***UNIVERSITY DATABASE MANAGEMENT SYSTEM***

**Group Members**

(Section 2)

Archit Agrawal (202051213)

Kalash Singh Jadoun (202051097)

Sarang Nagar (202051168)

Sahil Kumar Rai (202051167)

Sushil Kumar Patel (202051188)

Reporting TA: Amit Dwivedi

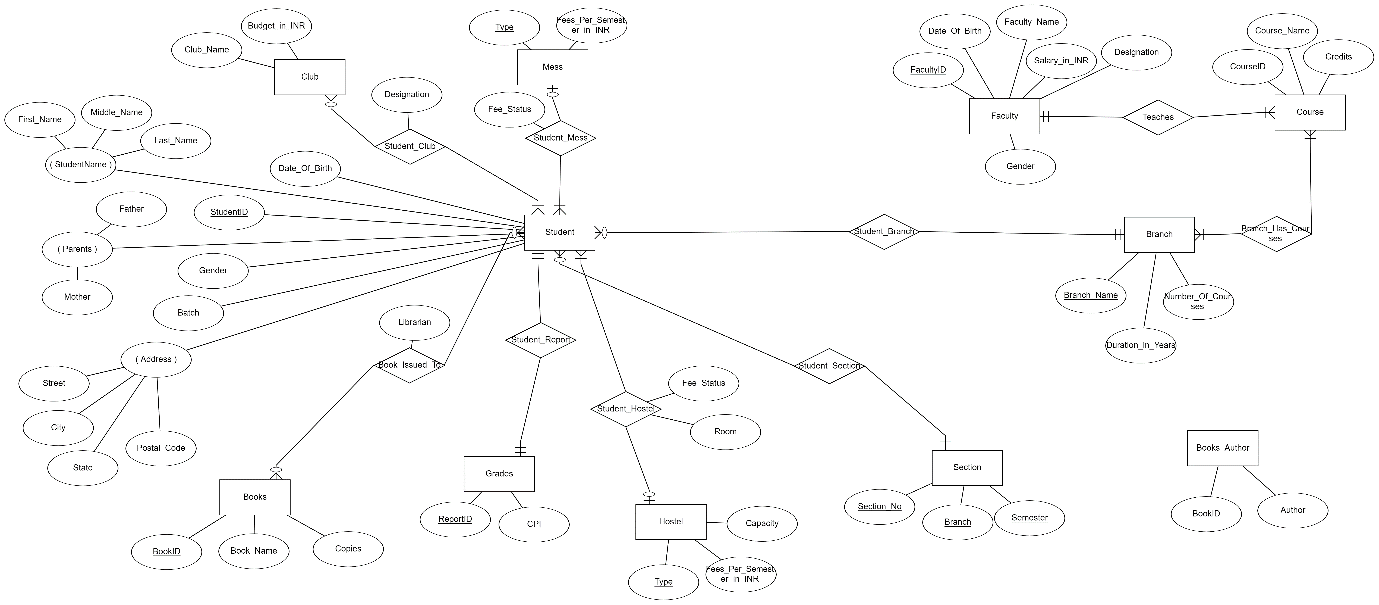
***INTRODUCTION***

University Database Management System is a MySQL database designed to handle the university records. The University Database Management System creates, manages and performs all the activities related to the database of a given university.

The database consists of information about the university, branches, students, faculties, courses, library, clubs etc. The main aim of this project is to manage the database in such a way that information can be retrieved and modified in an efficient way.

* **This Final Report Includes**:
* ER Diagram
* MySQL Database
* Web Application to access the Database from Client Side
* Functional Dependencies
* Closure of Functional Dependencies
* Minimal Cover of Functional Dependencies
* Normalization

***Entity-Relationship Diagram***

******

(The ER diagram is also submitted separately in the zip file as it is not clearly visible here.)

We have deleted several entities from the database because of several issues. Our model is depicted by the ER Diagram in the above image.

***MySQL Database and Web Application***

The code and setup for the database and web application is provided separately. All the steps are clearly mentioned to run the database in a separate file.

***Closure and Minimal Cover of Functional Dependencies***

In this section, we have mentioned all the functional dependencies that are present in our database. We have formed their closures and minimal covers. Since, there are a large number of schemas, we have shown how to find closure and minimal cover for some schemas.

* The **closure of a set of functional dependency** F, denoted by , is the set of all functional dependencies that can be derived from F using the Armstrong’s Rules.

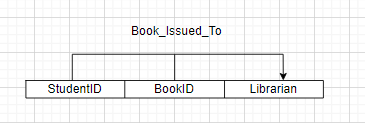
**Note:** Reflexive Functional Dependencies on a schema such as etc. are also a part of , but these dependencies are trivial and hence will not be written in . We have mainly written the transitive, pseudo-transitive and union rule dependencies, as they can be non-trivial. The candidate keys will be highlighted in red.

* The **minimal cover , of a set F of functional dependencies** is the minimum set of dependencies such that .

**Steps to find Minimal Cover of set F of functional dependencies:**

* Using decomposition rule, replace all functional dependencies of the form by and .
* Check for any extraneous attribute in the new set of functional dependencies. If there are any such attributes, remove them.
* Check for any redundant dependency. If there is any, remove them.
* **In Schema “Book\_Issued\_To”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | StudentID | BookID | Librarian |
| **Given Name** | A | B | C |

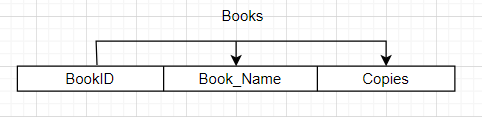
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, they will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Books”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | BookID | Book\_Name | Copies |
| **Given Name** | A | B | C |

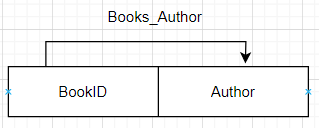
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Books\_Author”:**

We want the following functional dependencies to hold.



|  |  |  |
| --- | --- | --- |
| **Original Name** | BookID | Author |
| **Given Name** | A | B |

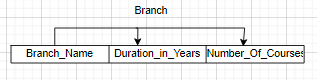
Therefore, the set F of functional dependencies contains:

* The closure of F, , are:
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Branch”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | Branch\_Name | Duration\_in\_Years | Number\_Of\_Courses |
| **Given Name** | A | B | C |

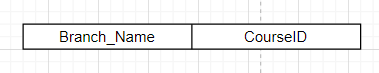
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Branch\_Has\_Course”:**

Only Trivial Functional Dependencies exists, therefore .



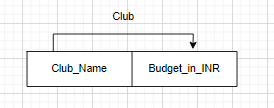
|  |  |  |
| --- | --- | --- |
| **Original Name** | Branch\_Name | CourseID |
| **Given Name** | A | B |

* The closure of F, , is:
* Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.
* The minimal cover for F is:

Since , hence,

* **In Schema “Club”:**

We want the following functional dependencies to hold.



|  |  |  |
| --- | --- | --- |
| **Original Name** | Club\_Name | Budget\_in\_INR |
| **Given Name** | A | B |

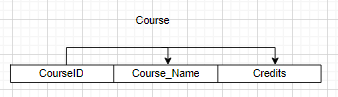
Therefore, the set F of functional dependencies contains:

* The closure of F, , is:
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Course”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | CourseID | Course\_Name | Credits |
| **Given Name** | A | B | C |

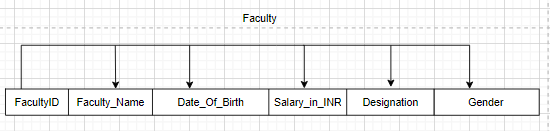
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Faculty”:**

We want the following functional dependencies to hold.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Original Name** | FacultyID | Faculty\_Name | Date\_Of\_Birth | Salary\_in\_INR | Designation | Gender |
| **Given Name** | A | B | C | D | E | F |

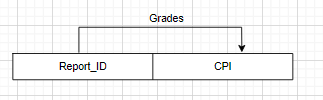
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* are derived using Union Rule.
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency. This can be checked in the following way.
* Remove the dependency and find the closure of A using remaining dependencies in F. Since, , which does not contain B. Hence, is not a redundant functional dependency.
* Similarly, we can test for other functional dependencies in F.

Hence,

* **In Schema “Grades”:**

We want the following functional dependencies to hold.



|  |  |  |
| --- | --- | --- |
| **Original Name** | ReportID | CPI |
| **Given Name** | A | B |

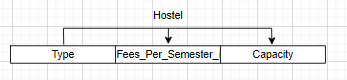
Therefore, the set F of functional dependencies contains:

* The closure of F, , is:
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Hostel”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | Type | Fees\_Per\_Semester\_in\_INR | Capacity |
| **Given Name** | A | B | C |

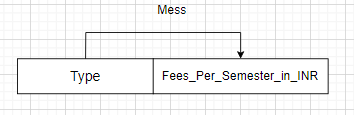
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Mess”:**

We want the following functional dependencies to hold.



|  |  |  |
| --- | --- | --- |
| **Original Name** | ReportID | CPI |
| **Given Name** | A | B |

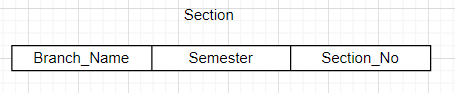
Therefore, the set F of functional dependencies contains:

* The closure of F, , is:
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Section”:**

Only Trivial Functional Dependencies exists, therefore .



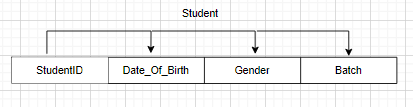
|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | Branch\_Name | Semester | Section\_No |
| **Given Name** | A | B | C |

* Few dependencies from the closure of F, , are:
* Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.
* The minimal cover for F is:

Since , hence,

* **In Schema “Student”:**

We want the following functional dependencies to hold.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Original Name** | StudentID | Date\_Of\_Birth | Gender | Batch |
| **Given Name** | A | B | C | D |

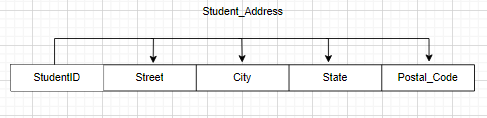
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency. This can be checked in the following way.
* Remove the dependency and find the closure of A using remaining dependencies in F. Since, , which does not contain B. Hence, is not a redundant functional dependency.
* Similarly, we can test for other functional dependencies in F.

Hence,

* **In Schema “Student\_Address”:**

We want the following functional dependencies to hold.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Original Name** | StudentID | Street | City | State | Postal\_Code |
| **Given Name** | A | B | C | D | E |

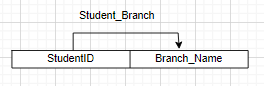
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* are derived using Union Rule.
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Student\_Branch”:**

We want the following functional dependencies to hold.



|  |  |  |
| --- | --- | --- |
| **Original Name** | StudentID | Branch\_Name |
| **Given Name** | A | B |

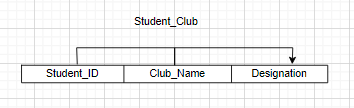
Therefore, the set F of functional dependencies contains:

* The closure of F, , is:
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Student\_Club”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | StudentID | Club\_Name | Designation |
| **Given Name** | A | B | C |

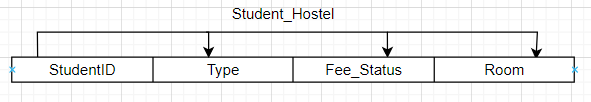
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* are derived using Union Rule.
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Student\_Hostel”:**

We want the following functional dependencies to hold.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Original Name** | StudentID | Type | Fee\_Status | Room |
| **Given Name** | A | B | C | D |

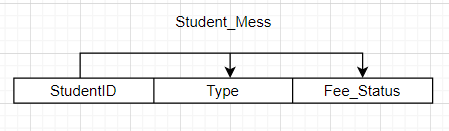
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency. This can be checked in the following way.
* Remove the dependency and find the closure of A using remaining dependencies in F. Since, , which does not contain B. Hence, is not a redundant functional dependency.
* Similarly, we can test for other functional dependencies in F.

Hence,

* **In Schema “Student\_Mess”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | StudentID | Type | Fee\_Status |
| **Given Name** | A | B | C |

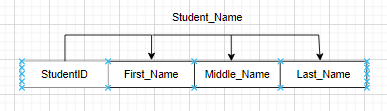
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Student\_Name”:**

We want the following functional dependencies to hold.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Original Name** | StudentID | First\_Name | Middle\_Name | Last\_Name |
| **Given Name** | A | B | C | D |

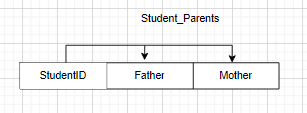
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency. This can be checked in the following way.
* Remove the dependency and find the closure of A using remaining dependencies in F. Since, , which does not contain B. Hence, is not a redundant functional dependency.
* Similarly, we can test for other functional dependencies in F.

Hence,

* **In Schema “Student\_Parents”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | StudentID | Type | Fee\_Status |
| **Given Name** | A | B | C |

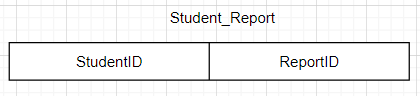
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* Functional dependencies are derived using the Augmentation Rule on the dependency . More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* is derived using Union Rule on
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Student\_Report”:**

Only Trivial Functional Dependencies exists, therefore .



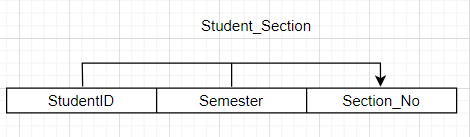
|  |  |  |
| --- | --- | --- |
| **Original Name** | StudentID | ReportID |
| **Given Name** | A | B |

* The closure of F, , is:
* Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.
* The minimal cover for F is:

Since , hence,

* **In Schema “Student\_Section”:**

We want the following functional dependencies to hold.



|  |  |  |  |
| --- | --- | --- | --- |
| **Original Name** | StudentID | Semester | Section\_No |
| **Given Name** | A | B | C |

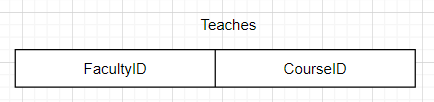
Therefore, the set F of functional dependencies contains:

* Few dependencies from the closure of F, , are:
* More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
* No Transitive Dependency
* are derived using Union Rule.
* The minimal cover for F is:
* F does not contain any extraneous attribute.
* F does not contain any redundant dependency.

Hence,

* **In Schema “Teaches”:**

Only Trivial Functional Dependencies exists, therefore .



|  |  |  |
| --- | --- | --- |
| **Original Name** | FacultyID | CourseID |
| **Given Name** | A | B |

* The closure of F, , is:
* Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.
* The minimal cover for F is:

Since , hence,

***Normalization***

In this section, we have justified the normalization in our database for the relations. Below is our justification of normality for each relation.

***For “Book\_Issued\_To”:***

* 1NF:- Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF:- FDs:- {StudentID, BookId 🡪 Librarian}. This is already an FFD (since there is only one FD). Therefore, table is in 2NF.
* 3NF:- There is only one FD in the table therefore there is no Transitive dependency in the table. Hence, it is in 3NF.
* BCNF:- Since, {student ID, BookID} is the primary key of the table, implies that it is a super key. Hence, the table is in BCNF.

***For “Books”:***

* 1NF:- Since, all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF:- FDs:- {BookID 🡪 Book\_Name, Author, Copies}. This is already an FFD (since there is only one FD). Therefore, table is in 2NF.
* 3NF:- There is only one FD in the table therefore there is no Transitive dependency in the table. Hence, it is in 3NF.
* BCNF:- Since BookID is the primary key of the table, implies that it is a super key. Hence, the table is in BCNF.

***For “Branch”:***

* 1NF:- Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF:- FDs:- {Branch\_Name 🡪 Duration\_in\_Years, Number\_of\_Course}. This is already an FFD (since there is only one FD). Therefore, table is in 2NF.
* 3NF:- There is only one FD in the table therefore there is no Transitive dependency in the table. Hence, it is in 3NF.
* BCNF:- Since Branch\_Name is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Club”:***

* 1NF:- Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF:- FDs: - {Club\_Name 🡪 Budget\_in\_INR}. This is already an FFD (since there is only one FD). Therefore, table is in 2NF.
* 3NF:- There is only one FD in the table therefore there is no Transitive dependency in the table. Hence, it is in 3NF.
* BCNF:- Since Club\_Name is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Course”:***

* 1NF:- Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF:- FDs:- {CourseID 🡪 Course\_Name, Credits}. This is already an FFD (since there is only one FD). Therefore, table is in 2NF.
* 3NF:- There is only one FD in the table therefore there is no Transitive dependency in the table. Hence, it is in 3NF.
* BCNF:- Since CourseID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Faculty”:***

* 1NF:- Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF:- FDs: - {FacultyID 🡪 > Faculty\_Name, Date\_of\_Birth, Salary\_in\_INR, Designation, Gender}. This is already an FFD. (since there is only one FD). Therefore, table is in 2NF.
* 3NF:- There is only one FD in the table therefore there is no Transitive dependency in the table. Hence, it is in 3NF.
* BCNF:- Since FacultyID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Grades”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {Report\_ID->CPI} This is already an FFD. (since there is only one FD). Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since Report\_ID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_Hostel”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->Type,Fee\_Per\_Semester,Fee\_Status}. This is already an FFD. (since there is only one FD). Therefore, table is in 2NF.
* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since StudentID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Mess”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {Type -> Fees\_Per\_Semester\_in\_INR}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since Type is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Section”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->Branch\_Name,Semester,Section\_no}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since Section is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_ Address”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->Street,city,State,postal code}. This is already an FFD. (since there is only one FD). Therefore, table is in 2NF.
* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since student ID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_Branch”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID -> Branch\_Name}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since StudentID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student \_Club”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID,Club\_Name -> Designation}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since student ID and Club\_Name is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_Hostel”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->Type,Fee\_Per\_Semester,Fee\_Status}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since StudentID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_Mess”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->Type,Fess\_Per\_Semseter,Fee\_Status}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since student ID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student Name”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->First\_Name,Middle\_Name,Last\_Name}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since student ID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_ Parents”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID ->Father ,Mother}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since student ID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.

***For “Student\_Section”:***

* 1NF: - Since all the attributes are atomic and single valued, therefore the table is already in 1NF.
* 2NF: - FDs: - {StudentID -> Semester,Branch\_Name,Section}

           This is already an FFD. (since there is only one FD)

           Therefore, table is in 2NF.

* 3NF: - There is only one FD in the table therefore there is no Transitive dependency in the table.      Hence, it is in 3NF.
* BCNF: - Since StudentID is the primary key of the table, implies that it is a super key. Hence the table is in BCNF.