1. (5 points) Consider the following schema for an e-commerce application:

Diver\( (\text{did}, \text{firstname}, \text{lastname}, \text{age}, \text{expertise\_level}) \)
Purchases\( (\text{did}, \text{eid}, \text{date}) \)
Equipment\( (\text{eid}, \text{name}, \text{price}) \)

Come up with a good physical database design to support a workload that consists of the following expected mix of queries:

\[
\begin{align*}
\text{Q0:} & \quad \text{SELECT * FROM Equipment WHERE } \text{eid} = \text{value1}; \\
\text{Q1:} & \quad \text{SELECT * FROM Diver WHERE } \text{firstname} \text{ LIKE} \ '%\text{value1}\%'; \\
\text{Q2:} & \quad \text{SELECT price, COUNT(*) FROM Equipment GROUP BY price;} \\
\text{Q3:} & \quad \text{SELECT E.name, E.price FROM Equipment E, Purchases P, Diver D WHERE D.age = value1 AND E.eid = P.eid AND D.did = P.did;} \\
\text{Q4:} & \quad \text{SELECT * FROM Equipment WHERE name LIKE 'value1%'}
\end{align*}
\]

Suppose this data will be residing in a DBMS that only offers B+ Trees as its choice of index type. Come up with a good set of indexes for the data and the workload above – indicating, for each recommended index, the indexed columns and the motivating queries in the table below. Use the first column of the table to number the indexes from 1 to N for quick reference in subsequent parts of this problem. (Not all rows need to be filled in.)

<table>
<thead>
<tr>
<th>Index Number</th>
<th>Indexed Table</th>
<th>Indexed Column(s)</th>
<th>Motivating Query(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Equipment</td>
<td>eid</td>
<td>Q0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(2) (3 points) Your design above should have at least one entry for each of the database's tables. Indicate below (by index number) what are your index option(s) to be considered for a clustered index, if any, and briefly say why.

(i) Clustered index(es) for Diver:

(ii) Clustered index(es) for Purchases:

(iii) Clustered index(es) for Equipment:

3. (2 points) In NOSQL world, instead of having three relations, we could insert a bag of (eid,date) pairs in Diver or a bag of (did,date) in Equipment table. Briefly mention why keeping the relations as it is (relational case) would be better for this example than any of the mentioned NOSQL alternatives.