**Main Queries:**

1. **Query for top 50 most popular National Park trails in US**

create table dev\_project.public.top\_50 as (with nfp as (select max(popularity) as popularity, area\_name from nparks\_final group by area\_name),

nfa as (select \* from nparks\_final where nparks\_final.popularity in (select popularity from nfp) and nparks\_final.area\_name in (select area\_name from nfp)),

nfb as (select \* from nv\_final where "unit name" in (select area\_name from nfa) and yearraw = '2016')

select \* from nparks\_final where area\_name in (select "unit name" from nfb) and popularity in (select popularity from nfp) order by popularity desc limit 5);

A screenshot of a computer

Description automatically generated

1. **Query for getting all the parks linked along with their weather airport code, city and distance from it.**

create table distance\_list as (select distinct(top\_50.name), top\_50.lat, weather\_final.locationlat, weather\_final.locationlng, top\_50.lon, top\_50.city\_name, top\_50.state\_name, weather\_final.airportcode, weather\_final.city, (3959 \* acos(cos(radians(top\_50.lat)) \* cos(radians(weather\_final.locationlat)) \* cos(radians(weather\_final.locationlng) - radians(top\_50.lon)) + sin(radians(top\_50.lat)) \* sin(radians(weather\_final.locationlat )))) as distance from top\_50, weather\_final where top\_50.lat between weather\_final.locationlat-1 and weather\_final.locationlat+1 and top\_50.lon between weather\_final.locationlng-1 and weather\_final.locationlng+1 order by name);

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. **Query to know how many cities is a trail near to. (Considered max distance ~80 miles as limit)**

create view trail\_to\_city\_count as select count(\*) as city\_to\_trail\_count, avg(distance), trail\_name from (select distinct(top\_50.name) as trail\_name, top\_50.city\_name, top\_us\_cities.city as popular\_city, top\_50.state\_name, top\_us\_cities.population, (3959 \* acos(cos(radians(top\_50.lat)) \* cos(radians(top\_us\_cities.lat)) \* cos(radians(top\_us\_cities.lon) - radians(top\_50.lon)) + sin(radians(top\_50.lat)) \* sin(radians(top\_us\_cities.lat )))) as distance, top\_50.lat as np\_lat, top\_50.lon as np\_lon, top\_us\_cities.lat as city\_lat, top\_us\_cities.lon as city\_lon from dev\_project.public.top\_50, dev\_project.public.top\_us\_cities where top\_50.lat between top\_us\_cities.lat-1 and top\_us\_cities.lat+1 and top\_50.lon between top\_us\_cities.lon-1 and top\_us\_cities.lon+1 order by top\_50.name asc, distance desc) group by trail\_name order by trail\_name asc WITH NO SCHEMA BINDING;

Graphical user interface, application

Description automatically generated

1. **Query to get list of cities near each park**

create view park\_to\_city as select distinct(top\_50.area\_name) as park\_name, top\_50.city\_name, top\_us\_cities.city as popular\_city, top\_50.state\_name, top\_us\_cities.population, (3959 \* acos(cos(radians(top\_50.lat)) \* cos(radians(top\_us\_cities.lat)) \* cos(radians(top\_us\_cities.lon) - radians(top\_50.lon)) + sin(radians(top\_50.lat)) \* sin(radians(top\_us\_cities.lat )))) as distance, top\_50.lat as np\_lat, top\_50.lon as np\_lon, top\_us\_cities.lat as city\_lat, top\_us\_cities.lon as city\_lon from dev\_project.public.top\_50, dev\_project.public.top\_us\_citites\_new as top\_us\_cities where top\_50.lat between top\_us\_cities.lat-1 and top\_us\_cities.lat+1 and top\_50.lon between top\_us\_cities.lon-1 and top\_us\_cities.lon+1 order by distance desc WITH NO SCHEMA BINDING;

Graphical user interface

Description automatically generated

1. **Query to get list of cities near each state park trail**

create view spark\_to\_city as select distinct(sparks\_final.name) as park\_name,

sparks\_final.city\_name,

sparks\_final.state\_name,

top\_us\_cities.city as popular\_city,

top\_us\_cities.population,

(3959 \* acos(cos(radians(sparks\_final.lat)) \* cos(radians(top\_us\_cities.lat)) \* cos(radians(top\_us\_cities.lon) - radians(sparks\_final.lng)) + sin(radians(sparks\_final.lat)) \* sin(radians(top\_us\_cities.lat )))) as distance,

sparks\_final.lat as sp\_lat,

sparks\_final.lng as sp\_lon,

top\_us\_cities.lat as city\_lat,

top\_us\_cities.lon as city\_lon,

sparks\_final.composite

from dev\_project.public.sparks\_final, dev\_project.public.top\_us\_citites\_new as top\_us\_cities

where sparks\_final.lat between top\_us\_cities.lat-1 and top\_us\_cities.lat+1

and sparks\_final.lng between top\_us\_cities.lon-1 and top\_us\_cities.lon+1

-- and sparks\_final.state\_name = 'California'

order by park\_name desc WITH NO SCHEMA BINDING;

A screenshot of a computer

Description automatically generated with medium confidence

1. **Query that gives list of parks alongwith its nearby popular cities and national park visitations per year change and each city weather yearly average composite scores**

create table dev\_project.public.weather\_parks\_cities as select park\_to\_city.park\_name, park\_to\_city.popular\_city, park\_to\_city.population, park\_to\_city.distance, visits\_final.visitor\_usage\_2019, visits\_final.visitor\_usage\_2018, visits\_final.visitor\_usage\_2017, visits\_final."%increase\_2018", visits\_final."%increase\_2019",avg\_comp\_compare.avg\_2016,avg\_comp\_compare.avg\_2017,avg\_comp\_compare.avg\_2018,avg\_comp\_compare.avg\_2019 from park\_to\_city,visits\_final,avg\_comp\_compare where visits\_final.name=park\_to\_city.park\_name and park\_to\_city.popular\_city= avg\_comp\_compare.city order by park\_to\_city.park\_name ;

Graphical user interface

Description automatically generated with medium confidence

1. **Visitations per year increase**

A screenshot of a computer

Description automatically generated with medium confidence

1. **Top U.S. cities**

A screenshot of a computer

Description automatically generated with medium confidence