D5A

1)

2)

Case 1:

A new query plan assumes that the salary values for the existing employees does not change; the only change to the table is addition of new employees.

The same department cannot exist in multiple cities

And that the number of Departments will be significantly less than the number of employees.

In that case, we could define a composite clustered index on [department\_no, created\_on] on the Employee table for 2 reasons:

1. We can keep track of the average salary of employees in each department till a certain time period: Once the average query is called, we only need evaluate average for the employees added after the last time the average query was called, and normalize with the previous result accordingly
2. There do not need to be lookups to the department table while the average is being calculated since we are only grouping by the department number. We can do a join on that result to match the dept numbers to the location names

This has the benefit of not having to query the entire dataset to find the average, just reusing the old value and adjusting it with the average of rows created after that point in time.

And for lookups, this becomes easy as we have both the values required (department number and created\_on) in our clustered index.

Case 2:

Assumptions:

1. the department table is significantly smaller than employee table, and a memory block can almost completely occupy the department table.
2. Department table won’t have duplicate department\_ids and will not need to be sorted.
3. The same department cannot exist in multiple cities. i.e. there will only be 1 tuple in the department table that has a city name.

Do a hash join on the employee table.

Generate hash table/partitions for employee and department.

Stream employee hash partitions into memory and store the department hash table in memory. Do a lookup on matching hash values and the ones that qualify, can be retrieved through the hash function.

3) Query to report number of rows in each table for your current schema

Select table\_name, num\_rows from all\_tables

This is ‘fairly’ accurate, gives the rowcount for the last time the dbms statistics were checked.