Experiment 5: Analytics in Action

Overview

Analytical queries, results cache, and cloning to work through typical tasks. Many query tools exist but for this experiment we'll use Snowflake native. In the previous experiments, we loaded data into two tables using Snowflake's bulk loader (COPY command) and the warehouse COMPUTE_WH. Now we are going to pretend we are analytics users at Citi Bike who need to query data in those tables using the worksheet and the second warehouse ANALYTICS_WH.

"Real World" Roles and Querying
In the "real world" the analytics users would likely have a different role then SYSADMIN; to keep the experiment simple we are going to stay with the SYSADMINrole for this experiment.

Also, in the "real-world" querying would typically be done with a business intelligence product like Tableau, Looker, PowerBI, etc. Or for more advanced analytics, data science products like Spark or R can query Snowflake. Basically any technology that leverages JDBC/ODBC can run analytics on the data in Snowflake. But to keep this experiment simple, all queries are being done via the Snowflake worksheet.

5.1 Execute SELECT Statements and Result Cache

5.1.1 Go the Worksheets tab. Within the worksheet, make sure you set your context appropriately:

Role: SYSADMIN
Warehouse: ANALYTICS_WH (L)
Database: CITIBIKE
Schema = PUBLIC

5.1.2 Run the query below to see a sample of the trips data

```
select * from trips limit 20;
```

Note: If you didn't reload the data after truncating in the previous experiment you'll see no data in the table. Check back in the last steps in Experiment 4 to ensure you loaded the data again.

Note: If for some reason the database was not create as SYSADMIN then you may receive an error.

Results Data Preview

Query ID SQL 4.05s 20 rows

Filter result... Copy

Row	TRIPDURATION	STARTTIME	STOPTIME	START_STATION...	START_STATION_NAME	START_STATION_LATITUDE
1	518	2016-09-18 ...	2016-09-18 ...	3108	Nassau Ave & Russell St	40.72557
14	1898	2016-09-18 ...	2016-09-18 ...	3345	Madison Ave & E 99 St	40.789485416
13	1231	2016-09-18 ...	2016-09-18 ...	3168	Central Park West & W 8...	40.78472675
8	510	2016-09-18 ...	2016-09-18 ...	3160	Central Park West & W 7...	40.77896784
17	847	2016-09-18 ...	2016-09-18 ...	3423	West Drive & Prospect P...	40.661063372

5.1.3 First let's look at some basic hourly statistics on Citi Bike usage. Run the query below in the worksheet. It will show for each hour the number of trips, average trip duration, and average trip distance.

```
select date_trunc('hour', starttime) as "date",
count(*) as "num trips",
avg(tripduration)/60 as "avg duration (mins)",
avg(haversine(start_station_latitude, start_station_longitude, end_station_latitude,
                        end_station_longitude)) as "avg distance (km)"
from trips
group by 1 order by 1;
```

Results Data Preview Open History

Query ID SQL 984ms 44,295 rows

Filter result... Copy Columns

Row	date	num trips	avg duration (mins)	avg distance (km)
1	2013-06-01 00:00:00.000	152	56.058442983333	2.127971476
2	2013-06-01 01:00:00.000	102	26.525163400000	2.067906273
3	2013-06-01 02:00:00.000	67	36.119900500000	2.31784827
4	2013-06-01 03:00:00.000	41	44.485365850000	2.349126632
5	2013-06-01 04:00:00.000	16	23.278125000000	1.840026007

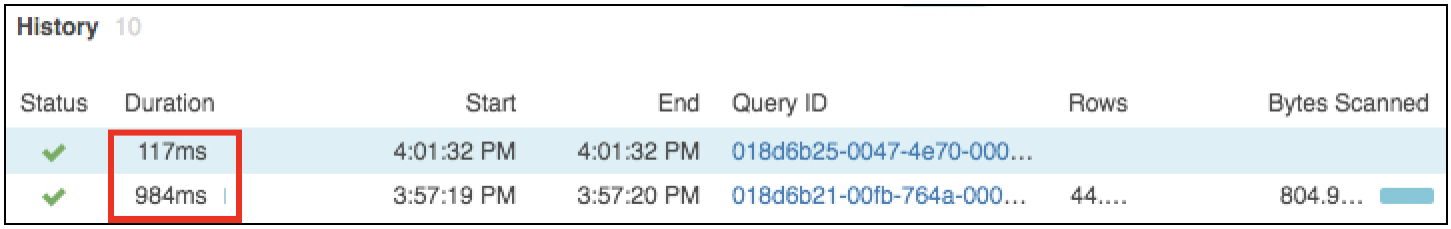
5.1.4 Snowflake has a result cache that holds the results of every query executed in the past 24 hours. These are available across warehouses, so query results returned to one user are available to any other user on the system who executes the same query, provided

the underlying data has not changed. Not only do these repeated queries return extremely fast, but they also use no compute credits.

Let's see the result cache in action by running the exact same query again.

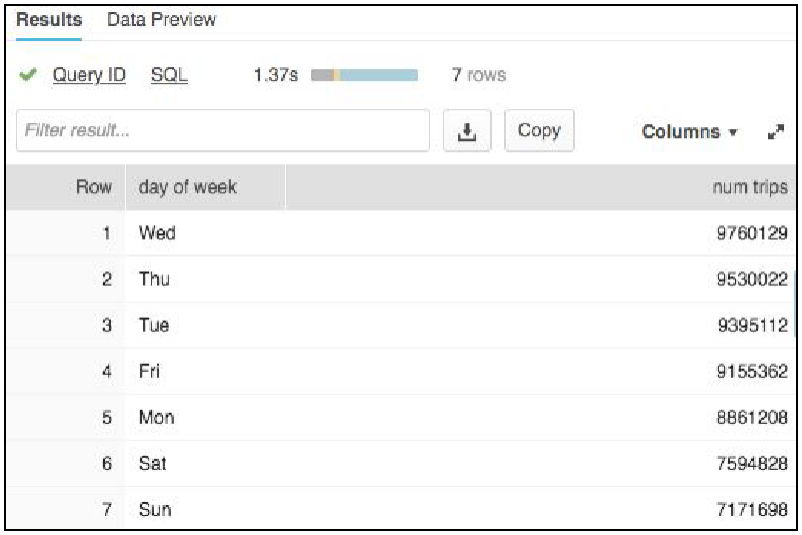
```
select date_trunc('hour', starttime) as "date",
count(*) as "num trips",
avg(tripduration)/60 as "avg duration (mins)",
avg(haversine(start_station_latitude, start_station_longitude, end_station_latitude,
                        end_station_longitude)) as "avg distance (km)"
from trips
group by 1 order by 1;
```

In the History window note that the query runs significantly faster now because the results have been cached.



5.1.5 Next, let's run this query to see which days of the week are the busiest:

```
select
    dayname(starttime) as "day of week",
    count(*) as "num trips"
from trips
group by 1 order by 2 desc;
```



Row	day of week	num trips
1	Wed	9760129
2	Thu	9530022
3	Tue	9395112
4	Fri	9155362
5	Mon	8861208
6	Sat	7594828
7	Sun	7171698

5.2 Clone a Table

Snowflake allows you to create clones, also known as “zero-copy clones” of tables, schemas, and databases in seconds. A snapshot of data present in the source object is taken when the clone is created, and is made available to the cloned object. The cloned object is writable, and is independent of the clone source. That is, changes made to either the source object or the clone object are not part of the other.

A popular use case for zero-copy cloning is to clone a production environment for use by Development & Testing to do testing and experimentation on without (1) adversely impacting the production environment and (2) eliminating the need to set up and manage two separate environments for production and Development & Testing.

Zero-Copy Cloning FTW!
A massive benefit is that the underlying data is not copied; just the metadata/pointers to the underlying data change. Hence “zero-copy” and storage requirements are not doubled when data is cloned. Most data warehouses cannot do this; for Snowflake it is easy!

5.2.1 Run the following command in the worksheet to create a development (dev) table

```
create table trips_dev clone trips
```

5.2.2 If closed, expand the database objects browser on the left of the worksheet. Click the small Refresh button in the left-hand panel and expand the object tree under the Citibike database. Check that you can see a new table under the CITIBIKE database named TRIPS_DEV. The development team now can do whatever they want with this table, including even deleting it, without having any impact on the TRIPS table or any other object.