Part 1

1.

Before split: $\frac{1}{2} \times \log 2 + \frac{1}{2} \times \log 2 = 1$

Split on HasJob:

$$0: \frac{1}{3} \times \log 3 + \frac{2}{3} \times \log \frac{3}{2} = 0.918$$

$$1: \frac{3}{5} \times \log \frac{5}{3} + \frac{2}{5} \times \log \frac{5}{2} = 0.971$$

$$H(Y|X) = \frac{3}{8} \times 0.918 + \frac{5}{8} \times 0.971 = 0.951$$

Info gained: $H(Y) - H(Y|X) = 0.049$

Split on HasFamily:

$$0: \frac{1}{4} \times \log 4 + \frac{3}{4} \times \log \frac{4}{3} = 0.81$$

$$1: \frac{3}{4} \times \log \frac{4}{3} + \frac{1}{4} \times \log 4 = 0.81$$

$$H(Y|X) = 0.5 \times 0.81 + 0.5 \times 0.81 = 0.81$$

Info gained: $H(Y) - H(Y|X) = 0.19$

Split on IsAbove30years:

$$0: \frac{1}{2} \times \log 2 + \frac{1}{2} \times \log 2 = 1$$

$$1: \frac{1}{2} \times \log 2 + \frac{1}{2} \times \log 2 = 1$$

$$H(Y|X) = \frac{2}{8} \times 1 + \frac{6}{8} \times 1 = 1$$

Info gained: $H(Y) - H(Y|X) = 0$

HasFamily has the most info gained so split on HasFamily.

2.

$$H(S|X) = 0.7 \times \log \frac{1}{0.7} + 0.2 \times \log \frac{1}{0.2} + 0.1 \times \log \frac{1}{0.1} = 1.157$$

This is the minimum capacity required to transmit this signal without data loss.

Part 2

1.

Bag of words takes the unique words (words that are not stop words) in the dataset and put them in a list. They are arranged in an unordered way such that each sentence is only represented by the words in it. Word2vec store the words in a vector space, assigning each word with a vector. These vectors are used to capture the context of a sentence, in which each word's meaning is predicted by its surrounding words. As a result, words that appear in similar context will be closer together in the vector space (measured by cosine similarity). The bag-of-words model is easier to construct, saving computing resources. While the bag-of-words model ignore the order of the words, the word2vec model study the arrangements of words to better understand the document.

2.

Word vectors are vectors of number that represent the weights of words or phrases in a document. By putting word vectors in a vector space, the similarities between words and how the words are used are discovered. The method of creating distributional representations of the words using word vectors in a high-dimensional vector space is called word embedding. The embedding of a word depends on the words around it, and how often it appears together with other words, so that the probability of this occurrence can be measured.

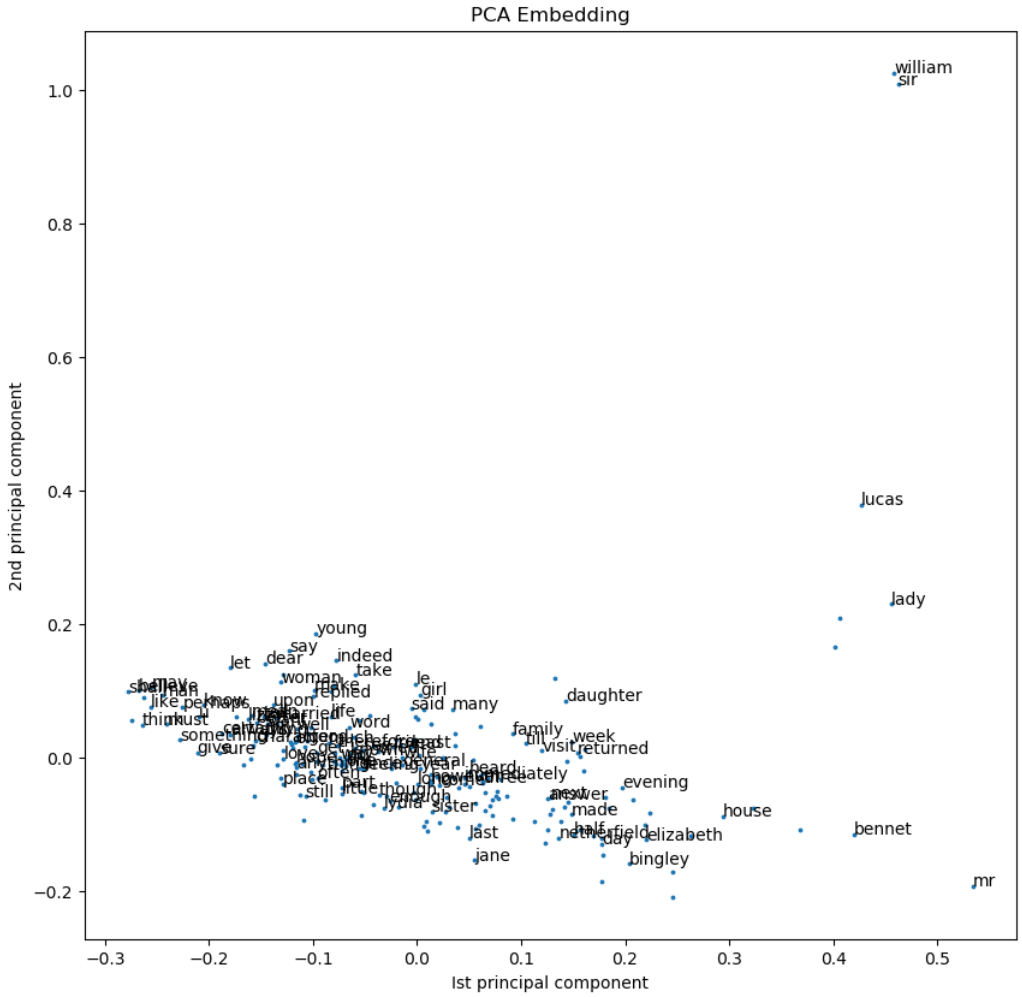
3.

A corpus is a large set of texts. It records the occurrences of each word in different grammar structures, so that a speech pattern can be established. To help recognizing the speech patterns, words are labeled with tags (n. adj.) and stemmed to allow comparisons.

4.

Each sentence is cleaned and preprocessed using the following steps: removing HTML tags, removing special characters, converting letters to lower cases, splitting words and removing stop words, lemmatizing, and rejoining all the remaining words together. Next, this new character string will be put into a word2vec training model, which considers 10 words around each individual word as context. Each word vector has a length of 300 features, and only words that appear more than 40 times in total are considered. The model was trained for 20 iterations. Eventually, there were 234 vocabularies in the model.

Two components were used in the PCA analysis. This model seemed to set greetings (sir, lady and names like William, Lucas) aside, while putting other words (verbs and nouns) together. The first component was effective in grouping the words, but most of the words in this model were very close in the second component.




Evaluation:

Similarity: “Man” and “woman” have similarity = 0.996. “see” and “visit” have similarity = 0.993. “two” and “mother” have similarity = 0.989. The model cannot separate common words.

Most similar: [lady woman] and [man] gave words like Collin, lucas and other irrelevant words.

[sir, man] and [woman] listed several male names before mentioning “lady” and “daughter”

“Love” is similar to “regard”, “heart” and “marriage”.

“visit” is similar to “return”, but also other irrelevant words like “family”, “week”.

Doesn’t match: In [man woman ok kill], “man” doesn’t match.

In [think come like lucas], “lucas” doesn’t match.

In conclusion: the model is trained to identify words that are used to call a person.

Part 3

1. Simple selects:

1) SELECT * FROM parents

```
-> ('abraham', 'barack') ('abraham', 'clinton') ('delano', 'herbert')
('eisenhower', 'fillmore') ('fillmore', 'abraham') ('fillmore', 'delano')
('fillmore', 'grover')
```

2) SELECT parent,child

FROM parents

WHERE parent = "abraham"

➔ `('abraham', 'barack') ('abraham', 'clinton')`

3) SELECT child

FROM parents

WHERE child LIKE '%e%'

➔ `('herbert',) ('fillmore',) ('delano',) ('grover',)`

4) SELECT DISTINCT parent

FROM parents

ORDER BY parent DESC

➔ `('fillmore',) ('eisenhower',) ('delano',) ('abraham',)`

5) SELECT a.child, b.child

FROM parents a, parents b

WHERE a.parent=b.parent AND a.child > b.child

➔ `('clinton', 'barack') ('delano', 'abraham') ('grover', 'abraham')`
`('grover', 'delano')`

2. Joins

1) SELECT count(*)

FROM dogs

WHERE fur ="short"

➔ `(3,)`

2) SELECT parents.parent

FROM parents INNER JOIN dogs ON parents.child =dogs.name AND dogs.fur ="curly"

➔ `('eisenhower',) ('delano',)`

3) SELECT a.child,b.child

FROM (parents INNER JOIN dogs ON parents.child =dogs.name) a, (parents INNER JOIN dogs ON parents.child =dogs.name) b

WHERE a.fur=b.fur AND b.parent=a.child

➔ `('abraham', 'clinton')`

3. Aggregate functions

1) SELECT kind, MIN(weight)

FROM animals

➔ `('parrot', 6)`

2) SELECT AVG(legs), AVG(weight)

FROM animals

➔ `(3.0, 2009.3333333333333)`

3) SELECT a.kind, a.legs, a.weight

FROM animals a

WHERE a.legs > 2 AND a.weight <20

➔ `('cat', 4, 10) ('ferret', 4, 10)`

4) SELECT legs, AVG(weight) as average_weights

FROM animals

GROUP BY legs

➔ `(2, 4005.3333333333335) (4, 13.333333333333334)`