# Quiz 1: Share Your Comuna-Level Summary Tables

IELE756 – Preparación y Análisis de Datos

Leo Ferres, PhD

April 16, 2026

## Quiz 1: Share Your Comuna-Level Summary Tables

**Points**: 2
**Released**: Thursday, April 16, 2026 (together with Tarea 3)
**Due**: Monday, April 20, 2026, 23:59
**Submission**: Canvas – see Part 3 below

---

## Goal

Tarea 3 asks you to fit ecological regression models at the comuna level. Your own team has two or three comunas, which is not a dataset. To fix this, **every team** must publish the summary tables they built in Tarea 1 and Tarea 2 in a **single agreed format**, so that every team can concatenate all 21 teams' outputs into a class-wide master of roughly 50 RM comunas.

This is a short, mechanical, **compliance** quiz. If your Tarea 1 and Tarea 2 outputs are already tidy, it will take you 20 to 30 minutes. The quiz exists because the moment one team uses a different column name or dtype, everyone's Tarea 3 breaks.

There is no analysis in Quiz 1. You are packaging, validating, and uploading. Nothing else.

---

# Part 1: The Three Files You Must Produce

Name them **exactly** as shown. `NN` is your two-digit, zero-padded team number from `groups.md` (team 7 becomes `team07`, team 21 becomes `team21`):

- `census_teamNN.csv`
- `eno_teamNN.csv`
- `grd_teamNN.csv`

One row per comuna you were assigned. If your team has three comunas, every CSV has three rows.

## 1.1 Census schema (`census_teamNN.csv`)

Columns, **in this order**, with these dtypes:

| Column | Dtype | Source |
|---|---|---|
| `codigo_comuna` | int64 | Tarea 1 Part 4 |
| `nombre_comuna` | string | Tarea 1 Part 4 |
| `pop_total` | int64 | Tarea 1 Part 4 |
| `pop_chilean` | int64 | Tarea 1 Part 4 |
| `pop_foreign` | int64 | Tarea 1 Part 4 |
| `pct_foreign` | float64 | Tarea 1 Part 4 |
| `median_age_chilean` | float64 | Tarea 1 Part 4 |
| `median_age_foreign` | float64 | Tarea 1 Part 4 |
| `mean_schooling_chilean` | float64 | Tarea 1 Part 4 |
| `mean_schooling_foreign` | float64 | Tarea 1 Part 4 |
| `emp_rate_chilean` | float64 | Tarea 1 Part 4 |
| `emp_rate_foreign` | float64 | Tarea 1 Part 4 |
| `dependency_ratio` | float64 | Tarea 1 Part 4 |

`pct_foreign` is a proportion expressed as a **percentage** (0 to 100), not a fraction (0 to 1). Same for the two employment-rate columns and the dependency ratio.

## 1.2 ENO schema (`eno_teamNN.csv`)

| Column | Dtype | Source |
|---|---|---|
| `codigo_comuna` | int64 | Tarea 2 Part A.4 |
| `nombre_comuna` | string | Tarea 2 Part A.4 |
| `eno_total` | int64 | Tarea 2 Part A.4 |
| `eno_chilean` | int64 | Tarea 2 Part A.4 |
| `eno_foreign` | int64 | Tarea 2 Part A.4 |
| `eno_desconocido` | int64 | Tarea 2 Part A.4 |
| `eno_top3_diseases` | string | Tarea 2 Part A.4 |

| Column | Dtype | Source |
|---|---|---|
| `eno_rate_per_10k` | float64 | Tarea 2 Part A.4 |

`eno_top3_diseases` is a **single string** with the three names joined by `" | "` (space, pipe, space), e.g. `"Coqueluche | Sifilis | Tuberculosis"`. Do not use a comma separator.

### 1.3 GRD schema (`grd_teamNN.csv`)

| Column | Dtype | Source |
|---|---|---|
| `codigo_comuna` | int64 | Tarea 2 Part B.4 |
| `nombre_comuna` | string | Tarea 2 Part B.4 |
| `grd_total` | int64 | Tarea 2 Part B.4 |
| `grd_chilean` | int64 | Tarea 2 Part B.4 |
| `grd_foreign` | int64 | Tarea 2 Part B.4 |
| `grd_pct_foreign` | float64 | Tarea 2 Part B.4 |
| `grd_mean_los` | float64 | Tarea 2 Part B.4 |
| `grd_mean_los_chilean` | float64 | Tarea 2 Part B.4 |
| `grd_mean_los_foreign` | float64 | Tarea 2 Part B.4 |
| `grd_mean_severity` | float64 | Tarea 2 Part B.4 |
| `grd_mortality_rate` | float64 | Tarea 2 Part B.4 |
| `grd_top3_chapters` | string | Tarea 2 Part B.4 |
| `grd_rate_per_10k` | float64 | Tarea 2 Part B.4 |

`grd_pct_foreign` and `grd_mortality_rate` are **percentages** (0 to 100), not fractions. `grd_top3_chapters` uses the same `" | "` separator as above.

### 1.4 Missing values

If a cell is genuinely unknown (for example, zero foreign-born in a comuna so `mean_schooling_foreign` is undefined), write it as an empty cell, not `NaN`, not `-99`, not `"NA"`. pandas will read empty cells as `NaN` on load, which is what everyone expects.

---

## Part 2: Validate Before You Upload

Paste the block below into a new cell in your Tarea 2 notebook, edit the two lines marked `EDIT THIS`, and run it. The script checks column names, order, dtypes, row counts, and comuna codes. If **any** assertion fails, fix the CSV and rerun until the script prints "All three files OK". Do not upload files that fail the script.

```python
import pandas as pd
from pathlib import Path

# ----- EDIT THIS -----
TEAM = 7                          # your team number from groups.md
MY_COMUNAS = [13119, 13122]       # codigo_comuna values for your team
# ---------------------

CENSUS_COLS = [
    ("codigo_comuna", "int"),
    ("nombre_comuna", "str"),
    ("pop_total", "int"),
    ("pop_chilean", "int"),
    ("pop_foreign", "int"),
    ("pct_foreign", "float"),
    ("median_age_chilean", "float"),
    ("median_age_foreign", "float"),
    ("mean_schooling_chilean", "float"),
    ("mean_schooling_foreign", "float"),
    ("emp_rate_chilean", "float"),
    ("emp_rate_foreign", "float"),
    ("dependency_ratio", "float"),
]

ENO_COLS = [
    ("codigo_comuna", "int"),
    ("nombre_comuna", "str"),
    ("eno_total", "int"),
    ("eno_chilean", "int"),
    ("eno_foreign", "int"),
    ("eno_desconocido", "int"),
    ("eno_top3_diseases", "str"),
    ("eno_rate_per_10k", "float"),
]

GRD_COLS = [
    ("codigo_comuna", "int"),
    ("nombre_comuna", "str"),
    ("grd_total", "int"),
    ("grd_chilean", "int"),
    ("grd_foreign", "int"),
    ("grd_pct_foreign", "float"),
    ("grd_mean_los", "float"),
    ("grd_mean_los_chilean", "float"),
    ("grd_mean_los_foreign", "float"),
    ("grd_mean_severity", "float"),
```

```
    ("grd_mortality_rate", "float"),
    ("grd_top3_chapters", "str"),
    ("grd_rate_per_10k", "float"),
]

def _check_dtype(series, kind, col):
    if kind == "int":
        assert pd.api.types.is_integer_dtype(series), \
            f"{col}: expected int, got {series.dtype}"
    elif kind == "float":
        assert pd.api.types.is_float_dtype(series) or \
               pd.api.types.is_integer_dtype(series), \
            f"{col}: expected float, got {series.dtype}"
    elif kind == "str":
        assert series.dtype == object, \
            f"{col}: expected string, got {series.dtype}"

def validate(path, schema, my_comunas):
    path = Path(path)
    assert path.exists(), f"{path} does not exist"
    df = pd.read_csv(path)

    expected_cols = [c for c, _ in schema]
    assert list(df.columns) == expected_cols, (
        f"{path}: columns must be exactly {expected_cols}, "
        f"got {list(df.columns)}"
    )

    for col, kind in schema:
        _check_dtype(df[col], kind, col)

    assert len(df) == len(my_comunas), (
        f"{path}: expected {len(my_comunas)} rows, got {len(df)}"
    )
    assert set(df["codigo_comuna"]) == set(my_comunas), (
        f"{path}: codigo_comuna must be exactly {set(my_comunas)}, "
        f"got {set(df['codigo_comuna'])}"
    )
    assert df["codigo_comuna"].is_unique, \
        f"{path}: codigo_comuna must be unique"

    print(f"OK  {path.name:30s}  rows={len(df)}  cols={len(df.columns)}")
    return df

team_tag = f"team{TEAM:02d}"
validate(f"census_{team_tag}.csv", CENSUS_COLS, MY_COMUNAS)
```

```
validate(f"eno_{team_tag}.csv",    ENO_COLS,    MY_COMUNAS)
validate(f"grd_{team_tag}.csv",    GRD_COLS,    MY_COMUNAS)
print("All three files OK -- ready to upload.")
```

### 2.1 How to adjust your Tarea 1 and Tarea 2 outputs

Most teams will only need to rename a few columns and fix dtypes. If your Tarea 1 summary already used the column names listed in Section 1.1, you are mostly done. Common fixes:

- **Column rename**: `df = df.rename(columns={"old_name": "new_name"})`.
- **Reorder columns**: `df = df[expected_cols]`.
- **Coerce integer columns that became floats**: `df["pop_total"] = df["pop_total"].astype("int64")`. If a column has missing values, first decide whether the missing rows are real (if so, use pandas `Int64` nullable dtype; the validator accepts that).
- **Percentages vs fractions**: if you stored `pct_foreign` as `0.08`, multiply by 100.
- **Write the file**: `df.to_csv(f"census_team{TEAM:02d}.csv", index=False, encoding="utf-8")`. Do not write the pandas index.

---

## Part 3: Where to Upload

Submit the three CSVs to **two** Canvas places. Both are required and both are checked automatically:

1. **Quiz 1 assignment page** on Canvas. Attach the three CSVs directly to your submission. This is how the staff records the grade.
2. **Class-wide group "IELE756 Tarea 3 data pool"**, under the Canvas **Files** tab of that group. Every student in the course is already a member of this group; you reach it from the course left sidebar: **People** -> **View User Groups** -> **IELE756 Tarea 3 data pool** -> **Files**. Upload your three CSVs directly into the root of the group's Files area (no subfolders). This is how every other team gets your files.

Only one member of the pair needs to submit. The other member's name must appear in the submission comment on the Quiz 1 assignment page.

If you edit your files after submitting (for example, because you noticed an error), replace them in **both** places. To replace in the group Files area, upload the corrected CSV with the **same file name**; Canvas will prompt you to overwrite. Confirm the overwrite. Do **not** upload a second file with a `team07-v2.csv`-style name.

**Do not delete** other teams' files from the group Files area. You have the permission to do so, and doing so will be treated as academic misconduct.

## Part 4: How Tarea 3 Consumes the Pool

This section is informational. You do not need to do it for Quiz 1, but you will need it for Tarea 3 Part 0. It is here so you understand why the schemas are strict.

After the Quiz 1 deadline, every team opens the "IELE756 Tarea 3 data pool" group's Files tab, selects all files, and downloads them as a single ZIP (Canvas offers a "Download" button above a multi-selection). Extract into one folder, for example `shared/`:

```
shared/
  census_team01.csv
  census_team02.csv
  ...
  eno_team01.csv
  eno_team02.csv
  ...
  grd_team01.csv
  grd_team02.csv
  ...
```

Then a one-liner per dataset builds the class master:

```python
import pandas as pd
import glob

def pool(pattern):
    frames = [pd.read_csv(p) for p in sorted(glob.glob(pattern))]
    master = pd.concat(frames, ignore_index=True)
    # sanity checks
    dupes = master[master.duplicated("codigo_comuna", keep=False)]
    if len(dupes):
        print(f"WARNING: duplicate comunas in {pattern}:")
        print(dupes.sort_values("codigo_comuna"))
    print(f"{pattern}: {len(master)} rows, "
          f"{master['codigo_comuna'].nunique()} unique comunas")
    return master

census_master = pool("shared/census_team*.csv")
eno_master    = pool("shared/eno_team*.csv")
grd_master    = pool("shared/grd_team*.csv")
```

If a `codigo_comuna` appears in two teams' outputs (it should not, since the group assignments are disjoint), inspect the rows, choose one by an explicit rule,

and document your choice in your Tarea 3 notebook.

---

## Grading

Quiz 1 is worth **2 points**, graded **all-or-nothing**:

- **2 / 2** – all three CSVs pass the validation script, are named correctly, and are uploaded to **both** the Quiz 1 assignment page and the "IELE756 Tarea 3 data pool" group Files area before the deadline.
- **0 / 2** – anything else.

A late Quiz 1 hurts the whole class (other teams cannot finish their Tarea 3 master until yours is in). The standard 2-day extension policy does **not** apply to Quiz 1. Please submit on time.

---

Typeset with: `pandoc assignments/Quiz1.md -o assignments/Quiz1.pdf --pdf-engine=pdflatex && evince assignments/Quiz1.pdf &`