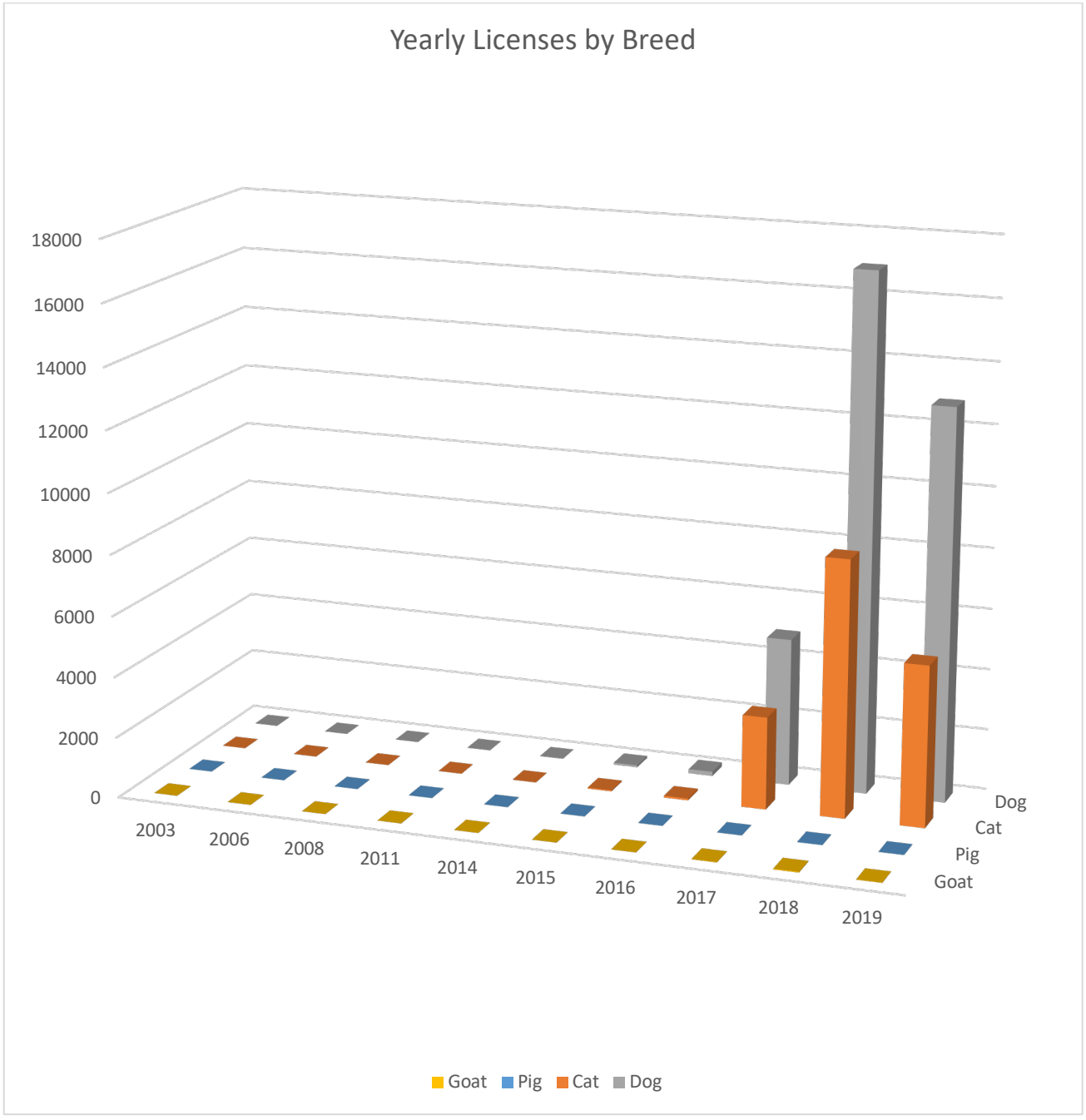
Question #1: What zip code has the most Golden Retrievers?

```
SELECT
  zip_code,
  count(*)
FROM
  seattle_license
WHERE
  primary_breed = 'Retriever, Golden'
  or secondary_breed = 'Retriever, Golden'
GROUP BY
  zip_code
ORDER BY
  count(*) desc
LIMIT 1;
```

```
+----------+----------+
| zip_code | count(*) |
+----------+----------+
| 98115    |      224 |
+----------+----------+
```

You get the same zip code, but fewer dogs, if you only look at primary breed.

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



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.

I thought I would be slick and write a program as an example here, because I wanted to work with the JSON formatted data from the REST API, rather than the CSV file. CSV data tends to be more open to interpretation and requires more cleansing.

As it turned out, you can only get 1000 rows from the API without a login token. So, I adapted that program to load JSON or CSV files as well.

The general method is thus:

1. Load the source data set from the API, CSV file or JSON file.
2. Load the current local data set from persistence.
3. Compare the data in the source data set to the persisted data set.
   3.1. If the license number is not found, this is new data and is inserted into the data store.
   3.2. If the license number is found, and it has changed from the version in the database, it is updated.

There are some performance enhancement that could be made. Since the data set is roughly 3M, and there is not enough data to make accurate estimates of growth, I am assuming that the largest data set will be roughly 25000 records per year, which will be a 5 times increase over 10 years. This means that the data set will be roughly 15M. Keeping 15M in memory would not be an issue, but pulling 250,000 rows from the database may be problematic.

1. The REST API returns the license_issue_date as a datetime. If we know the max license_issue_date from the data store, we can simply insert any record newer than that datetime, rather than scanning the existing licenses for it's existence. While we are small enough to cache the persisted data set this is not going to speed processing, but when we are using a database query to determine existence, it will be a performance gain.

2. The record is determined to have changed if any column value has changed. The comparison could be limited to a subset of the attribute to speed this up, or the cache and new records could be structured in the identical manner and a hash of the records would determine difference.

3. If the data set gets too large to query efficiently, the program could run as a daemon process, only loading records from the data store at startup, and polling the license source on an interval. This app would effectively become a pub/sub application with only one consumer.

4. This process is using a single insert with bind variables. It could be adapted to perform a bulk insert, assuming new data volume became problema