

Working with Data

Session 3 — Can I trust the answer?

Block 6 — Did every value survive? Bronze, silver, gold

Richer than the world?

Block 5's answer: the largest economy in 2024, at purchasing-power parity, is **China**.

Now divide GDP by people: **GDP per person**. The World Bank's pages have a row for the whole world too.

In 2024, is China's GDP per person above the world's, or below it?

Lab 5, layer by layer

	bronze: as it arrived	silver: one record, one row	gold: shaped for the question
Your team	the four pages, as the API sent them	population : 3,975 rows	places_2024 : 217 places
The report	the same four pages	report_population : page 1 only, 1,000 rows	report_board : one row, 19 places

Did every record arrive? The report's silver never read three of the four pages.

Bronze, silver, gold: the words as industry uses them

Layer	What it is	In this block
bronze	the data exactly as it arrived, never edited: every later table can be rebuilt from it	the World Bank's pages, read into <code>gdp_bronze</code> and <code>population_bronze</code> : every column still text
silver	cleaned and checked: the right types, consistent and explicit units, duplicates and missing values handled, sometimes joined; still at the detail of the records	<code>gdp_silver</code> and <code>population_silver</code> : the same rows, each value made a number
gold	business-ready: shaped for the reports and the questions people ask, often aggregated	<code>gold</code> : GDP per person, 2024

A convention, not a law: the names, and how many layers, vary. **A layer is not a guarantee:** this course counts between them.

Cell 1 · Bronze: four records, as the API sent them

Four records of the GDP pages, for 2024, followed through every layer: **China, Cuba, Singapore, World**.

cell 1

```
con.sql("""  
    SELECT * FROM gdp_bronze  
    WHERE country IN ('China', 'Cuba', 'Singapore', 'World')  
    AND year = '2024'  
    ORDER BY country  
""").df()
```

	country	year	value
0	China	2024	33592044935159.1
1	Cuba	2024	NaN
2	Singapore	2024	812252052277.036
3	World	2024	174159721560706

Cell 2 · Silver: the value made a number with **CAST**

cell 2

```
con.sql("""
    CREATE OR REPLACE VIEW gdp_silver AS
    SELECT country,
           CAST(year AS INTEGER) AS year,
           CAST(value AS DOUBLE) AS gdp
    FROM gdp_bronze
""")
con.sql("""
    SELECT * FROM gdp_silver
    WHERE country IN ('China', 'Cuba', 'Singapore', 'World') AND year = 2024
    ORDER BY country
""").df()
```

	country	year	gdp
0	China	2024	33,592,044,935,159.10
1	Cuba	2024	NaN
2	Singapore	2024	812,252,052,277.04
3	World	2024	174,159,721,560,706.00

Predict: on four rows, how many rows, how many numbers?

The same four rows of silver: China, Cuba, Singapore, World. Cuba's `gdp` is `NULL`.

```
SELECT COUNT(*) AS rows, COUNT(gdp) AS numbers
FROM gdp_silver
WHERE country IN ('China', 'Cuba', 'Singapore', 'World')
      AND year = 2024
```

Predict: what does `rows` print, and what does `numbers` print?

On four rows — what it printed

cell 3

```
con.sql("""  
    SELECT COUNT(*) AS rows, COUNT(gdp) AS numbers  
    FROM gdp_silver  
    WHERE country IN ('China', 'Cuba', 'Singapore', 'World') AND year = 2024  
""").df()
```

	rows	numbers
0	4	3

4 rows, 3 numbers. A **NULL** is still a row; **COUNT(gdp)** skips it.

Cell 4 · Silver for population: the same step, as a **BIGINT**

```
cell 4 con.sql("""
        CREATE OR REPLACE VIEW population_silver AS
        SELECT country,
               CAST(year AS INTEGER) AS year,
               CAST(value AS BIGINT) AS population
        FROM population_bronze
        """)
con.sql("""
        SELECT * FROM population_silver
        WHERE country IN ('China', 'Cuba', 'Singapore', 'World') AND year = 2024
        ORDER BY country
        """).df()
```

	country	year	population
0	China	2024	1408975000
1	Cuba	2024	10979783
2	Singapore	2024	6036860
3	World	2024	8140897523

Predict: the four, in gold

Cell 5: gold, two silvers joined. Then our four records.

```
CREATE OR REPLACE VIEW gold AS
SELECT g.country, g.gdp / p.population AS gdp_per_person
FROM gdp_silver AS g
JOIN population_silver AS p
  ON p.country = g.country AND p.year = g.year
WHERE g.year = 2024 AND g.gdp IS NOT NULL
```

Predict: how many of the four does gold print? And your answer from the start: is China above the world, or below?

The four, in gold — what it printed

```
cell 5 con.sql("""
CREATE OR REPLACE VIEW gold AS
SELECT g.country, g.gdp / p.population AS gdp_per_person
FROM gdp_silver AS g
JOIN population_silver AS p
  ON p.country = g.country AND p.year = g.year
WHERE g.year = 2024 AND g.gdp IS NOT NULL
""")
con.sql("SELECT * FROM gold WHERE country IN ('China', 'Cuba', 'Singapore', 'World') ORDER BY country").df()
```

	country	gdp_per_person
0	China	23,841.48
1	Singapore	134,548.76
2	World	21,393.18

Three rows. China 23,841.48 per person; World 21,393.18: China is above. Cuba has no GDP number, so the **WHERE** line left it out, on purpose.

Cell 6 · Bronze against silver: did every row, and every number, survive?

```
cell 6 display(con.sql("SELECT COUNT(*) AS rows, COUNT(value) AS numbers FROM gdp_bronze").df())  
display(con.sql("SELECT COUNT(*) AS rows, COUNT(gdp) AS numbers FROM gdp_silver").df())
```

	rows	numbers
0	3975	3681

	rows	numbers
0	3975	3681

On your own: the rows with no GDP number

In your notes, alone: the whole query, `SELECT` to the end.

In silver, how many of 2024's rows have no GDP number?

`gdp_silver`'s columns: `country`, `year`, `gdp`.

The rows with no GDP number — what it printed

```
cell 7 display(con.sql("""
    SELECT COUNT(*) - COUNT(gdp) AS no_number
    FROM gdp_silver
    WHERE year = 2024
""").df())
display(con.sql("""
    SELECT COUNT(*) AS no_number
    FROM gdp_silver
    WHERE year = 2024 AND gdp IS NULL
""").df())
```

	no_number
0	23

	no_number
0	23

Cell 8 · Population as an **INTEGER**, and **CAST** stops

People come in whole numbers, so **INTEGER** looks right. It holds up to 2,147,483,647.

```
cell 8 con.sql("""  
    CREATE OR REPLACE VIEW population_int AS  
    SELECT country,  
           CAST(year AS INTEGER) AS year,  
           CAST(value AS INTEGER) AS population  
    FROM population_bronze  
    """)  
con.sql("SELECT COUNT(population) FROM population_int").df()
```

ConversionException: Conversion Error: Could not convert string '3552306404' to INT32 when casting from source column value

An error, at the first value it cannot hold: 3,552,306,404 people, a group's. The error is a help: it names the value.

Predict: the shortcut

Cell 9: `TRY_CAST` never errors: what it cannot read becomes `NULL`.

```
CREATE OR REPLACE VIEW population_int AS
SELECT country,
       CAST(year AS INTEGER) AS year,
       TRY_CAST(value AS INTEGER) AS population
FROM population_bronze
```

Predict: bronze has 3,960 population numbers. Does `population_int` have the same number, or fewer?

The shortcut — what it printed

cell 9

```
con.sql("""
    CREATE OR REPLACE VIEW population_int AS
    SELECT country,
           CAST(year AS INTEGER)      AS year,
           TRY_CAST(value AS INTEGER) AS population
    FROM population_bronze
""")
display(con.sql("SELECT COUNT(value) AS numbers_in_bronze FROM population_bronze").df())
display(con.sql("SELECT COUNT(population) AS numbers_in_silver FROM population_int").df())
```

	numbers_in_bronze
0	3960

	numbers_in_silver
0	3810

3,960 in bronze, 3,810 in silver: 150 numbers gone, and no error.

Predict: does it matter for your question?

Cell 10: gold's query, with `population_int` in place of `population_silver`, on our four.

```
SELECT g.country, g.gdp / p.population AS gdp_per_person
FROM gdp_silver AS g
JOIN population_int AS p
  ON p.country = g.country AND p.year = g.year
WHERE g.year = 2024 AND g.gdp IS NOT NULL
  AND g.country IN ('China', 'Cuba', 'Singapore', 'World')
ORDER BY g.country
```

Predict: what does it print for China, and for World? Can it still say whether China is above the world?

Gold on the shortcut — what it printed

cell 10

```
con.sql("""
    SELECT g.country, g.gdp / p.population AS gdp_per_person
    FROM gdp_silver AS g
    JOIN population_int AS p
      ON p.country = g.country AND p.year = g.year
    WHERE g.year = 2024 AND g.gdp IS NOT NULL
      AND g.country IN ('China', 'Cuba', 'Singapore', 'World')
    ORDER BY g.country
""").df()
```

	country	gdp_per_person
0	China	23,841.48
1	Singapore	134,548.76
2	World	NaN

China 23,841.48, as before. World: no number. Is China above the world? This gold cannot say, and nothing errored.

What you just saw

1. **Bronze** is the data as it arrived, never edited. **Silver** is cleaned and checked, still one row per record. **Gold** is shaped for the question people ask.
2. A **NULL** is still a row. `COUNT(*)` counts rows; `COUNT(gdp)` counts the rows that hold a number.
3. **A layer is not a guarantee.** At each step, two questions: did every record arrive, and did every value survive? The first count that drops, with no reason you can name, is at the step to repair.

Homework 3 assigned — Waste per person: two files, three layers

The question: **municipal waste generated per person, for the 27 EU members, in 2024**, answered **alone**: Eurostat's totals, divided by the World Bank's population. Due **Saturday 31 October, 23:59**, on Moodle. Plan for about 3 hours.

The starter is `earino/ecbs5294-hw03-per-person` on GitHub; its README has the clone line, for home. **The setups that read both files are supplied**, with the same code as Labs 5 and 6.

1. **Did every record arrive?** (*your count against the pages' `total`*)
2. **Did every value survive?** (*waste silver with `CAST`, and your count before and after*)
3. **Gold: waste per person, 2024** (*silver and population joined; a row for every member, a number where there is one*)
4. **Can you trust this answer?** (*a colleague's gold: it runs, and one number in the report is wrong*)

Every query is yours. Under every question: your own count, and one sentence. Then three lines of reflection. The notebook is the whole submission.

The final exam

Closed book, individual, in person, about 70 minutes; 50% of the course grade, in a separate session after Homework 3 is due. The date comes from the program calendar, on Moodle. One A4 sheet of notes, handwritten by you, both sides; no laptop, no phone, no AI.

Part	What	Points
A	Ten multiple-choice questions on short scenarios	40
B	Four queries you write by hand, scored clause by clause	40
C	One diagnosis note, ending with the two sentences you would send a manager	20

The same skills as the homeworks, on new scenarios.

Getting ready for it

- **This evening, on Moodle:** the **mock exam**, in the final's format. Sit it like the exam: closed book, your one sheet, about 70 minutes.
- **Also this evening:** the **Knowledge check 3 retry**, one query, ungraded. Answer in the Moodle text box by **Wednesday 21 October, 23:59**; back by **Friday 23 October**, with a model answer and one correction from a TA.
- **Monday 2 November:** the mock's self-check key, with Homework 3's.
- Part B is marked like the knowledge checks: against a model query, clause by clause.
- **Start your sheet now.** Making it is revision.

5-minute stand-and-stretch — start the Lab 6 clone first

Roles change for Lab 6. **The new keyboard person clones**, on the laptop that will be the team's work laptop:

```
git clone https://github.com/earino/ecbs5294-lab06-layers.git  
cd ecbs5294-lab06-layers  
uv sync
```

Then leave the laptop. Back in five.

Lab 6 — Waste per person: bronze, silver, gold

Where this lab goes: Eurostat's municipal waste file: a new kind of bronze, a text file in which every cell is text, sometimes with a letter after the number. Section A shows you its cells. You make silver and gold, count between the layers, and find who generates the most waste per person. Then, in section D, a colleague's report on the same file answers the same question, and ranks **15 places**. Nothing errors.

First minute: the clone finished? The keyboard opens the folder in VS Code, opens `notebooks/report.ipynb`, picks the `.venv` kernel, types your names in the *Your team* cell, and runs the three setup cells. Then top to bottom, **running each cell as you come to it**.

Your work, in `report.ipynb`. **A:** look at bronze; one count. **B:** your silver, and your count of bronze against it. **C:** your gold, the answer, and one more look of your choice. **D:** the colleague's report: the count that shows where its places went, the cells it lost, and the repair. **E:** the check, supplied.

The syntax is on one page: `REFERENCE-sql.md`. The team next to you at minute 33. The checkpoint is the notebook, on Moodle.

The last ten minutes — explain it to the team next to you

Two teams, one laptop at a time. At your laptop, finished or not, start with one result you found interesting; then the last step you reached; if you repaired the report, which layer was wrong and what the check printed. About two minutes.

The other team asks one question. If you reached the report: **why was it wrong, not just where?** If not: **what is one row of what you showed?** Then swap laptops.

Then the what-if, both teams together: **the colleague's same TRY_CAST , on the waste file in thousand tonnes (THS_T) instead of per person: would its 2024 ranking of waste generated hold all 25 places with a number? What would you count to know?** Commit to an answer before anyone runs anything.

Unsure, or the teams disagree? **Hands up.**

The lab ends here, finished or not. Before you leave, the keyboard uploads the notebook as it is to Moodle; everyone else pastes the *Your team* line.