Raymond Lin
304937942
CS 143

Homework 3

1. SELECT

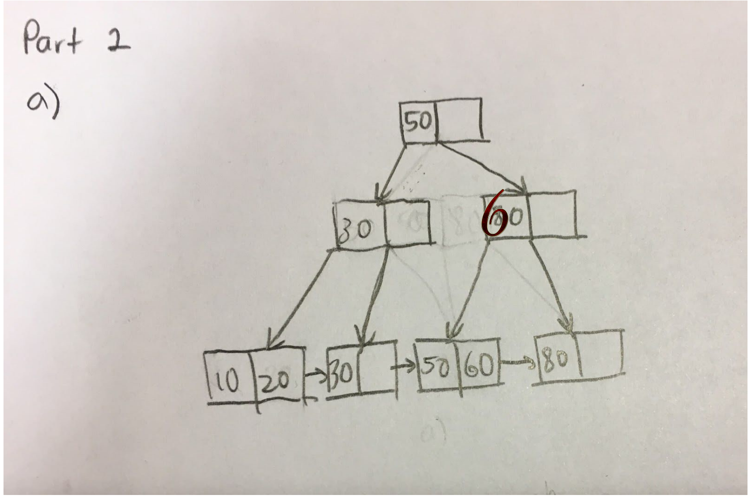
```
company,
        SUM(CASE value WHEN 'agile-dev' THEN 1 ELSE 0 END) AS fast_paced,
        SUM(CASE value WHEN 'benefit-company' THEN 1 ELSE 0 END) AS
            benefit_company,
        SUM(CASE value WHEN 'bonded-by-product' THEN 1 ELSE 0 END) AS
            bonded_by_product,
        SUM(CASE value WHEN 'continuous-delivery' THEN 1 ELSE 0 END) AS
            continuous_delivery,
        SUM(CASE value WHEN 'creative-innovative' THEN 1 ELSE 0 END) AS
            creative_innovative,
        SUM(CASE value WHEN 'cross-dep' THEN 1 ELSE 0 END) AS cross_dep,
        SUM(CASE value WHEN 'customer-first' THEN 1 ELSE 0 END) AS
            customer_first,
        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 'engages-community' THEN 1 ELSE 0 END) AS
            engages_community,
        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 'fast-paced' THEN 1 ELSE 0 END) AS fast_paced,
        SUM(CASE value WHEN 'feedback' THEN 1 ELSE 0 END) AS feedback,
        SUM(CASE value WHEN 'flat-organization' THEN 1 ELSE 0 END) AS
            flat_organization,
        SUM(CASE value WHEN 'flex-hours' THEN 1 ELSE 0 END) AS flex_hours,
        SUM(CASE value WHEN 'friends-outside-work' THEN 1 ELSE 0 END) AS
            friends_outside_work,
        SUM(CASE value WHEN 'good-beer' THEN 1 ELSE 0 END) AS good_beer,
        SUM(CASE value WHEN 'impressive-teammates' THEN 1 ELSE 0 END) AS
            impressive_teammates,
        SUM(CASE value WHEN 'inclusive' THEN 1 ELSE 0 END) AS inclusive,
        SUM(CASE value WHEN 'internal-mobility' THEN 1 ELSE 0 END) AS
            internal_mobility,
SUM(CASE value WHEN 'internal-promotion' THEN 1 ELSE 0 END) AS
    internal_promotion,
SUM(CASE value WHEN 'interns' THEN 1 ELSE 0 END) AS interns,
SUM(CASE value WHEN 'junior-devs' THEN 1 ELSE 0 END) AS junior_devs,
SUM(CASE value WHEN 'light-meetings' THEN 1 ELSE 0 END) AS
    light_meetings,
SUM(CASE value WHEN 'lunch-together' THEN 1 ELSE 0 END) AS
    lunch_together,
SUM(CASE value WHEN 'many-hats' THEN 1 ELSE 0 END) AS many_hats,
SUM(CASE value WHEN 'new-tech' THEN 1 ELSE 0 END) AS new_tech,
SUM(CASE value WHEN 'office-layout' THEN 1 ELSE 0 END) AS
    office_layout,
SUM(CASE value WHEN 'open-communication' THEN 1 ELSE 0 END) AS
    open_communication,
SUM(CASE value WHEN 'open-source' THEN 1 ELSE 0 END) AS open_source,
SUM(CASE value WHEN 'pair-programs' THEN 1 ELSE 0 END) AS
    pair_programs,
SUM(CASE value WHEN 'parents' THEN 1 ELSE 0 END) AS ideal_for_parents,
SUM(CASE value WHEN 'personal-growth' THEN 1 ELSE 0 END) AS
    personal_growth,
SUM(CASE value WHEN 'physical-wellness' THEN 1 ELSE 0 END) AS
    physical_wellness,
SUM(CASE value WHEN 'product-driven' THEN 1 ELSE 0 END) AS
    product_driven,
SUM(CASE value WHEN 'project-ownership' THEN 1 ELSE 0 END) AS
    project_ownership,
SUM(CASE value WHEN 'psychologically-safe' THEN 1 ELSE 0 END) AS
    psychologically_safe,
SUM(CASE value WHEN 'quality-code' THEN 1 ELSE 0 END) AS
    quality_code,
SUM(CASE value WHEN 'rapid-growth' THEN 1 ELSE 0 END) AS
    rapid_growth,
SUM(CASE value WHEN 'remote-ok' THEN 1 ELSE 0 END) AS remote_ok,
SUM(CASE value WHEN 'retention' THEN 1 ELSE 0 END) AS retention,
SUM(CASE value WHEN 'risk-taking' THEN 1 ELSE 0 END) AS risk_taking,
SUM(CASE value WHEN 'safe-env' THEN 1 ELSE 0 END) AS safe_env,
SUM(CASE value WHEN 'team-oriented' THEN 1 ELSE 0 END) AS
    team_oriented,
SUM(CASE value WHEN 'worklife-balance' THEN 1 ELSE 0 END) AS
    worklife_balance
```

FROM hw3.keyvalues
GROUP BY company;

Number of columns: 47 (1 for company + 46 different values)
Number of rows: 67 (67 different companies)

2.

a.



b. max number of keys in a tree of height h: $3^{(h-1)} * 2$

worst case complexity of finding lower bound (in this case, 10): $O(log(3^{(h-1)} * 2))$

worst case complexity of finding values that satisfy the range (in this case, 10 < A < 50): $O(n_1 + log(3^{(h-1)} * 2))$

Range query is not on entire search key. Each record likely to be on a different block because of ordering of records in a file. Worst case: $O(n_1 h)$

c. worst case complexity of finding values that satisfy two ranges of different keys (in this case, 10 < A < 50 and 5 < B < 10): $O(n_2 n_1 + log(3^{(h-1)} * 2))$

Same as above, since we just check each tuple that satisfies the first condition to see if they satisfy the second. Worst case: $O(n_1 h)$

d. As long as n_1 and n_2 are small, the index would be efficient in finding records that satisfy both conditions.

The index is efficient when $n_1 = n_2$ because no extra records are output in the first stage.

3.

a. A hash data structure is not ideal for range queries because the data is not sorted and not stored in order. As a result, if you are looking for a range, you must compute the hash value on all the keys within that range and enter disk for each hash value, because each key within the range may not hash to the same bucket. This increases block I/O.

b. If we could only allocate C contiguous blocks, we could implement a hash table that is much larger than C blocks by using multi-level hashing. So for example, a certain set of keys would all hash to the same value, and this corresponds to a pointer that points to another level of blocks that contain all of the key value pairs. Then, we would use a different hash function on the key to find the pointer that points to the exact location of the record.
   Make a list that contains entries of the different C block groups. Locating a block requires 2 steps: use block number to find block address and then look for tuple within block.