PROJECT DEFENSE PRESENTATION

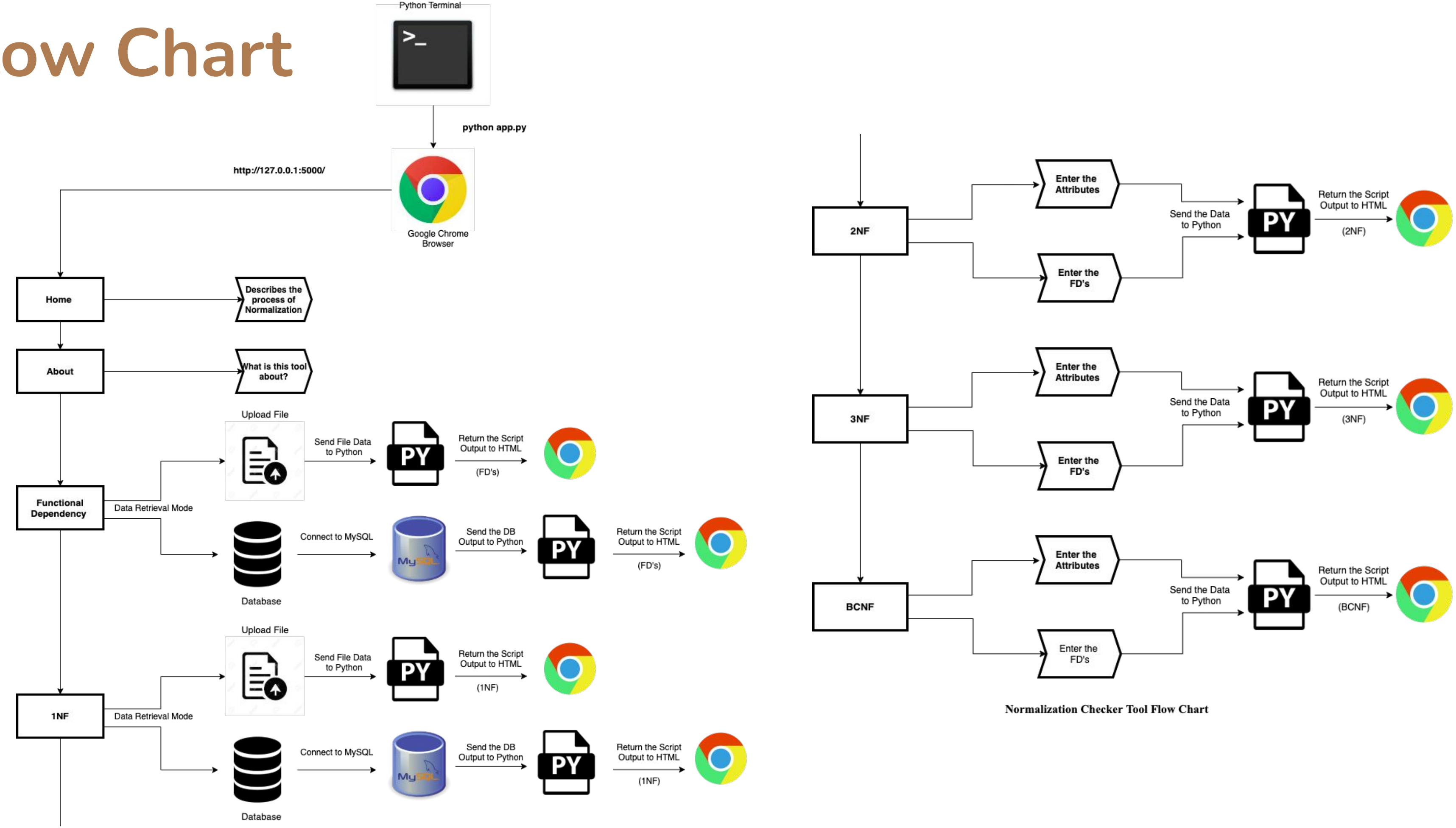
Normalization Checker Tool For Relational Databases

SARANYA PANDIARAJ | POOJA MALAGE

GANGADHAR KAVURI | MERCY BOSE

Data analysis & Discussion

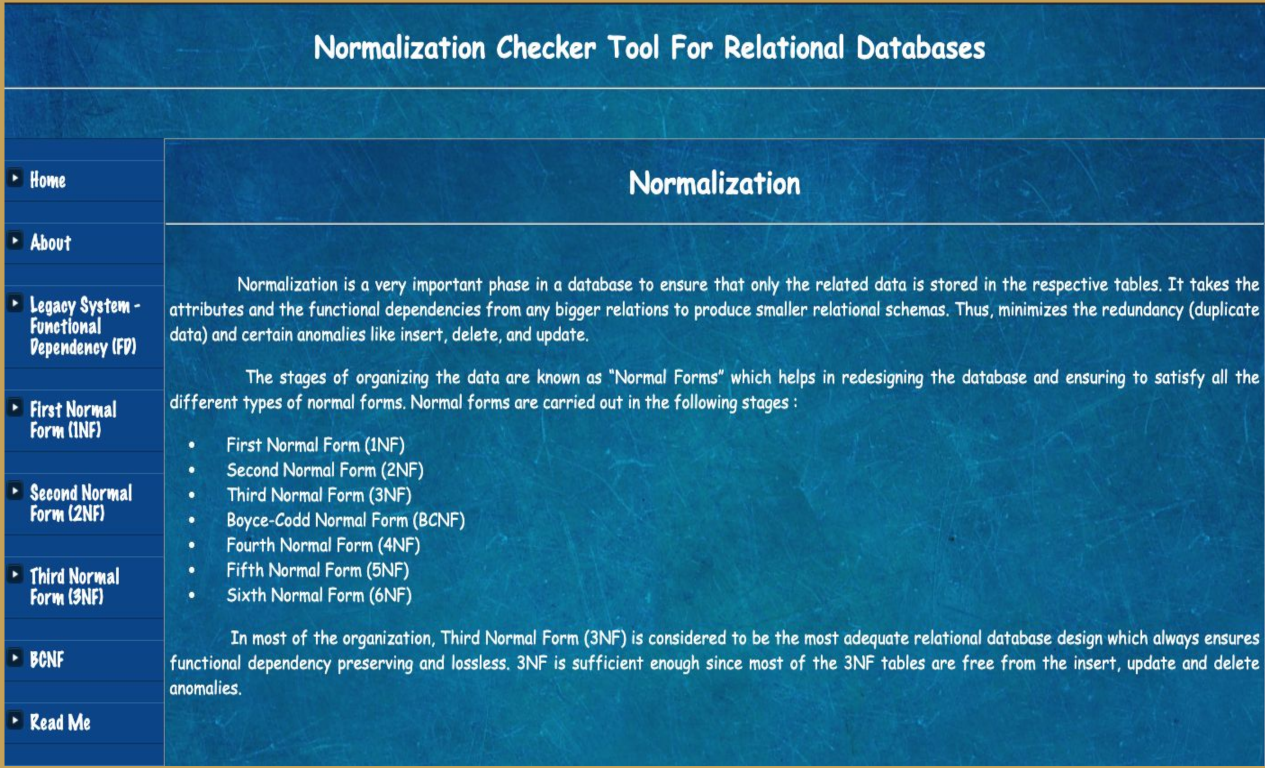
Flow Chart



Normalization Checker Tool Flow Chart

Output Generation

The tool is designed where a user is guided through stages of the Normalization design process till BCNF as shown below



Normalization Checker Tool For Relational Databases

Normalization

Normalization is a very important phase in a database to ensure that only the related data is stored in the respective tables. It takes the attributes and the functional dependencies from any bigger relations to produce smaller relational schemas. Thus, minimizes the redundancy (duplicate data) and certain anomalies like insert, delete, and update.

The stages of organizing the data are known as "Normal Forms" which helps in redesigning the database and ensuring to satisfy all the different types of normal forms. Normal forms are carried out in the following stages :

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)
- Sixth Normal Form (6NF)

In most of the organization, Third Normal Form (3NF) is considered to be the most adequate relational database design which always ensures functional dependency preserving and lossless. 3NF is sufficient enough since most of the 3NF tables are free from the insert, update and delete anomalies.

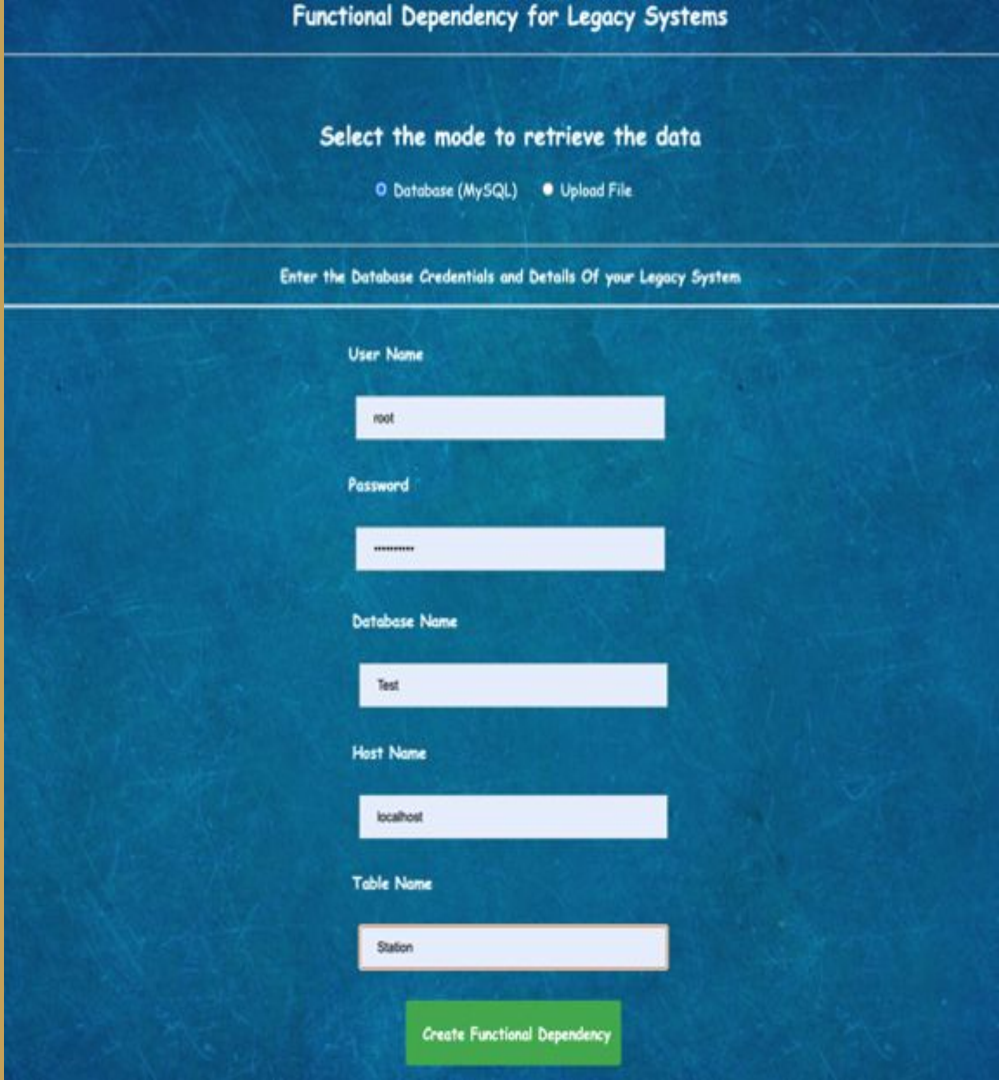
Output Analysis

Analysis of output for the below normal forms and also the FD's

- **Legacy System - Functional Dependency (FD)**
- **First Normal Form (1NF)**
- **Second Normal Form (2NF)**
- **Third Normal Form (3NF)**
- **Boyce-Codd Normal Form (BCNF)**

Legacy System - Functional Dependency

Data Retrieval -> Database



Functional Dependency for Legacy Systems

Select the mode to retrieve the data

○ Database (MySQL) ● Upload File

Enter the Database Credentials and Details Of your Legacy System

User Name

root

Password

••••••••••

Database Name

Test

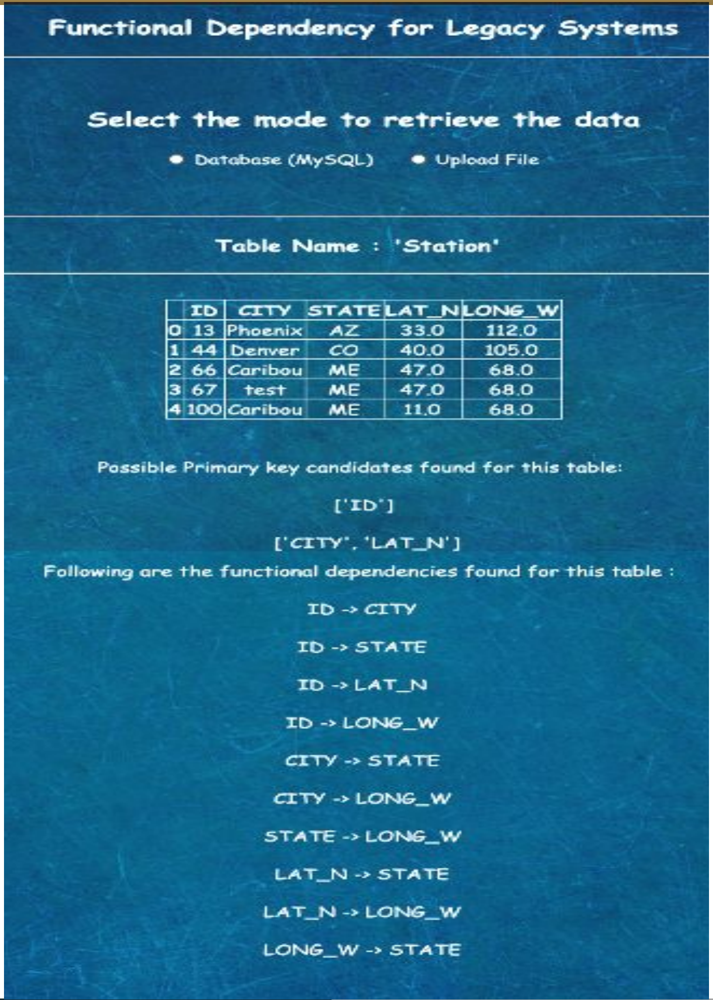
Host Name

localhost

Table Name

Station

FD Output (Table)



Functional Dependency for Legacy Systems

Select the mode to retrieve the data

● Database (MySQL) ● Upload File

Table Name : 'Station'

	ID	CITY	STATE	LAT_N	LONG_W
0	13	Phoenix	AZ	33.0	112.0
1	44	Denver	CO	40.0	105.0
2	66	Caribou	ME	47.0	68.0
3	67	test	ME	47.0	68.0
4	100	Caribou	ME	11.0	68.0

Possible Primary key candidates found for this table:

['ID']

['CITY', 'LAT_N']

Following are the functional dependencies found for this table :

ID -> CITY

ID -> STATE

ID -> LAT_N

ID -> LONG_W

CITY -> STATE

CITY -> LONG_W

STATE -> LONG_W

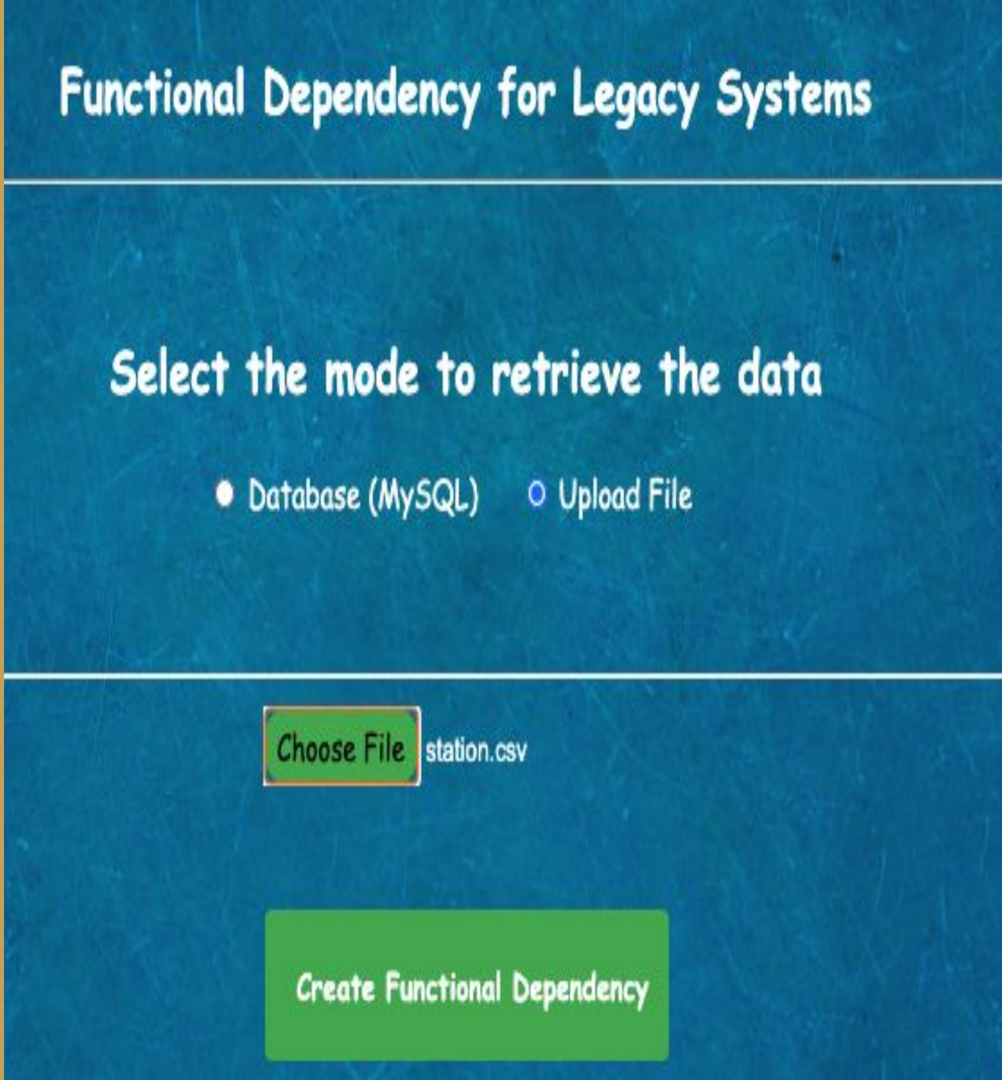
LAT_N -> STATE

LAT_N -> LONG_W

LONG_W -> STATE

Legacy System - Functional Dependency

Data Retrieval -> Upload File



Functional Dependency for Legacy Systems

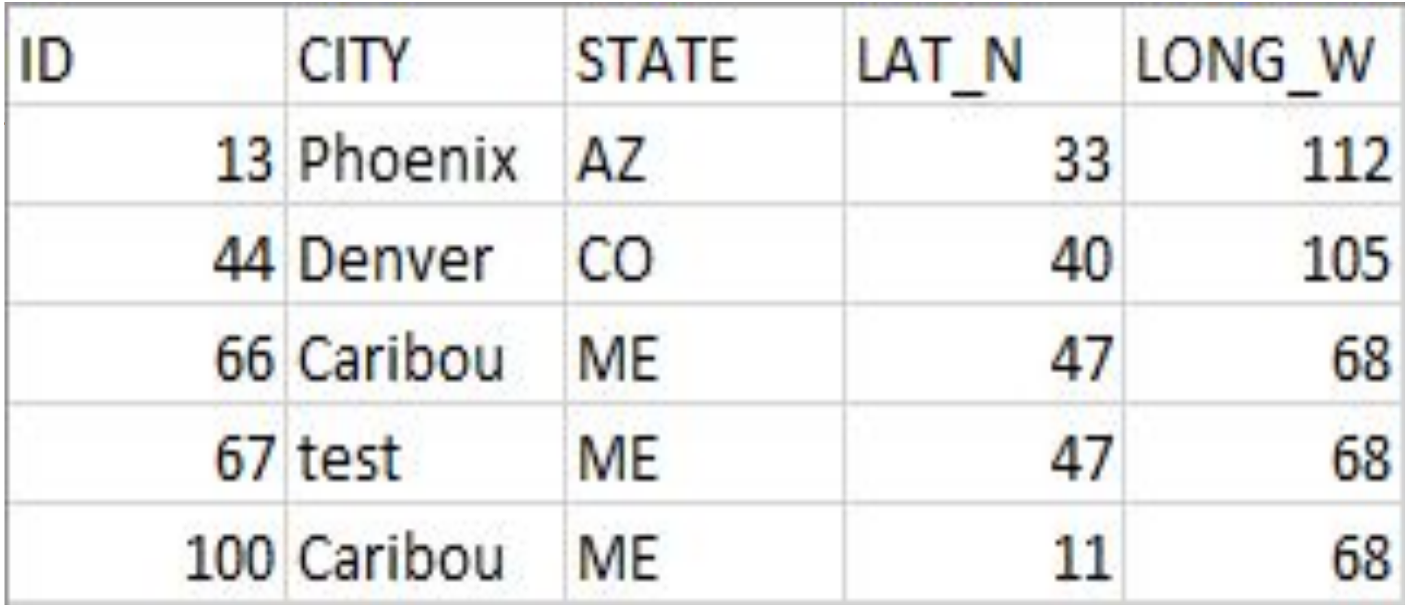
Select the mode to retrieve the data

● Database (MySQL) ○ Upload File

Choose File station.csv

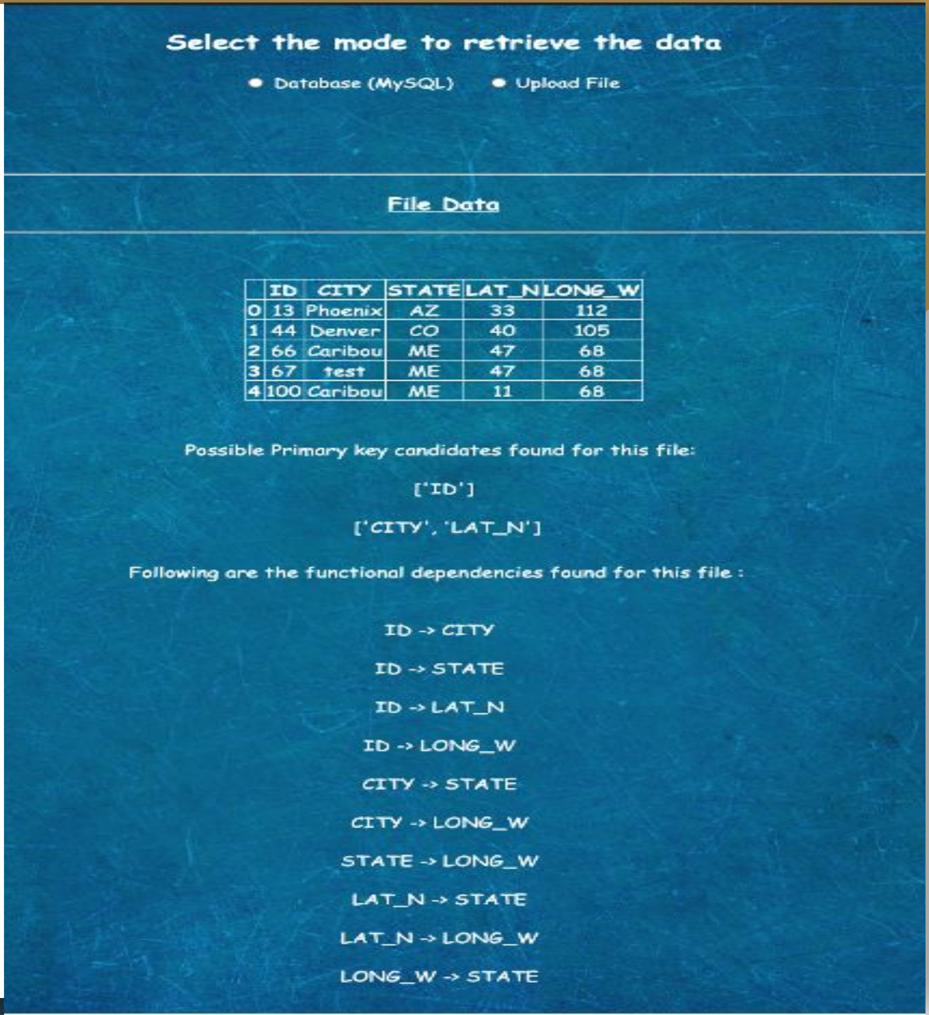
Create Functional Dependency

Station - Example File



ID	CITY	STATE	LAT_N	LONG_W
13	Phoenix	AZ	33	112
44	Denver	CO	40	105
66	Caribou	ME	47	68
67	test	ME	47	68
100	Caribou	ME	11	68

FD Output (File)



Select the mode to retrieve the data

● Database (MySQL)   ● Upload File

File Data

	ID	CITY	STATE	LAT_N	LONG_W
0	13	Phoenix	AZ	33	112
1	44	Denver	CO	40	105
2	66	Caribou	ME	47	68
3	67	test	ME	47	68
4	100	Caribou	ME	11	68

Possible Primary key candidates found for this file:

['ID']

['CITY', 'LAT_N']

Following are the functional dependencies found for this file :

ID -> CITY

ID -> STATE

ID -> LAT_N

ID -> LONG_W

CITY -> STATE

CITY -> LONG_W

STATE -> LONG_W

LAT_N -> STATE

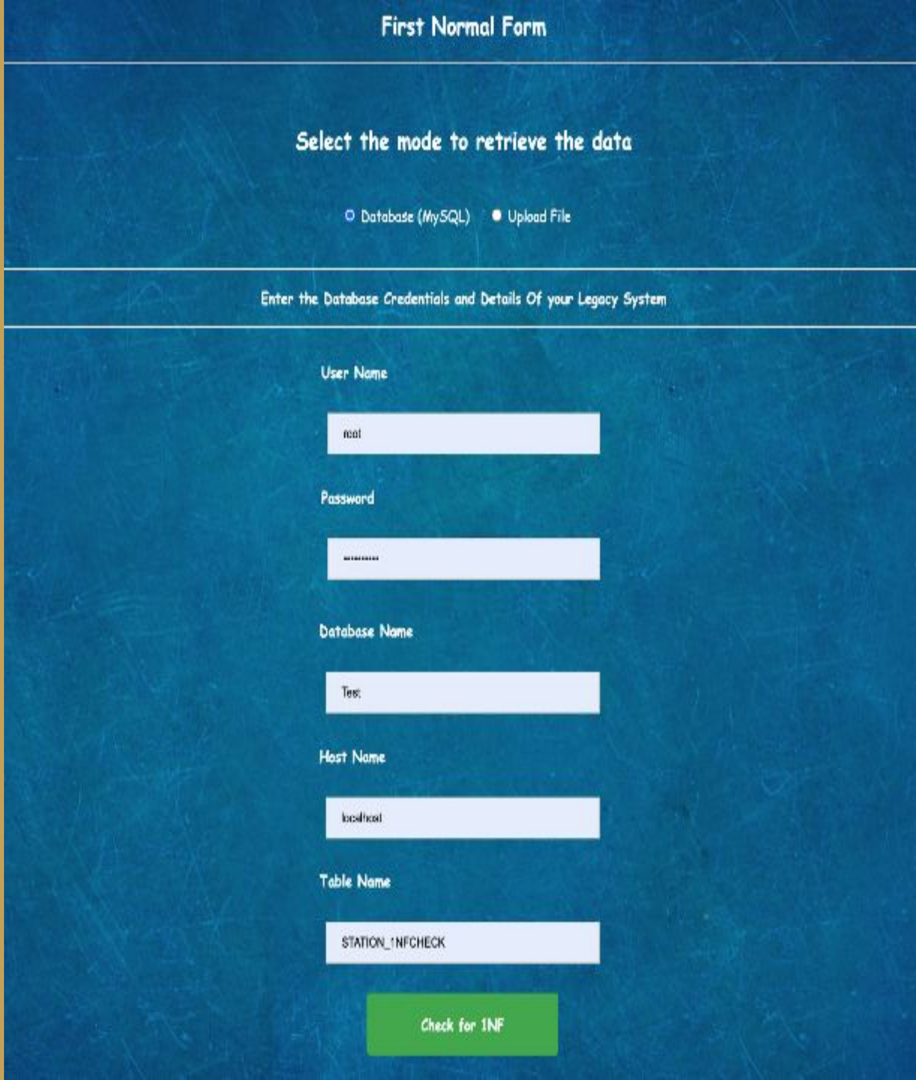
LAT_N -> LONG_W

LONG_W -> STATE

First Normal Form (1NF)

Data Retrieval Mode ->

Database



First Normal Form

Select the mode to retrieve the data

○ Database (MySQL) ● Upload File

Enter the Database Credentials and Details Of your Legacy System

User Name	root
Password	•••••••••
Database Name	Test
Host Name	localhost
Table Name	STATION_1NFCHECK

Output for 1NF: Table

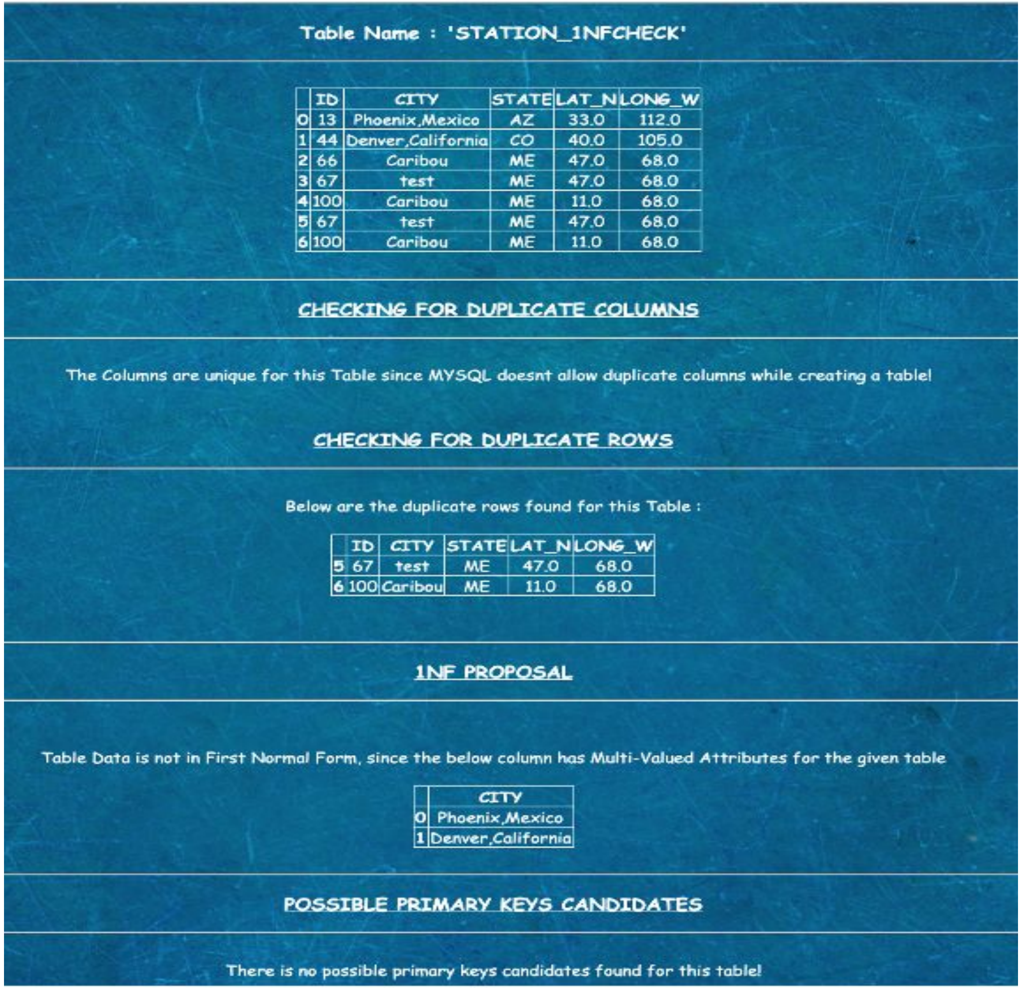


Table Name : 'STATION_1NFCHECK'

	ID	CITY	STATE	LAT_N	LONG_W
0	13	Phoenix,Mexico	AZ	33.0	112.0
1	44	Denver,California	CO	40.0	105.0
2	66	Caribou	ME	47.0	68.0
3	67	test	ME	47.0	68.0
4	100	Caribou	ME	11.0	68.0
5	67	test	ME	47.0	68.0
6	100	Caribou	ME	11.0	68.0

CHECKING FOR DUPLICATE COLUMNS

The Columns are unique for this Table since MYSQL doesnt allow duplicate columns while creating a table!

CHECKING FOR DUPLICATE ROWS

Below are the duplicate rows found for this Table :

	ID	CITY	STATE	LAT_N	LONG_W
5	67	test	ME	47.0	68.0
6	100	Caribou	ME	11.0	68.0

1NF PROPOSAL

Table Data is not in First Normal Form, since the below column has Multi-Valued Attributes for the given table

	CITY
0	Phoenix,Mexico
1	Denver,California

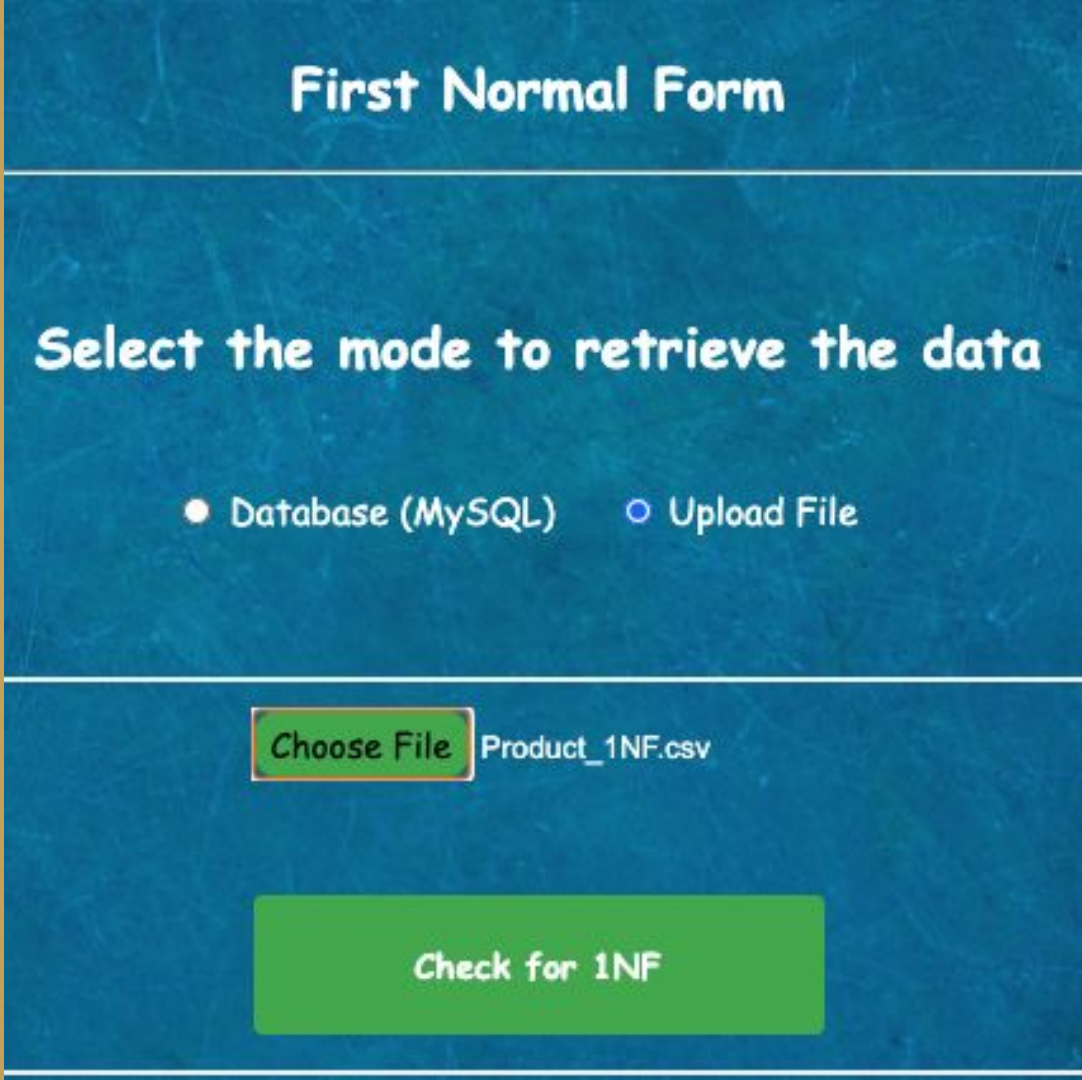
POSSIBLE PRIMARY KEYS CANDIDATES

There is no possible primary keys candidates found for this table!

First Normal Form (1NF)

Data Retrieval Mode ->

Upload File



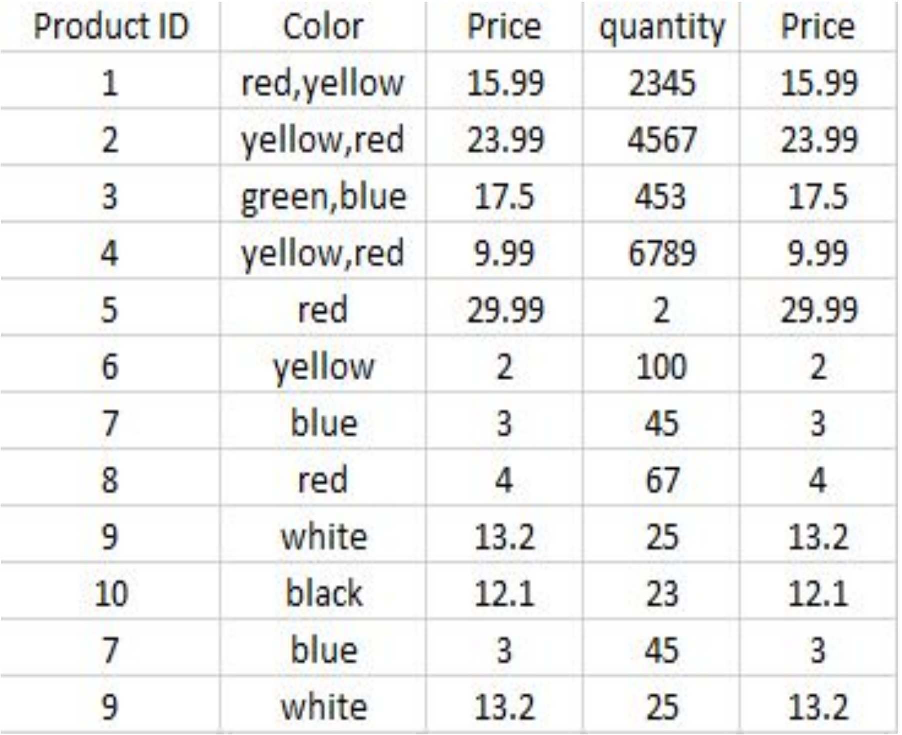
First Normal Form

Select the mode to retrieve the data

○ Database (MySQL) ● Upload File

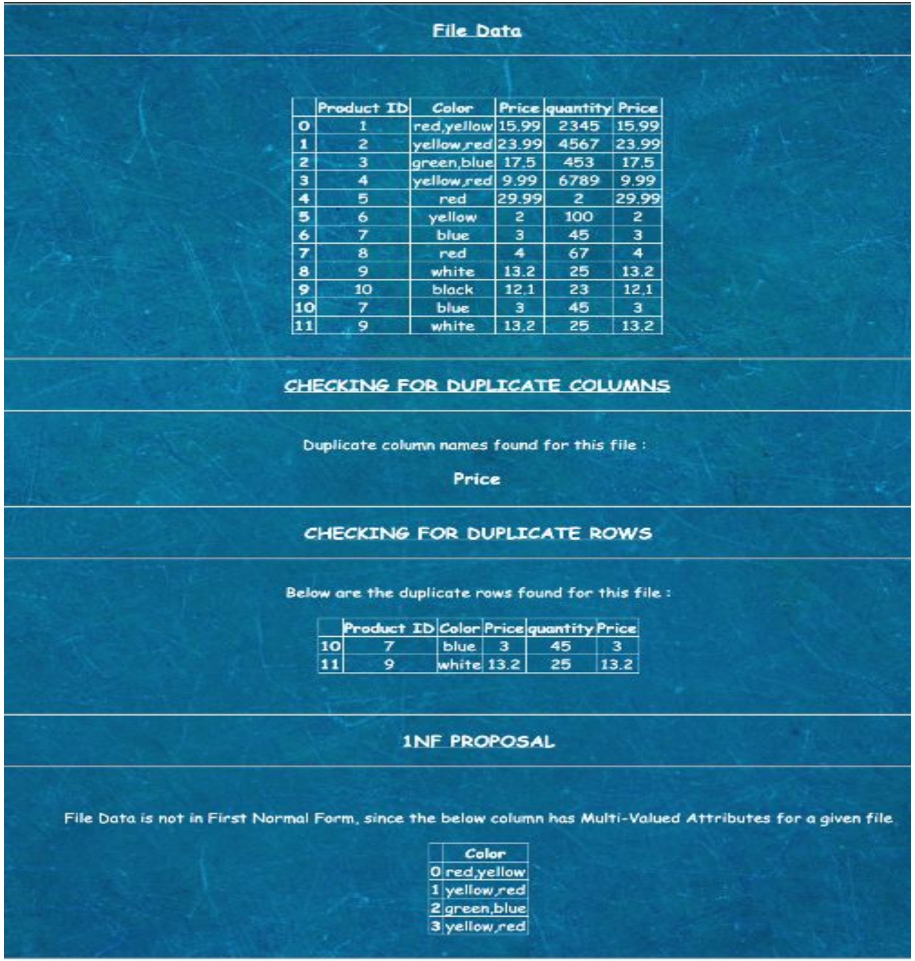
Choose File Product_1NF.csv

Product_1NF
Example File



Product ID	Color	Price	quantity	Price
1	red,yellow	15.99	2345	15.99
2	yellow,red	23.99	4567	23.99
3	green,blue	17.5	453	17.5
4	yellow,red	9.99	6789	9.99
5	red	29.99	2	29.99
6	yellow	2	100	2
7	blue	3	45	3
8	red	4	67	4
9	white	13.2	25	13.2
10	black	12.1	23	12.1
7	blue	3	45	3
9	white	13.2	25	13.2

Output for 1NF : File



File Data

	Product ID	Color	Price	quantity	Price
0	1	red,yellow	15.99	2345	15.99
1	2	yellow,red	23.99	4567	23.99
2	3	green,blue	17.5	453	17.5
3	4	yellow,red	9.99	6789	9.99
4	5	red	29.99	2	29.99
5	6	yellow	2	100	2
6	7	blue	3	45	3
7	8	red	4	67	4
8	9	white	13.2	25	13.2
9	10	black	12.1	23	12.1
10	7	blue	3	45	3
11	9	white	13.2	25	13.2

CHECKING FOR DUPLICATE COLUMNS

Duplicate column names found for this file :

Price

CHECKING FOR DUPLICATE ROWS

Below are the duplicate rows found for this file :

	Product ID	Color	Price	quantity	Price
10	7	blue	3	45	3
11	9	white	13.2	25	13.2

1NF PROPOSAL

File Data is not in First Normal Form, since the below column has Multi-Valued Attributes for a given file.

	Color
0	red,yellow
1	yellow,red
2	green,blue
3	yellow,red

Input for 2NF, 3NF, BCNF

Attributes

{beer,brewery,strength,city,region,warehouse,quantity}

Functional Dependencies

beer->brewery

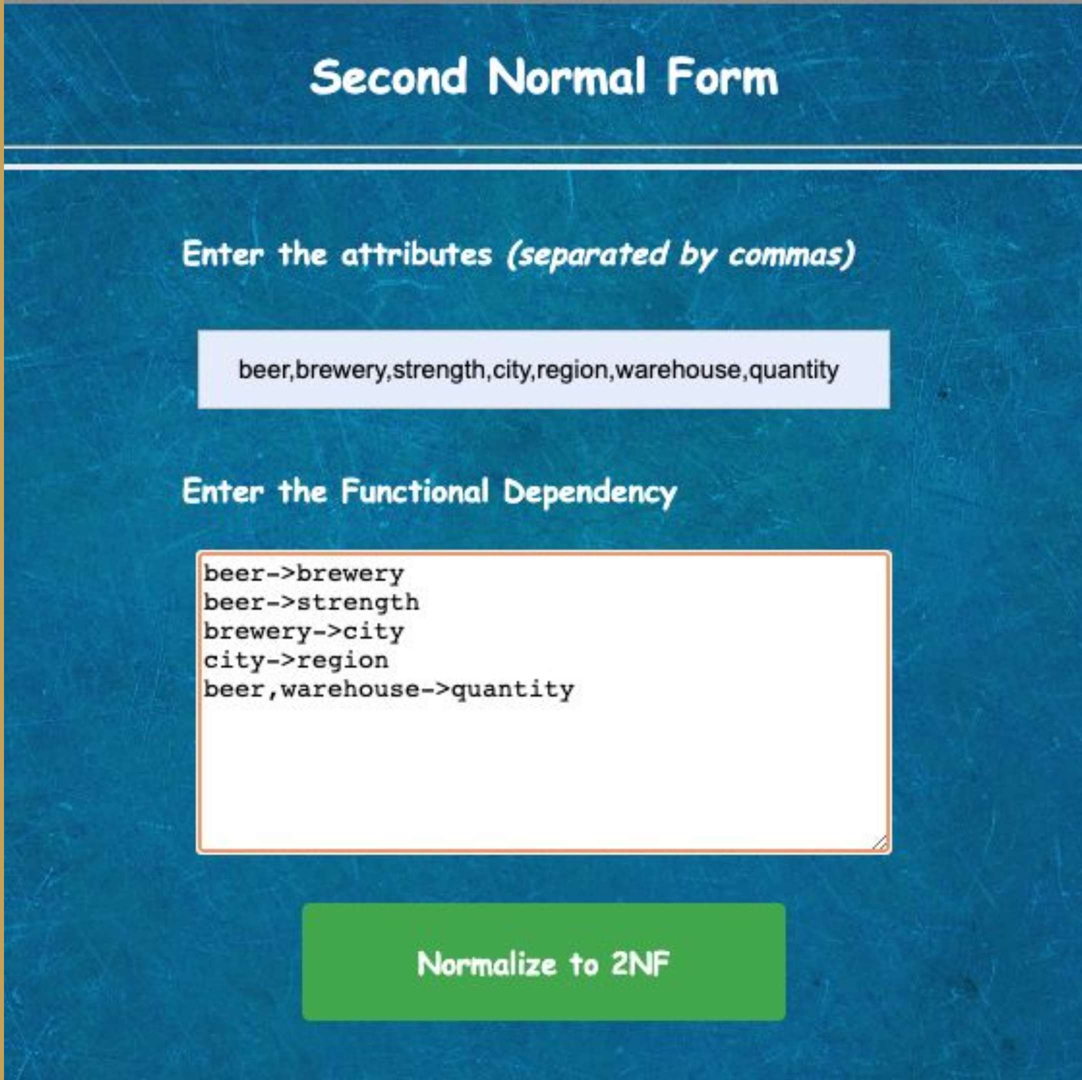
beer->strength

brewery->city

city->region

beer,warehouse->quantity

Second Normal Form (2NF)



Second Normal Form

Enter the attributes *(separated by commas)*

beer,brewery,strength,city,region,warehouse,quantity

Enter the Functional Dependency

```
beer->brewery
beer->strength
brewery->city
city->region
beer,warehouse->quantity
```

Output for 2NF

Merge the below functional dependencies to create a new relation

{'brewery', 'beer'} : ['beer'] -> ['brewery']

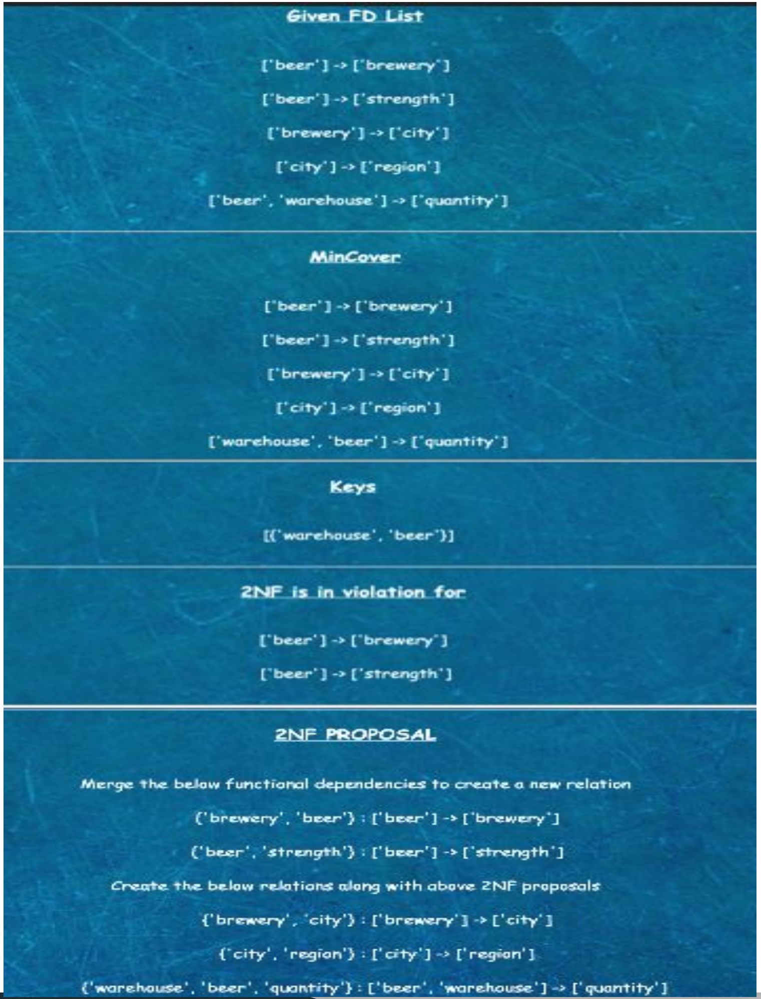
{'strength', 'beer'} : ['beer'] -> ['strength']

Create the below relations along with above 2NF proposals

{'brewery', 'city'} : ['brewery'] -> ['city']

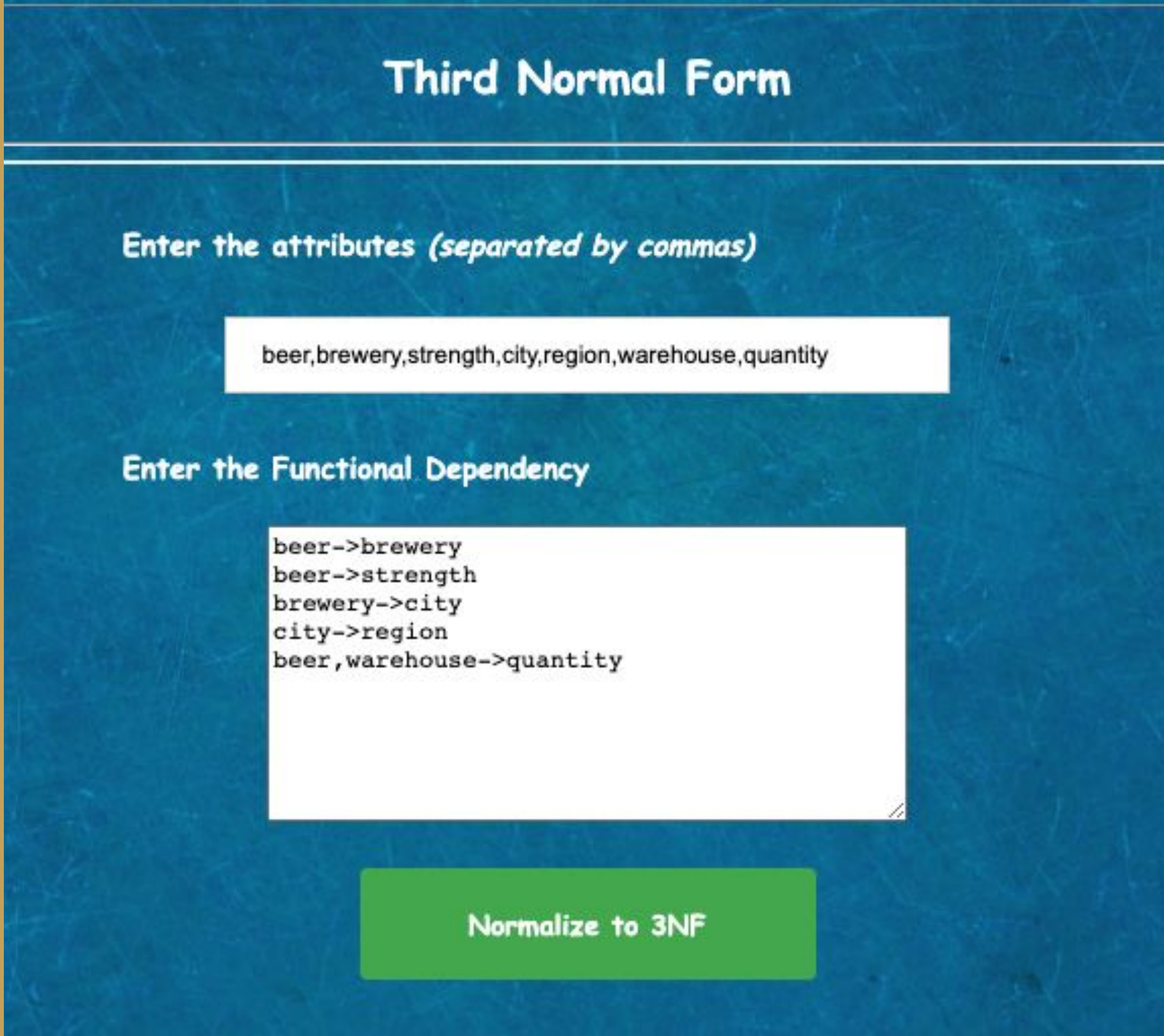
{'region', 'city'} : ['city'] -> ['region']

{'warehouse', 'quantity', 'beer'} : ['beer', 'warehouse'] -> ['quantity']



```
Given FD List

['beer'] -> ['brewery']
['beer'] -> ['strength']
['brewery'] -> ['city']
['city'] -> ['region']
['beer', 'warehouse'] -> ['quantity']

MinCover

['beer'] -> ['brewery']
['beer'] -> ['strength']
['brewery'] -> ['city']
['city'] -> ['region']
['warehouse', 'beer'] -> ['quantity']

Keys

[{'warehouse', 'beer'}]

2NF is in violation for

['beer'] -> ['brewery']
['beer'] -> ['strength']

2NF PROPOSAL

Merge the below functional dependencies to create a new relation
{'brewery', 'beer'} : ['beer'] -> ['brewery']
{'beer', 'strength'} : ['beer'] -> ['strength']
Create the below relations along with above 2NF proposals
{'brewery', 'city'} : ['brewery'] -> ['city']
{'city', 'region'} : ['city'] -> ['region']
{'warehouse', 'beer', 'quantity'} : ['beer', 'warehouse'] -> ['quantity']
```

Third Normal Form (3NF)



Third Normal Form

Enter the attributes *(separated by commas)*

beer,brewery,strength,city,region,warehouse,quantity

Enter the Functional Dependency

```
beer->brewery
beer->strength
brewery->city
city->region
beer,warehouse->quantity
```

Output for 3NF

{'brewery', 'beer'} ['beer'] -> ['brewery']

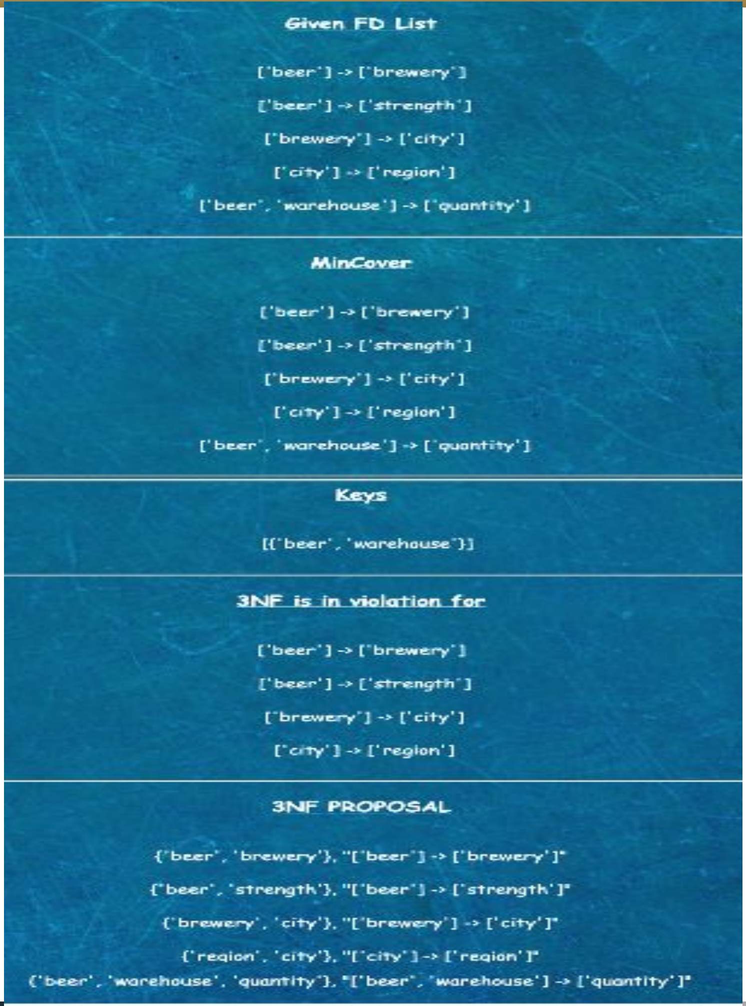
{'strength', 'beer'} ['beer'] -> ['strength']

{'brewery', 'city'} ['brewery'] -> ['city']

{'region', 'city'} ['city'] -> ['region']

{'warehouse', 'quantity', 'beer'}

['warehouse', 'beer'] -> ['quantity']



Given FD List

['beer'] -> ['brewery']

['beer'] -> ['strength']

['brewery'] -> ['city']

['city'] -> ['region']

['beer', 'warehouse'] -> ['quantity']

MinCover

['beer'] -> ['brewery']

['beer'] -> ['strength']

['brewery'] -> ['city']

['city'] -> ['region']

['beer', 'warehouse'] -> ['quantity']

Keys

[{'beer', 'warehouse'}]

3NF is in violation for

['beer'] -> ['brewery']

['beer'] -> ['strength']

['brewery'] -> ['city']

['city'] -> ['region']

3NF PROPOSAL

{'beer', 'brewery'}, "['beer'] -> ['brewery']"

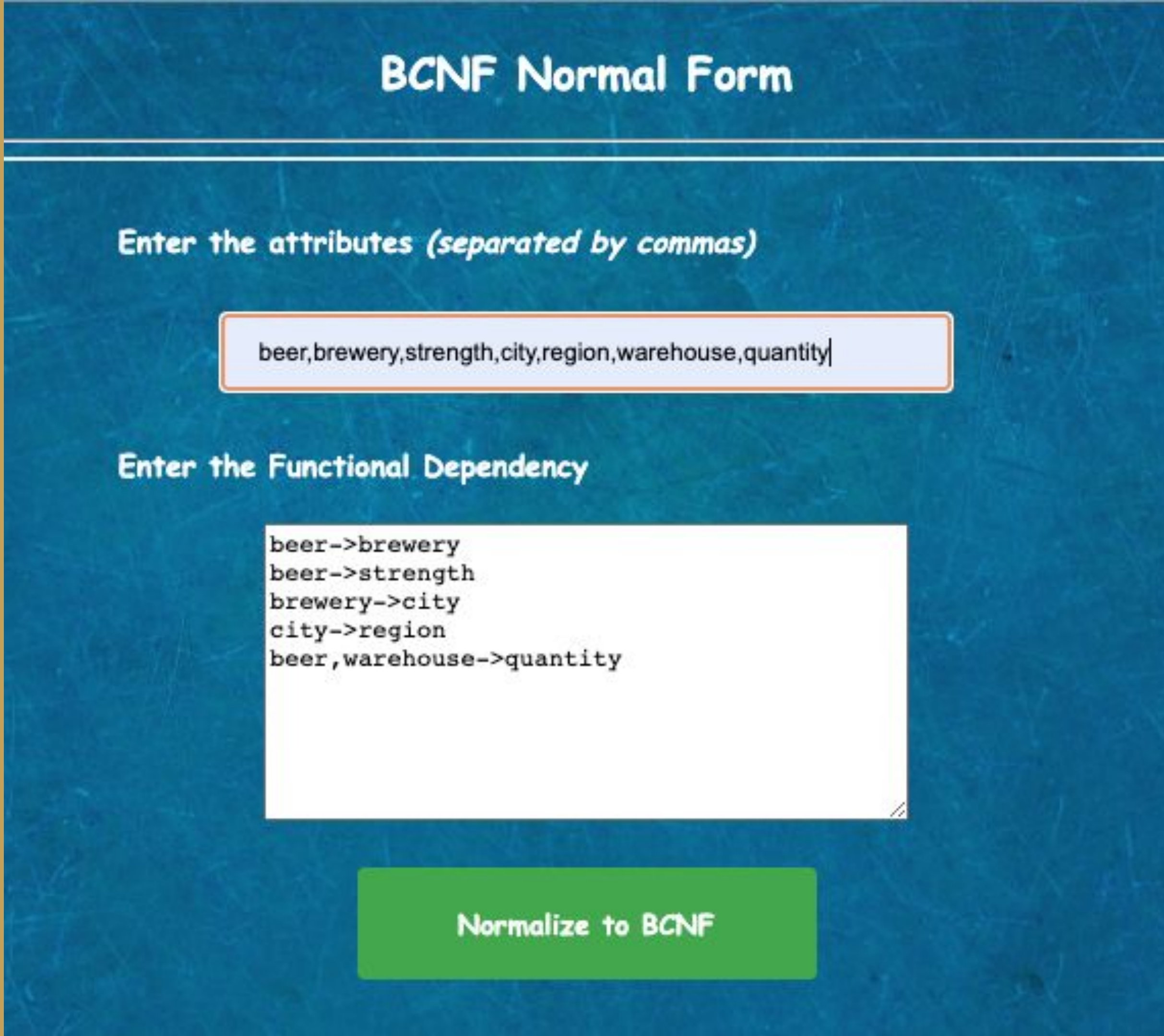
{'beer', 'strength'}, "['beer'] -> ['strength']"

{'brewery', 'city'}, "['brewery'] -> ['city']"

{'region', 'city'}, "['city'] -> ['region']"

{'beer', 'warehouse', 'quantity'}, "['beer', 'warehouse'] -> ['quantity']"

Boyce Codd Normal Form (BCNF)



BCNF Normal Form

Enter the attributes *(separated by commas)*

beer,brewery,strength,city,region,warehouse,quantity

Enter the Functional Dependency

```
beer->brewery
beer->strength
brewery->city
city->region
beer,warehouse->quantity
```

Output for BCNF

{'region', 'city'} ['city'] -> ['region']

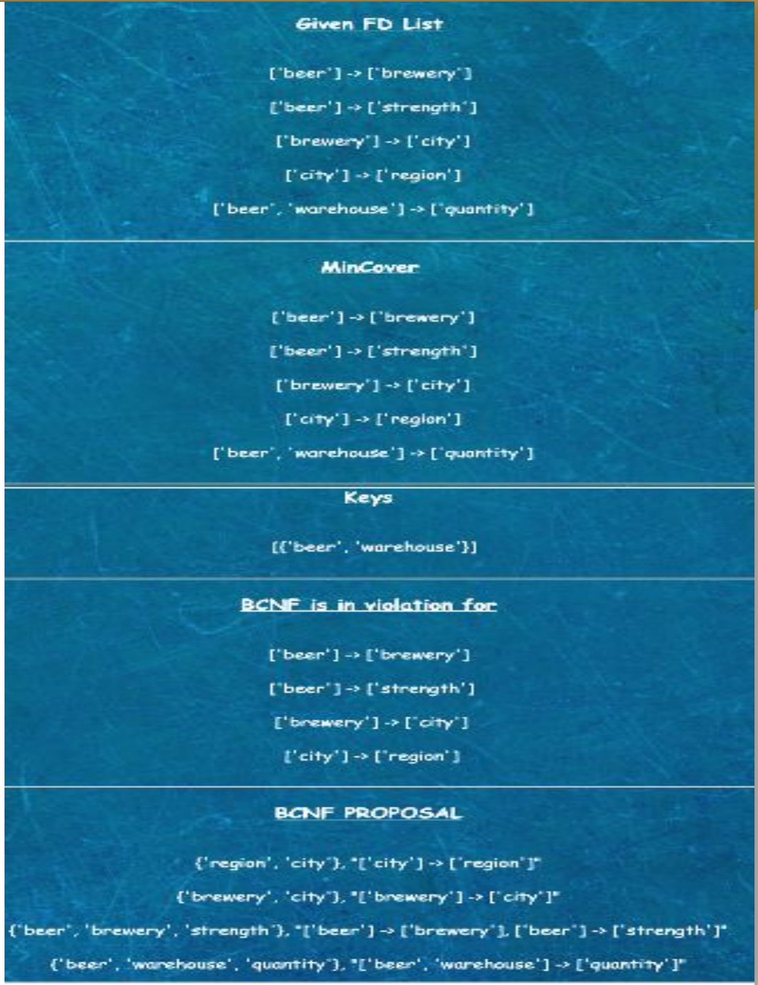
{'brewery', 'city'} ['brewery'] -> ['city']

{'strength', 'brewery', 'beer'}

['beer'] -> ['brewery'], ['beer'] -> ['strength']

{'warehouse', 'quantity', 'beer'}

['warehouse', 'beer'] -> ['quantity']



Given FD List

['beer'] -> ['brewery']

['beer'] -> ['strength']

['brewery'] -> ['city']

['city'] -> ['region']

['beer', 'warehouse'] -> ['quantity']

MinCover

['beer'] -> ['brewery']

['beer'] -> ['strength']

['brewery'] -> ['city']

['city'] -> ['region']

['beer', 'warehouse'] -> ['quantity']

Keys

[{'beer', 'warehouse'}]

BCNF is in violation for

['beer'] -> ['brewery']

['beer'] -> ['strength']

['brewery'] -> ['city']

['city'] -> ['region']

BCNF PROPOSAL

{'region', 'city'}, "['city'] -> ['region']"

{'brewery', 'city'}, "['brewery'] -> ['city']"

{'beer', 'brewery', 'strength'}, "['beer'] -> ['brewery'], ['beer'] -> ['strength']"

{'beer', 'warehouse', 'quantity'}, "['beer', 'warehouse'] -> ['quantity']"

Compare output against hypothesis

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result
TC01	Check the input for 1NF	Upload the data into tool to check for 1NF	There should be No repetition groups. Where there are users to be warned.	Resultant output from 1NF produced no repetitive groups. Where there are repetitive groups, warning messages will be displayed.
TC02	Check for any duplicate data present in the relation	Retrieve the table data and check if the data is not being duplicated in a relation by executing the duplicate check query.	There should not be any duplicate data in a given relation. Where there are, users to be warned.	Where there were duplicates, warning messages have been displayed.
TC03	Check the result if it satisfies minimal cover requirements	Enter the FD's, for the attributes to check if the result satisfies minimal cover.	There should be a minimal cover for the specified input.	Generated output produced at least one cover as expected per minimal cover algorithm.

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result
TC04	Check if the result satisfies 2NF requirements	A. Specify the Relational attributes B. Specify Functional dependencies C. Run the code to check the output.	The resultant code should never have any partial functional dependencies	Generated output matches
TC05	Check if the result satisfies 3NF properties	Compare the result with the business specified requirements.	1. There should not be any transitive dependencies. 2. There should not be any loss of information.	Implementation of 3NF algorithm resulted in: 1. Removal of Transitive dependencies 2. It ensured loss less decomposition.
TC06	Check if the result satisfies BCNF properties		All the dependencies other inclusive of non-prime attributes have to be removed	BCNF decomposition is as expected and resulted in removal of dependencies where determinants were not necessarily candidate keys.

We conclude that the designed normalization model for checking and proposing appropriate normalization up to BCNF meets the expectations and therefore meets the hypothesis stated before designing the normalization model.

Abnormal case explanation

- Attributes mentioned in FD which are not part of relation set- decomposition won't proceed further
- Attributes for functional dependency is not in singleton form
- Tool validates those scenario, abort normalization process and handle the error
- Error message: Functional dependency has attributes that are not in relation set

Static Regression

Since our project involves automation of normalization and we do not have any quantitative variables to calculate or predict, this is not applicable for this project

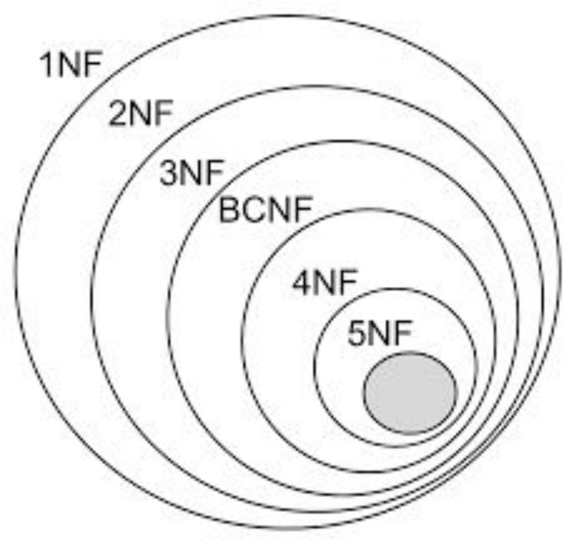
Discussion

- Normal forms generated till BCNF implemented using existing algorithms
- All the normal forms generated till BCNF have been validated and are supported by the tool
- System performs a complete decomposition where no extra attributes are added or removed

Discussion

- Our system handles only singleton values for attributes on the right hand side of the specified functional dependencies
- Extraneous dependencies - not being specified in the original relation set does not include the attributes for normalization
- Cyclic dependencies- Refrain from having them in case of many attributes in view of performance and accuracy

Conclusions & Recommendations



Summary & Conclusion

Only have a good relational database system is not enough

Normalization checker tool developed in Python can be used -

- ➢ To analyse existing ER models
- ➢ Understand how normalization works
- ➢ Check normalization for legacy databases
- ➢ Check normal forms for given relationships and functional dependencies
- ➢ Receive Proposals of normalization for 1NF, 2NF, 3NF and BCNF

Higher normal forms ≠ Highly efficient systems always.

So- Be wise in your design considerations

Recommendations for future studies

Increase the scope of testing the tool with more use cases

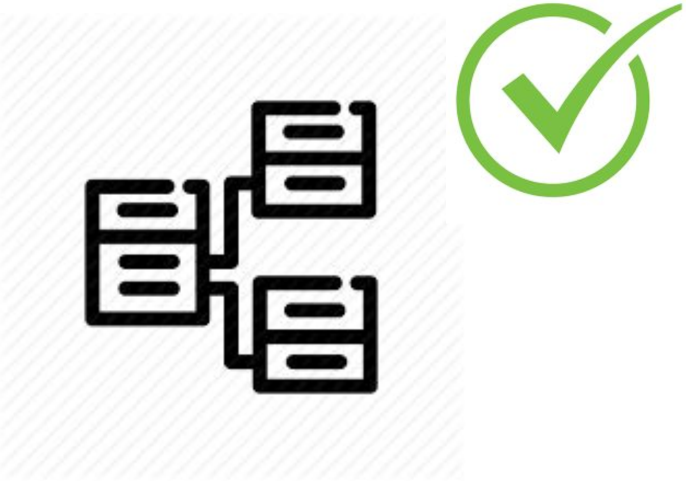
Handle right side split of functional dependency

Test for compatibility with other files, formats and databases

Develop functionality for higher normal forms - 4NF and 5NF

Develop a way to denormalize data if required to improve performance for a system in special scenarios

Demo of our tool



THANK YOU!