

```
CREATE DATABASE ms198db
USE ms198db
```

-- Q1: Who has seen a flower at Alaska Flat?

```
SELECT DISTINCT s.person
FROM SIGHTINGS s
WHERE s.location = 'Alaska Flat';
```

	person
1	Donna
2	Helen
3	Jennifer
4	John
5	Maria
6	Michael
7	Robert
8	Sandra

--Q2: Who has seen the same flower at both Moreland Mill and at Steve Spring?

```
SELECT s.person
FROM SIGHTINGS s
WHERE s.location = 'Moreland Mill' AND EXISTS (
    SELECT * FROM SIGHTINGS s2
    WHERE s2.name = s.name AND s2.location = 'Steve Spring'
);
```

Output		person	person 2
1 row			
		person	
1		Jennifer	

--Q3: What is the scientific name for each of the different flowers that have been sighted by either Michael or Robert above 8250 feet in elevation?

```
SELECT DISTINCT f1.genus, f1.species
FROM flowers f1, sightings s, features f2
WHERE f1.comname = s.name
  AND f2.location = s.location
  AND (s.person = 'Michael' OR s.person = 'Robert')
  AND f2.elev > 8250
```

genus	species
Chaenactis	douglasii
Fremontodendron	californicum
Lilium	pardalinum
Polemonium	californicum
Streptanthus	diversifolius
Triteleia	laxa
Viola	quercetorum
Viola	sheltonii
Zigadenus	venenosus

--Q4: Which maps hold a location where someone has seen Alpine penstemon in August?

```
SELECT DISTINCT f.map
FROM FEATURES f JOIN SIGHTINGS s on f.location = s.location
WHERE MONTH(sighted) = 8 AND name = 'Alpine penstemon';
```

map
1 Claraville
2 Walker Pass

--Q5: Which genus have more than one species recorded in the SSWC database?

```
SELECT f.genus
FROM FLOWERS f
GROUP BY genus
HAVING COUNT(*) > 1;
```

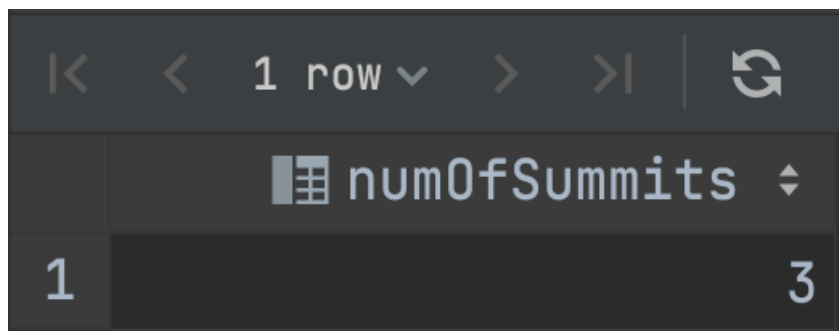


A screenshot of a database query result interface. At the top, there are navigation arrows and a dropdown menu showing '4 rows'. Below this is a table with a single column labeled 'genus'. The table contains four rows of data: '1 Gilia', '2 Mimulus', '3 Penstemon', and '4 Viola'.

	genus
1	Gilia
2	Mimulus
3	Penstemon
4	Viola

--Q6: How many summits are on the Sawmill Mountain map?

```
SELECT COUNT(*)
FROM FEATURES f
WHERE f.map = 'Sawmill Mountain' AND f.class='Summit'
GROUP BY map;
```



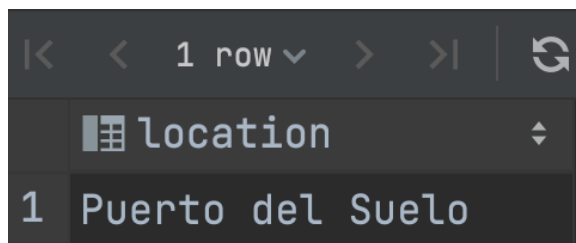
A screenshot of a database query result interface. At the top, there are navigation arrows and a dropdown menu showing '1 row'. Below this is a table with a single column labeled 'numOfSummits'. The table contains one row of data: '1 3'.

	numOfSummits
1	3

--Q7: What is the furthest south location that James has seen a flower?
"Furthest south" means lowest latitude.

```
CREATE VIEW AGG as
SELECT s.location, latitude
FROM SIGHTINGS s JOIN FEATURES f on s.location = f.location
WHERE s.person = 'James'
```

```
SELECT TOP (1) a.location
FROM AGG a
ORDER BY a.latitude ASC;
```



The screenshot shows a database query result interface. At the top, there are navigation controls: a left arrow, a right arrow, a dropdown menu showing '1 row', and a refresh icon. Below this, there is a table with one column labeled 'location' and one row containing the value 'Puerto del Suelo'.

	location
1	Puerto del Suelo

--Q8: Who has not seen a flower at a location of class Tower?

```
SELECT DISTINCT s.person
FROM SIGHTINGS s JOIN FEATURES f on s.location = f.location
WHERE NOT EXISTS(
    SELECT *
    FROM SIGHTINGS s2 JOIN FEATURES f2 on s2.location =
f2.location
    WHERE s.person = s2.person AND class = 'Tower'
);
```



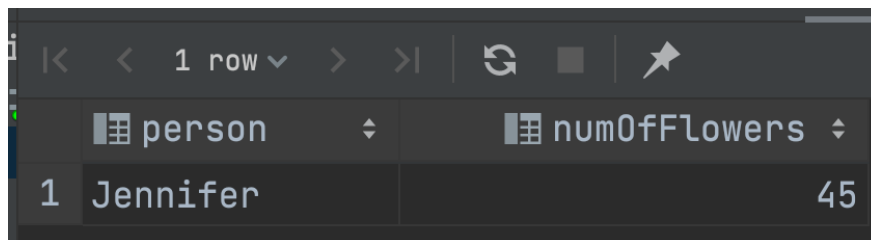
A screenshot of a database query result interface. At the top, there are navigation controls: a left arrow, a right arrow, and a dropdown menu showing '10 rows'. Below this is a table with a single column labeled 'person'. The table contains 10 rows of names, numbered 1 through 10 on the left. The names are: Brad, Donna, Helen, James, Jennifer, John, Pete, Robert, Sandra, and Tim.

	person
1	Brad
2	Donna
3	Helen
4	James
5	Jennifer
6	John
7	Pete
8	Robert
9	Sandra
10	Tim

--Q9: Who has seen flowers at the most distinct locations, and how many flowers was that?

```
CREATE VIEW AGG_9 AS
SELECT TOP(1) s.person
FROM SIGHTINGS s
GROUP BY s.person
ORDER BY COUNT(DISTINCT s.location) DESC
```

```
SELECT s.person, COUNT(DISTINCT s.name) AS numOfFlowers
FROM SIGHTINGS s, AGG_9 a_9
WHERE a_9.person = s.person
GROUP BY s.person
```



	person	numOfFlowers
1	Jennifer	45

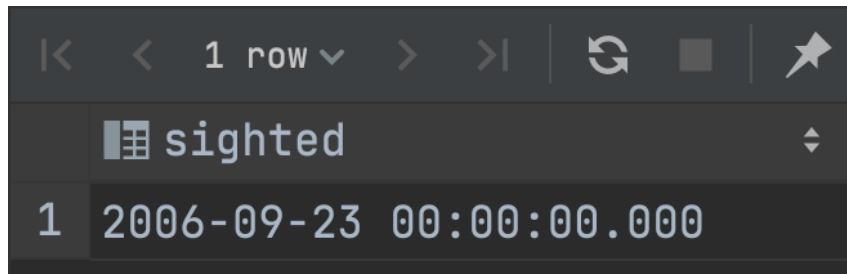
--Q10: For those people who have seen all of the flowers in the SSWC database, what was the date at which they saw their last unseen flower? In other words, at which date did they finish observing all of the flowers in the database

```

CREATE VIEW AGG_10 as
SELECT s.person, COUNT(DISTINCT s.name) AS discover_cnt
FROM SIGHTINGS s
GROUP BY s.person

SELECT TOP(1) s10.sighted
FROM SIGHTINGS s10 JOIN AGG_10 as a10 ON s10.person =
a10.person
WHERE discover_cnt = (SELECT COUNT(f.flow_id) FROM FLOWERS f)
ORDER BY s10.sighted DESC

```



The screenshot shows a database query result viewer with a dark theme. At the top, there is a navigation bar with icons for back, forward, and search, along with a dropdown menu showing '1 row'. Below the navigation bar, there is a table with one column labeled 'sighted'. The table contains one row with the value '2006-09-23 00:00:00.000'.

	sighted
1	2006-09-23 00:00:00.000

--Q11: For Jennifer, compute the fraction of her 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

```

CREATE VIEW AGG_11 as
SELECT DATENAME(month, s.sighted) AS inMonth, COUNT(*) AS
times
FROM SIGHTINGS s
WHERE s.person = 'Jennifer'
GROUP BY DATENAME(month, s.sighted)

SELECT all.inMonth, 1.0*all.times/SUM(all.times) OVER () AS
Percentage
FROM AGG_11 all

```

6 rows		
	inMonth	Percentage
1	April	0.015625000000
2	August	0.117187500000
3	July	0.218750000000
4	June	0.351562500000
5	May	0.242187500000
6	September	0.054687500000

--Q12: Whose set of flower sightings is most similar to John'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

```
CREATE VIEW NumIntersectsWithJohn AS
SELECT s2.person, COUNT(DISTINCT s2.name) AS intersectsNum
FROM SIGHTINGS s1, SIGHTINGS s2
WHERE s1.person = 'John' AND s1.name = s2.name
GROUP BY s2.person
```

```
CREATE VIEW BothNumAdded1 AS
SELECT s4.person, COUNT(DISTINCT s3.name) + COUNT(DISTINCT
s4.name) AS addedNum
FROM SIGHTINGS s3, SIGHTINGS s4
WHERE s3.person = 'John'
GROUP BY s4.person
```

```
CREATE VIEW NumUnionsWithJohn1 AS
SELECT n2.person, n2.addedNum-n1.intersectsNum AS unionNum
FROM NumIntersectsWithJohn n1, BothNumAdded1 n2
WHERE n1.person = n2.person
```

```
SELECT TOP(1) n4.person, 1.0*n3.intersectsNum/n4.unionNum AS
Jaccard_idx
FROM NumIntersectsWithJohn n3, NumUnionsWithJohn1 n4
WHERE n3.person = n4.person AND n3.person <> 'John'
ORDER BY Jaccard_idx DESC;
```

1 row			
	person		Jaccard_idx
1	James		0.434782608695

