment elevation myocardial infarction"

 [39] "Rupture of chordae tendinae due to and following acute
myocardial infarction"
 [40] "Stented coronary artery"
```

```
 [41] "Thrombosis of atrium, auricular appendage, and ventricle due to
and following acute myocardial infarction"
 [42] "Myocardial infarction due to demand ischemia"

 [43] "Multi vessel coronary artery disease"

 [44] "Mechanical complication of coronary bypass"

 [45] "Left coronary artery occlusion"

 [46] "Unstable angina co-occurrent and due to coronary
arteriosclerosis"
 [47] "Acute ST segment elevation myocardial infarction of inferior
wall"
 [48] "Congenital anomaly of coronary artery"

 [49] "Arteriosclerosis of autologous arterial coronary artery bypass
graft"
 [50] "Old myocardial infarction"

 [51] "Arteriosclerosis of arterial coronary artery bypass graft"

 [52] "Atrial septal defect due to and following acute myocardial
infarction"
 [53] "Double coronary vessel disease"

 [54] "Prinzmetal angina"

 [55] "Acute myocardial infarction of lateral wall"

 [56] "Rupture of papillary muscle as current complication following
acute myocardial infarction"
 [57] "Acute myocardial infarction of septum"

 [58] "Single coronary vessel disease"

 [59] "Non-obstructive atherosclerosis of coronary artery"

 [60] "Dissection of coronary artery"

 [61] "Aneurysm of coronary vessels"

 [62] "Post percutaneous transluminal coronary angioplasty"

 [63] "Coronary artery finding"

 [64] "Coronary artery stent thrombosis"

 [65] "Post infarct angina"
```

```
 [66] "Coronary bypass graft finding"

 [67] "Coronary artery fistula to left atrium"

 [68] "Significant coronary bypass graft disease"

 [69] "Postmyocardial infarction syndrome"

 [70] "Chronic total occlusion of coronary artery"

 [71] "Arteriosclerosis of coronary artery bypass graft of
transplanted heart"
 [72] "Pericarditis secondary to acute myocardial infarction"

 [73] "Rupture of cardiac wall without hemopericardium as current
complication following acute myocardial infarction"
 [74] "Normal coronary arteries"

 [75] "Arteriosclerosis of autologous coronary artery bypass graft"

 [76] "Acute non-ST segment elevation myocardial infarction"

 [77] "Acute coronary artery occlusion not resulting in myocardial
infarction"
 [78] "Acute myocardial infarction of apical-lateral wall"

 [79] "Coronary arteriosclerosis"

 [80] "Acute non-Q wave infarction"

 [81] "Acute subendocardial infarction"

 [82] "Arteriosclerosis of coronary artery bypass graft"

 [83] "Anomalous origin of coronary artery"

 [84] "Acute myocardial infarction"

 [85] "Anomalous origin of right coronary artery"

 [86] "Patient post percutaneous transluminal coronary angioplasty"

 [87] "Aortocoronary bypass graft present"

 [88] "Angina co-occurrent and due to coronary arteriosclerosis"

 [89] "Acute ST segment elevation myocardial infarction"

 [90] "Arteriosclerosis of nonautologous coronary artery bypass graft"

 [91] "Disorder of coronary artery"
```

```
 [92] "Atherosclerosis of coronary artery without angina pectoris"

 [93] "Coronary atherosclerosis"

 [94] "Coronary artery graft present"

 [95] "Acute myocardial infarction of inferolateral wall"

 [96] "Coronary arteriosclerosis following coronary artery bypass
graft"
 [97] "Drug coated stent in branch of right coronary artery"

 [98] "Drug coated stent in anterior descending branch of left
coronary artery"
 [99] "Stent in anterior descending branch of left coronary artery"

[100] "Drug coated stent in circumflex branch of left coronary artery"
```

```
ehr_codes2 <- ehr_codes |>
    group_by(person_id) |>
    summarize(distinct_count = n_distinct(datetime))

case_definition2 <- ehr_codes2[ehr_codes2$distinct_count>=2,]
nrow(case_definition2)
```

```
[1] 32452
```

Get cases from survey data

```
library(tidyverse)
library(bigrquery)

# This query represents dataset "cad_survey" for domain "survey" and
was generated for All of Us Controlled Tier Dataset v8
dataset_70858888_survey_sql <- paste("
    SELECT
        answer.person_id,
        answer.survey_datetime,
        answer.survey,
        answer.question_concept_id,
        answer.question,
        answer.answer_concept_id,
        answer.answer,
        answer.survey_version_concept_id,
        answer.survey_version_name
    FROM
        `ds_survey` answer
    WHERE
```

```
        (
            question_concept_id IN (836876)
        )
        AND (
            answer.PERSON_ID IN (SELECT
                distinct person_id
            FROM
                `cb_search_person` cb_search_person
            WHERE
                cb_search_person.person_id IN (SELECT
                    person_id
                FROM
                    `cb_search_person` p
                WHERE
                    has_whole_genome_variant = 1 )
                AND cb_search_person.person_id IN (SELECT
                    person_id
                FROM
                    `cb_search_person` p
                WHERE
                    has_ehr_data = 1 ) )
        )", sep="")

# Formulate a Cloud Storage destination path for the data exported
from BigQuery.
# NOTE: By default data exported multiple times on the same day will
overwrite older copies.
#       But data exported on a different days will write to a new
location so that historical
#       copies can be kept as the dataset definition is changed.
survey_70858888_path <- file.path(
  Sys.getenv("WORKSPACE_BUCKET"),
  "bq_exports",
  Sys.getenv("OWNER_EMAIL"),
  strftime(lubridate::now(), "%Y%m%d"),  # Comment out this line if
you want the export to always overwrite.
  "survey_70858888",
  "survey_70858888_*.csv")
message(str_glue('The data will be written to {survey_70858888_path}.
Use this path when reading ',
                 'the data into your notebooks in the future.'))

# Perform the query and export the dataset to Cloud Storage as CSV
files.
# NOTE: You only need to run `bq_table_save` once. After that, you can
#       just read data from the CSVs in Cloud Storage.
bq_table_save(
  bq_dataset_query(Sys.getenv("WORKSPACE_CDR"),
dataset_70858888_survey_sql, billing = Sys.getenv("GOOGLE_PROJECT")),
```

```
  survey_70858888_path,
  destination_format = "CSV")


# Read the data directly from Cloud Storage into memory.
# NOTE: Alternatively you can `gsutil -m cp {survey_70858888_path}` to
copy these files
#       to the Jupyter disk.
read_bq_export_from_workspace_bucket <- function(export_path) {
  col_types <- cols(survey = col_character(), question =
col_character(), answer = col_character(), survey_version_name =
col_character())
  bind_rows(
    map(system2('gsutil', args = c('ls', export_path), stdout = TRUE,
stderr = TRUE),
        function(csv) {
          message(str_glue('Loading {csv}.'))
          chunk <- read_csv(pipe(str_glue('gsutil cat {csv}')),
col_types = col_types, show_col_types = FALSE)
          if (is.null(col_types)) {
            col_types <- spec(chunk)
          }
          chunk
        }))
}
survey_df <-
read_bq_export_from_workspace_bucket(survey_70858888_path)

dim(survey_df)
```

```
The data will be written to gs://fc-secure-672eeb92-4859-4ed9-9f59-
d4349f3534a0/bq_exports/micah_hysong@researchallofus.org/20250825/
survey_70858888/survey_70858888_*.csv. Use this path when reading the
data into your notebooks in the future.

Loading
gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/bq_exports/micah_h
ysong@researchallofus.org/20250825/survey_70858888/
survey_70858888_000000000000.csv.


[1] 110228      9
```

```
cad<-survey_df[survey_df$answer == "Including yourself, who in your
family has had coronary artery/coronary heart disease? - Self",]
case_definition3<-unique(cad$person_id)
length(case_definition3)
```

```
[1] 7114
```

Get all Cases

```
cases01<-unique(c(case_definition1, case_definition2$person_id))
length(cases01)
cases01<-unique(c(cases01, case_definition3))
length(cases01)
```

```
[1] 32501
```

```
[1] 34435
```

Get controls

```
# This snippet assumes that you run setup first

# This code copies a file from your Google Bucket into a dataframe

# replace 'test.csv' with the name of the file in your google bucket
(don't delete the quotation marks)
name_of_file_in_bucket <- 'Demographic_and_ancestry_covariates.csv'

################################################################################
##
##
################## DON'T CHANGE FROM HERE
################################
##
################################################################################
##

# Get the bucket name
my_bucket <- Sys.getenv('WORKSPACE_BUCKET')

# Copy the file from current workspace to the bucket
system(paste0("gsutil cp ", my_bucket, "/data/",
name_of_file_in_bucket, " ."), intern=T)

# Load the file into a dataframe
demographics  <- read_csv(name_of_file_in_bucket)
```

```
character(0)
```

```
Rows: 162193 Columns: 28
── Column specification
────────────────────────────────────────────────────────
Delimiter: ","
chr  (8): SexGender, income, education, where_born, military,
healthcare, di...
dbl  (9): person_id, race_unknown, age_today, LGBTQIA, ehr_length,
```

```
relative_...
lgl  (8): AIAN, Asian, Black, Mid, Multiple, PI, White, His
date (3): date_of_birth, min_date, max_date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

```
controls <- demographics$person_id[!(demographics$person_id %in%
ehr_codes$person_id)]
controls <- controls[!(controls %in% case_definition1)]
controls <- controls[!(controls %in% case_definition3)]
length(controls)
```

```
[1] 136140
```

```
# Assign status01
demographics$status01 <- ifelse(demographics$person_id %in% controls,
0,
                                ifelse(demographics$person_id %in%
cases01, 1, NA))

df_cases <- data.frame(
  person_id = cases01,
  status = 1
)

df_controls <- data.frame(
  person_id = controls,
  status = 0
)

final_df <- rbind(df_cases, df_controls)
nrow(final_df)
n_distinct(final_df$person_id)
```

```
[1] 170575
```

```
[1] 170575
```

```
table(final_df$status)
```

```
     0      1
136140  34435
```

```
# This snippet assumes that you run setup first

# This code saves your dataframe into a csv file in a "data" folder in
Google Bucket

# Replace df with THE NAME OF YOUR DATAFRAME
```

```r
my_dataframe <- final_df

# Replace 'test.csv' with THE NAME of the file you're going to store
in the bucket (don't delete the quotation marks)
destination_filename <- 'eMERGE_CAD_case_control_df.csv'

#########################################################################
##
##
################## DON'T CHANGE FROM HERE
################################
##
#########################################################################
##

# store the dataframe in current workspace
write_excel_csv(my_dataframe, destination_filename)

# Get the bucket name
my_bucket <- Sys.getenv('WORKSPACE_BUCKET')

# Copy the file from current workspace to the bucket
system(paste0("gsutil cp ./", destination_filename, " ", my_bucket,
"/data/"), intern=T)

# Check if file is in the bucket
system(paste0("gsutil ls ", my_bucket, "/data/*.csv"), intern=T)
```

```
character(0)
```

```
[1]
"gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/data/Demographic_
and_ancestry_covariates.csv"
[2]
"gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/data/all_demograp
hics.csv"
[3]
"gs://fc-secure-672eeb92-4859-4ed9-9f59-d4349f3534a0/data/eMERGE_CAD_c
ase_control_df.csv"
```