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

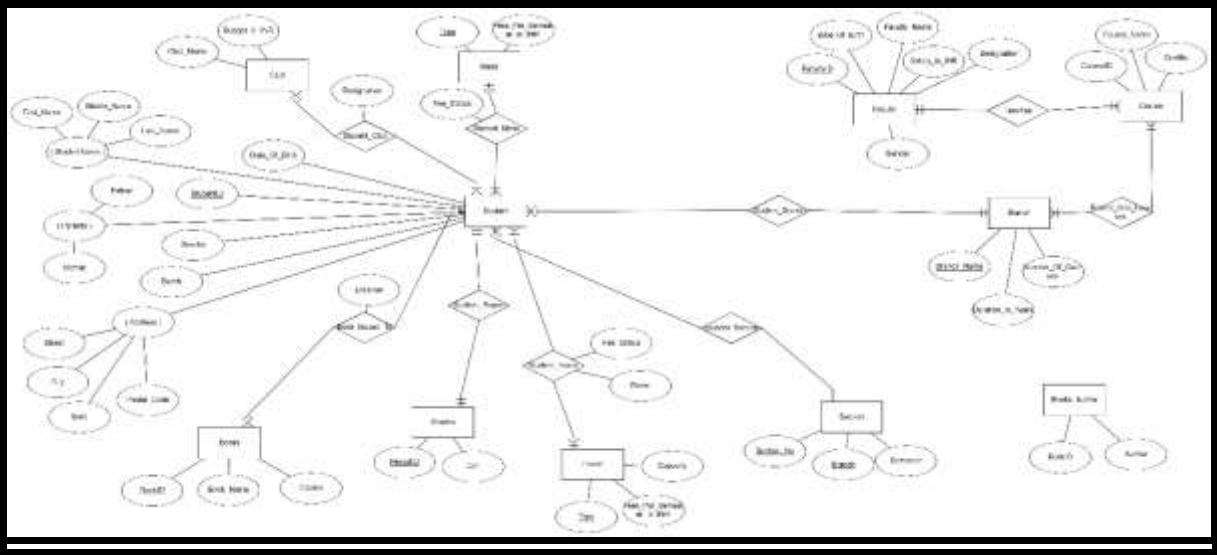
University Database Management System is a MySQL database designed to handle the university records. The University Database Management System creates, manages and performs all the activities related to the database of a given university.

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.

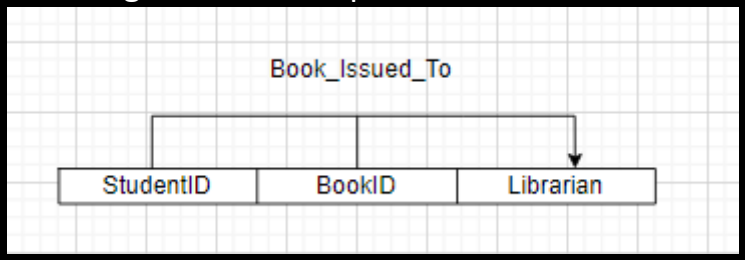
- ⇨ The **closure of a set of functional dependency** F, denoted by F^+, is the set of all functional dependencies that can be derived from F using the Armstrong's Rules.
  Note: Reflexive Functional Dependencies on a schema $\{A, B, C, D, ..\}$ such as $\{A \rightarrow A, AB \rightarrow A,\ ABC \rightarrow AB\}$ etc. are also a part of F^+, but these dependencies are trivial and hence will not be written in F^+. 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 F_c, of a set F of functional dependencies** is the minimum set of dependencies such that $F^+ = F_c^+$.

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

  - ❖ Using decomposition rule, replace all functional dependencies of the form $A \rightarrow BC$ by $A \rightarrow B$ and $A \rightarrow C$.
  - ❖ 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:

$$AB \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, AB \rightarrow AB, ABC \rightarrow AB, AB \rightarrow C, ABC \rightarrow AC, AB \rightarrow ABC\}$$

- Functional dependencies $\{ABC \rightarrow AC\}$ are derived using the Augmentation Rule on the dependency $\{AB \rightarrow A\}$. More dependencies can be derived using Augmentation Rule (however, they will be trivial).
- No Transitive Dependency
- $\{AB \rightarrow AC, AB \rightarrow ABC\}$ is derived using Union Rule on $\{AB \rightarrow A, AB \rightarrow C\}$.

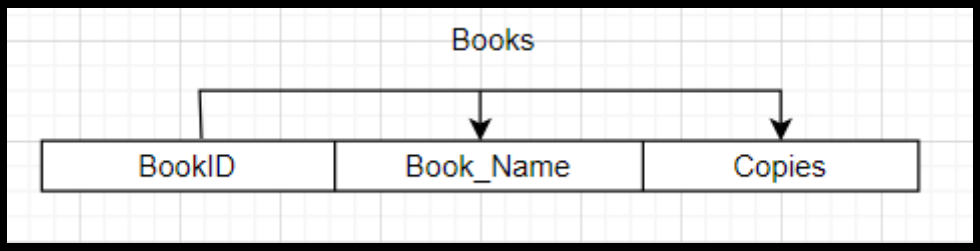
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, A \rightarrow ABC\}$$

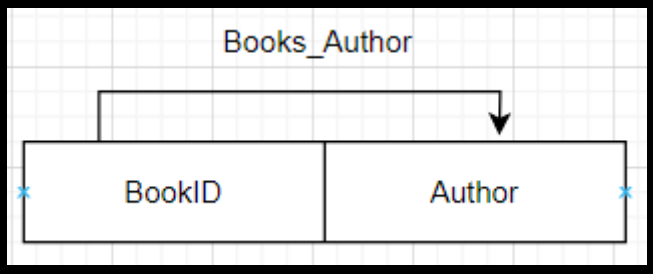
- Functional dependencies $\{AC \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC, A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B, A \rightarrow C\}$.

➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **<u>In Schema "Books_Author":</u>**
  We want the following functional dependencies to hold.



Original Name	BookID	Author
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

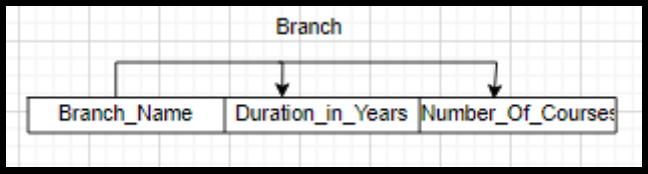
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C\}$.

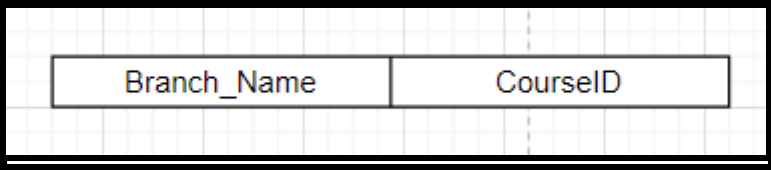
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Branch_Has_Course":**

Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	Branch_Name	CourseID
Given Name	A	B

➔ The closure of F, F^+, is:

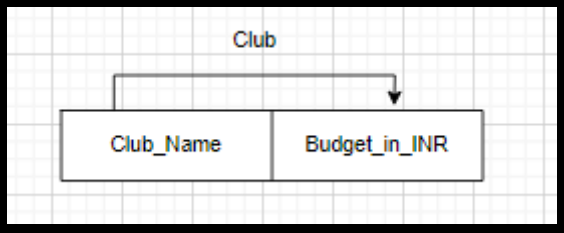
$$F^+ = \{A \to A, B \to B, AB \to A, AB \to B, AB \to AB\}$$

- Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:
Since $F = \emptyset$, hence, $F_c = \emptyset$

- **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:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

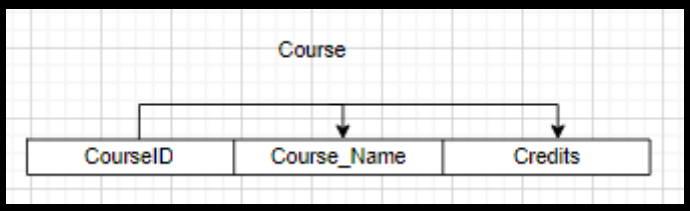
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B, A \rightarrow C\}$.

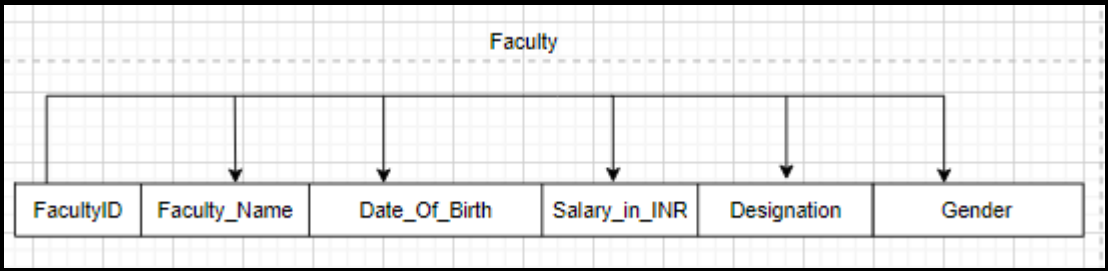
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$
$$A \rightarrow E$$
$$A \rightarrow F$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, A \rightarrow F, AC \rightarrow BC, AF \rightarrow BF, A \rightarrow BC, A \rightarrow ABCDEF\}$$

- Functional dependencies $\{AC \rightarrow BC, AF \rightarrow BF\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC, A \rightarrow ABCDEF\}$ are derived using Union Rule.

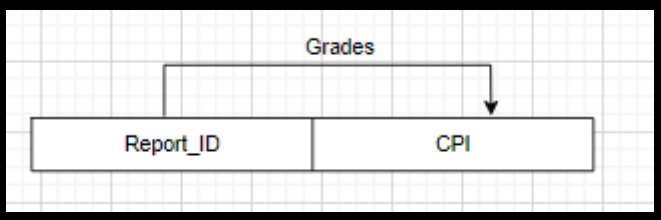
➔ The minimal cover F_c for F is:

- 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 $A \rightarrow B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D, E, F\}$, which does not contain B. Hence, $A \rightarrow B$ is not a redundant functional dependency.
  - Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

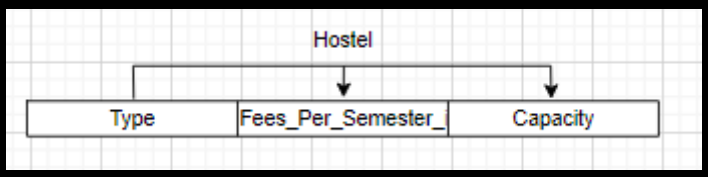
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency

- $\{A \rightarrow BC\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C\}$.

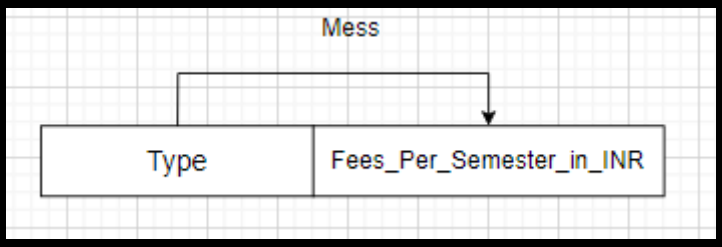
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **<u>In Schema "Mess":</u>**

We want the following functional dependencies to hold.



Original Name	ReportID	CPI
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

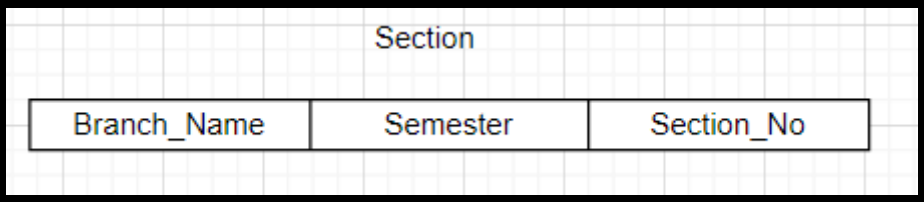
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Section":**

  Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	Branch_Name	Semester	Section_No
Given Name	A	B	C

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, B \rightarrow B, ABC \rightarrow AB, ABC \rightarrow ABC\}$$

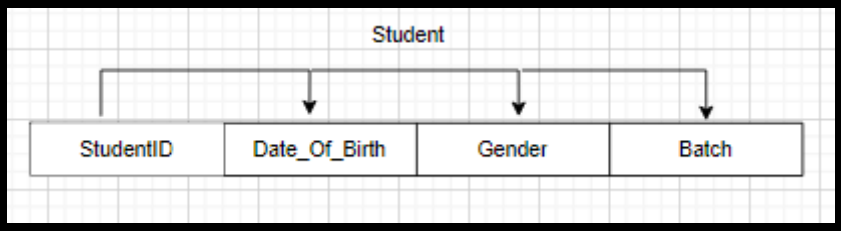
➢ Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:

Since $F = \emptyset$, hence, $F_c = \emptyset$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \to A, AB \to A, ABC \to AB, A \to B, A \to C, A \to D, AC \to BC, AD \to BD, A \to BCD, A \to ABCD\}$$

- Functional dependencies $\{AC \to BC, AD \to BD\}$ are derived using the Augmentation Rule on the dependency $\{A \to B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \to BCD, A \to ABCD\}$ is derived using Union Rule on $\{A \to B, A \to C, A \to D\}$.

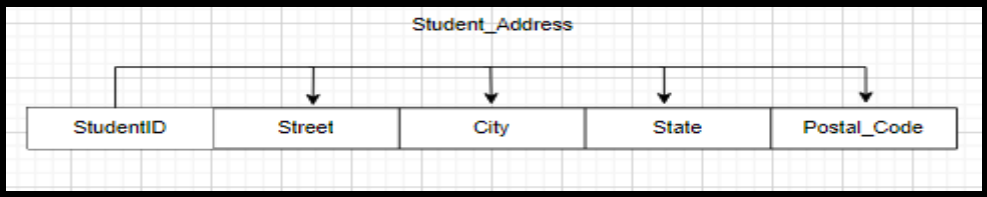
➔ The minimal cover F_c for F is:

- 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 $A \to B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D\}$, which does not contain B. Hence, $A \to B$ is not a redundant functional dependency.
  - Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$
$$A \rightarrow E$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, AC \rightarrow BC, AE \rightarrow BE, A \rightarrow BC, A \rightarrow ABCDE\}$$

- Functional dependencies $\{AC \rightarrow BC, AE \rightarrow BE\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC, A \rightarrow ABCDE\}$ are derived using Union Rule.

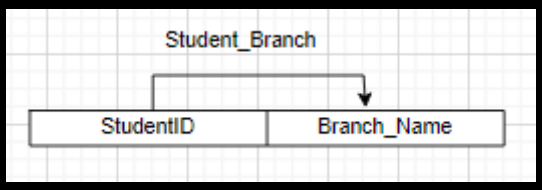
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

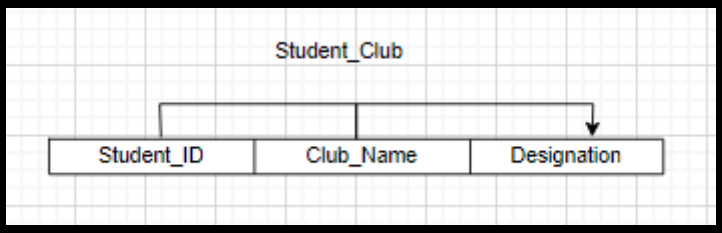
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **<u>In Schema "Student_Club":</u>**

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:

$$AB \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, AB \rightarrow C, AB \rightarrow AC, AB \rightarrow ABC\}$$

- More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{AB \rightarrow AC,\ AB \rightarrow ABC\}$ are derived using Union Rule.

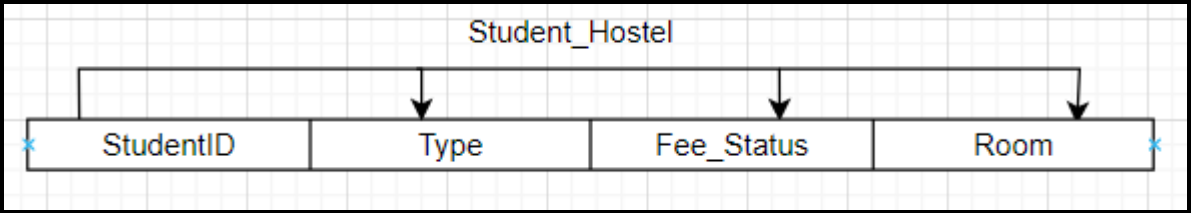
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, AC \rightarrow BC, AD \rightarrow BD, A \rightarrow BCD, A \rightarrow ABCD\}$$

- Functional dependencies $\{AC \rightarrow BC, AD \rightarrow BD\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BCD, A \rightarrow ABCD\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$.

➔ The minimal cover F_c for F is:

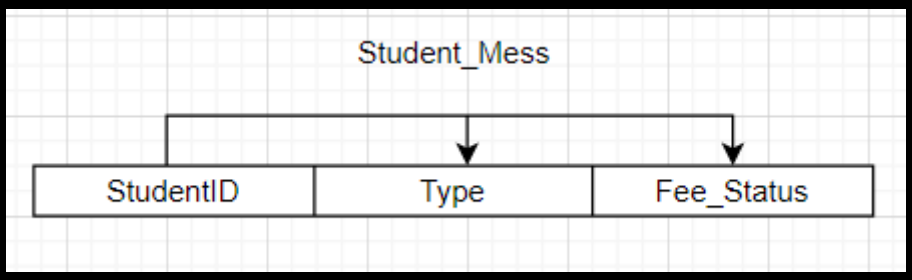
- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $A \rightarrow B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D\}$, which does not contain B. Hence, $A \rightarrow B$ is not a redundant functional dependency.

- Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow BC, A \rightarrow ABC\}$$

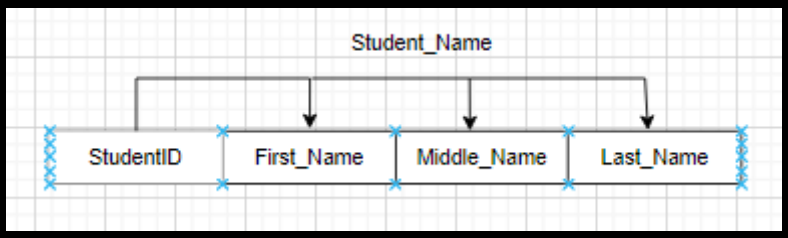
- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC, A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B, A \rightarrow C\}$.

➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **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:

$$A \to B$$
$$A \to C$$
$$A \to D$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \to A, AB \to A, ABC \to AB, A \to B, A \to C, A \to D, AC \to BC, AD \to BD, A \to ABCD\}$$

- Functional dependencies $\{AC \to BC, AD \to BD\}$ are derived using the Augmentation Rule on the dependency $\{A \to B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \to ABCD\}$ is derived using Union Rule on $\{A \to B, A \to C, A \to D\}$.

➔ The minimal cover F_c for F is:

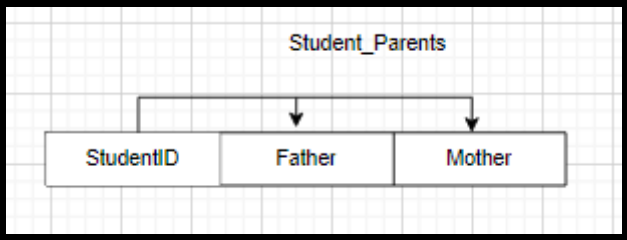
- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $A \to B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D\}$, which does not contain B. Hence, $A \to B$ is not a redundant functional dependency.

- Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **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:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C\}$.

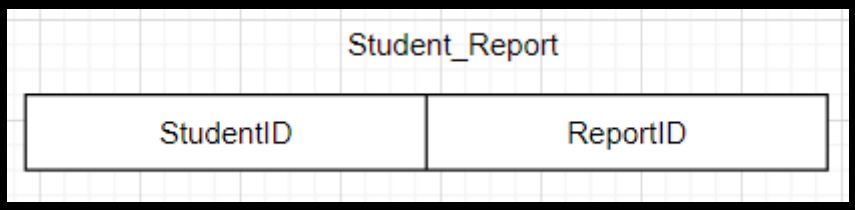
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Student_Report":**

  Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	StudentID	ReportID
Given Name	A	B

➔ The closure of F, F^+, is:

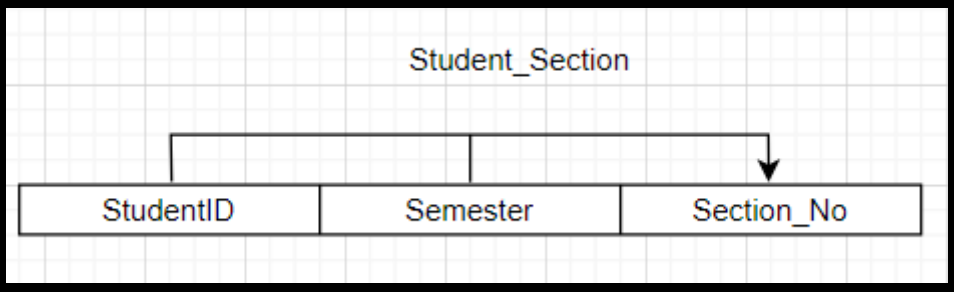
$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, AB \rightarrow AB\}$$

➢ Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:
Since $F = \emptyset$, hence, $F_c = \emptyset$

- **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:

$$AB \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, AB \rightarrow C, AB \rightarrow AC, AB \rightarrow ABC\}$$

- More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{AB \rightarrow AC,\ AB \rightarrow ABC\}$ are derived using Union Rule.

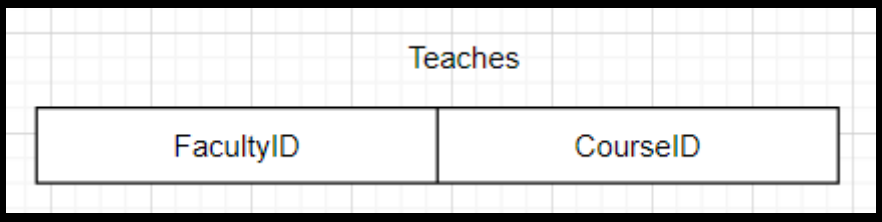
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Teaches":**

Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	FacultyID	CourseID
Given Name	A	B

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, AB \rightarrow AB\}$$

- Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:
Since $F = \emptyset$, hence, $F_c = \emptyset$

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.