1. How many entries do you have in your database who have applied for Spring 2025?

Answer:

Entries for Spring 2025: 107

Query Used:

```python
def count_spring_2025_entries(conn):
    """Counts number of applicants for Spring 2025.
    Original Question: How many entries do you have in your database who have applied for Fall 2024?
    """
    cur = conn.cursor()
    cur.execute("""
        SELECT COUNT(*) FROM applicants
        WHERE term ILIKE '%Spring 2025%'
    """)
    result = cur.fetchone()[0]
    cur.close()
    print(f"Entries for Spring 2025: {result}")
    return result
```

Rationale:

This query counts the total number of rows in the applicants table where the term column contains the phrase "Spring 2025". Specifically, I used ILIKE to ensure matches regardless of capitalization or formatting (case-insensitive matching). Even though during the creation of the original database, the dates are formatted in a specific way, case-insensitive matching makes the query robust to differences in how the term value is stored.

---

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

Answer:

Percentage of international students: 55.03%

Query Used:

```python
def percent_international(conn):
    """Calculates percentage of international students (not American or Other).
    Original Question: What percentage of entries are from international students (not American or Other) (to two decimal places)?
    """
    cur = conn.cursor()
    cur.execute("""
        SELECT COUNT(*) FROM applicants
        WHERE us_or_international ILIKE '%International%'
    """)
    international = cur.fetchone()[0]
    cur.execute("SELECT COUNT(*) FROM applicants")
    total = cur.fetchone()[0]
    percent = (international / total) * 100 if total > 0 else 0
    cur.close()
    print(f"Percentage of international students: {percent:.2f}%")
    return round(percent, 2)
```

Rationale:

The first query counts all applicants labeled as international (case-insensitive) in the us_or_international field. The second query counts the total number of applicants. This allows for easy percentage calculation by dividing the international applicant count by the total applicant count.

---

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

Answer:

Average GPA: 3.73

Average GRE: 264.07

Average GRE V: 159.83

Average GRE AW: 8.81

Query:

```
def average_metrics(conn):
    """Finds the average GPA, GRE, GRE V, and GRE AW for applicants who provided each metric.

    For each metric, calculates the average using only applicants who provided that metric.
    Original Question: What is the average GPA, GRE, GRE V, GRE AW of applicants who provide these metrics?
    """
    cur = conn.cursor()
    # Individual averages, using only non-null values for each metric
    cur.execute("SELECT AVG(gpa) FROM applicants WHERE gpa IS NOT NULL")
    gpa = cur.fetchone()[0]
    cur.execute("SELECT AVG(gre) FROM applicants WHERE gre IS NOT NULL")
    gre = cur.fetchone()[0]
    cur.execute("SELECT AVG(gre_v) FROM applicants WHERE gre_v IS NOT NULL")
    gre_v = cur.fetchone()[0]
    cur.execute("SELECT AVG(gre_aw) FROM applicants WHERE gre_aw IS NOT NULL")
    gre_aw = cur.fetchone()[0]
    cur.close()
    print(f"Average GPA: {gpa:.2f}, GRE: {gre:.2f}, GRE V: {gre_v:.2f}, GRE AW: {gre_aw:.2f}")
    return gpa, gre, gre_v, gre_aw
```

Rationale:

This query uses AVG() to calculate the average for each of the four numerical metrics. Because in the original database, values that were not provided were filled in with **null**, we can consider only the rows where the value is provided for each numerical metric by using **IS NOT NULL**, and compute the averages from those applicants only.

---

4. What is the average GPA of American students in Spring 2025?

Answer:

Average GPA (American, Spring 2025): 3.63

Query Used:

```python
def average_gpa_american_spring_2025(conn):
    """Finds average GPA of American students who applied for Spring 2025.
    Original Question: What is their average GPA of American students in Fall 2024?
    """
    cur = conn.cursor()
    cur.execute("""
        SELECT AVG(gpa)
        FROM applicants
        WHERE us_or_international ILIKE '%American%'
        AND term ILIKE '%Spring 2025%'
        AND gpa IS NOT NULL
    """)
    result = cur.fetchone()[0]
    cur.close()
    print(f"Average GPA (American, Spring 2025): {result:.2f}")
    return result
```

Rationale:

This query combines many of the query functions used above. I used AVG() to calculate the average from the resulting selections, locating applicant data only whose us_or_international field contains "American" (case-insensitive), applied for "Spring 2025", and reported a GPA. Similarly, as missing data is indicated by **null,** using **IS NOT** NULL ensures that the averaging only occurs over valid records. The **AND** logic ensures that all conditions must be satisfied, which is what we want.

---

5. What percent of entries for Spring 2025 are Acceptances (to two decimal places)?

Answer:

Percent Acceptances (Spring 2025): 54.21%

Query Used:

```python
def percent_acceptances_spring_2025(conn):
    """Percent of Spring 2025 entries that are Acceptances.
    Original Question: What percent of entries for Fall 2024 are Acceptances (to two decimal places)?
    """
    cur = conn.cursor()
    cur.execute("""
        SELECT COUNT(*) FROM applicants
        WHERE term ILIKE '%Spring 2025%'
    """)
    total = cur.fetchone()[0]
    cur.execute("""
        SELECT COUNT(*) FROM applicants
        WHERE term ILIKE '%Spring 2025%'
        AND status ILIKE '%Accepted%'
    """)
    accepted = cur.fetchone()[0]
    percent = (accepted / total) * 100 if total > 0 else 0
    cur.close()
    print(f"Percent Acceptances (Spring 2025): {percent:.2f}%")
    return round(percent, 2)
```

Rationale:

The first query counts accepted applicants for Spring 2025 (case-insensitive), and the second query counts all applicants for that term the **AND** logic in the second query further narrows the query from the first query by including an additional condition where the field status must be accepted. This allows for simple percentage calculate where it is the accepted count divided by the total, times 100, getting the percentage acceptance for that cycle

---

6. What is the average GPA of applicants who applied for Spring 2025 who are Acceptances?

Answer:

Average GPA (Accepted, Spring 2025): 3.57

Query Used:

```python
def average_gpa_accepted_spring_2025(conn):
    """Average GPA of accepted applicants who applied for Spring 2025.
    Original Question: What is the average GPA of applicants who applied for Fall 2024 who are Acceptances?"""
    cur = conn.cursor()
    cur.execute("""
        SELECT AVG(gpa)
        FROM applicants
        WHERE term ILIKE '%Spring 2025%'
        AND status ILIKE '%Accepted%'
        AND gpa IS NOT NULL
    """)
    result = cur.fetchone()[0]
    cur.close()
    print(f"Average GPA (Accepted, Spring 2025): {result:.2f}")
    return result
```

Rationale:

This query uses AVG() to calculate the average GPA from the returned data, where the database is filtered for applicants who applied for Spring 2025 (case-insensitive), were accepted (case-insensitive), and reported a GPA (**IS NOT NULL**). The **AND** logic ensures that only those who satisfy all the conditions are included.

---

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

Answer:

JHU Masters Computer Science applicants: 11

Query Used:

```
def count_jhu_cs_masters(conn):
    """Counts entries for JHU, masters, Computer Science.
    Original Question: How many entries are from applicants who applied to JHU for a masters degrees in Computer Science?"""
    cur = conn.cursor()
    cur.execute("""
        SELECT COUNT(*) FROM applicants
        WHERE (university ILIKE '%JHU%'
               OR university ILIKE '%Johns Hopkins%'
               OR university ILIKE '%John Hopkins%'
               OR university ILIKE '%John Hopkin%'
               OR university ILIKE '%Johns Hopkin%')
        AND (degree ILIKE '%Master%' OR degree ILIKE '%MS%' OR degree ILIKE '%Masters%')
        AND program ILIKE '%Computer Science%'
    """)
    result = cur.fetchone()[0]
    cur.close()
    print(f"JHU Masters Computer Science applicants: {result}")
    return result
```

Rationale:

This query counts applicants whose university is JHU (matching either "JHU" or "Johns Hopkins", with other common misspelling variations for flexibility), who are applying for a master's degree (matching "Master", “Masters”, or "MS"), and whose program is Computer Science. Using ILIKE allows for case-insensitive matching and accommodates slight variations in data entry.

Question:
Having carried out this assignment, please write two paragraphs about the inherent limitations of carrying out analytics over anonymously submitted data items. Did the analytic responses surprise you? How does this different from standards? For example, the average GRE quantitative reasoning score was 157 for 2023-2023 and was nearly 165 for grad school entries submitted (see sample output). Why do you think that is? What might cause this to occur? Please place your essay into a file called limitations.pdf.

There are inherent limitations within anonymously submitted data, especially regarding accuracy, representativeness, and, as a result, bias. Precisely because the data relies on voluntary self-reporting, there is no way to verify the authenticity or accuracy of the information provided by each respondent. Some entries may be exaggerated, incomplete, fabricated, or even irrelevant, depending on how free responses are allowed. This introduces other problems, such as data-entry errors or inconsistent formatting. For example, while scraping the data, there was one entry that seemed suspicious in both validity and intent—the University listed was "42 US," with the comment: “i can't kll u without authorization, but i can make sure u get so fcked up it reminds u hard of all the times u raped some student.” While other metrics and degrees were mentioned, the non-existent university and inappropriate comment make this entry untrustworthy. Anonymity also prevents the detection of duplicate or fraudulent submissions, as there is no unique identifier or way to trace submissions back to a single individual.

As a result, anonymously submitted data samples may not be representative of the whole population. In the case with Grad Café, the submissions may not accurately reflect the overall applicant pool. Those who choose to submit their results may systematically differ from those who do not, potentially introducing self-selection bias and leading to surprising analytic results. For example, as mentioned in the question, the calculated average GRE scores in past datasets were much higher than official national averages—an average GRE Quantitative score of nearly 165 in the submissions, compared to the national average of about 157 in 2023. This discrepancy likely arises from several factors: students who score highly may be more motivated to publicly share their results, while those with lower scores may choose not to submit or may underreport. There is also a possibility of selective reporting, where only “successful” applicants (those with acceptances or strong profiles) submit their data. As a result, analytics from such datasets can differ significantly from official statistics, and the findings should be interpreted with caution, keeping in mind the non-random, self-selected nature of the sample and the lack of data validation.