

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

## Output

The screenshot shows a database IDE interface. The top panel displays a SQL query in a file named `SG275_QUERIES.sql`. The query is:

```
-- 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';
```

The bottom panel shows the output of the query, which is a table with 8 rows. The table is titled `PERSON` and contains the following names:

PERSON
Donna
He'len
Jennifer
John
Maria
Michael
Robert
Sandra

The IDE also shows the Database Explorer on the left, indicating the connection to `@classdb.ad.rice.edu` and the database `DB_SG275`. The Services panel at the bottom left shows the execution of the query `SG275_QUERIES.sql` in 368 ms.

```
-- 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 database IDE interface with the following components:

- Database Explorer:** Shows the database structure for `@classdb.ad.rice.edu`, including `DB_SG275` (0 of 13), `master` (0 of 13), and `Server Objects`.
- SG275\_QUERIES.sql:** The active file containing the SQL query:
 

```
-- 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 -- same flower
WHERE s1.LOCATION = 'Moreland Mill'
   AND s2.LOCATION = 'Steve Spring';
```
- Services:** A list of database services with their execution times:
  - `sightings.sql`: 489 ms
  - `SG275_QUERIES.sql`: 461 ms
  - `createtables.sql`: 700 ms
  - `people.sql`: 280 ms
  - `flowers.sql`
- Output:** The result of the query is displayed in a table with the following structure:
 

PERSON
Jennifer
- Footer:** Shows the file path `/ > Users > gkrs_17 > Downloads > COMP_543_ToolsAndModelsForDS > SG275_QUERIES.sql` and statistics: `8:1 (280 chars, 8 line breaks) LF UTF-8 4 spaces`.

```
-- 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 IDE interface. The top panel displays the SQL query being executed. The bottom panel shows the output of the query, which is a table with two columns: GENUS and SPECIES. The output contains 14 rows of data.

**Database Explorer:** The left sidebar shows the database structure. The selected database is @classdb.ad.rice.edu. The selected schema is DB\_SG275. The selected table is SG275\_QUERIES.sql.

**SQL Query:** The query is as follows:

```
-- 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 output is a table with two columns: GENUS and SPECIES. The output contains 14 rows of data.

GENUS	SPECIES
1	Fremontodendron californicum
2	Polemonium californicum
3	Sphenosciadium capitellatum
4	Penstemon davidsonii
5	Triphysaria eriantha
6	Arenaria kingii
7	Triteleia laxa
8	Castilleja lineariloba
9	Gilia mediomontana
10	Mimulus primuloides
11	Viola sheltonii
12	Asclepias speciosa
13	Lomatium torreyi
14	Zigadenus venenosus

The bottom status bar shows the file path: /Users/gkrs\_17/Downloads/COMP\_543\_ToolsAndModelsForDS/SG275\_QUERIES.sql. The status bar also shows the file size: 19:1 (398 chars, 12 line breaks) and the encoding: UTF-8.

```
-- 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 database IDE interface with the following components:

- Database Explorer:** Shows a connection to `@classdb.ad.rice.edu` with a database `DB_SG275` containing tables `master` and `Server Objects`.
- SG275\_QUERIES.sql:** The SQL query is displayed in the editor, with line numbers 32 through 41. The query is:
 

```
-- 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;
```
- Services:** A list of database services is shown, including `sightings.sql` (469 ms), `SG275_QUERIES.sql` (397 ms), `createtables.sql` (700 ms), `people.sql` (280 ms), and `flowers.sql`.
- Output:** The query result is displayed in a table with one row:
 

MAP
Sawmill Mountain
- Status Bar:** Shows the file path `Users > gkrs_17 > Downloads > COMP_543_ToolsAndModelsForDS > SG275_QUERIES.sql` and the file size `34:1 (226 chars, 6 line breaks)` in `UTF-8` encoding with `4 spaces`.

```
-- 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 database IDE interface. On the left, the 'Database Explorer' pane shows a connection to '@classdb.ad.rice.edu' with a tree view containing 'DB\_SG275', 'master', and 'Server Objects'. Below this, the 'Services' pane shows a list of files: 'sightings.sql' (469 ms), 'SG275\_QUERIES.sql' (395 ms), 'createtables.sql' (700 ms), 'people.sql' (280 ms), and 'flowers.sql'. The main editor pane displays a SQL query in a file named 'SG275\_QUERIES.sql'. The query is:   
-- 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;   
Below the query, there is a comment: -- How many mines are on the Claraville map?   
The 'Output' pane at the bottom right shows the results of the query, titled 'Which genus have mor...in the SSWC database?'. It displays a table with one column, 'GENUS', and four rows of data: 1. Gilia, 2. Mimulus, 3. Penstemon, and 4. Viola. The status bar at the bottom indicates '4 rows' and '43:1 (158 chars, 5 line breaks)'. The bottom of the IDE shows the file path: 'Users > gkrs\_17 > Downloads > COMP\_543\_ToolsAndModelsForDS > SG275\_QUERIES.sql'.

```
-- 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;

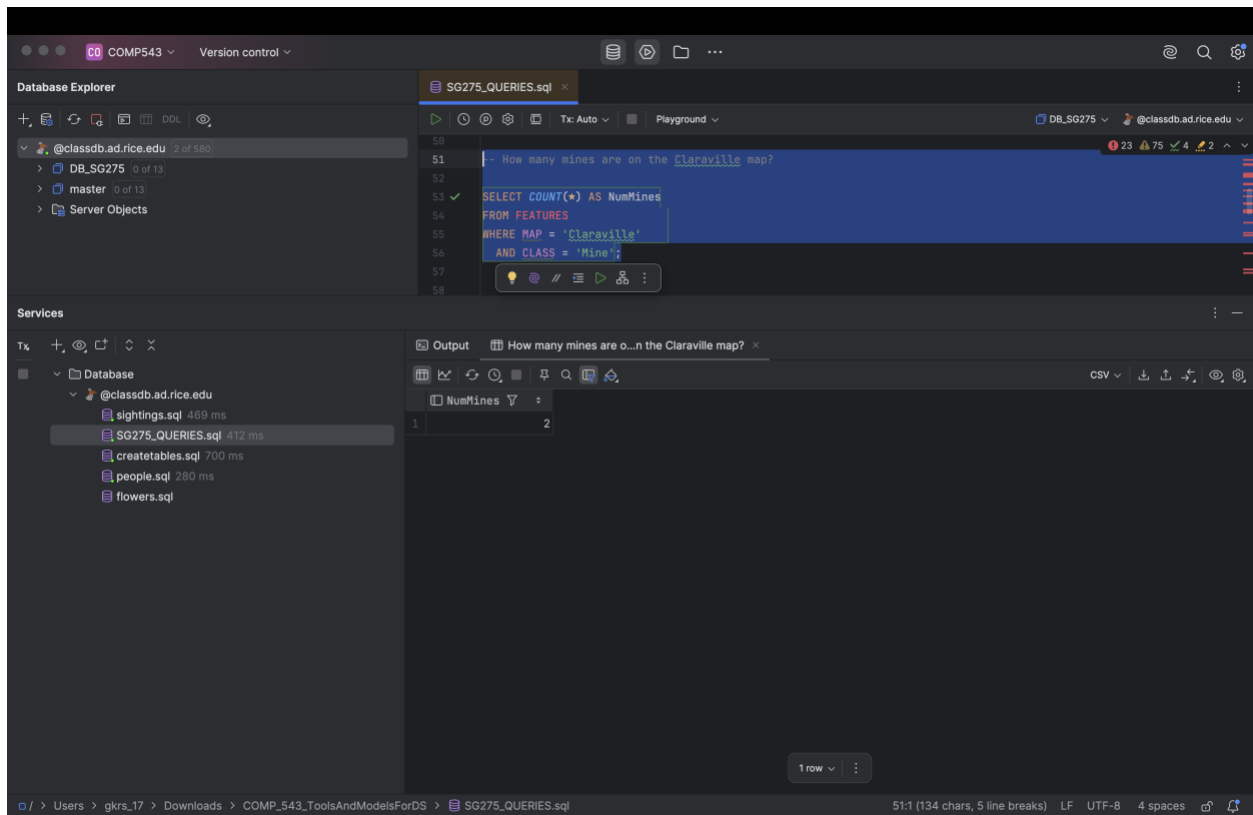
-- How many mines are on the Claraville map?
```

GENUS
1. Gilia
2. Mimulus
3. Penstemon
4. 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 IDE interface with a dark theme. The top panel displays a SQL query in a file named `SG275_QUERIES.sql`. The query is as follows:

```
-- How many mines are on the Claraville map?

SELECT COUNT(*) AS NumMines
FROM FEATURES
WHERE MAP = 'Claraville'
      AND CLASS = 'Mine';
```

The bottom panel shows the output of the query. It displays a table with one column, `NumMines`, and one row with the value `2`.

NumMines
2

The interface also includes a `Database Explorer` on the left, showing a connection to `@classdb.ad.rice.edu` with a database `DB_SG275`. The `Services` panel on the left lists several SQL files, including `SG275_QUERIES.sql`, which is currently selected. The status bar at the bottom indicates the file path and the query's length: `51:1 (134 chars, 5 line breaks) 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 IDE interface with the following components:

- Database Explorer:** Shows a tree view of the database structure. The selected database is `@classdb.ad.rice.edu`. Under it, there are two databases: `DB_SG275` (0 of 13) and `master` (0 of 13). Under `DB_SG275`, there are two server objects: `sightings.sql` (469 ms) and `SG275_QUERIES.sql` (474 ms).
- SG275\_QUERIES.sql:** The main editor window showing the SQL query. The query is: 

```
-- 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;
```
- Services:** A section on the left showing a list of services. The selected service is `LOCATION`. The output of the query is displayed in the `Output` pane.
- Output:** The output pane shows the result of the query. The output is: 

```
1 | Frog Meadows Campground
```

The status bar at the bottom indicates the file path: `Users > gkrs_17 > Downloads > COMP_543_ToolsAndModelsForDS > SG275_QUERIES.sql`. The status bar also shows the file size: `59:1 (266 chars, 7 line breaks)`, the encoding: `UTF-8`, and the line endings: `LF`.

```
-- 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 IDE interface. The top panel displays a SQL query in a file named `SG275_QUERIES.sql`. The query is designed to find people who have not seen a flower at a location of class 'Spring'. The query text is as follows:

```
-- 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'
);
```

The bottom panel shows the output of the query. The title bar indicates the query name: "Who has not seen a flower at a location of class Spring?". The output is displayed in a table with the following data:

PERSON
1 Donna
2 John
3 Sandra
4 Robert

The status bar at the bottom indicates that 4 rows were returned. The file path at the bottom of the IDE is `D:\Users\gkrs_17\Downloads\COMP_543_ToolsAndModelsForDS\SG275_QUERIES.sql`.



```

-- 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 IDE interface. The top panel displays the SQL query being executed. The bottom panel shows the output of the query, which is a table with three columns: PERSON, DistinctLocations, and DistinctFlowers. The output shows one row for 'Brad' with 3 distinct locations and 3 distinct flowers.

**Database Explorer:** @classdb.ad.rice.edu, DB\_SG275, master, Server Objects.

**Services:** Database, @classdb.ad.rice.edu, sightings.sql (469 ms), SG275\_QUERIES.sql (402 ms), createtables.sql (700 ms), people.sql (280 ms), flowers.sql.

**Output:** flowers was that? x

PERSON	DistinctLocations	DistinctFlowers
1 Brad	3	3

1 row

82:1 (496 chars, 14 line breaks) LF UTF-8 4 spaces

```
-- 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 the SQL Server Enterprise Manager interface. The left pane displays the Database Explorer with the following structure:

- @classdb.ad.rice.edu (2 of 580)
  - DB\_SG275 (0 of 13)
    - Server Objects

The right pane shows the execution of the query from the file SG275\_QUERIES.sql. The query is as follows:

```
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;
```

The bottom pane shows the output of the query, which is a single row with the following data:

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

The status bar at the bottom indicates the file path: > Users > gkrs\_17 > Downloads > COMP\_543\_ToolsAndModelsForDS > SG275\_QUERIES.sql. The file size is 102:1 (748 chars, 22 line breaks). The encoding is UTF-8 with 4 spaces.

```

-- 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 database IDE interface. The top panel displays the SQL query being executed. The bottom panel shows the output of the query, which is a table with two columns: MonthName and Fraction. The output shows three rows of data for the months of May, June, and July.

**SQL Query:**

```

-- 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;

```

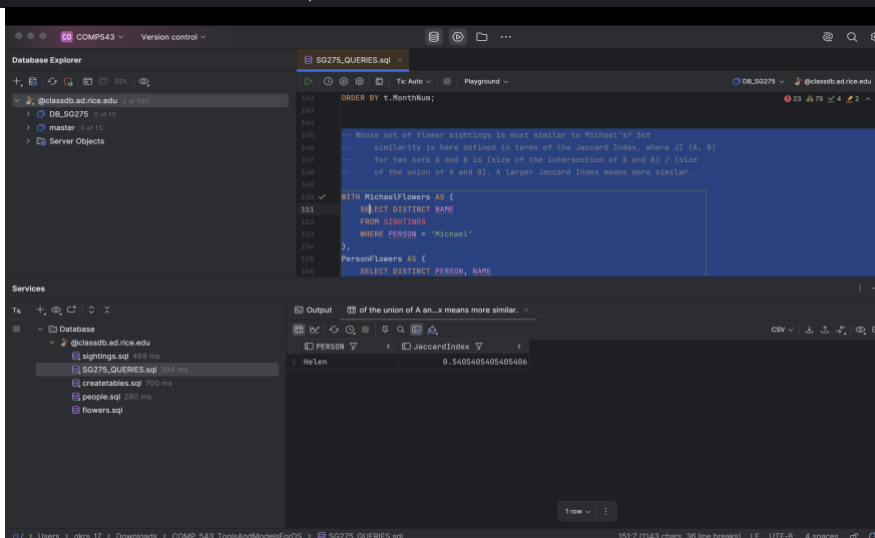
**Output:**

MonthName	Fraction
May	0.1000000000
June	0.5000000000
July	0.4000000000

The IDE also shows a Database Explorer on the left with a tree view of the database structure, including tables like sightings.sql, createtables.sql, people.sql, and flowers.sql. The bottom status bar indicates the file path and character encoding.

```
-- 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;
```



Output