1. 索引

在关系型数据库中，索引和散列可以提高查询性能。以下是我建议在此数据库中对哪些数据字段进行索引或散列的建议，以及相应的解释：

Program:

Program\_ID (Primary Key) - 主键通常会自动创建索引。

University\_ID (Foreign Key) - 作为外键，该字段经常用于连接查询，因此建议创建索引。

University:

University\_ID (Primary Key) - 主键通常会自动创建索引。

Standardized\_Test:

Program\_ID (Primary Key, Foreign Key) - 作为主键和外键，该字段应该被索引。在这种情况下，不需要单独为外键创建索引，因为主键已经被索引。

Employment:

Program\_ID (Primary Key, Foreign Key) - 作为主键和外键，该字段应该被索引。在这种情况下，不需要单独为外键创建索引，因为主键已经被索引。

这里的建议主要基于以下原因：

主键通常已经自动创建了索引，以便快速查找特定记录。

外键用于连接查询，因此对它们进行索引可以加快连接操作的速度。

在考虑对哪些字段进行散列时，通常会选择那些用于执行高度选择性查询的字段。然而，在这个数据库中，没有明显的候选字段需要散列。一般来说，散列适用于大型数据库和那些需要快速查找和插入的字段。在这种情况下，索引可以满足大多数性能需求

1. FD MVD NF

在这个数据库中，我们有四个表：Program，University，Standardized\_Test，和Employment。以下是将E-R图转换为关系模式（Relation Schema）的过程，其中包括主键、外键、函数依赖以及多值依赖的明确指示，并证明设计处于良好的标准化形式。

Program (已经是第三范式):

Program\_ID (Primary Key)

University\_ID (Foreign Key, referencing University)

Complexity\_Level

Program\_Name

Early\_DDL

Final\_DDL

Program\_Link

GPA

Recommendation\_Min\_Amount

Recommendation\_Requirement

Tuition\_Fee

Undergraduate\_Institution\_Level

函数依赖：

Program\_ID → All other attributes (完全函数依赖)

University\_ID → Program\_Name (部分函数依赖)

University (已经是第三范式):

University\_ID (Primary Key)

University\_Name

函数依赖：

University\_ID → University\_Name (完全函数依赖)

Standardized\_Test (Weak Entity, depends on Program, 已经是第三范式):

Program\_ID (Primary Key, Foreign Key, referencing Program)

GRE\_Required

GRE\_Cut\_off\_Point

TOEFL\_Accepted

TOEFL\_Requirement

IELTS\_Accepted

IELTS\_Requirement

函数依赖：

Program\_ID → All other attributes (完全函数依赖)

Employment (Weak Entity, depends on Program, 已经是第三范式):

Program\_ID (Primary Key, Foreign Key, referencing Program)

Avg\_Salary\_Per\_Year

Employment\_Rate

Employer\_Satisfaction

Company

Industry

函数依赖：

Program\_ID → All other attributes (完全函数依赖)

所有表都处于第三范式（3NF），原因如下：

在第一范式（1NF）中，所有表中的属性都是原子性的。

在第二范式（2NF）中，所有表的非主键属性都完全依赖于主键（没有部分依赖）。

在第三范式（3NF）中，不存在传递依赖。也就是说，非主键属性之间没有依赖关系。

这个设计遵循了数据库设计的良好实践，并且是标准化的。