**Question #1**: What zip code has the most Golden Retrievers?

Answer: 98115

Comments: The following queries uncovered the answer. If secondary breed was included, 98115 still came out on top.

MySQL syntax

SELECT zip\_code

, count(\*)

FROM pet\_license

WHERE species='Dog'

AND primary\_breed = 'Retriever, Golden'

GROUP BY zip\_code

ORDER BY count(\*) desc

LIMIT 1;

T-SQL syntax

SELECT TOP 1 zip

, count(\*)

FROM dbo.tblPetLicense

WHERE species='Dog'

AND primaryBreed = 'Retriever, Golden'

GROUP BY zip

ORDER BY count(\*) desc

…included secondary breed if it became relevant

SELECT TOP 1 zip

, count(\*)

FROM dbo.tblPetLicense

WHERE species='Dog'

AND (

primaryBreed = 'Retriever, Golden'

OR

secondaryBreed = 'Retriever, Golden'

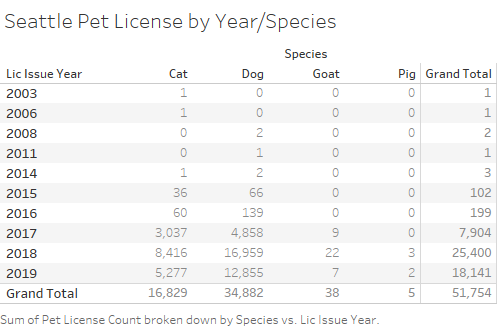
)

GROUP BY zip

ORDER BY count(\*) desc

**Question #2**: Via a visualization method of your choosing, make a chart showing the number of licenses per year per species.

Answer:

****

**Question #3**: How would you extend this job to continuously capture pet license data? A description of your thoughts here are sufficient, you do not need to write this code.

Answer:

What are the business requirements? I like to ask first because depending upon the answer, the solution could be very expensive or very inexpensive. The more “real-time” the requirement is, the more expensive the solution generally. If checking for new data once per day is sufficient, then the solution is very inexpensive. Once per hour is even cheap.

To capture the data continuously in near real-time, I would first research the possibility of using Kafka to monitor a Socrata endpoint and if this is possible and within budget, I would create a Producer from that http endpoint. I know database tables can be monitored along with various other types of data stores. I have no idea if Kafka has the ability to “watch” a socrata endpoint for updates.

If this was not possible, I would set up a scheduled task to fire/check for new data as frequently as would be allowed. To prevent throttling or limitations on frequency, I would consider the possibility of obtaining an app\_token from the producer of the data.

For scalability overall I would employ AWS technologies.

If Kafka is feasible, then use its capability to monitor the source and store raw files at S3 and push processed data into SQS for its ultimate destination in some RDS data store.

If Kafka is untenable, I would consider writing a lamba function to check for/pull new data (on a schedule) into an S3 bucket in its raw form (for replay/rebuild). I would also write a second lambda function to monitor the S3 bucket to process and store the data in its target, logging the operation. For super scalability with super high volume, employ SQS to insure imports are successful and transactionally processed once and only once.