Worker Satisfaction Survey — Data De-identification Workbook

Data De-identification Advisor

June 08, 2026

Contents

Introduction

This workbook guides you through de-identifying the **Apple Grower Satisfaction Survey** (`WorkerSatisfaction_300rows.` 300 rows, 16 columns).

Learning objectives

- Identify privacy risks in survey data
- Apply de-identification steps in R using **dplyr**
- Export a shareable dataset and a separate data key file

Workflow

1. **Setup** — load packages and set file paths
2. **Load and assess** — inspect raw data and review privacy risks
3. **Transform** — apply four de-identification steps (pseudonymization, aggregation, anonymization)
4. **Verify and deliver** — run QA checks and export output files

De-identification steps (Part 3)

Step	Technique	Variables
1	Pseudonymization	`worker_id`, `email_id`, `owner_id`
2	Aggregation	`age` (date of birth → age bands)
3	Pseudonymization	`city`, `orchard_id`
4	Anonymization	`immigration_stat`, `username_id`, `comments`

Terminology

Technique	Meaning in this workbook
Pseudonymization	Replace identifiers with coded values; a **data key** can reverse the mapping (Steps 1 and 3)
Aggregation	Group a detailed value into broader categories (Step 2: DOB → age band)
Anonymization	Irreversibly remove or transform data so individuals cannot be re-identified; **no key file** (Step 4)

Output files

- `WorkerSatisfaction_300rows_deidentified.xlsx` — for research (e.g. public access. . .)
- `data_key_file/WorkerSatisfaction_data_key_DUMMY.xlsx` — for authorized personnel only; store separately from the de-identified data (dummy example)

Do not edit the raw source file (`WorkerSatisfaction_300rows.xlsx`).

Roadmap

Part	Section	What happens
1	Setup	Install packages, set paths
2	Load and assess	Read raw data; review risks by de-identification step
3	Transform	Steps 1–4: pseudonymize → aggregate → pseudonymize → anonymize
4	Verify and deliver	QA checks → export de-identified file → export data key

Step	Technique	What to look for (Part 2)	What to do (Part 3)
1	Pseudonymization	Names, emails, owner names	Replace with `Worker_01`, `Email_01`, `Owner_01`, . . . (reversible via key)
2	Aggregation	`age` is date of birth	Group into 5-year age bands
3	Pseudonymization	City and orchard names	Replace with `City_01`, `Orchard_01`, . . . (reversible via key)
4	Anonymization	Immigration status, handles, free text	Pool small groups; remove `username_id` and `comments` (irreversible)

Part 1 — Setup

R and RStudio (workshop prerequisite)
This workbook assumes you have **R** and **RStudio** installed and can open and run `.Rmd` files. If you have not set these up yet, follow the UBC Library Research Commons guide:
Installing R and RStudio

Required packages

Package	Purpose
`readxl`	Read the original Excel survey file
`dplyr`	Transform data (native pipe `|>`)
`writexl`	Export Excel outputs

Run **Install packages** once if a package is missing.

```
# install.packages("readxl")
# install.packages("dplyr")
# install.packages("writexl")
# install.packages(c("readxl", "dplyr", "writexl"))
```

```
library(readxl)
library(dplyr)
library(writexl)
```

```
DATA_FILE   <- "WorkerSatisfaction_300rows.xlsx"
OUTPUT_FILE <- "WorkerSatisfaction_300rows_deidentified.xlsx"
KEY_DIR     <- "data_key_file"
KEY_FILE    <- file.path(KEY_DIR, "WorkerSatisfaction_data_key_DUMMY.xlsx")
K_THRESHOLD <- 5
```

k-anonymity threshold: categories with fewer than 5 records will be pooled before anonymization.

```
cat("K_THRESHOLD =", K_THRESHOLD, "\n")
```

```
## K_THRESHOLD = 5
```

```
cat("Data key file:", KEY_FILE, "\n")
```

```
## Data key file: data_key_file/WorkerSatisfaction_data_key_DUMMY.xlsx
```

Part 2 — Load and Assess

Load the unmodified survey and review what makes the data identifiable.

2.1 Load raw data

```
raw <- read_excel(DATA_FILE, sheet = "Unmodified data")

cat("Raw data:", nrow(raw), "rows ×", ncol(raw), "columns\n")
```

```
## Raw data: 300 rows × 16 columns
```

```r
head(raw, 10)
```

```
## # A tibble: 10 x 16
##    worker_id        email_id age                 immigration_stat city  province
##    <chr>            <chr>    <dttm>              <chr>            <chr> <chr>
##  1 James Strange    james.s~ 1998-09-10 00:00:00 Non-immigrant    Kelo~ B.C.
##  2 Maya Liya        maya.li~ 1994-08-04 00:00:00 Immigrant        Kelo~ B.C.
##  3 Amelio Beal      amelio.~ 1995-01-02 00:00:00 Non-permanent r~ West~ B.C.
##  4 Cara Sahara      cara.sa~ 1991-03-22 00:00:00 Non-permanent r~ West~ B.C.
##  5 Neiv Rieg        neivr@g~ 1990-07-15 00:00:00 Non-permanent r~ Vict~ B.C.
##  6 Troy Ahoy        tahoy@y~ 1988-06-06 00:00:00 Immigrant        Vict~ B.C.
##  7 Dave Mahew       dmahew@~ 2000-11-11 00:00:00 Non-immigrant    Vern~ B.C.
##  8 Jamie Thomas     jamiet@~ 1992-12-18 00:00:00 Immigrant        Vern~ B.C.
##  9 Betty Stills     bettys5~ 1997-09-06 00:00:00 Non-permanent r~ Cobb~ B.C.
## 10 Enrique Iglasias ei123@h~ 1985-10-16 00:00:00 Non-permanent r~ Cobb~ B.C.
## # i 10 more variables: orchard_id <chr>, owner_id <chr>, username_id <chr>,
## #   sns_worked <dbl>, sat_hrs <dbl>, trt_workers <dbl>, trt_manager <dbl>,
## #   cmf_manager <dbl>, sat_work_overall <dbl>, comments <chr>
```

2.2 Step 1 — Direct identifiers (pseudonymization)

These variables point to a specific person. They will be **pseudonymized** in Step 1 (not dropped) so rows stay linkable within the dataset without exposing real names or contact details. A data key allows authorized staff to reverse the mapping.

Variable	Why it is risky	Pseudonym prefix
`worker_id`	Full name (300 unique values)	`Worker_001`, ...
`email_id`	Email address (300 unique values)	`Email_001`, ...
`owner_id`	Orchard owner name (12 unique values)	`Owner_01`, ...

2.3 Step 2 — Date of birth (aggregation)

The column `age` stores **date of birth**, not a numeric age. Exact DOB is a strong quasi-identifier when combined with location and employer.

```r
data.frame(
  earliest     = as.character(min(as.Date(raw$age))),
  latest       = as.character(max(as.Date(raw$age))),
  unique_dates = length(unique(raw$age))
)
```

```
##     earliest     latest unique_dates
## 1 1970-02-20 2000-11-11          295
```

Exact DOB will be replaced with **5-year age bands** in Step 2 (aggregation).

2.4 Step 3 — Geography and orchards (pseudonymization)

`city` has 12 levels; `province` is always B.C. Small groups are easier to re-identify. Orchard names link respondents to a specific workplace. Both will be pseudonymized in Step 3.

```r
raw |> count(city, sort = TRUE)
```

```
## # A tibble: 12 x 2
##    city             n
```

```
##    <chr>        <int>
##  1 Victoria        33
##  2 Summerland      32
##  3 Keremeos        31
##  4 Kelowna         29
##  5 West Kelowna    29
##  6 Peachland       27
##  7 Cobble Hill     26
##  8 Vernon          24
##  9 Naramata        18
## 10 Oliver          18
## 11 Osoyoos         17
## 12 Penticton       16
```

Note: If any city had fewer than 5 records, we would pool it into `"Other BC community"` before pseudonymizing.

```
raw |> count(orchard_id, sort = TRUE)
```

```
## # A tibble: 12 x 2
##    orchard_id              n
##    <chr>               <int>
##  1 Billie's Apples        33
##  2 Okanagan Fresh         32
##  3 Valley View Apples     31
##  4 Applejacks             29
##  5 West Kelowna Apples    29
##  6 Lakeside Groves        27
##  7 Country Apples         26
##  8 Bon Appletit           24
##  9 Golden Valley Farms    18
## 10 Ridgeline Orchard      18
## 11 Peak Harvest Co        17
## 12 Sunrise Orchards       16
```

Note: Replace orchard names with non-descriptive codes (`Orchard_01`, ...).

2.5 Step 4 — Immigration status, handles, and free text (anonymization)

These variables cannot be safely shared in their original form. **Anonymization** is irreversible — no data key is created for these fields.

`immigration_stat`

Small category counts increase re-identification risk. Pool any group below the k-anonymity threshold into a broader label.

```
raw |> count(immigration_stat, sort = TRUE)
```

```
## # A tibble: 3 x 2
##   immigration_stat           n
##   <chr>                  <int>
## 1 Non-permanent resident   107
## 2 Non-immigrant            103
## 3 Immigrant                 90
```

```r
rare_immigration <- raw |>
  count(immigration_stat) |>
  filter(n < K_THRESHOLD) |>
  pull(immigration_stat)

cat("Rare immigration groups (n <", K_THRESHOLD, "):\n")
```

```
## Rare immigration groups (n < 5 ):
```

```r
rare_immigration
```

```
## character(0)
```

> **Note:** In this dataset all immigration groups meet the threshold. The pooling logic in Step 4 still runs as a safeguard.

`username_id`

Social-media handles are quasi-identifiers (265 unique values; 36 missing). They will be **removed** in Step 4 — anonymization by deletion.

`comments`

The `comments` field can name people, employers, platforms, and birth years — even after other columns are cleaned.

```r
raw |>
  filter(comments != "None") |>
  slice(1:10) |>
  select(comments)
```

```
## # A tibble: 10 x 1
##    comments
##    <chr>
##  1 I get that I'm a farm worker and grinding is part of the job, but it'd be ni~
##  2 I don't mind the actual work, but the people I work with and the owners like~
##  3 I only like this job because I can make good money, everything else sucks. I~
##  4 The hours and work are hard, but it's good money and I love the team we have~
##  5 Hard but great work
##  6 I'm an apple picker, it is what it is
##  7 It's a good summer job to help me pay for school, definitely not something I~
##  8 The owner's wife is such a sweetheart, but I don't like the owner himself.  ~
##  9 I hate this job and everybody who works here.  the guys I work with are awfu~
## 10 I don't love the job but I stay for the money and the routine.
```

> **Note:** Remove `comments` entirely in Step 4. Free text cannot be reliably anonymized without manual review.

Part 3 — Transform

Apply each step from Part 2 in order. Run chunks sequentially; each step shows **10 rows** of the updated dataframe.

```r
step1_ids <- c("worker_id", "email_id", "owner_id")
```

Step 1: Pseudonymization for direct identifiers

Replace `worker_id`, `email_id`, and `owner_id` with non-descriptive codes. Store mappings in the data key (Part 4).

```
step1 <- raw |>
  mutate(
    worker_id = paste0("Worker_", sprintf("%03d", as.integer(factor(worker_id)))),
    email_id  = paste0("Email_",  sprintf("%03d", as.integer(factor(email_id)))),
    owner_id  = paste0("Owner_",  sprintf("%02d", as.integer(factor(owner_id))))
  )

cat("Step 1:", nrow(step1), "rows ×", ncol(step1), "columns\n")
```

```
## Step 1: 300 rows × 16 columns
```

```
head(step1, 10)
```

```
## # A tibble: 10 x 16
##    worker_id  email_id  age                 immigration_stat      city  province
##    <chr>      <chr>     <dttm>              <chr>                 <chr> <chr>
##  1 Worker_114 Email_110 1998-09-10 00:00:00 Non-immigrant         Kelo~ B.C.
##  2 Worker_170 Email_163 1994-08-04 00:00:00 Immigrant             Kelo~ B.C.
##  3 Worker_010 Email_011 1995-01-02 00:00:00 Non-permanent reside~ West~ B.C.
##  4 Worker_035 Email_033 1991-03-22 00:00:00 Non-permanent reside~ West~ B.C.
##  5 Worker_194 Email_193 1990-07-15 00:00:00 Non-permanent reside~ Vict~ B.C.
##  6 Worker_276 Email_261 1988-06-06 00:00:00 Immigrant             Vict~ B.C.
##  7 Worker_054 Email_057 2000-11-11 00:00:00 Non-immigrant         Vern~ B.C.
##  8 Worker_117 Email_114 1992-12-18 00:00:00 Immigrant             Vern~ B.C.
##  9 Worker_022 Email_021 1997-09-06 00:00:00 Non-permanent reside~ Cobb~ B.C.
## 10 Worker_064 Email_061 1985-10-16 00:00:00 Non-permanent reside~ Cobb~ B.C.
## # i 10 more variables: orchard_id <chr>, owner_id <chr>, username_id <chr>,
## #   sns_worked <dbl>, sat_hrs <dbl>, trt_workers <dbl>, trt_manager <dbl>,
## #   cmf_manager <dbl>, sat_work_overall <dbl>, comments <chr>
```

```
step1 |>
  select(all_of(step1_ids)) |>
  slice(1:5)
```

```
## # A tibble: 5 x 3
##   worker_id  email_id  owner_id
##   <chr>      <chr>     <chr>
## 1 Worker_114 Email_110 Owner_03
## 2 Worker_170 Email_163 Owner_03
## 3 Worker_010 Email_011 Owner_08
## 4 Worker_035 Email_033 Owner_08
## 5 Worker_194 Email_193 Owner_02
```

Step 2: Aggregation for the age variable

Replace exact date of birth with **5-year age bands**.

```
to_age_band <- function(dob) {
  yrs <- as.integer(difftime(Sys.Date(), as.Date(dob), units = "days") / 365.25)
  if (yrs < 25)      "18-24"
  else if (yrs < 35) "25-34"
  else if (yrs < 45) "35-44"
```

```
  else if (yrs < 55) "45-54"
  else               "55+"
}

step2 <- step1 |>
  mutate(age_band = sapply(age, to_age_band)) |>
  select(-age) |>
  relocate(age_band, .before = immigration_stat)

cat("Step 2:", nrow(step2), "rows ×", ncol(step2), "columns\n")
```

```
## Step 2: 300 rows × 16 columns
```

```
head(step2, 10)
```

```
## # A tibble: 10 x 16
##    worker_id  email_id  age_band immigration_stat      city  province orchard_id
##    <chr>      <chr>     <chr>    <chr>                 <chr> <chr>    <chr>
##  1 Worker_114 Email_110 25-34    Non-immigrant         Kelo~ B.C.     Applejacks
##  2 Worker_170 Email_163 25-34    Immigrant             Kelo~ B.C.     Applejacks
##  3 Worker_010 Email_011 25-34    Non-permanent reside~ West~ B.C.     West Kelo~
##  4 Worker_035 Email_033 35-44    Non-permanent reside~ West~ B.C.     West Kelo~
##  5 Worker_194 Email_193 35-44    Non-permanent reside~ Vict~ B.C.     Billie's ~
##  6 Worker_276 Email_261 35-44    Immigrant             Vict~ B.C.     Billie's ~
##  7 Worker_054 Email_057 25-34    Non-immigrant         Vern~ B.C.     Bon Apple~
##  8 Worker_117 Email_114 25-34    Immigrant             Vern~ B.C.     Bon Apple~
##  9 Worker_022 Email_021 25-34    Non-permanent reside~ Cobb~ B.C.     Country A~
## 10 Worker_064 Email_061 35-44    Non-permanent reside~ Cobb~ B.C.     Country A~
## # i 9 more variables: owner_id <chr>, username_id <chr>, sns_worked <dbl>,
## #   sat_hrs <dbl>, trt_workers <dbl>, trt_manager <dbl>, cmf_manager <dbl>,
## #   sat_work_overall <dbl>, comments <chr>
```

Step 3: Pseudonymization for city and orchard_id

Pseudonymize `city` and `orchard_id` with non-descriptive codes.

```
step3 <- step2 |>
  mutate(
    city       = paste0("City_",    sprintf("%02d", as.integer(factor(city)))),
    orchard_id = paste0("Orchard_", sprintf("%02d", as.integer(factor(orchard_id))))
  )

cat("Step 3:", nrow(step3), "rows ×", ncol(step3), "columns\n")
```

```
## Step 3: 300 rows × 16 columns
```

```
head(step3, 10)
```

```
## # A tibble: 10 x 16
##    worker_id  email_id  age_band immigration_stat      city  province orchard_id
##    <chr>      <chr>     <chr>    <chr>                 <chr> <chr>    <chr>
##  1 Worker_114 Email_110 25-34    Non-immigrant         City~ B.C.     Orchard_01
##  2 Worker_170 Email_163 25-34    Immigrant             City~ B.C.     Orchard_01
##  3 Worker_010 Email_011 25-34    Non-permanent reside~ City~ B.C.     Orchard_12
##  4 Worker_035 Email_033 35-44    Non-permanent reside~ City~ B.C.     Orchard_12
##  5 Worker_194 Email_193 35-44    Non-permanent reside~ City~ B.C.     Orchard_02
```

```
##  6 Worker_276 Email_261 35-44    Immigrant             City~ B.C.     Orchard_02
##  7 Worker_054 Email_057 25-34    Non-immigrant         City~ B.C.     Orchard_03
##  8 Worker_117 Email_114 25-34    Immigrant             City~ B.C.     Orchard_03
##  9 Worker_022 Email_021 25-34    Non-permanent reside~ City~ B.C.     Orchard_04
## 10 Worker_064 Email_061 35-44    Non-permanent reside~ City~ B.C.     Orchard_04
## # i 9 more variables: owner_id <chr>, username_id <chr>, sns_worked <dbl>,
## #   sat_hrs <dbl>, trt_workers <dbl>, trt_manager <dbl>, cmf_manager <dbl>,
## #   sat_work_overall <dbl>, comments <chr>
```

```
step3 |> count(city, sort = TRUE)
```

```
## # A tibble: 12 x 2
##    city        n
##    <chr>   <int>
##  1 City_11    33
##  2 City_09    32
##  3 City_03    31
##  4 City_02    29
##  5 City_12    29
##  6 City_07    27
##  7 City_01    26
##  8 City_10    24
##  9 City_04    18
## 10 City_05    18
## 11 City_06    17
## 12 City_08    16
```

```
step3 |> count(orchard_id, sort = TRUE)
```

```
## # A tibble: 12 x 2
##    orchard_id     n
##    <chr>      <int>
##  1 Orchard_02    33
##  2 Orchard_07    32
##  3 Orchard_11    31
##  4 Orchard_01    29
##  5 Orchard_12    29
##  6 Orchard_06    27
##  7 Orchard_04    26
##  8 Orchard_03    24
##  9 Orchard_05    18
## 10 Orchard_09    18
## 11 Orchard_08    17
## 12 Orchard_10    16
```

Step 4: Anonymization for immigration_stat, username_id, and comments

Anonymization is **irreversible** — removed columns and pooled labels are not stored in the data key.

```
step4 <- step3 |>
  mutate(
    immigration_stat = if_else(
      immigration_stat %in% rare_immigration,
      "Other immigration status",
      immigration_stat
    )
```

```
  ) |>
  select(-username_id, -comments)

deid <- step4

cat("Step 4 - final:", nrow(deid), "rows ×", ncol(deid), "columns\n")
```

```
## Step 4 - final: 300 rows × 14 columns
```

```
head(deid, 10)
```

```
## # A tibble: 10 x 14
##    worker_id  email_id  age_band immigration_stat      city  province orchard_id
##    <chr>      <chr>     <chr>    <chr>                 <chr> <chr>    <chr>
##  1 Worker_114 Email_110 25-34    Non-immigrant         City~ B.C.     Orchard_01
##  2 Worker_170 Email_163 25-34    Immigrant             City~ B.C.     Orchard_01
##  3 Worker_010 Email_011 25-34    Non-permanent reside~ City~ B.C.     Orchard_12
##  4 Worker_035 Email_033 35-44    Non-permanent reside~ City~ B.C.     Orchard_12
##  5 Worker_194 Email_193 35-44    Non-permanent reside~ City~ B.C.     Orchard_02
##  6 Worker_276 Email_261 35-44    Immigrant             City~ B.C.     Orchard_02
##  7 Worker_054 Email_057 25-34    Non-immigrant         City~ B.C.     Orchard_03
##  8 Worker_117 Email_114 25-34    Immigrant             City~ B.C.     Orchard_03
##  9 Worker_022 Email_021 25-34    Non-permanent reside~ City~ B.C.     Orchard_04
## 10 Worker_064 Email_061 35-44    Non-permanent reside~ City~ B.C.     Orchard_04
## # i 7 more variables: owner_id <chr>, sns_worked <dbl>, sat_hrs <dbl>,
## #   trt_workers <dbl>, trt_manager <dbl>, cmf_manager <dbl>,
## #   sat_work_overall <dbl>
```

```
deid |> count(immigration_stat, sort = TRUE)
```

```
## # A tibble: 3 x 2
##   immigration_stat           n
##   <chr>                  <int>
## 1 Non-permanent resident   107
## 2 Non-immigrant            103
## 3 Immigrant                 90
```

What changed

Step	Technique	Variables
1	Pseudonymization	`worker_id` → `Worker_001` …; `email_id` → `Email_001` …; `owner_id` → `Owner_01` …
2	Aggregation	`age` (DOB) → `age_band` (5-year groups)
3	Pseudonymization	`city` → `City_01` …; `orchard_id` → `Orchard_01` …
4	Anonymization	`immigration_stat` pooled if below threshold; `username_id` and `comments` removed

Part 4 — Verify and Deliver

4.1 QA checks

Confirm the de-identified data is safe to export. All checks should show `TRUE`.

```
data.frame(
  check = c(
    "Step 1: direct IDs pseudonymized",
    "Step 2: no exact DOB column",
    "Step 3: cities pseudonymized",
    "Step 3: orchards pseudonymized",
    "Step 4: comments removed",
    "Step 4: username_id removed",
    "Row count unchanged"
  ),
  passed = c(
    !any(unique(raw$worker_id) %in% deid$worker_id),
    !"age" %in% names(deid),
    !any(unique(raw$city) %in% deid$city),
    !any(unique(raw$orchard_id) %in% deid$orchard_id),
    !"comments" %in% names(deid),
    !"username_id" %in% names(deid),
    nrow(deid) == nrow(raw)
  )
)
```

```
##                              check passed
## 1 Step 1: direct IDs pseudonymized   TRUE
## 2      Step 2: no exact DOB column   TRUE
## 3     Step 3: cities pseudonymized   TRUE
## 4   Step 3: orchards pseudonymized   TRUE
## 5         Step 4: comments removed   TRUE
## 6      Step 4: username_id removed   TRUE
## 7               Row count unchanged   TRUE
```

```
data.frame(
  check = c(
    "Emails pseudonymized",
    "Owners pseudonymized"
  ),
  passed = c(
    !any(unique(raw$email_id) %in% deid$email_id),
    !any(unique(raw$owner_id) %in% deid$owner_id)
  )
)
```

```
##                  check passed
## 1 Emails pseudonymized   TRUE
## 2 Owners pseudonymized   TRUE
```

```
data.frame(
  check = c(
    "No rare immigration groups in output",
    "age_band present"
  ),
```

```
  passed = c(
    all(table(deid$immigration_stat) >= K_THRESHOLD),
    "age_band" %in% names(deid)
  )
)
```

```
##                                 check passed
## 1 No rare immigration groups in output   TRUE
## 2                     age_band present   TRUE
```

```
data.frame(
  rows               = nrow(deid),
  columns            = ncol(deid),
  smallest_city      = min(table(deid$city)),
  smallest_age_band  = min(table(deid$age_band)),
  smallest_orchard   = min(table(deid$orchard_id))
)
```

```
##   rows columns smallest_city smallest_age_band smallest_orchard
## 1  300      14            16                14               16
```

Note: Each group should ideally have at least 5 records, or be pooled into a broader category.

4.2 Export de-identified dataset

This is the file you can share for analysis. It does **not** include the data key.

```
write_xlsx(deid, OUTPUT_FILE)

cat("Exported:", OUTPUT_FILE, "\n")
```

```
## Exported: WorkerSatisfaction_300rows_deidentified.xlsx
```

```
cat("Rows:", nrow(deid), "| Columns:", ncol(deid), "\n\n")
```

```
## Rows: 300 | Columns: 14
```

```
cat("Preview of exported data:\n")
```

```
## Preview of exported data:
```

```
head(deid, 10)
```

```
## # A tibble: 10 x 14
##    worker_id  email_id  age_band immigration_stat      city  province orchard_id
##    <chr>      <chr>     <chr>    <chr>                 <chr> <chr>    <chr>
##  1 Worker_114 Email_110 25-34    Non-immigrant         City~ B.C.     Orchard_01
##  2 Worker_170 Email_163 25-34    Immigrant             City~ B.C.     Orchard_01
##  3 Worker_010 Email_011 25-34    Non-permanent reside~ City~ B.C.     Orchard_12
##  4 Worker_035 Email_033 35-44    Non-permanent reside~ City~ B.C.     Orchard_12
##  5 Worker_194 Email_193 35-44    Non-permanent reside~ City~ B.C.     Orchard_02
##  6 Worker_276 Email_261 35-44    Immigrant             City~ B.C.     Orchard_02
##  7 Worker_054 Email_057 25-34    Non-immigrant         City~ B.C.     Orchard_03
##  8 Worker_117 Email_114 25-34    Immigrant             City~ B.C.     Orchard_03
##  9 Worker_022 Email_021 25-34    Non-permanent reside~ City~ B.C.     Orchard_04
## 10 Worker_064 Email_061 35-44    Non-permanent reside~ City~ B.C.     Orchard_04
## # i 7 more variables: owner_id <chr>, sns_worked <dbl>, sat_hrs <dbl>,
## #   trt_workers <dbl>, trt_manager <dbl>, cmf_manager <dbl>,
```

```
## #   sat_work_overall <dbl>
```

4.3 Data key file (DUMMY — illustrative only)

Keeping the data key file well protected is essential because it is the sole mechanism that can re-identify individuals in otherwise de-identified research data, and its compromise can undermine confidentiality protections and cause harm to participants. In the UBC environment, the key file must be stored separately from research data on UBC-approved secure systems, with access restricted to authorized personnel only and never stored on personal devices or unapproved cloud services.

Here is an example of a data key file for the work described above; this is a **dummy file provided for illustrative purposes only**, and in real-world research settings, such files must never be shared on public systems.

File path: data_key_file/WorkerSatisfaction_data_key_DUMMY.xlsx — **one sheet** (`data_key`) with all mappings in a single table:

Column	Purpose
`variable`	Which field was pseudonymized (`worker_id`, `email_id`, `city`, ...)
`original_value`	Value in the raw file
`pseudonym_code`	Code used in the de-identified file
`notes`	Optional step label

To **restore** a de-identified file: for each `variable`, join `pseudonym_code` in the de-identified data back to `original_value` using this table (never merge the key into the shareable dataset).

The key holds **mappings for pseudonymized identifiers only** (Steps 1 and 3). Anonymized fields (`username_id`, `comments`) are not included — they cannot be reversed.

```
worker_key <- raw |>
  distinct(worker_id) |>
  arrange(worker_id) |>
  transmute(
    original_value = worker_id,
    pseudonym_code = paste0("Worker_", sprintf("%03d", row_number()))
  )

email_key <- raw |>
  distinct(email_id) |>
  arrange(email_id) |>
  transmute(
    original_value = email_id,
    pseudonym_code = paste0("Email_", sprintf("%03d", row_number()))
  )

owner_key <- raw |>
  distinct(owner_id) |>
  arrange(owner_id) |>
  transmute(
    original_value = owner_id,
    pseudonym_code = paste0("Owner_", sprintf("%02d", row_number()))
  )

orchard_key <- raw |>
```

```
  distinct(orchard_id) |>
  arrange(orchard_id) |>
  transmute(
    original_value = orchard_id,
    pseudonym_code = paste0("Orchard_", sprintf("%02d", row_number()))
  )

city_key <- raw |>
  distinct(city) |>
  arrange(city) |>
  transmute(
    original_value = city,
    pseudonym_code = paste0("City_", sprintf("%02d", row_number()))
  )

# One table: stack every mapping for simple lookup and restore
data_key <- bind_rows(
  worker_key  |> mutate(variable = "worker_id",  notes = NA_character_),
  email_key   |> mutate(variable = "email_id",   notes = NA_character_),
  owner_key   |> mutate(variable = "owner_id",   notes = NA_character_),
  orchard_key |> mutate(variable = "orchard_id", notes = NA_character_),
  city_key    |> mutate(variable = "city",       notes = NA_character_)
) |>
  select(variable, original_value, pseudonym_code, notes) |>
  arrange(variable, original_value)

cat("Data key preview (first 10 rows):\n")
```

```
## Data key preview (first 10 rows):
```

```
head(data_key, 10)
```

```
## # A tibble: 10 x 4
##    variable original_value pseudonym_code notes
##    <chr>    <chr>          <chr>          <chr>
##  1 city     Cobble Hill    City_01        <NA>
##  2 city     Kelowna        City_02        <NA>
##  3 city     Keremeos       City_03        <NA>
##  4 city     Naramata       City_04        <NA>
##  5 city     Oliver         City_05        <NA>
##  6 city     Osoyoos        City_06        <NA>
##  7 city     Peachland      City_07        <NA>
##  8 city     Penticton      City_08        <NA>
##  9 city     Summerland     City_09        <NA>
## 10 city     Vernon         City_10        <NA>
```

```
cat("\nRows by variable:\n")
```

```
##
## Rows by variable:
```

```
data_key |> count(variable)
```

```
## # A tibble: 5 x 2
##   variable       n
##   <chr>      <int>
```

```
## 1 city          12
## 2 email_id     300
## 3 orchard_id    12
## 4 owner_id      12
## 5 worker_id    300
```

```
dir.create(KEY_DIR, showWarnings = FALSE)

write_xlsx(list(data_key = data_key), KEY_FILE)

cat("Data key exported:", KEY_FILE, "\n")
```

```
## Data key exported: data_key_file/WorkerSatisfaction_data_key_DUMMY.xlsx
```

```
cat("Single sheet: data_key (", nrow(data_key), " mapping rows )\n", sep = "")
```

```
## Single sheet: data_key (636 mapping rows )
```

```
cat("Store separately from", OUTPUT_FILE, "- never on public systems.\n")
```

```
## Store separately from WorkerSatisfaction_300rows_deidentified.xlsx - never on public systems.
```

Closing Summary

Step	Technique	What changed
1	Pseudonymization	`worker_id`, `email_id`, `owner_id` → pseudonym codes (key file)
2	Aggregation	`age` (DOB) → `age_band` (5-year groups)
3	Pseudonymization	`city` → `City_01` …; `orchard_id` → `Orchard_01` …
4	Anonymization	`immigration_stat` pooled if below threshold; `username_id` and `comments` removed

The de-identified dataset keeps all satisfaction and treatment ratings for analysis while substantially lowering re-identification risk. **The data key is highly sensitive** — it links pseudonym codes back to real names, emails, and locations for Steps 1 and 3 only. Protect and store it separately.