Part 1
There are 67 rows and 47 columns (one for the company name and 46 for the value).
Query:

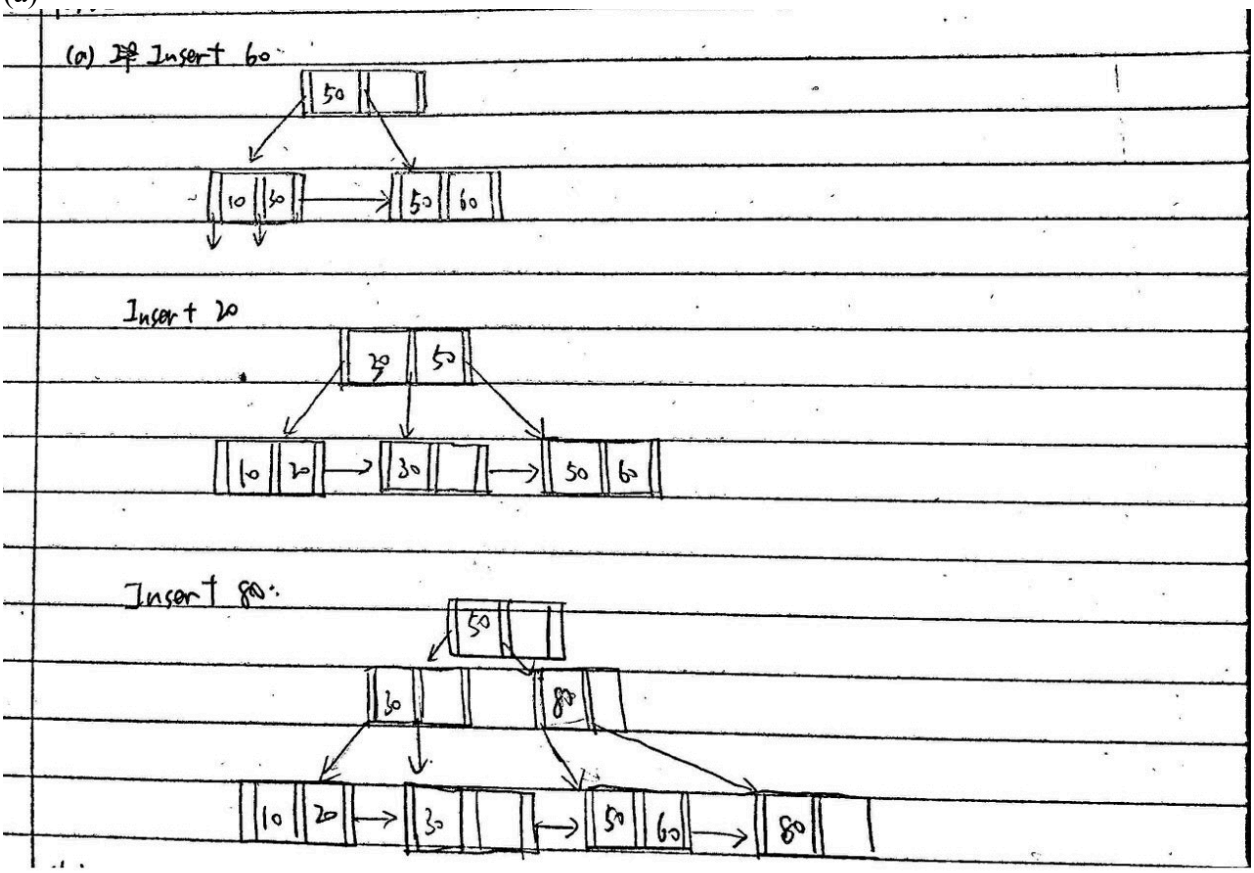
```
--SELECT value
--FROM hw3.keyvalues
--GROUP BY value;
--Calculate number of values

SELECT company,
	SUM(CASE value WHEN 'retention' THEN 1 ELSE 0 END) AS retention,
	SUM(CASE value WHEN 'bonded-by-product' THEN 1 ELSE 0 END) AS
bonded_by_product,
	SUM(CASE value WHEN 'new-tech' THEN 1 ELSE 0 END) AS new_tech,
	SUM(CASE value WHEN 'continuous-delivery' THEN 1 ELSE 0 END) AS
continuous_delivery,
	SUM(CASE value WHEN 'internal-mobilit' THEN 1 ELSE 0 END) AS internal_mobilit,
	SUM(CASE value WHEN 'agile-dev' THEN 1 ELSE 0 END) AS agile_dev,
	SUM(CASE value WHEN 'engineering-driven' THEN 1 ELSE 0 END) AS
engineering_driven,
	SUM(CASE value WHEN ' eq-iq' THEN 1 ELSE 0 END) AS  eq_iq,
	SUM(CASE value WHEN ' project-ownership' THEN 1 ELSE 0 END)AS
project_ownership,
	SUM(CASE value WHEN 'cross-dep' THEN 1 ELSE 0 END) AS cross_dep,
	SUM(CASE value WHEN ' flat-organization' THEN 1 ELSE 0 END) AS
flat_organization,
	SUM(CASE value WHEN 'feedback' THEN 1 ELSE 0 END) AS feedback,
	SUM(CASE value WHEN 'fast-pace' THEN 1 ELSE 0 END) AS fast_pace,
	SUM(CASE value WHEN 'risk-taking' THEN 1 ELSE 0 END) AS risk_taking,
	SUM(CASE value WHEN 'rapid-growth' THEN 1 ELSE 0 END) AS rapid_growth,
	SUM(CASE value WHEN 'good-beer' THEN 1 ELSE 0 END) AS good_beer,
	SUM(CASE value WHEN 'parents' THEN 1 ELSE 0 END) AS parents,
	SUM(CASE value WHEN 'open-sourcet' THEN 1 ELSE 0 END) AS open_source,
	SUM(CASE value WHEN 'friends-outside-work' THEN 1 ELSE 0 END) AS
friends_outside_work,
	SUM(CASE value WHEN 'product-drive' THEN 1 ELSE 0 END) AS product_drive,
	SUM(CASE value WHEN 'team-oriented' THEN 1 ELSE 0 END) AS team_oriented,
	SUM(CASE value WHEN 'internal-promotion' THEN 1 ELSE 0 END) AS
internal_promotion,
	SUM(CASE value WHEN 'benefit-company' THEN 1 ELSE 0 END) AS
benefit_company,
	SUM(CASE value WHEN 'physical-wellness' THEN 1 ELSE 0 END) AS
physical_wellness,
	SUM(CASE value WHEN ' many-hats' THEN 1 ELSE 0 END) AS  many_hats,
	SUM(CASE value WHEN 'open-communication' THEN 1 ELSE 0 END) AS
open_communication,
	SUM(CASE value WHEN 'data-driven' THEN 1 ELSE 0 END) AS data_driven,
```

```
        SUM(CASE value WHEN 'diverse-team' THEN 1 ELSE 0 END) AS diverse_team,
        SUM(CASE value WHEN 'quality-code' THEN 1 ELSE 0 END) AS quality_code,
        SUM(CASE value WHEN 'lunch-together' THEN 1 ELSE 0 END) AS lunch_together,
        SUM(CASE value WHEN 'office-layout' THEN 1 ELSE 0 END) AS  office_layout,
        SUM(CASE value WHEN 'flex-hours' THEN 1 ELSE 0 END) AS flex_hours,
        SUM(CASE value WHEN 'customer-first' THEN 1 ELSE 0 END) AS customer_first,
        SUM(CASE value WHEN 'pair-programs' THEN 1 ELSE 0 END) AS pair_programs,
        SUM(CASE value WHEN 'psychologically-safe' THEN 1 ELSE 0 END) AS
psychologically_safe,
        SUM(CASE value WHEN 'engages-community' THEN 1 ELSE 0 END) AS
engages_community,
        SUM(CASE value WHEN 'personal-growth' THEN 1 ELSE 0 END) AS
personal_growth,
        SUM(CASE value WHEN 'creative-innovative' THEN 1 ELSE 0 END) AS
creative_innovative,
        SUM(CASE value WHEN 'worklife-balance' THEN 1 ELSE 0 END) AS
worklife_balance,
        SUM(CASE value WHEN 'inclusive' THEN 1 ELSE 0 END) AS inclusive,
        SUM(CASE value WHEN 'junior-dev' THEN 1 ELSE 0 END) AS junior_dev,
        SUM(CASE value WHEN 'safe-env' THEN 1 ELSE 0 END) AS safe_env,
        SUM(CASE value WHEN 'remote-ok' THEN 1 ELSE 0 END) AS remote_ok,
        SUM(CASE value WHEN 'impressive-teammates' THEN 1 ELSE 0 END) AS
impressive_teammates,
        SUM(CASE value WHEN 'light-meetings' THEN 1 ELSE 0 END) AS light_meetings,
        SUM(CASE value WHEN 'interns' THEN 1 ELSE 0 END) AS interns


FROM hw3.keyvalues
GROUP BY company;
```
Part 2

(a)



(b) $n_1 * h$

(c) It is possible that each value could be in different blocks. Thus, the worst case of cost is $n_1 * h$.

(d) $n_1 = n_2$

Part 3

(a) Why is a hash structure not the best choice for a search key on which range queries are likely?

A hash structure is not the best choice when we do range query. Because in hashing structure we need to traverse all buckets to find the location as they do not store consecutively in the buckets.

(b) Our description of static hashing assumes that a large contiguous stretch of disk blocks can be allocated to a static hash table. Suppose you can allocate only C contiguous blocks. Suggest how to implement the hash table, if it can be much larger than C blocks. Access to a block should still be efficient.

We can store the address of the address of every n*C block, which means we store the address of the first block, then the Cth, then 2C and so on. Therefore, when we need to get the address of block k, we can calculate it by the address stored for k/n*C, and then plus it with k%C.