Added Questions:

How many total applicants are in the scraped database?

```
# Query total number of applicants in the database
total_applicants = fetch_value(
    connection,
    """
    SELECT COUNT(*)
    FROM grad_applications;
    """
)
```

This was a simple query. I used COUNT(*) to count the number of rows in the database. For my set-up this is identical to the number of total applicants. FROM was used to specify I wanted to count from the grad_applications table.

How many accepted applicants for Fall 2026 gave their GPA (two decimal places)?

```
# Query percentage of accepted Fall 2026 applicants who reported a GPA
accepted_fall_2026_gpa_pct = fetch_value(
    connection,
    """
    SELECT
        COALESCE(
            ROUND(
                100.0 *
                COUNT(*) FILTER (WHERE gpa IS NOT NULL AND gpa > 0)
                / NULLIF(COUNT(*), 0),
                2
            ),
            0
        )
    FROM grad_applications
    WHERE term = 'Fall 2026'
      AND LOWER(status) LIKE 'accepted:%';
    """
)
```

This query uses SELECT to denote that a single value will be returned. FROM specifies that the table we are searching is the grad_appliations table. WHERE filters to only entries that have the term ‘Fall 2026’. LOWER is used to remove case-sensitivity and LIKE is used to find statuses that start with accepted. The nested command achieves the following: COUNT(*) once again counts the rows but is filtered using WHERE to find only rows where the GPA is greater than 0 and not Null (aka someone reported a GPA), NULLIF(COUNT(*), 0) is used to prevent dividing by zero in case there are no accepted applicants. The overarching equation divides the total sum of the GPAs by the number of applicants and multiplies by 100 to get a percent. ROUND is used to round to two decimal places. COALESCE returns a result of 0 if instead of Null in case the result is Null.

How many rejected applicants for Fall 2026 gave their GPA (two decimal places)?

```
# Query percentage of rejected Fall 2026 applicants who reported a GPA
rejected_fall_2026_gpa_pct = fetch_value(
    connection,
    """
    SELECT
        COALESCE(
            ROUND(
                100.0 *
                COUNT(*) FILTER (WHERE gpa IS NOT NULL AND gpa > 0)
                / NULLIF(COUNT(*), 0),
                2
            ),
            0
        )
    FROM grad_applications
    WHERE term = 'Fall 2026'
      AND LOWER(status) LIKE 'rejected:%';
    """
)
```

This query uses SELECT to denote that a single value will be returned. FROM specifies that the table we are searching is the grad_appliations table. WHERE filters to only entries that have the term 'Fall 2026'. LOWER is used to remove case-sensitivity and LIKE is used to find statuses that start with rejected. The nested command achieves the following: COUNT(*) once again counts the rows but is filtered using WHERE to find only rows where the GPA is greater than 0 and not Null (aka someone reported a GPA), NULLIF(COUNT(*), 0) is used to prevent dividing by zero in case there are no rejected applicants. The overarching equation divides the total sum of the GPAs by the number of applicants and multiplies by 100 to get a percent. ROUND is used to round to two decimal places. COALESCE returns a result of 0 if instead of Null in case the result is Null.

Given Questions:

How many entries do you have in your database who have applied for Fall 2026?

```
# Query total number of Fall 2026 applications
fall_2026_count = fetch_value(
    connection,
    "SELECT COUNT(*) FROM grad_applications WHERE term = 'Fall 2026';"
)
```

This query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table and WHERE to filter in on term's with 'Fall 2026'.

What percentage of entries are from international students (not American or Other) (to two decimal places)?

```
# Query total number of applications across all terms
total_count = fetch_value(
    connection,
```

```
    "SELECT COUNT(*) FROM grad_applications;"
)
# Query total number of international applicants
international_count = fetch_value(
    connection,
    """
    SELECT COUNT(*)
    FROM grad_applications
    WHERE LOWER(us_or_international) = 'international';
    """
)

# Calculate percentage of international applicants, rounded to 2 decimal places
international_pct = (
    round((international_count / total_count) * 100, 2)
    if total_count and total_count > 0
    else 0
)
```

This query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table and WHERE to filter in on 'international' results.

The math part uses the value found above in conjunction with the total count value to calculate the percentage. It includes logic to ensure that 0's are not throw into the mix.

What is the average GPA, GRE, GRE V, GRE AW of applicants who provide these metrics?

```
# Query average GPA, GRE, GRE Verbal, and GRE Analytical Writing scores
avg_gpa, avg_gre, avg_gre_v, avg_gre_aw = fetch_row(
    connection,
    """
    SELECT
        AVG(gpa),
        AVG(gre),
        AVG(gre_v),
        AVG(gre_aw)
    FROM grad_applications;
    """
)

# Preserve NULL values if no GPA data exists
avg_gpa = avg_gpa if avg_gpa is not None else None
# Preserve NULL values if no GRE data exists
avg_gre = avg_gre if avg_gre is not None else None
# Preserve NULL values if no GRE verbal data exists
avg_gre_v = avg_gre_v if avg_gre_v is not None else None
# Preserve NULL values if no GRE analytical writing data exists
avg_gre_aw = avg_gre_aw if avg_gre_aw is not None else None
```

In this query, SELECT has multiple things under it so it is saying that it will return 4 results in this case. AVG is used to calculate the average of a column and automatically ignores Null values. FROM is used to say we are pulling from the grad_application table.

The python part at the bottom is preserving Null values if they are encountered (which they will be).

What is their average GPA of American students in Fall 2026?

```
# Query average GPA for US applicants applying for Fall 2026
avg_gpa_us_fall_2026 = fetch_value(
    connection,
    """
    SELECT AVG(gpa)
    FROM grad_applications
    WHERE term = 'Fall 2026'
      AND LOWER(us_or_international) = 'us'
      AND gpa IS NOT NULL;
    """
)

# Preserve NULL if no qualifying GPA values exist
avg_gpa_us_fall_2026 = (
    avg_gpa_us_fall_2026
    if avg_gpa_us_fall_2026 is not None
    else None
)
```

SELECT in this case is showing one value that uses AVG to calculate the GPA from filtered rows. FROM is pulling from the grad_applications table. Where is filtering in on the term ‘Fall 2026’, LOWER is used to prevent case sensitivity and the equals is finding us applicants, the following line is saying to include only lines where the GPA is present.

The python part keeps the value as None if the query returns a Null value.

What percent of entries for Fall 2025 are Acceptances (to two decimal places)?

```
# Query total number of Fall 2025 applications
fall_2025_total = fetch_value(
    connection,
    "SELECT COUNT(*) FROM grad_applications WHERE term = 'Fall 2025';"
)

# Query number of accepted Fall 2025 applications
fall_2025_accept = fetch_value(
    connection,
    """
    SELECT COUNT(*)
    FROM grad_applications
    WHERE term = 'Fall 2025'
      AND LOWER(status) LIKE 'accepted%';
    """
)

# Calculate Fall 2025 acceptance rate, rounded to 2 decimal places
fall_2025_accept_pct = (
    round((fall_2025_accept / fall_2025_total) * 100, 2)
    if fall_2025_total and fall_2025_total > 0
    else 0
)
```

The first query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table and WHERE to filter in on term's with 'Fall 2025'.

The second query uses SELECT COUNT (*) to return a value from the row count, FROM to indicate it is the grad_applicantions table, WHERE to hone in on the term 'Fall 2025', LOWER to prevent case sensitivity and LIKE to find applicants who status start with accepted.

The python part uses those values to compute a percentage and returns 0 if there are no applications.

What is the average GPA of applicants who applied for Fall 2025 who are Acceptances?

```
# Query average GPA of accepted Fall 2025 applicants
avg_gpa_fall_2025_accept = fetch_value(
    connection,
    """
    SELECT AVG(gpa)
    FROM grad_applications
    WHERE term = 'Fall 2025'
      AND LOWER(status) LIKE 'accepted%'
      AND gpa IS NOT NULL;
    """
)

# Preserve NULL if no accepted GPA values exist
avg_gpa_fall_2025_accept = (
    avg_gpa_fall_2025_accept
    if avg_gpa_fall_2025_accept is not None
    else None
)
```

The query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table, WHERE to filter in on term's with 'Fall 2025', LOWER to remove case sensitivity, LIKE to find status that starts with accepted, and only incudes GPAs that are not Null.

The python part preserves the Null state if there are no GPA values.

How many entries are from applicants who applied to JHU for a masters degrees in Computer Science?

```
# Query number of JHU Computer Science master's applications
jhu_cs_masters = fetch_value(
    connection,
    """
    SELECT COUNT(*)
    FROM grad_applications
    WHERE LOWER(degree) = 'masters'
      AND (
            LOWER(program) LIKE '%computer science%'
         OR LOWER(program) LIKE '%comp sci%'
```

```
        OR LOWER(program) = '%cs%'
        OR LOWER(program) LIKE '%computer-science%'
        OR LOWER(program) LIKE '%computerscience%'
        OR LOWER(program) LIKE '%csci%'
      )
      AND (
            LOWER(program) LIKE '%johns hopkins%'
        OR LOWER(program) LIKE '%john hopkins%'
        OR LOWER(program) LIKE '%jhu%'
        OR LOWER(program) LIKE '%johns-hopkins%'
        OR LOWER(program) LIKE '%john hopkins university%'
        OR LOWER(program) LIKE '%johns hopkins university%'
        OR LOWER(program) LIKE '%johns hopkins univ%'
        OR LOWER(program) LIKE '%johns hopkins univeristy%'
        OR LOWER(program) LIKE '%johns hopkins univrsity%'
        OR LOWER(program) LIKE '%johns hopkins univertiy%'
        OR LOWER(program) LIKE '%johns hopkins universty%'
        OR LOWER(program) LIKE '%johns hopkins u%'
        OR LOWER(program) LIKE '%johs hopkins%'
        OR LOWER(program) LIKE '%jonhs hopkins%'
        OR LOWER(program) LIKE '%johns hopkinss%'
        OR LOWER(program) LIKE '%john hopkinss%'
      );
    """
)

)
```

The query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table, WHERE to filter in on masters degrees and LOWER to remove case sensitivity. It then goes into a series of LIKE searches that search for possible misspellings of a program name or university.

How many entries from 2026 are acceptances from applicants who applied to Georgetown University, MIT, Stanford University, or Carnegie Mellon University for a PhD in Computer Science?

```
# Count Fall 2026 PhD CS acceptances at target schools (RAW fields)
fall_2026_cs_accept = fetch_value(
    connection,
    """
    SELECT COUNT(*)
    FROM grad_applications
    WHERE term = 'Fall 2026'
      AND LOWER(status) LIKE 'accepted%'
      AND LOWER(degree) = 'phd'

      -- Program matching (CS variants, RAW program)
      AND (
            LOWER(program) LIKE '%computer science%'
        OR LOWER(program) LIKE '%comp sci%'
        OR LOWER(program) = '%cs%'
        OR LOWER(program) LIKE '%computer-science%'
        OR LOWER(program) LIKE '%computerscience%'
      )

      -- University matching (RAW program instead of university)
      AND (
```

```
            LOWER(program) LIKE '%georgetown%'
        OR LOWER(program) LIKE '%george town%'
        OR LOWER(program) LIKE '%geoerge town%'
        OR LOWER(program) LIKE '%george-town%'
        OR LOWER(program) LIKE '%georgetown university%'
        OR LOWER(program) LIKE '%george town university%'
        OR LOWER(program) LIKE '%geoerge town university%'
        OR LOWER(program) LIKE '%georgetown univeristy%'
        OR LOWER(program) LIKE '%georgetown univrsity%'
        OR LOWER(program) LIKE '%georgetown unversity%'
        OR LOWER(program) LIKE '%georgetown univercity%'
        OR LOWER(program) LIKE '%georgetown univ%'

        OR LOWER(program) LIKE '%george town univeristy%'
        OR LOWER(program) LIKE '%geoerge town univeristy%'
        OR LOWER(program) LIKE '%george town univrsity%'
        OR LOWER(program) LIKE '%geoerge town univrsity%'
        OR LOWER(program) LIKE '%mit%'
        OR LOWER(program) LIKE '%m.i.t%'
        OR LOWER(program) LIKE '%massachusetts institute of technology%'
        OR LOWER(program) LIKE '%massachusetts inst of technology%'
        OR LOWER(program) LIKE '%institute of technology (mit)%'
        OR LOWER(program) LIKE '%mass tech%'

        OR LOWER(program) LIKE '%stanford%'
        OR LOWER(program) LIKE '%standford%'
        OR LOWER(program) LIKE '%stanfod%'
        OR LOWER(program) LIKE '%stanforrd%'
        OR LOWER(program) LIKE '%stanford university%'
        OR LOWER(program) LIKE '%standford university%'
        OR LOWER(program) LIKE '%stanford univeristy%'
        OR LOWER(program) LIKE '%stanford univrsity%'
        OR LOWER(program) LIKE '%stanford univ%'

        OR LOWER(program) LIKE '%carnegie mellon%'
        OR LOWER(program) LIKE '%carnegie melon%'
        OR LOWER(program) LIKE '%carnegiemelon%'
        OR LOWER(program) LIKE '%carnegie-mellon%'
        OR LOWER(program) LIKE '%carnegi mellon%'
        OR LOWER(program) LIKE '%carnigie mellon%'
        OR LOWER(program) LIKE '%carnegie mellon university%'
        OR LOWER(program) LIKE '%carnegie melon university%'
        OR LOWER(program) LIKE '%carnegie mellon univeristy%'
        OR LOWER(program) LIKE '%carnegie mellon univrsity%'
        OR LOWER(program) LIKE '%carnegie mellon univ%'

        OR LOWER(program) LIKE '%cmu%'
    );
  """
)
```

The query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table, WHERE to filter in on the term Fall 2026, LIKE to find accepted status, and = to find the type of degree. It then goes into a series of LIKE searches that search for possible misspellings of a program name or university.

Do you numbers for question 8 change if you use LLM Generated Fields (rather than your downloaded fields)?

```
# Count same acceptances using LLM-generated fields
# Fall 2026 PhD CS acceptances (LLM-generated fields)
fall_2026_cs_accept_llm = fetch_value(
    connection,
    """
    SELECT COUNT(*)
    FROM grad_applications
    WHERE term = 'Fall 2026'
      AND LOWER(status) LIKE 'accepted%'
      AND LOWER(degree) = 'phd'

      -- Program matching (CS variants)
      AND (
            LOWER(llm_generated_program) LIKE '%computer science%'
         OR LOWER(llm_generated_program) LIKE '%comp sci%'
         OR LOWER(llm_generated_program) = '%cs%'
         OR LOWER(llm_generated_program) LIKE '%computer-science%'
         OR LOWER(llm_generated_program) LIKE '%computerscience%'
      )

      -- University matching
      AND (
            LOWER(llm_generated_university) LIKE '%georgetown%'
         OR LOWER(llm_generated_university) LIKE '%george town%'
         OR LOWER(llm_generated_university) LIKE '%geoerge town%'
         OR LOWER(llm_generated_university) LIKE '%george-town%'
         OR LOWER(llm_generated_university) LIKE '%georgetown university%'
         OR LOWER(llm_generated_university) LIKE '%george town university%'
         OR LOWER(llm_generated_university) LIKE '%geoerge town university%'
         OR LOWER(llm_generated_university) LIKE '%georgetown univeristy%'
         OR LOWER(llm_generated_university) LIKE '%georgetown univrsity%'
         OR LOWER(llm_generated_university) LIKE '%georgetown unversity%'
         OR LOWER(llm_generated_university) LIKE '%georgetown univercity%'
         OR LOWER(llm_generated_university) LIKE '%georgetown univ%'
         OR LOWER(llm_generated_university) LIKE '%george town univeristy%'
         OR LOWER(llm_generated_university) LIKE '%geoerge town univeristy%'
         OR LOWER(llm_generated_university) LIKE '%george town univrsity%'
         OR LOWER(llm_generated_university) LIKE '%geoerge town univrsity%'

         OR LOWER(llm_generated_university) LIKE '%mit%'
         OR LOWER(llm_generated_university) LIKE '%m.i.t%'
         OR LOWER(llm_generated_university) LIKE '%massachusetts institute of
technology%'
         OR LOWER(llm_generated_university) LIKE '%massachusetts inst of technology%'
         OR LOWER(llm_generated_university) LIKE '%institute of technology (mit)%'
         OR LOWER(llm_generated_university) LIKE '%mass tech%'

         OR LOWER(llm_generated_university) LIKE '%stanford%'
         OR LOWER(llm_generated_university) LIKE '%standford%'
         OR LOWER(llm_generated_university) LIKE '%stanfod%'
         OR LOWER(llm_generated_university) LIKE '%stanforrd%'
         OR LOWER(llm_generated_university) LIKE '%stanford university%'
         OR LOWER(llm_generated_university) LIKE '%standford university%'
         OR LOWER(llm_generated_university) LIKE '%stanford univeristy%'
         OR LOWER(llm_generated_university) LIKE '%stanford univrsity%'
         OR LOWER(llm_generated_university) LIKE '%stanford univ%'

         OR LOWER(llm_generated_university) LIKE '%carnegie mellon%'
         OR LOWER(llm_generated_university) LIKE '%carnegie melon%'
         OR LOWER(llm_generated_university) LIKE '%carnegiemelon%'
         OR LOWER(llm_generated_university) LIKE '%carnegie-mellon%'
         OR LOWER(llm_generated_university) LIKE '%carnegi mellon%'
         OR LOWER(llm_generated_university) LIKE '%carnigie mellon%'
```

```
        OR LOWER(llm_generated_university) LIKE '%carnegie mellon university%'
        OR LOWER(llm_generated_university) LIKE '%carnegie melon university%'
        OR LOWER(llm_generated_university) LIKE '%carnegie mellon univeristy%'
        OR LOWER(llm_generated_university) LIKE '%carnegie mellon univrsity%'
        OR LOWER(llm_generated_university) LIKE '%carnegie mellon univ%'
        OR LOWER(llm_generated_university) LIKE '%cmu%'
    );
  """
)
```

The query uses SELECT to denote that a single value will be returned, COUNT(*) to count the rows, FROM to specific that we are looking at the grad_application table, WHERE to filter in on the term Fall 2026, LIKE to find accepted status, and = to find the type of degree. It then goes into a series of LIKE searches that search for possible misspellings of a program name or university.