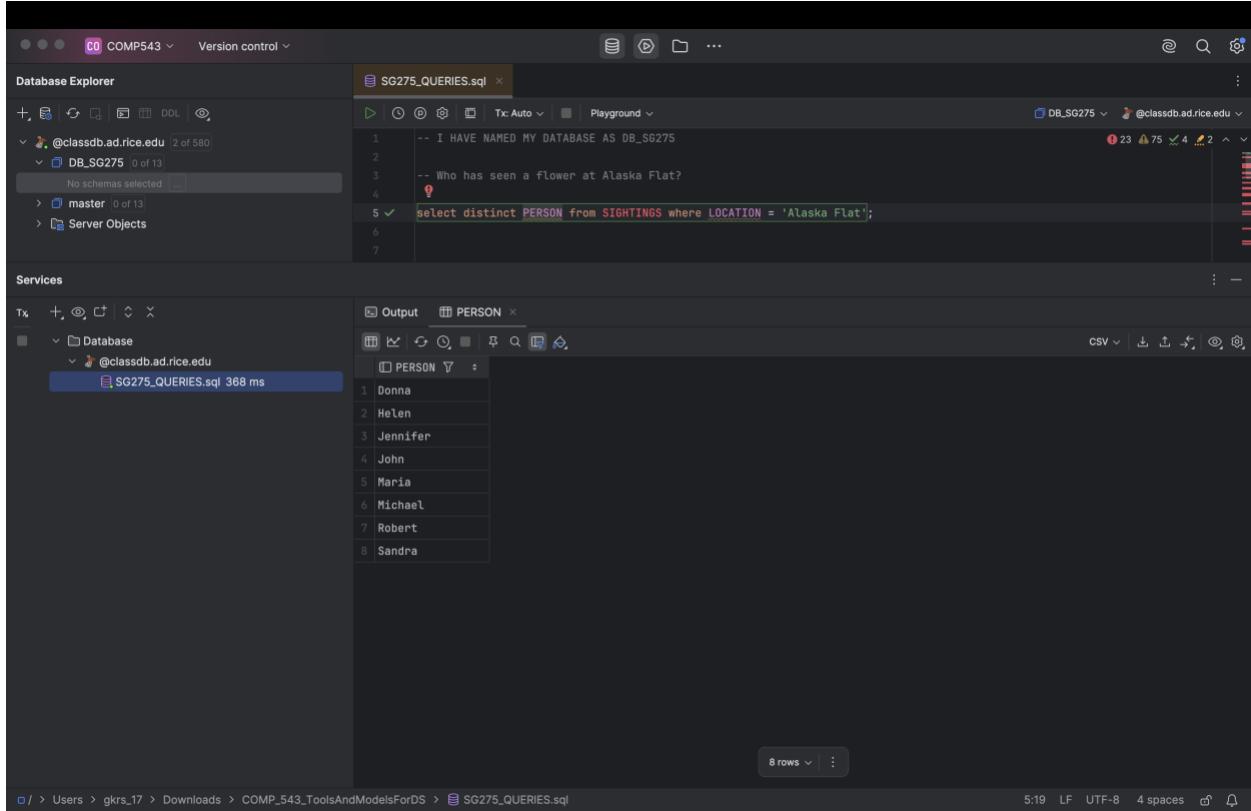


```
-- Who has seen a flower at Alaska Flat?  
  
SELECT DISTINCT PERSON FROM SIGHTINGS WHERE LOCATION = 'Alaska Flat';
```

Output



The screenshot shows a SQL development environment with the following interface elements:

- Database Explorer:** Shows the database structure. A connection to `@classdb.ad.rice.edu` is selected, displaying the `DB_SG275` database with 13 objects. The `master` database is also listed.
- Query Editor:** Contains the SQL query:

```
-- I HAVE NAMED MY DATABASE AS DB_SG275  
-- Who has seen a flower at Alaska Flat?  
SELECT DISTINCT PERSON FROM SIGHTINGS WHERE LOCATION = 'Alaska Flat';
```
- Results Grid:** Shows the output of the query, titled "PERSON". The results are:

PERSON
1 Donna
2 Helen
3 Jennifer
4 John
5 Maria
6 Michael
7 Robert
8 Sandra
- Status Bar:** At the bottom, it shows the file path `Users > gkrs_17 > Downloads > COMP_543.ToolsAndModelsForDS > SG275_QUERIES.sql`, the execution time `368 ms`, and the timestamp `5:19`.

```
-- Who has seen the same flower at both Moreland Mill and at Steve Spring?

SELECT DISTINCT s1.PERSON
FROM SIGHTINGS s1
JOIN SIGHTINGS s2
    ON s1.PERSON = s2.PERSON
    AND s1.NAME = s2.NAME
WHERE s1.LOCATION = 'Moreland Mill'
    AND s2.LOCATION = 'Steve Spring';
```

Output

The screenshot shows a SQL development environment with several panes:

- Database Explorer:** Shows the database connection @classdb.ad.rice.edu and the DB_SG275 schema.
- SQL Editor:** The query being run is highlighted in a code editor window titled SG275_QUERIES.sql.
- Services:** Shows a list of transactions and files, including sightings.sql, SG275.QUERIES.sql, createtables.sql, people.sql, and flowers.sql.
- Output:** A results pane showing the output of the query. It displays a single row with the value "Jennifer" under the column "PERSON".

```
-- What is the scientific name for each of the different flowers that have
--      been sighted by either Michael or Robert below 7250 feet in elevation?

SELECT DISTINCT f.GENUS, f.SPECIES
FROM SIGHTINGS s
JOIN FLOWERS f
    ON s.NAME = f.COMNAME
JOIN PEOPLE p
    ON s.PERSON = p.PERSON
JOIN FEATURES fe
    ON s.LOCATION = fe.LOCATION
WHERE p.PERSON IN ('Michael', 'Robert')
    AND fe.ELEV < 7250;
```

Output

The screenshot shows a database management interface with the following details:

- Top Bar:** COMP543, Version control.
- Database Explorer:** Shows a connection to @classdb.ad.rice.edu (DB_SG275) with 13 objects.
- Query Editor:** A tab labeled SG275.QUERIES.sql contains the provided SQL query.
- Output Window:** Shows the results of the query, which lists 14 flower species with their scientific names.

GENUS	SPECIES
Fremontodendron	californicum
Polemonium	californicum
Sphenocleadium	capitellatum
Penstemon	davidsonii
Triphysaria	eriantha
Arenaria	kingii
Triteleia	laxa
Castilleja	lineariloba
Gilia	mediomontana
Mimulus	primuloides
Viola	sheltonii
Asclepias	speciosa
Lomatium	torreyi
Zigadenus	venenosus

```
-- Which maps hold a location where someone has seen Alpine penstemon in June?

SELECT DISTINCT fe.MAP
FROM SIGHTINGS s
JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
WHERE s.NAME = 'Alpine penstemon'
    AND MONTH(s.SIGHTED) = 6;
```

Output

The screenshot shows a SQL development environment with several panes:

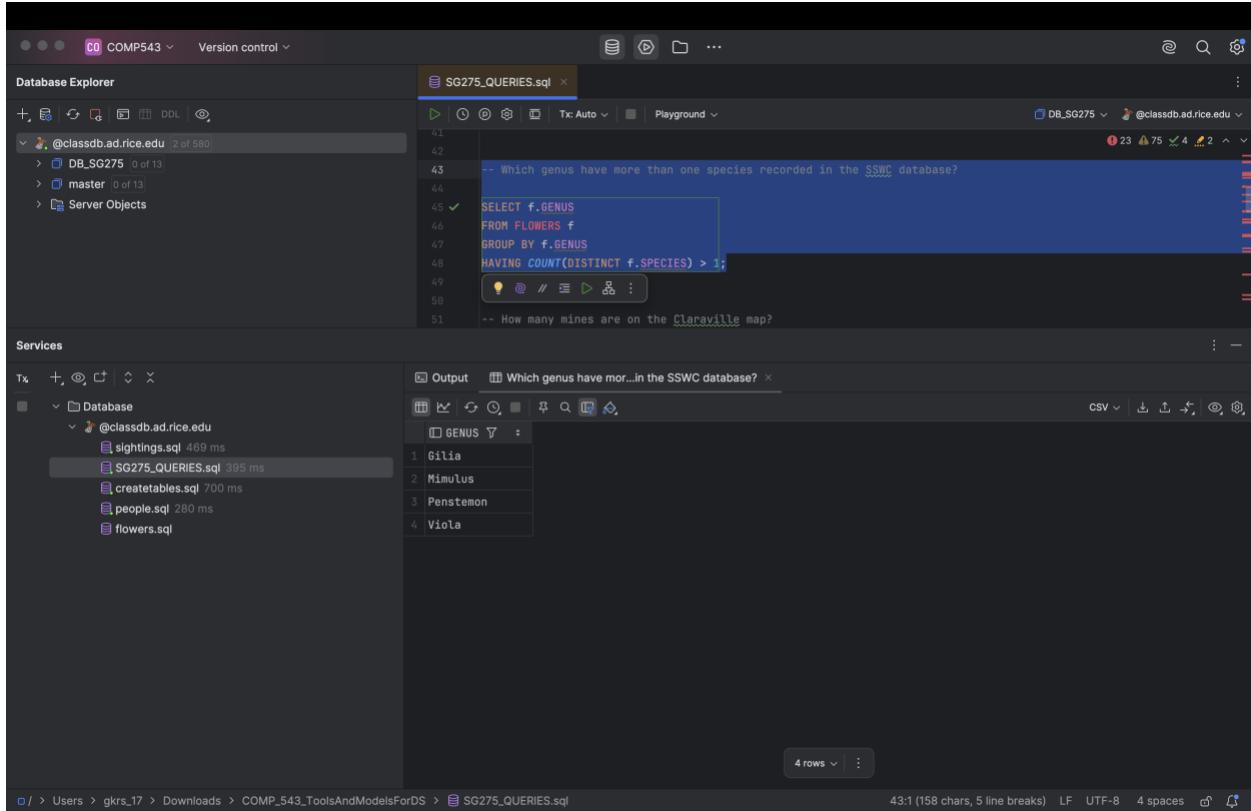
- Database Explorer:** Shows the connection to `@classdb.ad.rice.edu` and the database `DB_SG275`. It lists files like `sightings.sql`, `SG275.QUERIES.sql`, `createtables.sql`, `people.sql`, and `flowers.sql`.
- Editor:** The query `-- Which maps hold a location where someone has seen Alpine penstemon in June?` is written in the editor.
- Playground:** The query is executed, and the results are displayed in a table:

MAP
Sawmill Mountain

- Services:** Shows the transaction history with entries for `sightings.sql`, `SG275.QUERIES.sql`, `createtables.sql`, `people.sql`, and `flowers.sql`.
- Output:** A preview of the result table is shown.
- Bottom Status Bar:** Displays the file path `/Users/gkrs_17/Downloads/COMP_543_ToolsAndModelsForDS/SG275.QUERIES.sql`, character count `34:1 (226 chars, 6 line breaks)`, encoding `UTF-8`, and other metadata.

```
-- Which genus have more than one species recorded in the SSWC database?  
  
SELECT f.GENUS  
FROM FLOWERS f  
GROUP BY f.GENUS  
HAVING COUNT(DISTINCT f.SPECIES) > 1;
```

Output



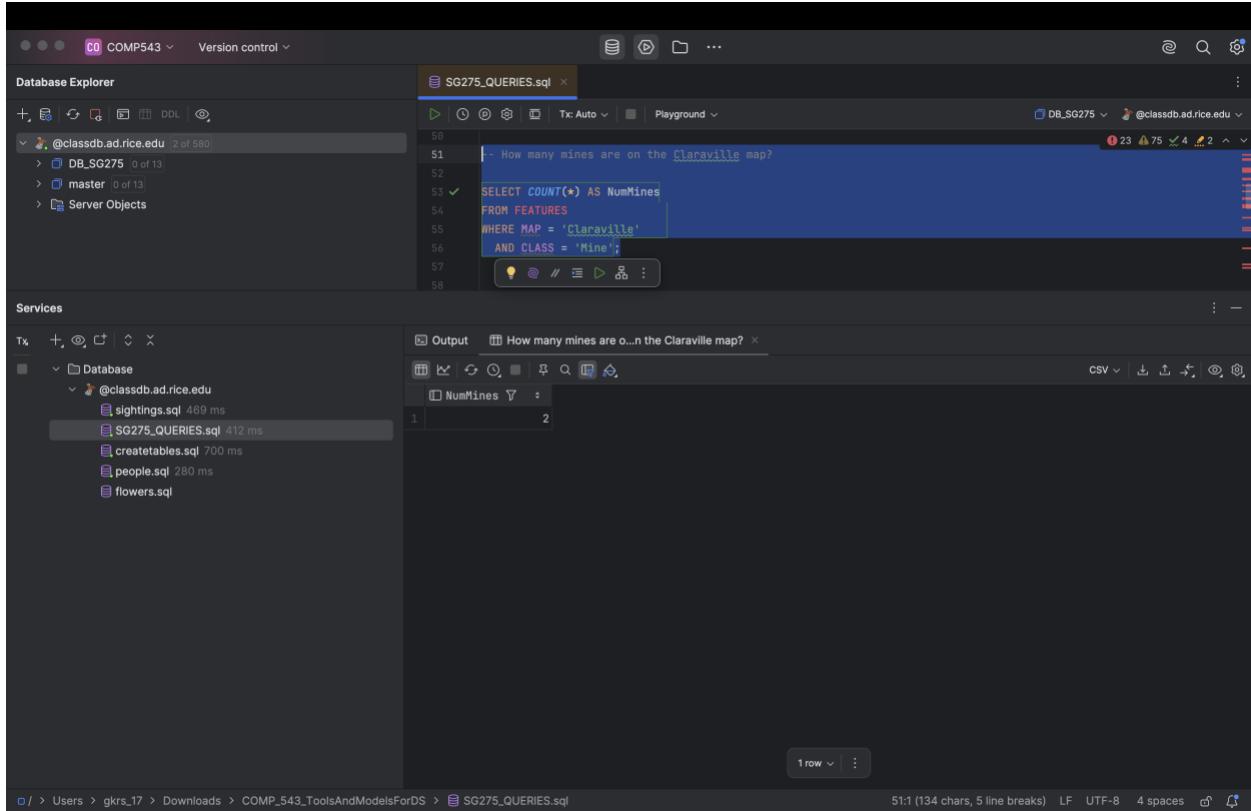
The screenshot shows a SQL development environment with the following interface elements:

- Top Bar:** COMP543, Version control.
- Database Explorer:** Shows the connection @classdb.ad.rice.edu, database DB_SG275, and tables master and Server Objects.
- Editor Area:** A code editor titled "SG275.QUERIES.sql" containing the provided SQL query. The output of the query is visible below the code.
- Output Area:** An "Output" tab titled "Which genus have mor...in the SSWC database?" displays the results of the query in a table format.
- Services Area:** Shows a list of recent files and their execution times: sightings.sql (469 ms), SG275.QUERIES.sql (395 ms), createtables.sql (700 ms), people.sql (280 ms), and flowers.sql.
- Bottom Status Bar:** Shows the file path /Users/gkrs_17/Downloads/COMP_543_ToolsAndModelsForDS/SC275_QUERIES.sql, character count (43:1 (158 chars, 5 line breaks)), and encoding (LF UTF-8 4 spaces).

GENUS
Gilia
Mimulus
Penstemon
Viola

```
-- How many mines are on the Claraville map?  
  
SELECT COUNT(*) AS NumMines  
FROM FEATURES  
WHERE MAP = 'Claraville'  
AND CLASS = 'Mine';
```

Output

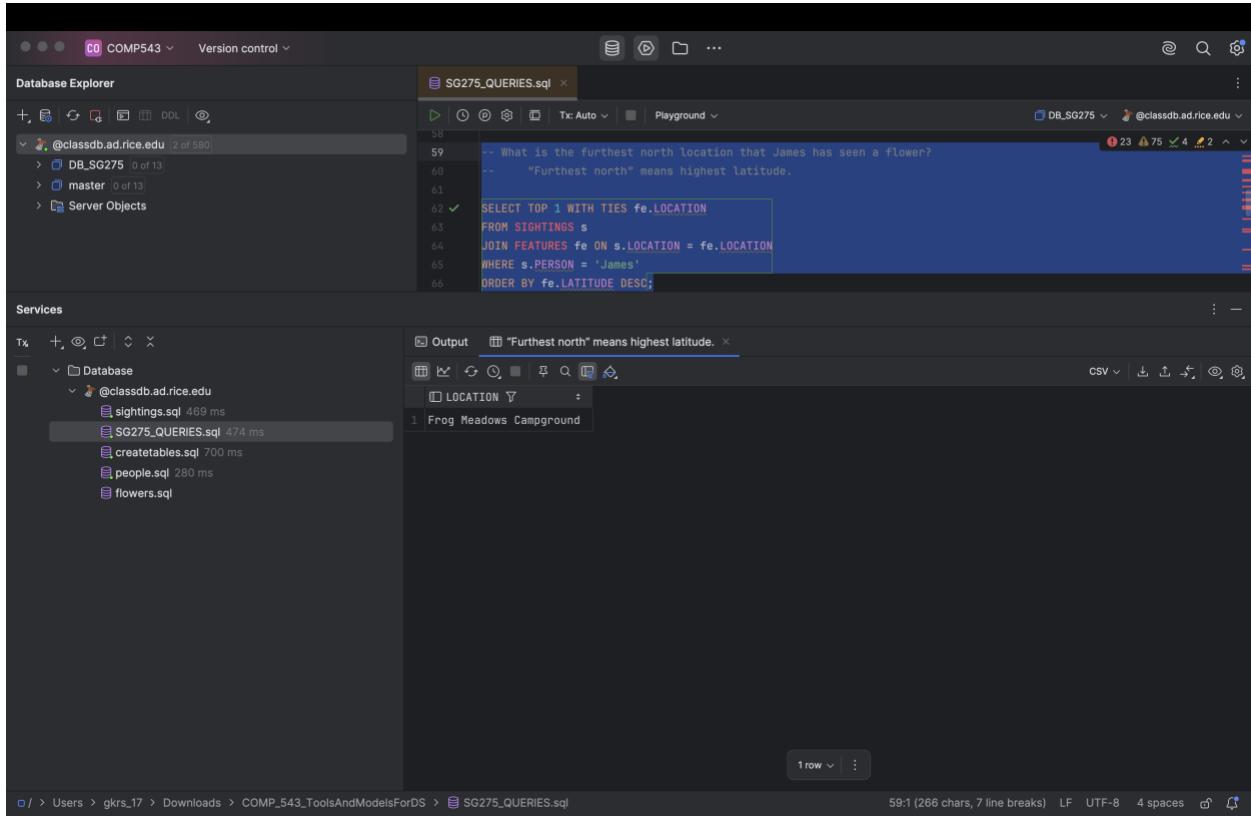


The screenshot shows a database management interface with the following details:

- Top Bar:** COMP543, Version control.
- Database Explorer:** Shows the connection @classdb.ad.rice.edu, the database DB_SG275 (0 of 13), and the master (0 of 13) and Server Objects nodes.
- Query Editor:** A tab labeled SG275.QUERIES.sql contains the SQL code from the previous block. The line "SELECT COUNT(*) AS NumMines" is highlighted.
- Output Window:** An Output tab displays the results of the query: "NumMines" with a value of 2.
- Services:** A sidebar on the left lists various SQL files: sightings.sql, SG275.QUERIES.sql, createtables.sql, people.sql, and flowers.sql.
- Bottom Status:** The status bar shows the path /Users/gkrs_17/Downloads/COMP_543_ToolsAndModelsForDS/SC275_QUERIES.sql, the line count 51:1 (134 chars, 5 line breaks), and encoding LF UTF-8 4 spaces.

```
-- What is the furthest north location that James has seen a flower?  
-- "Furthest north" means highest latitude.  
  
SELECT TOP 1 WITH TIES fe.LOCATION  
FROM SIGHTINGS s  
JOIN FEATURES fe ON s.LOCATION = fe.LOCATION  
WHERE s.PERSON = 'James'  
ORDER BY fe.LATITUDE DESC;
```

Output



The screenshot shows a SQL development environment with the following interface elements:

- Database Explorer:** Shows the database structure under the connection `@classdb.ad.rice.edu`, including the `DB_SG275` database and its `master` and `Server Objects`.
- Editor:** The query `SG275_QUERIES.sql` is open, containing the SQL code provided in the text block.
- Output Window:** The results of the query execution are displayed here. The output shows a single row:

LOCATION
Frog Meadows Campground
- Services:** A sidebar showing the execution history of various SQL files, with `SG275_QUERIES.sql` highlighted.
- Bottom Status Bar:** Displays the file path `/Users/gkrs_17/Downloads/COMP_543_ToolsAndModelsForDS/SG275_QUERIES.sql`, character count `59:1 (266 chars, 7 line breaks)`, encoding `LF - UTF-8`, and other metadata.

```
-- Who has not seen a flower at a location of class Spring?

SELECT p.PERSON
FROM PEOPLE p
WHERE NOT EXISTS (
    SELECT 1
    FROM SIGHTINGS s
    JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
    WHERE s.PERSON = p.PERSON
        AND fe.CLASS = 'Spring'
);
```

Output

The screenshot shows a database development environment with several panes:

- Database Explorer**: Shows the connection to `@classdb.ad.rice.edu` and the database `DB_SG275`.
- Services**: Shows the transaction history.
- SQL Editor**: The query `SG275_QUERIES.sql` is open, containing the SQL code provided above.
- Output**: The result of the query is displayed in a table:

	PERSON
1	Donna
2	John
3	Sandra
4	Robert

```

-- Who has seen flowers at the least distinct locations, and how many
distinct
--      flowers was that?

WITH per_person AS (
    SELECT p.PERSON,
           COUNT(DISTINCT s.LOCATION) AS DistinctLocations,
           COUNT(DISTINCT s.NAME)      AS DistinctFlowers
    FROM PEOPLE p
   LEFT JOIN SIGHTINGS s ON p.PERSON = s.PERSON
  GROUP BY p.PERSON
)
SELECT PERSON, DistinctLocations, DistinctFlowers
FROM per_person
WHERE DistinctLocations = (SELECT MIN(DistinctLocations) FROM per_person);

```

Output

The screenshot shows a database development environment with the following interface elements:

- Top Bar:** COMP543, Version control.
- Database Explorer:** Shows a connection to @classdb.ad.rice.edu with DB_SG275 selected. It lists tables like PEOPLE, SIGHTINGS, and others, along with stored procedures like sightings.sql, SG275.QUERIES.sql, createtables.sql, people.sql, and flowers.sql.
- Editor Area:** A code editor titled "SG275.QUERIES.sql" containing the provided SQL query. The code is syntax-highlighted with numbers on the left indicating line numbers.
- Output Area:** An "Output" tab titled "flowers was that?" displays the results of the query. The results table has columns PERSON, DistinctLocations, and DistinctFlowers. One row is shown for Brad, with values 3 for both columns.
- Bottom Status Bar:** Shows the file path /Users/gkrs_17/Downloads/COMP_543_ToolsAndModelsForDS/SG275.QUERIES.sql, character count (496), line count (14), and encoding (UTF-8).

PERSON	DistinctLocations	DistinctFlowers
Brad	3	3

```

-- For those people who have seen all of the flowers in the SSWC database,
-- what was the date at which they saw their last unseen flower?
-- In other words, at which date did they finish observing all of the
-- flowers in the database?

DECLARE @TotalFlowers INT = (SELECT COUNT(*) FROM FLOWERS);

WITH first_seen AS (
    SELECT s.PERSON, s.NAME, MIN(s.SIGHTED) AS FirstSeenDate
    FROM SIGHTINGS s
    GROUP BY s.PERSON, s.NAME
),
person_counts AS (
    SELECT PERSON, COUNT(*) AS FlowersSeenCount
    FROM first_seen
    GROUP BY PERSON
)
SELECT fs.PERSON,
       MAX(fs.FirstSeenDate) AS FinishedAllOn
FROM first_seen fs
JOIN person_counts pc ON fs.PERSON = pc.PERSON
WHERE pc.FlowersSeenCount = @TotalFlowers
GROUP BY fs.PERSON;

```

Output

The screenshot shows a SQL development environment with several panes:

- Database Explorer:** Shows the connection to `@classdb.ad.rice.edu` and the database `DB_SG275`.
- Editor:** The query `SG275_QUERIES.sql` is displayed, containing the SQL code provided above.
- Output:** The results of the query execution are shown in a table:

PERSON	FinishedAllOn
Maria	2006-09-23 00:00:00.000

```

-- For Tim, compute the fraction of his sightings on a per-month basis.
--     For example, we might get {(September, .12), (October, .74),
--     (November, .14)}. The fractions should add up to one across
-- all months.

WITH TimSightings AS (
    SELECT MONTH(SIGHTED) AS MonthNum,
           DATENAME(month, SIGHTED) AS MonthName
        FROM SIGHTINGS
       WHERE PERSON = 'Tim'
),
Total AS (
    SELECT COUNT(*) AS TotalCount FROM TimSightings
)
SELECT t.MonthName,
       COUNT(*) * 1.0 / tot.TotalCount AS Fraction
  FROM TimSightings t
CROSS JOIN Total tot
 GROUP BY t.MonthName, t.MonthNum, tot.TotalCount
 ORDER BY t.MonthNum;

```

Output

The screenshot shows a SQL development environment with the following interface elements:

- Database Explorer:** Shows the database connection @classdb.ad.rice.edu and the schema DB_SG275.
- Editor:** The query SG275_QUERIES.sql is open, containing the provided SQL code. The code is highlighted with syntax coloring.
- Output:** A table titled "all months." displays the results of the query. The table has two columns: MonthName and Fraction.

MonthName	Fraction
May	0.100000000000
June	0.500000000000
July	0.400000000000

At the bottom of the interface, there is a status bar showing the file path and some performance metrics.

```

-- Whose set of flower sightings is most similar to Michael's? Set
-- similarity is here defined in terms of the Jaccard Index, where JI (A,
B)
-- for two sets A and B is (size of the intersection of A and B) / (size
-- of the union of A and B). A larger Jaccard Index means more similar.

WITH MichaelFlowers AS (
    SELECT DISTINCT NAME
    FROM SIGHTINGS
    WHERE PERSON = 'Michael'
),
PersonFlowers AS (
    SELECT DISTINCT PERSON, NAME
    FROM SIGHTINGS
),
PersonCounts AS (
    SELECT PERSON, COUNT(*) AS PersonCount
    FROM PersonFlowers
    GROUP BY PERSON
),
MichaelCount AS (
    SELECT COUNT(*) AS MCount FROM MichaelFlowers
),
Intersections AS (
    SELECT pf.PERSON, COUNT(*) AS InterCount
    FROM PersonFlowers pf
    JOIN MichaelFlowers mf ON pf.NAME = mf.NAME
    GROUP BY pf.PERSON
)
SELECT TOP 1 WITH TIES
    pc.PERSON,
    CAST(ISNULL(ic.InterCount, 0) AS FLOAT) /
    (pc.PersonCount + mc.MCount - ISNULL(ic.InterCount, 0)) AS
JaccardIndex
FROM PersonCounts pc
LEFT JOIN Intersections ic ON pc.PERSON = ic.PERSON
CROSS JOIN MichaelCount mc
WHERE pc.PERSON <> 'Michael'
ORDER BY JaccardIndex DESC;

```

PERSON	JaccardIndex
Helen	0.5405405405406

Output