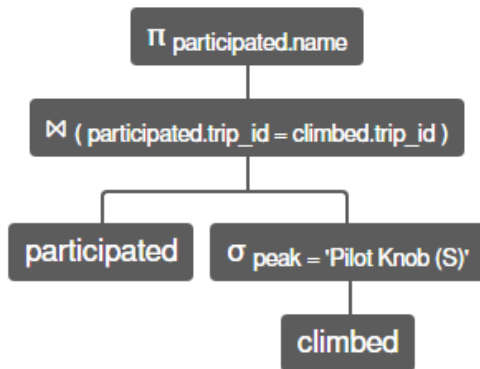


CIS4301 Assignment #3

Each problem spans about a page

1.

```
SELECT NAME
FROM PARTICIPATED
WHERE trip_id IN (SELECT trip_id
                  FROM CLIMBED
                  WHERE PEAK = 'Pilot Knob (S)');
```



$\pi_{\text{participated.name}} (\text{participated} \bowtie (\text{participated.trip_id} = \text{climbed.trip_id}) (\sigma_{\text{peak} = \text{'Pilot Knob (S)'}} (\text{climbed})))$

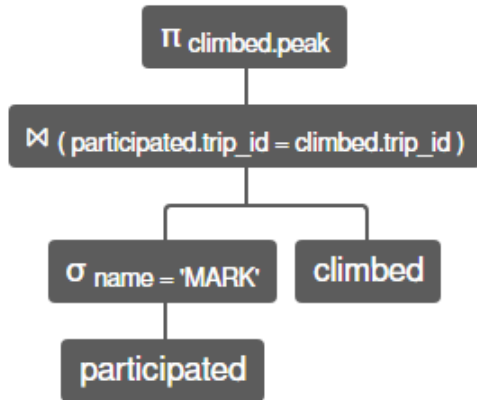
SQLite	
Table	
CLIMBED	
CLIMBER	
PARTICIPATED	
PEAK	
View	
PeakClimber	
PeakCount	
MariaDB	

```
1 SELECT NAME
2 FROM PARTICIPATED
3 WHERE trip_id IN (SELECT trip_id
4                   FROM CLIMBED
5                   WHERE PEAK = 'Pilot Knob (S)');
```

NAME
JOHN
MARK
MICHAEL

2.

```
SELECT DISTINCT peak
FROM   climbed,
       participated
WHERE  climbed.trip_id = participated.trip_id
AND    NAME = 'MARK';
```



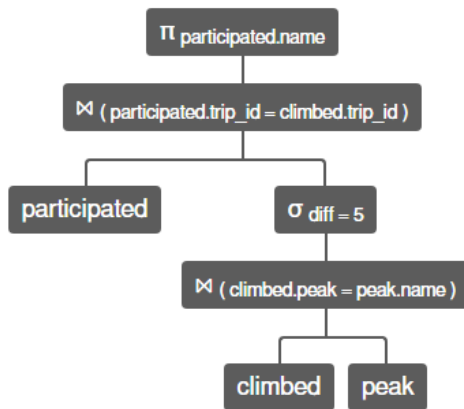
$\pi_{\text{climbed.peak}} ((\sigma_{\text{name} = \text{'MARK'}}(\text{participated})) \bowtie_{\text{participated.trip_id} = \text{climbed.trip_id}} (\text{climbed}))$

The screenshot shows a SQL IDE interface. On the left is a sidebar with a tree view containing 'Table' (CLIMBED, CLIMBER, PARTICIPATED), 'View' (PeakClimber, PeakCount), and 'MenuDB'. The main area is split into two panes. The top pane shows the SQL query: `SELECT DISTINCT peak FROM climbed, participated WHERE climbed.trip_id = participated.trip_id AND NAME = 'MARK';`. The bottom pane shows the results of the query, which is a list of peak names: Center Peak, North Maggie Mountain, Vihaback, Kearsarge Peak, Lion Rock, Midway Mountain, Mount Hale, Mount Langley, Pilot Knob (S), Dragon Peak, Mount Barnard, Mount Guyot, Mount Newcomb, South Osgood, Thor Peak, Jengara Mountain, Florence Peak, Joe Devil Peak, Mount McAfee, Mount Ricketts, Muah Mountain, Olancha Peak, Moses Mountain, Mount Williamson, and Heedham Mountain.

PEAK
Center Peak
North Maggie Mountain
Vihaback
Kearsarge Peak
Lion Rock
Midway Mountain
Mount Hale
Mount Langley
Pilot Knob (S)
Dragon Peak
Mount Barnard
Mount Guyot
Mount Newcomb
South Osgood
Thor Peak
Jengara Mountain
Florence Peak
Joe Devil Peak
Mount McAfee
Mount Ricketts
Muah Mountain
Olancha Peak
Moses Mountain
Mount Williamson
Heedham Mountain

3.

```
SELECT NAME
FROM PARTICIPATED
WHERE trip_id IN (SELECT trip_id
                  FROM CLIMBED,
                  PEAK
                  WHERE CLIMBED.peak = PEAK.name
                  AND PEAK.diff = 5);
```



π participated.name (participated \bowtie (participated.trip_id = climbed.trip_id) σ diff = 5 (climbed \bowtie (climbed.peak = peak.name) peak)))

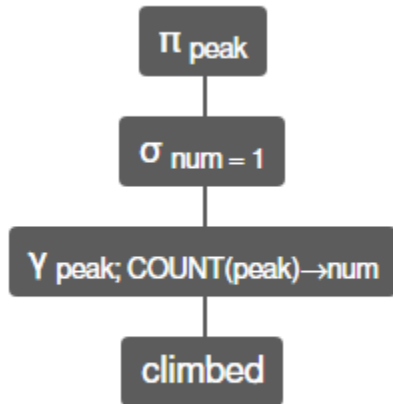
SQLite	
Table	
CLIMBED	
CLIMBER	
PARTICIPATED	
PEAK	
View	
PeakClimber	
PeakCount	
MarlaDB	
PostgreSQL	

SQLite	
1 SELECT NAME	
2 FROM PARTICIPATED	
3 WHERE trip_id IN (SELECT trip_id	
4 FROM CLIMBED,	
5 PEAK	
6 WHERE CLIMBED.peak = PEAK.name	
7 AND PEAK.diff = 5);	

NAME
JOHN
ELIZABETH
DONNA

4.

```
SELECT peak
FROM (SELECT peak,
      Count (peak) AS NUM
      FROM climbed
      GROUP BY peak
      HAVING num = 1);
```



$\pi_{\text{peak}} (\sigma_{\text{num} = 1} (\gamma_{\text{peak}; \text{COUNT}(\text{peak}) \rightarrow \text{num}} (\text{climbed})))$

File Link Run Export Import Sign in

SQLite

Table

- CLIMBED
- CLIMBER
- PARTICIPATED
- PEAK

View

- PeakClimber
- PeakCount

SQLite

```
1 SELECT peak
2 FROM (SELECT peak,
3      COUNT (peak) AS NUM
4      FROM climbed
5      GROUP BY peak
6      HAVING num = 1);
```

peak
Angora Mountain
Cartago Peak
Center Peak
Coyote Peaks
Crag Peak
Dragon Peak
Kern Peak
Lamont Peak
Lone Pine Peak
Mount Barnard
Mount Bradley
Mount Clarence King
Mount Gardiner
Mount Geneva
Mount LeConte
Mount Muir
Mount Pickering
Mount Stanford (S)
Mount Whitney
North Coast

Close AD

17 1

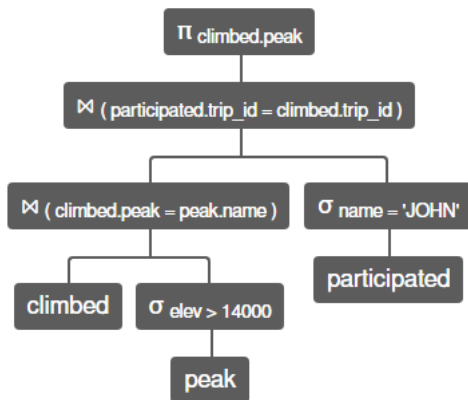
File	Link	Run	Export	Import	Sign in
SQLite					
Table					
CLIMBED					
CLIMBER					
PARTICIPATED					
PEAK					
View					
PeakClimber					
PeakCount					
MySQL					
PostgreSQL					
MS SQL					
Oracle					

5.

```

SELECT DISTINCT peak
FROM   climbed,
       participated,
       peak
WHERE  climbed.trip_id = participated.trip_id
       AND participated.NAME = 'JOHN'
       AND ( climbed.peak = peak.NAME
             AND peak.elev > '14000' );

```

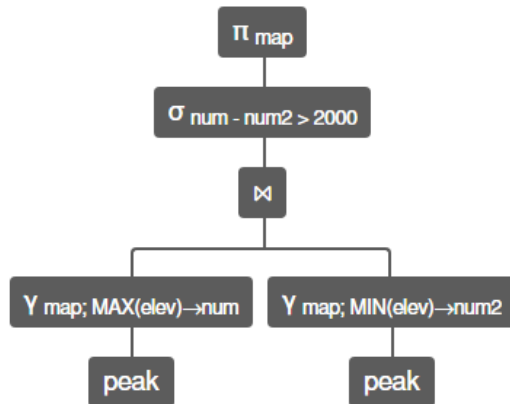


$\pi_{\text{climbed.peak}}((\text{climbed} \bowtie (\text{climbed.peak} = \text{peak.name}) (\sigma_{\text{elev} > 14000} \text{peak})) \bowtie (\text{participated.trip_id} = \text{climbed.trip_id}) (\sigma_{\text{name} = \text{'JOHN'}} (\text{participated})))$

File	Link	Run	Export	Import	Sign in
SQLite					
Table					
CLIMBED					
CLIMBER					
PARTICIPATED					
PEAK					
View					
PeakClimber					
PeakCount					
MySQL					
PostgreSQL					
MS SQL					
Oracle					

6.

```
SELECT map
FROM peak
GROUP BY map
HAVING ( Max(elev) - Min(elev) ) > 2000;
```

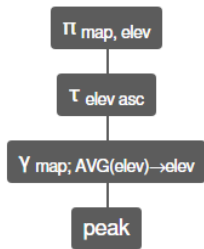


$\Pi_{\text{map}} (\sigma_{\text{num} - \text{num2} > 2000} ((\Upsilon_{\text{map}; \text{MAX}(\text{elev}) \rightarrow \text{num}} (\text{peak})) \bowtie (\Upsilon_{\text{map}; \text{MIN}(\text{elev}) \rightarrow \text{num2}} (\text{peak})))))$

MAP
Kearsarge Peak
Mount Whitney

7.

```
SELECT map,
       Avg (elev)
FROM   peak
GROUP BY map
ORDER BY Avg(elev);
```



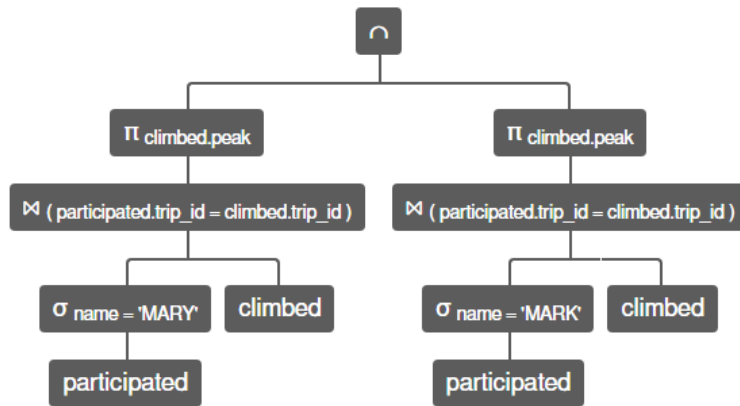
$\pi_{map, elev} (\tau_{elev \text{ asc}} Y_{map; AVG(elev) \rightarrow elev}(peak))$

SQLite	
Table	1: SELECT map,
CLIMBED	2: Avg (elev)
CLIMBER	3: FROM peak
PARTICIPATED	4: GROUP BY map
PEAK	5: ORDER BY Avg(elev);
MAP	
Onyx	6200
Lament Peak	7635
Ninemile Canyon	8000
Rockhouse Basin	8360
Owens Peak	8453
Keansarge Peak	8550.666666666666
Cannell Peak	8802
Silver City	9023
Crag Peak	9480
Monache Mtn	9533
Moses Mtn	9782.5
Sirretta Peak	9977
Kam Lake	10545
Bartlett	11016
Mt Silliman	11108
Lodgepole	11240
Olancha	11301.5
Kern Peak	11510
Sphinx Lakes	11717

SQLite	
Table	1: SELECT map,
CLIMBED	2: Avg (elev)
CLIMBER	3: FROM peak
PARTICIPATED	4: GROUP BY map
PEAK	5: ORDER BY Avg(elev);
MAP	
Sphinx Lakes	11717
Mineral King	12280.4
Triple Divide Peak	12657.375
Mt Clarence King	12838.375
Cirque Peak	12900
Mt Kaaveah	12945
Mt Brewer	13334.357142857143
Mount Whitney	13493.277777777777
Mt Langley	13561
Mt Williamson	13747.777777777777

8.

```
SELECT peak
FROM participated,
climbed
WHERE NAME = 'MARY'
AND climbed.trip_id = participated.trip_id
INTERSECT
SELECT peak
FROM participated,
climbed
WHERE NAME = 'MARK'
AND climbed.trip_id = participated.trip_id;
```



$\pi_{\text{climbed, peak}}((\sigma_{\text{name} = \text{'MARY'}}(\text{participated})) \bowtie (\text{participated.trip_id} = \text{climbed.trip_id})(\text{climbed})) \cap \pi_{\text{climbed, peak}}((\sigma_{\text{name} = \text{'MARK'}}(\text{participated})) \bowtie (\text{participated.trip_id} = \text{climbed.trip_id})(\text{climbed}))$

File

Link

Run

Export

Import

Sign in

SQLite

Table

CLIMBED

CLIMBER

PARTICIPATED

PEAK

MySQL

PostgreSQL

MS SQL

Oracle

Syntax

Close AD

```

1 SELECT peak
2 FROM participated,
3   climbed
4 WHERE NAME = 'MARY'
5   AND climbed.trip_id = participated.trip_id
6 INTERSECT
7 SELECT peak
8 FROM participated,
9   climbed
10 WHERE NAME = 'MARK'
11   AND climbed.trip_id = participated.trip_id;
```

PEAK

Dragon Peak

Joe Devel Peak

Kearsarge Peak

Lion Rock

Midway Mountain

Moses Mountain

Mount Bernard

Mount Guyot

Mount Hale

Mount Langley

Mount McAfee

Mount Newcomb

Mount Rixford

Mount Williamson

Mt. Shasta

Needham Mountain

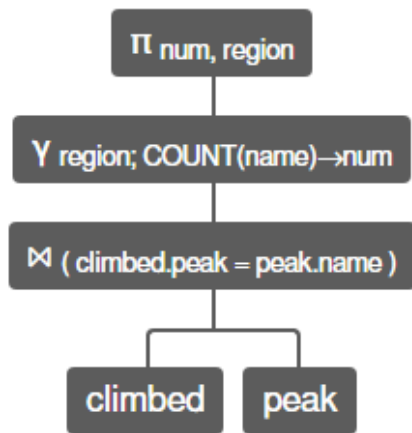
Olancho Peak

South Guard

Thor Peak

9.

```
SELECT Count (DISTINCT NAME),  
       region  
FROM   peak  
WHERE  NOT EXISTS (SELECT peak  
                   FROM   climbed  
                   WHERE  climbed.peak = peak.NAME)  
GROUP BY region  
ORDER BY region;
```

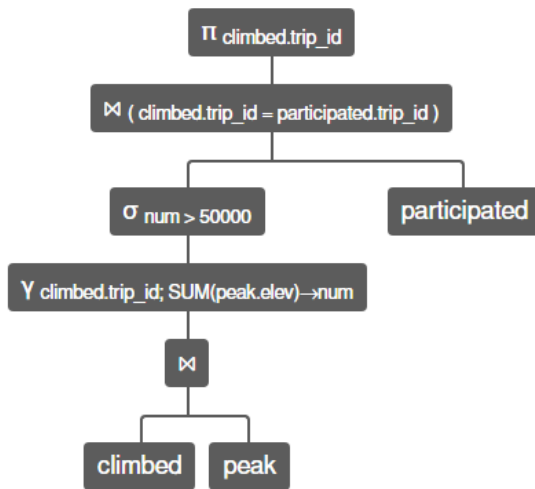


$\Pi \text{ num, region } (\sigma \text{ region; COUNT(name) } \rightarrow \text{num } (\text{climbed } \bowtie (\text{climbed.peak} = \text{peak.name}) (\text{peak})))$

File	Link	Run	Export	Import	Sign in	
SQLite						
Table						
CLIMBED						
CLIMBER						
PARTICIPATED						
PEAK						
MySQL						
PostgreSQL						
MS SQL						
Oracle						
Syntax						
Close AD						
1	SELECT COUNT (DISTINCT NAME),					
2	region					
3	FROM peak					
4	WHERE NOT EXISTS (SELECT peak					
5	FROM climbed					
6	WHERE climbed.peak = peak.NAME)					
7	GROUP BY region					
8	ORDER BY region;					
9						
10						
1	Count (DISTINCT NAME)					REGION
1						Corcoran to Whitney
3						Great Western Divide
9						Kearsarge and West
4						Kearsarge Pass
6						Kings Kern Divide
1						Mineral King
3						Olancho to Langley
3						Southern Sierra
5						Whitney to Williamson

10.

```
SELECT climbed.trip_id
FROM climbed
LEFT JOIN participated
    ON climbed.trip_id = participated.trip_id
LEFT JOIN peak
    ON climbed.peak = peak.NAME
GROUP BY climbed.trip_id
HAVING Sum (peak.elev) > 500000;
```

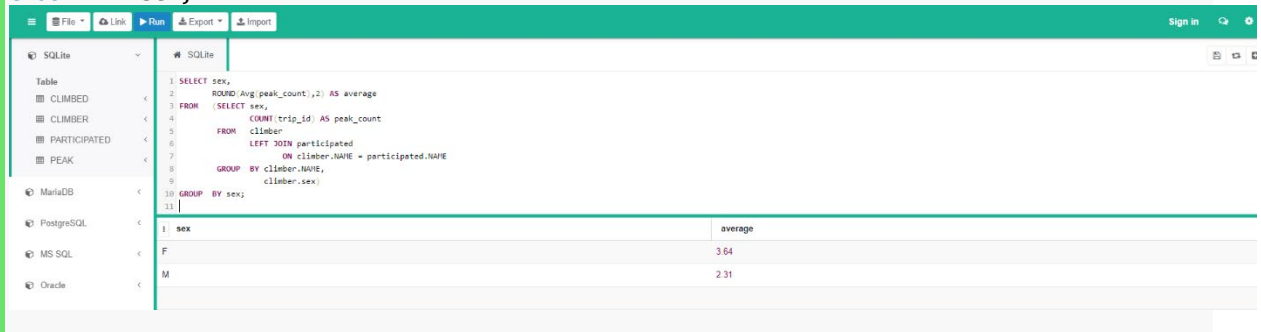


$\pi_{\text{climbed.trip_id}}((\sigma_{\text{num} > 50000}(\gamma_{\text{climbed.trip_id}; \text{SUM}(\text{peak.elev}) \rightarrow \text{num}}(\text{climbed} \bowtie \text{peak}))) \bowtie (\text{climbed.trip_id} = \text{participated.trip_id}) \text{ participated})$

The screenshot shows a SQL IDE interface. On the left, there is a table list with 'CLIMBED', 'CLIMBER', 'PARTICIPATED', and 'PEAK'. The main area displays the SQL query:
1 SELECT climbed.trip_id
2 FROM climbed
3 LEFT JOIN participated
4 ON climbed.trip_id = participated.trip_id
5 LEFT JOIN peak
6 ON climbed.peak = peak.NAME
7 GROUP BY climbed.trip_id
8 HAVING Sum (peak.elev) > 500000;
9
The results pane at the bottom shows a table with one column 'TRIP_ID' and three rows with values 8, 13, and 13.

11.

```
SELECT sex,  
       ROUND(Avg(peak_count),2) AS average  
FROM (SELECT sex,  
            COUNT(trip_id) AS peak_count  
      FROM climber  
      LEFT JOIN participated  
            ON climber.NAME = participated.NAME  
      GROUP BY climber.NAME,  
               climber.sex)  
GROUP BY sex;
```



The screenshot shows a database management interface. On the left, a sidebar lists tables: CLIMBED, CLIMBER, PARTICIPATED, and PEAK. The main area displays the SQL query from the previous block. Below the query editor, a results table is shown with the following data:

sex	average
F	3.64
M	2.31

12.

```
CREATE VIEW peakclimber
AS
  SELECT DISTINCT NAME,
                    peak
  FROM    climbed,
         participated
  WHERE   climbed.trip_id = participated.trip_id;
```

```
CREATE VIEW peakcount
AS
  SELECT NAME,
         Count(peak) AS peakNumber
  FROM   peakclimber
  GROUP  BY NAME;
```

```
SELECT NAME
FROM   peakclimber
WHERE  peak IN (SELECT peak
                FROM   peakclimber
                WHERE  NAME = 'MARIA')
        AND NAME != 'MARIA'
GROUP  BY NAME
HAVING Count(peak) IN (SELECT peaknumber
                      FROM   peakcount
                      WHERE  NAME = 'MARIA');
```

Results

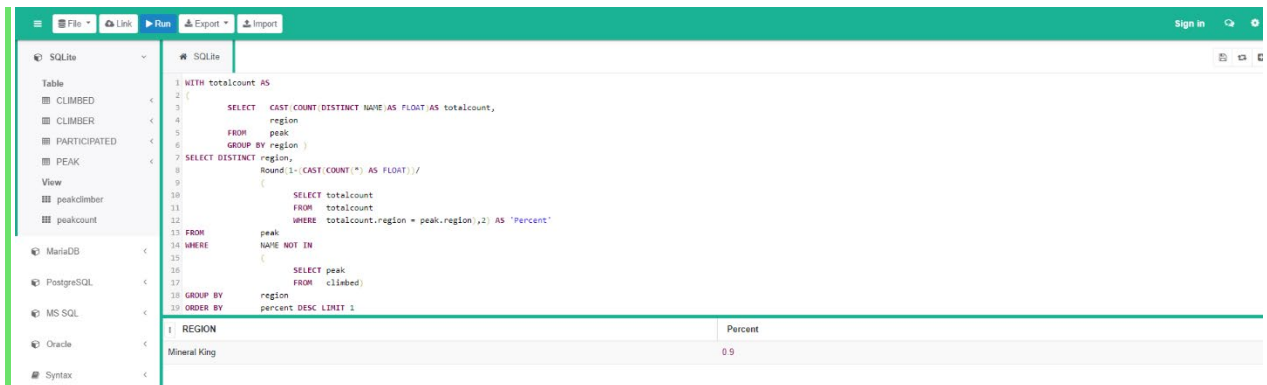
NAME
KENNETH
MARY

13.

```
WITH totalCount as(
  SELECT
    CAST(COUNT(distinct name)as FLOAT)as totalCount,region
FROM peak
  group by region
)

SELECT DISTINCT REGION,
  round(1-
(CAST(COUNT(*) as FLOAT))/(select totalCount from totalCount where totalCount.region = peak.re
gion),2) as 'Percent'
FROM PEAK
WHERE NAME not IN
(SELECT PEAK FROM CLIMBED)

GROUP BY region
order by percent desc
limit 1
```



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

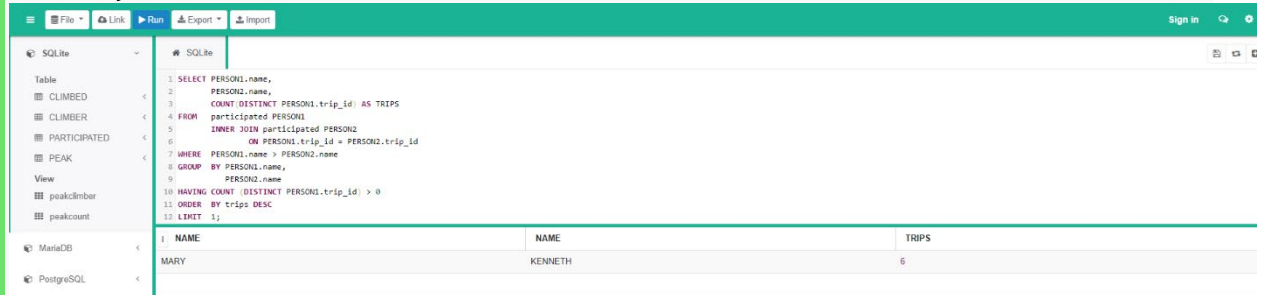
```
1 WITH totalCount AS
2 (
3     SELECT CAST(COUNT(DISTINCT NAME)AS FLOAT)AS totalCount,
4           region
5     FROM   peak
6     GROUP BY region )
7 SELECT DISTINCT region,
8     Round(1-(CAST(COUNT(*) AS FLOAT))/
9     (
10        SELECT totalCount
11        FROM   totalCount
12        WHERE  totalCount.region = peak.region),2) AS "Percent"
13 FROM     peak
14 WHERE    NAME NOT IN
15     (
16        SELECT peak
17        FROM   climbed)
18 GROUP BY region
19 ORDER BY percent DESC LIMIT 1
```

The results pane displays a single row of data:

REGION	Percent
Mineral King	0.9

14.

```
SELECT PERSON1.name,  
       PERSON2.name,  
       Count(DISTINCT PERSON1.trip_id) AS TRIPS  
FROM   participated PERSON1  
       INNER JOIN participated PERSON2  
           ON PERSON1.trip_id = PERSON2.trip_id  
WHERE  PERSON1.name > PERSON2.name  
GROUP BY PERSON1.name,  
         PERSON2.name  
HAVING Count (DISTINCT PERSON1.trip_id) > 0  
ORDER BY trips DESC  
LIMIT 1;
```



The screenshot shows a SQL IDE interface with a menu bar (File, Link, Run, Export, Import) and a sidebar on the left listing database objects (Table, CLIMBED, CLIMBER, PARTICIPATED, PEAK, View, peakclimber, peakcount). The main editor displays the SQL query from the previous block. Below the editor, the results are shown in a table with three columns: NAME, NAME, and TRIPS. The table contains one row of data.

NAME	NAME	TRIPS
MARY	KENNETH	6

15.

```
WITH table_peaks
  AS (SELECT climbed.peak,
             climbed.when_climbed,
             participated.NAME
       FROM climbed
       INNER JOIN participated
            ON participated.trip_id = climbed.trip_id)
SELECT DISTINCT NAME
FROM (SELECT t1.NAME
      FROM table_peaks t1
      LEFT JOIN table_peaks t2
            ON t1.NAME = t2.NAME
            AND ( t2.when_climbed > t1.when_climbed
                 OR ( t2.when_climbed = t1.when_climbed
                     AND t2.peak <> t1.peak ) )
            AND NOT ( t2.when_climbed > Date(
t1.when_climbed, 60) )) AS
query
GROUP BY NAME
HAVING Count (*) > 20;
```

The screenshot shows a SQL IDE interface with a green header bar containing 'File', 'Link', 'Run', 'Export', and 'Import' buttons, along with a 'Sign in' link. On the left, a sidebar lists database components: 'Table' (CLIMBED, CLIMBER, PARTICIPATED, PEAK), 'View' (peakclimber, peakcount), 'MariaDB', 'PostgreSQL', 'MS SQL', 'Oracle', and 'Syntax'. The main editor displays the SQL query from the previous block, with line numbers 1 through 20. Below the editor, the results of the query are shown as a table with one column, 'NAME', containing the following names: DONNA, ELIZABETH, JOHN, KENNETH, LINDA, MARK, MARY, PATRICIA, and STEVEN.

NAME
DONNA
ELIZABETH
JOHN
KENNETH
LINDA
MARK
MARY
PATRICIA
STEVEN