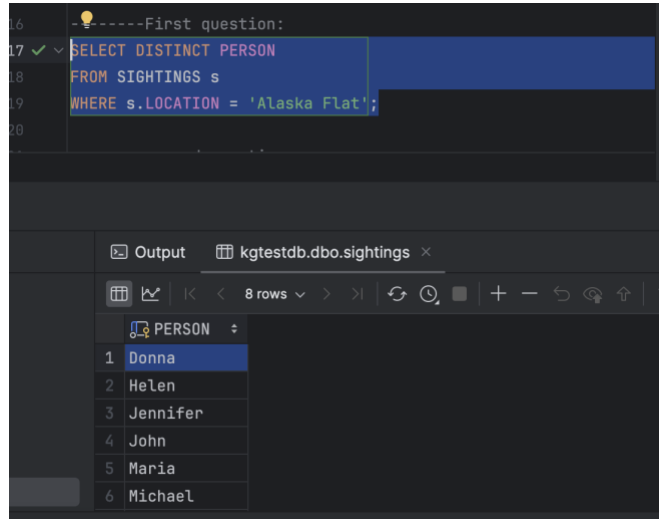


1. Who has seen a flower at Alaska Flat?

SQL Query:

```
SELECT DISTINCT PERSON
FROM SIGHTINGS s
WHERE s.LOCATION = 'Alaska Flat';
```

Executed Picture:



-----First question:

```
SELECT DISTINCT PERSON
FROM SIGHTINGS s
WHERE s.LOCATION = 'Alaska Flat';
```

PERSON
1 Donna
2 Helen
3 Jennifer
4 John
5 Maria
6 Michael

Solution:

Donna
Helen
Jennifer
John
Maria
Michael
Robert
Sandra

2. Who has seen the same flower at both Moreland Mill and at Steve Spring?

SQL Query:

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

Executed Picture:

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

Output kgtestdb.dbo.sightings x

PERSON
1 Jennifer

Solution:

Jennifer

3. 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?

SQL Query:

```
SELECT DISTINCT f.GENUS, f.SPECIES
FROM FLOWERS f
JOIN SIGHTINGS s ON f.COMNAME = s.NAME
JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
WHERE (s.PERSON = 'Michael' OR s.PERSON = 'Robert')
AND fe.ELEV > 8250;
```

Or

```
SELECT DISTINCT f.GENUS, f.SPECIES
FROM FLOWERS f
JOIN SIGHTINGS s ON f.COMNAME = s.NAME
JOIN PEOPLE p ON s.PERSON = p.PERSON
JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
WHERE (p.PERSON = 'Michael' OR p.PERSON = 'Robert')
AND fe.ELEV > 8250;
```

Executed Picture:

```
SELECT DISTINCT f.GENUS, f.SPECIES
FROM FLOWERS f
JOIN SIGHTINGS s ON f.COMNAME = s.NAME
JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
WHERE (s.PERSON = 'Michael' OR s.PERSON = 'Robert')
AND fe.ELEV > 8250;
```

Output kgtestdb.dbo.flowers

	GENUS	SPECIES
1	Chaenactis	douglasii
2	Fremontodendron	californicum
3	Lilium	pardalinum
4	Polemonium	californicum
5	Streptanthus	diversifolius
6	Triteleia	laxa

Solution:

Chaenactis,douglasii

Fremontodendron,californicum

Lilium,pardalinum

Polemonium,californicum

Streptanthus,diversifolius

Triteleia,laxa

Viola,quercetorum

Viola,sheltonii

Zigadenus,venenosus

4. Which maps hold a location where someone has seen Alpine penstemon in August?

SQL Query:

```
SELECT DISTINCT fe.MAP
FROM FEATURES fe
JOIN SIGHTINGS s ON fe.LOCATION = s.LOCATION
JOIN FLOWERS f ON s.NAME = f.COMNAME
WHERE f.COMNAME = 'Alpine penstemon' AND MONTH(s.SIGHTED) = 8;
```

Executed Picture:

```
--fourth question:
SELECT DISTINCT fe.MAP
FROM FEATURES fe
JOIN SIGHTINGS s ON fe.LOCATION = s.LOCATION
JOIN FLOWERS f ON s.NAME = f.COMNAME
WHERE f.COMNAME = 'Alpine penstemon' AND MONTH(s.SIGHTED) = 8;
```

Output kgtestdb.dbo.features x

MAP

1	Claraville
2	Walker Pass

Solution:

Claraville

Walker Pass

5. Which genus has more than one species recorded in the SSWC database?

SQL Query:

```
SELECT f.GENUS
FROM FLOWERS f
GROUP BY f.GENUS
HAVING COUNT(DISTINCT f.SPECIES) > 1;
```

Executed Picture:

```
SELECT f.GENUS
FROM FLOWERS f
GROUP BY f.GENUS
HAVING COUNT(DISTINCT f.SPECIES) > 1;
```

Output kgtestdb.dbo.flowers x

GENUS

1	Gilia
2	Mimulus
3	Penstemon
4	Viola

Solution:

Gilia

Mimulus

Penstemon

Viola

6. How many summits are on the Sawmill Mountain map?

SQL Query:

```
SELECT COUNT(*)
FROM FEATURES
WHERE CLASS = 'Summit' AND MAP = 'Sawmill Mountain';
```

Executed Picture:

The screenshot shows a SQL query execution interface. At the top, the query is entered: `SELECT COUNT(*) FROM FEATURES WHERE CLASS = 'Summit' AND MAP = 'Sawmill Mountain';`. Below the query, the results are displayed in a table with one row and one column, showing the count of 3. The interface includes a toolbar with various icons for editing and viewing the results.

Output
3

Solution:

3

7. What is the furthest south location where James has seen a flower? "Furthest south" means lowest latitude.

SQL Query:

```
SELECT TOP 1 s.LOCATION, fe.LATITUDE
FROM SIGHTINGS s
JOIN PEOPLE p ON s.PERSON = p.PERSON
JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
WHERE p.PERSON = 'James'
ORDER BY fe.LATITUDE asc;
```

Executed Picture:

```
62 -----seventh question:
63 ✓ SELECT TOP 1 s.LOCATION, fe.LATITUDE
64 FROM SIGHTINGS s
65 JOIN PEOPLE p ON s.PERSON = p.PERSON
66 JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
67 WHERE p.PERSON = 'James'
68 ORDER BY fe.LATITUDE asc;
```

Output Result 18 x

	LOCATION	LATITUDE
1	Puerto del Suelo	344937

Solution:

Puerto del Suelo,344937

8. Who has not seen a flower at a location of class Tower?

SQL Query:

```
SELECT DISTINCT p.PERSON
FROM PEOPLE p
WHERE p.PERSON NOT IN (
    SELECT DISTINCT s.PERSON
    FROM SIGHTINGS s
    JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
    WHERE fe.CLASS = 'Tower'
);
```

Executed Picture:

```

71 -----eighth question:
72 ✓ SELECT DISTINCT p.PERSON
73 FROM PEOPLE p
74 WHERE p.PERSON NOT IN (
75     SELECT DISTINCT s.PERSON
76     FROM SIGHTINGS s
77     JOIN FEATURES fe ON s.LOCATION = fe.LOCATION
78     WHERE fe.CLASS = 'Tower'
79 );

```

Output kgtestdb.dbo.people x

10 rows

	PERSON
1	Brad
2	Donna
3	Helen
4	James
5	Jennifer
6	John

Solution:

Brad
Donna
Helen
James
Jennifer
John
Pete
Robert
Sandra
Tim

9. Who has seen flowers at the most distinct locations, and how many flowers were that?

SQL Query:

```

WITH FlowerLocations AS (
    SELECT
        s.PERSON,
        COUNT(DISTINCT s.LOCATION) AS DistinctLocations,
        COUNT(DISTINCT s.NAME) AS DistinctFlowers
    FROM
        SIGHTINGS s
    GROUP BY
        s.PERSON
)

SELECT TOP 1
    PERSON,
    DistinctLocations,

```

```

DistinctFlowers
FROM
  FlowerLocations
ORDER BY
  DistinctLocations DESC, DistinctFlowers DESC;

```

Executed Picture:

The screenshot shows a SQL IDE with a query editor and a results pane. The query in the editor is as follows:

```

88 -----
89 WITH FlowerLocations AS (
90     SELECT
91         s.PERSON,
92         COUNT(DISTINCT s.LOCATION) AS DistinctLocations,
93         COUNT(DISTINCT s.NAME) AS DistinctFlowers
94     FROM
95         SIGHTINGS s
96     GROUP BY
97         s.PERSON
98 )
99
100 SELECT TOP 1
101     PERSON,
102     DistinctLocations,
103     DistinctFlowers
104 FROM
105     FlowerLocations
106 ORDER BY
107     DistinctLocations DESC, DistinctFlowers DESC;
108

```

The results pane shows a table with 30 columns. The first three columns are PERSON, DistinctLocations, and DistinctFlowers. The results are as follows:

	PERSON	DistinctLocations	DistinctFlowers
1	Jennifer	40	45

Solution:

(I even provided location along with no. of flowers)

Jennifer,40,45

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

SQL Query:

```

WITH FlowerCounts AS (
    SELECT
        s.PERSON,
        COUNT(DISTINCT s.NAME) AS TotalFlowerCount
    FROM
        SIGHTINGS s
    GROUP BY
        s.PERSON

```



```

)
SELECT
    s.PERSON,
    MAX(s.SIGHTED) AS LastUnseenFlowerDate
FROM
    SIGHTINGS s
JOIN
    FlowerCounts fc ON s.PERSON = fc.PERSON
WHERE
    fc.TotalFlowerCount = (SELECT COUNT(DISTINCT COMNAME) FROM FLOWERS)
GROUP BY
    s.PERSON;

```

Executed Picture:

-----10th question:

```

91 WITH FlowerCounts AS (
92     SELECT
93         s.PERSON,
94         COUNT(DISTINCT s.NAME) AS TotalFlowerCount
95     FROM
96         SIGHTINGS s
97     GROUP BY
98         s.PERSON
99 )
100 SELECT
101     s.PERSON,
102     MAX(s.SIGHTED) AS LastUnseenFlowerDate
103 FROM
104     SIGHTINGS s
105 JOIN
106     FlowerCounts fc ON s.PERSON = fc.PERSON
107 WHERE
108     fc.TotalFlowerCount = (SELECT COUNT(DISTINCT COMNAME) FROM FLOWERS)
109 GROUP BY
110     s.PERSON;
111

```

Output Result 24 x

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

Solution:

Maria,2006-09-23 00:00:00.000

- 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.

SQL Query:

```

WITH JenniferSightings AS (
    SELECT
        MONTH(s.SIGHTED) AS Month,
        DATENAME(MONTH, s.SIGHTED) AS MonthName,
        COUNT(*) AS SightingsCount
    FROM
        SIGHTINGS s
    WHERE
        s.PERSON = 'Jennifer'
    GROUP BY
        MONTH(s.SIGHTED), DATENAME(MONTH, s.SIGHTED)
)

```

```

)

SELECT
    Month,
    MonthName,
    CAST(SightingsCount AS DECIMAL) / SUM(CAST(SightingsCount AS DECIMAL))
OVER () AS Fraction
FROM
    JenniferSightings;

```

Executed Picture:

133 ✓ WITH JenniferSightings AS (1 2

134 SELECT

135 MONTH(s.SIGHTED) AS Month,

136 DATENAME(MONTH, s.SIGHTED) AS MonthName,

137 COUNT(*) AS SightingsCount

138 FROM

139 SIGHTINGS s

140 WHERE

141 s.PERSON = 'Jennifer'

142 GROUP BY

143 MONTH(s.SIGHTED), DATENAME(MONTH, s.SIGHTED)

144)

145 SELECT

146 Month,

147 MonthName,

148 CAST(SightingsCount AS DECIMAL) / SUM(CAST(SightingsCount AS DECIMAL)) OVER () AS Fraction

149 FROM

150 JenniferSightings;

JenniferSightings

Output Result 27 x

6 rows v

	Month	MonthName	Fraction
1	4	April	0.015625000000000000
2	8	August	0.117187500000000000
3	7	July	0.218750000000000000
4	6	June	0.351562500000000000
5	5	May	0.242187500000000000
6	9	September	0.054687500000000000

Solution:

(I even added Month for reference) -
 Month,MonthName,Fraction
 4,April,0.015625000000000000
 8,August,0.117187500000000000
 7,July,0.218750000000000000
 6,June,0.351562500000000000

5,May,0.24218750000000000000

9,September,0.05468750000000000000

12. Whose set of flower sightings is most similar to John's? The 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 similarities.

SQL Query:

```
CREATE VIEW JohnFlowers AS
SELECT DISTINCT NAME
FROM SIGHTINGS
WHERE PERSON = 'John';

CREATE VIEW IntersectionCounts AS
SELECT
    s.PERSON,
    COUNT(DISTINCT s.NAME) AS IntersectionCount
FROM
    SIGHTINGS s
JOIN
    JohnFlowers jf ON s.NAME = jf.NAME
WHERE
    s.PERSON <> 'John'
GROUP BY
    s.PERSON;

CREATE VIEW UnionCounts AS
SELECT
    s.PERSON,
    COUNT(DISTINCT s.NAME) + ISNULL(COUNT(DISTINCT jf.NAME), 0) AS UnionCount
FROM
    SIGHTINGS s
LEFT JOIN
    JohnFlowers jf ON s.NAME = jf.NAME
WHERE
    s.PERSON <> 'John'
GROUP BY
    s.PERSON;

CREATE VIEW JaccardIndex AS
SELECT
    ic.PERSON,
    ic.IntersectionCount,
    uc.UnionCount,
    CAST(ic.IntersectionCount AS DECIMAL) / NULLIF(uc.UnionCount, 0) AS
JaccardIndex
FROM
    IntersectionCounts ic
JOIN
    UnionCounts uc ON ic.PERSON = uc.PERSON;

SELECT TOP 1
```

```

PERSON,
IntersectionCount,
UnionCount,
JaccardIndex
FROM
    JaccardIndex
ORDER BY
    JaccardIndex DESC;

```

Executed Picture:

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL code:

```

333
334 ALTER VIEW JaccardIndex AS
335 SELECT
336     ic.PERSON,
337     ic.IntersectionCount,
338     uc.UnionCount,
339     CAST(ic.IntersectionCount AS DECIMAL) / NULLIF(uc.UnionCount, 0) AS JaccardIndex
340 FROM
341     IntersectionCounts ic
342 JOIN
343     UnionCounts uc ON ic.PERSON = uc.PERSON;
344
345 SELECT TOP 1
346     PERSON,
347     IntersectionCount,
348     UnionCount,
349     JaccardIndex
350 FROM
351     JaccardIndex
352 ORDER BY
353     JaccardIndex DESC;

```

The results pane at the bottom shows the output of the query, displaying a single row for the person 'Brad'.

	PERSON	IntersectionCount	UnionCount	JaccardIndex
1	Brad	3	6	0.500000000000

Solution:

Brad,3,6,0.500000000000