

## QUERY DESCRIPTION AND EXPLANATION

### 1. Total applicant count

```
SELECT COUNT (*) FROM applicants
```

Counts every row in applicants' table. Provide baseline context for how much data we have.

### 2. Fall 2026 Count

```
SELECT COUNT(*) FROM applicants WHERE term = 'Fall 2026'
```

Counts entries where the term field exactly matches "Fall 2026." Fall 2026 is the most recent and relevant admission cycle for analysis

### 3. International Percentage

```
SELECT ROUND(  
    100.0 * COUNT(*) FILTER (WHERE us_or_international = 'International')  
    / COUNT(*), 2  
) FROM applicants
```

Calculates what percentage of all entries are marked "International". Uses FILTER clause for a cleaner conditional counting in a single pass. Rounded to 2 decimal places for readability.

### 4. Average GPA, GRE, GRE V, GRE AW

```
SELECT  
    ROUND(AVG(gpa)::numeric, 2),  
    ROUND(AVG(gre)::numeric, 2),  
    ROUND(AVG(gre_v)::numeric, 2),  
    ROUND(AVG(gre_aw)::numeric, 2)  
FROM applicants
```

WHERE gpa IS NOT NULL OR gre IS NOT NULL

OR gre\_v IS NOT NULL OR gre\_aw IS NOT NULL

Calculates averages for all test metrics in one query. Combines four calculations into one query for efficiency. The WHERE clause ensures we only consider rows that have at least one metric. AVG() automatically ignores NULL values per column.

#### 5. Average GPA of American Students (Fall 2026)

```
SELECT ROUND(AVG(gpa)::numeric, 2)
```

```
FROM applicants
```

```
WHERE us_or_international = 'American'
```

```
AND term = 'Fall 2026'
```

```
AND gpa IS NOT NULL
```

Average GPA for American students who reported a GPA. Filters on three conditions to get a specific demographic slice. Explicit gpa IS NOT NULL ensures we only average actual values.

#### 6. Acceptance Percentage for Fall 2026

```
SELECT ROUND(
```

```
100.0 * COUNT(*) FILTER (WHERE status ILIKE 'Accepted%%')
```

```
/ COUNT(*), 2
```

```
) FROM applicants
```

```
WHERE term = 'Fall 2026'
```

Calculates what percentage of Fall 2026 entries have an acceptance status. Uses ILIKE 'Accepted%' because status values includes dates (e.g. "Accepted on 14 Feb"). The %% is Python string escaping for %.

#### 7. Average GPA of Accepted Applicants (Fall 2026)

```
SELECT ROUND(AVG(gpa)::numeric, 2)
```

FROM applicants

WHERE term = 'Fall 2026'

AND status ILIKE 'Accepted%%'

AND gpa IS NOT NULL

Averages GPA for accepted Fall 2026 applicants who reported a GPA. Allows comparison with another query to see if accepted students have higher GPAs.

#### 8. JHU Masters in Computer Science Count

SELECT COUNT(\*)

FROM applicants

WHERE llm\_generated\_university ILIKE '%%Hopkins%%'

AND llm\_generated\_program ILIKE '%%Computer Science%%'

AND degree = 'Masters'

Counts Masters applicants to Hopkins CS program. Uses %Hopkins% pattern instead of exact match to catch misspellings (“John Hopkins”, “John’s Hopkins”, etc) Uses LLM-standardized fields for cleaner matching.

#### 9. PhD CS Acceptances (Program Field)

SELECT COUNT(\*)

FROM applicants

WHERE term ILIKE '%%2026'

AND status ILIKE 'Accepted%%'

AND degree = 'PhD'

AND program ILIKE '%%Computer Science%%'

AND (program ILIKE '%%Georgetown University%%'

OR program ILIKE '%%Massachusetts Institute of Technology%%'

OR program ILIKE '%%Stanford University%%')

OR program ILIKE '%%Carnegie Mellon University%%')

Counts 2026 PhD CS acceptances at 4 elite schools using the raw program field. Searches the original unprocessed program field to show what is possible without LLM standardization Uses full university names for matching.

#### 10. PhD CS Acceptances (LLM Fields)

```
SELECT COUNT(*)
FROM applicants
WHERE term ILIKE '%%2026'
AND status ILIKE 'Accepted%%'
AND degree = 'PhD'
AND llm_generated_program ILIKE '%%Computer Science%%'
AND llm_generated_university IN (
    'Georgetown University',
    'Massachusetts Institute of Technology',
    'Stanford University',
    'Carnegie Mellon University'
)
```

Same as previous query but using LLM-standardized fields. Demonstrates the value of LLM standardization. Can use exact IN matching instead of pattern matching. In some cases, catching more entries due to normalized names.

#### 11. Top 10 Programs

```
SELECT llm_generated_program, COUNT(*) AS num_applicants
FROM applicants
WHERE llm_generated_program IS NOT NULL AND llm_generated_program != ''
GROUP BY llm_generated_program
```

ORDER BY num\_applicants DESC

LIMIT 10

Finds the 10 most frequently applied-to programs. Uses LLM-standardized fields so “CS”, “Comp Sci” and “Computer Science” all count together. GROUP BY aggregates, ORDER BY DESC ranks and LIMIT 10 caps results

## 12. Top 10 Universities

SELECT llm\_generated\_university, COUNT(\*) AS num\_applicants

FROM applicants

WHERE llm\_generated\_university IS NOT NULL AND llm\_generated\_university != "

GROUP BY llm\_generated\_university

ORDER BY num\_applicants DESC

LIMIT 10

Finds the 10 most frequently applied-to universities. Same logic as before. LLM standardization ensures “MIT” and “Massachusetts Institute of Technology” count together

## 13. Acceptance by degree Type

SELECT

degree,

COUNT(\*) AS total,

COUNT(\*) FILTER (WHERE status ILIKE 'Accepted%') AS accepted,

ROUND(

100.0 \* COUNT(\*) FILTER (WHERE status ILIKE 'Accepted%')

/ COUNT(\*), 2

) AS acceptance\_rate

FROM applicants

WHERE degree IN ('Masters', 'PhD', 'PsyD')

GROUP BY degree

ORDER BY degree

Calculates acceptance rate for each of the three main degree types. Uses FILTER to count total and accepted in one pass per group. Limits to Masters, PhD and PsyD since other degrees have minimal entries

#### 14. Acceptance rate by nationality

SELECT

us\_or\_international,

COUNT(\*) AS total,

COUNT(\*) FILTER (WHERE status ILIKE 'Accepted%') AS accepted,

ROUND(

100.0 \* COUNT(\*) FILTER (WHERE status ILIKE 'Accepted%')

/ COUNT(\*), 2

) AS acceptance\_rate

FROM applicants

WHERE us\_or\_international IN ('American', 'International')

GROUP BY us\_or\_international

ORDER BY us\_or\_international

Calculate acceptance rate for American vs International applicants. Excludes ~6330 entries with empty nationality. Same FILTER pattern for efficient grouped calculation.