Introduction to Database

Part 1

What is Database?

- A database is a collection of related data, stored in an organised or logical manner.

- The data could relate to (for example) a firm's customers; each customer data is a record. Each record will store the same items of data, for example, customer name, address … etc., and these are called attributes.

- Storing data in a database allows us to access and manage the data. Examples of databases in real-life include:
  - Patient medical records
  - Supermarket inventory
  - Contact list

Old School – Library Card Catalog

Physical to Digital

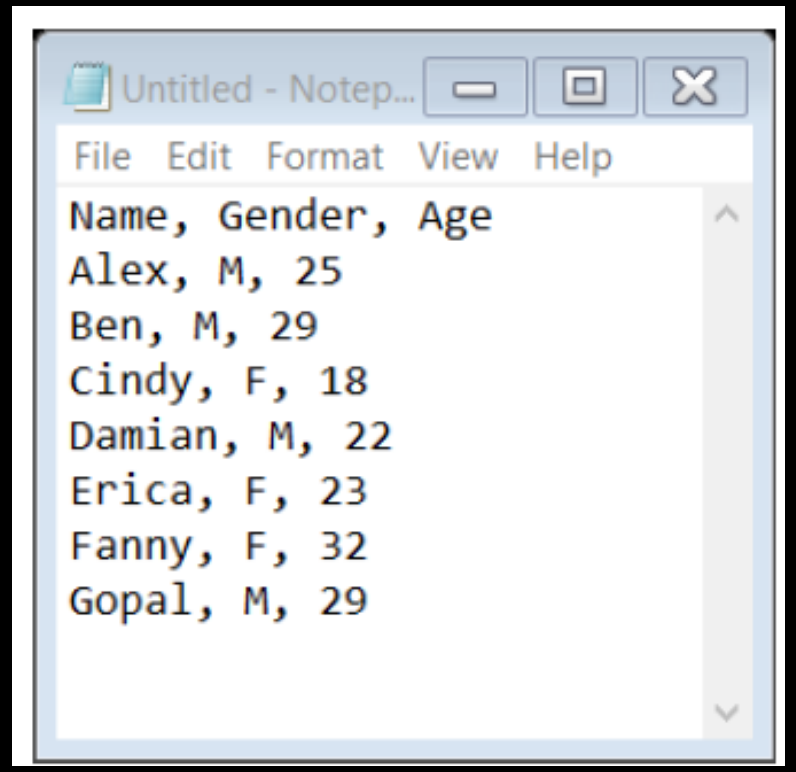
- Card Index System:
  - The cards would have a header line and the cards would then be filed in some determined order, for example, customer name filed in alphabetical order.
  - That was fine if we wanted to find a particular customer by name but what if we wanted to find the customer living at “13 The High Street” or the customer with the contact number ‘91234567’?
  - The major limitation of a manual card index is that the cards can only be held in one particular order and so can only be searched on one data item.
- Digital solution to store and manage large records

Flat File

- Using File-based approach (flat files) to manage data: e.g. text files, spread sheet

- Before database software, a database application would have been programmed and constructed using one or more flat files.

	A	B	C
1	**Name**	**Gender**	**Age**
2	Alex	M	25
3	Ben	M	29
4	Cindy	F	18
5	Damian	M	22
6	Erica	F	23
7	Fanny	F	32
8	Gopal	M	29
9			



Flat File - Limitations

- Record structure is fixed.
  - If we wanted to add a new field, it would require a considerable amount of redesign of the files and program code.
- No check on data integrity.
- There could be duplicated data in a flat file.

	A	B	C
1	**Name**	**Gender**	**Age**
2	Alex	M	25
3	Ben	M	29
4	Cindy	F	eighteen
5	Damian	M	22
6	Erica	F	23
7	Fanny	F	don't know
8	Damian	M	22
9			

Why relational database?

Table

A table is a two-dimensional representation of data stored in rows and columns. Tables are made up of records and fields.

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

A **record** is a complete set of data about a single item.
For this example, a record refers to the complete set of data of a particular student.

A **field/attribute** refers to one piece of data about a single item. For this example, the table has 4 fields – RegNo, Name, Gender and MobileNo.

Table Description

- A table description can be expressed as:
  - TableName (Attribute1, Attribute2, Attribute3, …)

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

STUDENT Table

STUDENT (RegNo, Name, Gender, MobileNo)

Relation

- In the relational database model, data is stored in relations and represented in the form of tuples (rows).
- A relational database is a collection of relational tables.
- A table is a relation if it fulfils the following conditions:
  - Values are atomic
  - Columns are of the same kind
  - Rows are unique (no duplicates)
  - The order of columns is insignificant
  - Each column must have a unique name

Record keys- Candidate Keys

A **candidate key** is defined as a minimal set of fields which can uniquely identify each record in a table.

It tells a particular record apart from another record.

A candidate key should never be NULL or empty.

There can be more than one candidate key for a table. A candidate key can also be a combination of more than one field.

Below is an example of a table `Student18S12`, showing data for students from civics class 18S12.

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

Candidate Key: {RegNo, Name}

Record keys – Primary Keys

A **primary key** is a candidate key that is most appropriate to become the main key for a table.

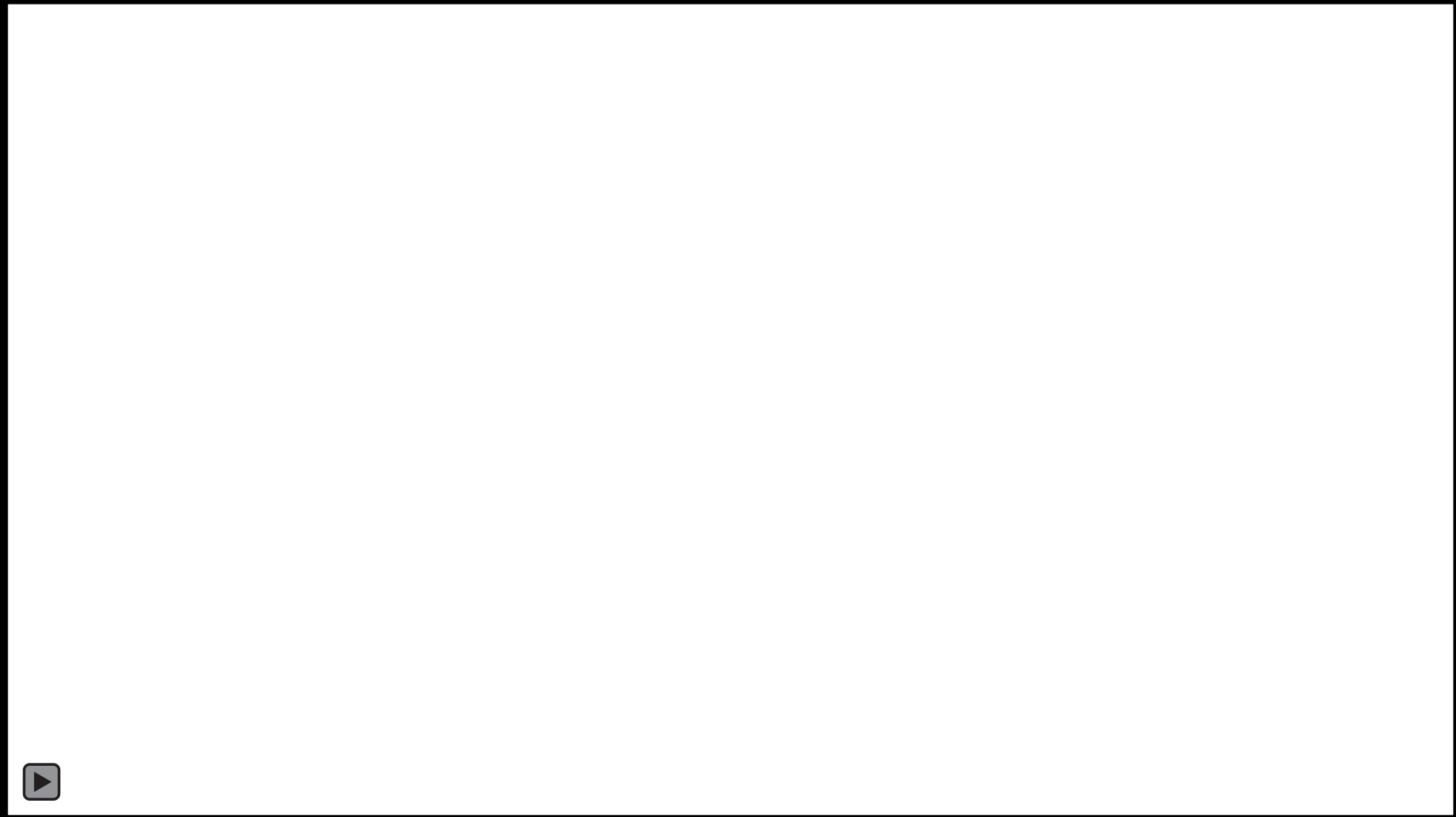
It uniquely identifies each record in a table and should not change over time.

That is, a primary key tells a particular record apart from another record.

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

Primary Key: {RegNo}

Candidate & Primary Keys



Record keys- Secondary Keys

Candidate keys that are not selected as primary key are known as **secondary keys**.

Candidate Key: {RegNo, Name}

Primary Key: {RegNo}

Secondary Key: {Name}

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

Record keys- Composite Keys

A **composite key** is a combination of two or more fields in a table that can be used to uniquely identify each record in a table.

Uniqueness is only guaranteed when the fields are combined. When taken individually, the fields do not guarantee uniqueness.

Consider the `Student` table. Now, a single field is not able to uniquely identify a record. A composite key composing 2 fields is needed uniquely identify a record.

Composite Key: {RegNo, CivicsClass}

RegNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Agnes	F	18S12
4	Aisha	F	18S12
5	Ajay	M	18S12
6	Alex	M	18S12
7	Alice	F	18S12
8	Amy	F	18S12
9	Andrew	M	18S12
10	Andy	M	18S12
1	Adam	M	18A10
2	Bala	M	18A10
3	Bee Lay	F	18A10
4	Ben	M	18A10
5	Boon Kiat	M	18A10
6	Boon Lim	M	18A10
7	Charles	M	18A10
8	Chee Seng	M	18A10
9	Cher Leng	F	18A10
10	Choo Tuan	M	18A10

Record keys- Foreign Keys

A **foreign key** is an attribute in one table that refers to the primary key in another table.

Another table, `ClassInfo`, as shown below, stores the information about each civics class.

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	BlkA-L2-3
18A10	Pauline Lee	BlkC-L1-4

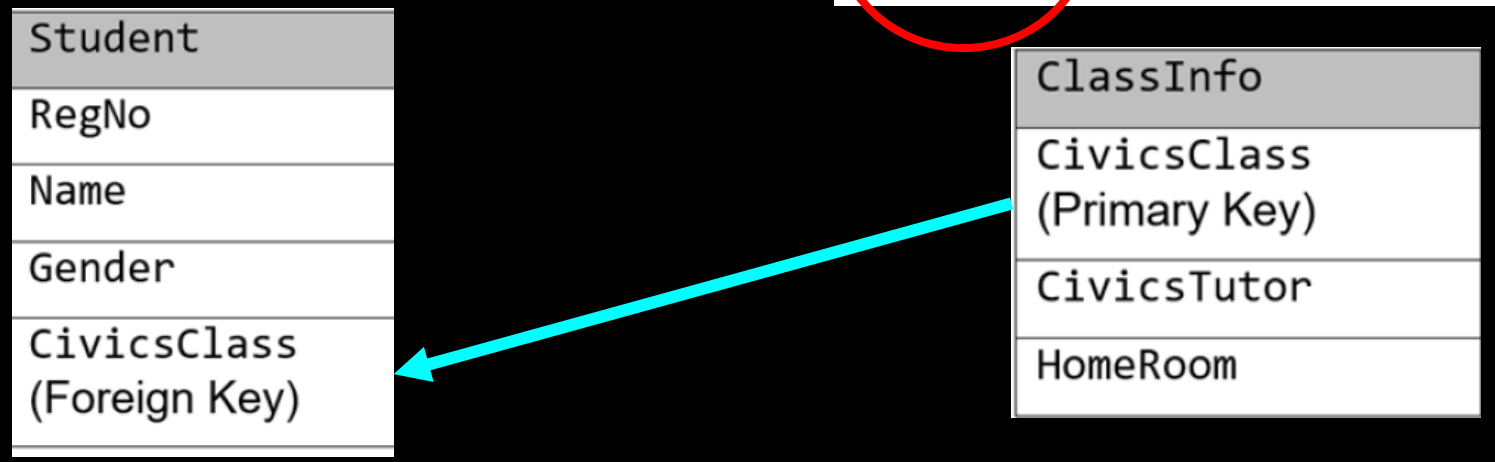
`CivicsClass` is chosen to be the primary key (PK) for table `ClassInfo`.

Record keys- Foreign Keys

RegNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Agnes	F	18S12
4	Aisha	F	18S12
5	Ajay	M	18S12
6	Alex	M	18S12
7	Alice	F	18S12
8	Amy	F	18S12
9	Andrew	M	18S12
10	Andy	M	18S12
1	Adam	M	18A10
2	Bala	M	18A10
3	Bee Lay	F	18A10
4	Ben	M	18A10
5	Boon Kiat	M	18A10
6	Boon Lim	M	18A10
7	Charles	M	18A10
8	Chee Seng	M	18A10
9	Cher Leng	F	18A10
10	Choo Tuan	M	18A10

Notice that the primary key in table `ClassInfo` (i.e., the `CivicsClass` field) is related or linked to the `CivicsClass` field in table `Student`.

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	BlkA-L2-3
18A10	Pauline Lee	BlkC-L1-4



This makes `CivicsClass` field in table `Student` a foreign key (FK).

Data Integrity

- Database management systems offer excellent validation support.
- The DBMS uses its data dictionary to perform validation checks on data entered into the database.
- The data dictionary is a depository of all the information about the basic database design and all objects which have been created for its use, for example, tables.
- Validation checks are set up at the table design stage and are effective every time a reference is made to that item of data including application programs which access data in that table.

Attributes	Type	Validation
RegNo	INTEGER	Unique, Not Null
Name	VARCHAR(32)	Not Null
Gender	CHAR(1)	Either ‘F’ or ‘M’
CivicsClass	CHAR(5)	Follows format YYCXX, where YY is the year, C the letter that represents the stream, XX the class code

Data Redundancy

- **Data redundancy** refers to the same data being stored more than once.

- Data for `CivicsClass` and `CivicsTutor` are repeated for students who are in the same civics class. This is data redundancy.

RegNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Agnes	F	18S12	Peter Lim
4	Aisha	F	18S12	Peter Lim
5	Ajay	M	18S12	Peter Lim
6	Alex	M	18S12	Peter Lim
7	Alice	F	18S12	Peter Lim
8	Amy	F	18S12	Peter Lim
9	Andrew	M	18S12	Peter Lim
10	Andy	M	18S12	Peter Lim

Data Redundancy

- Data Redundancy causes issues when inserting, updating and deleting data from the database.

RegNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Agnes	F	18S12	Peter Lim
4	Aisha	F	18S12	Peter Lim
5	Ajay	M	18S12	Peter Lim
6	Alex	M	18S12	Peter Lim
7	Alice	F	18S12	Peter Lim
8	Amy	F	18S12	Peter Lim
9	Andrew	M	18S12	Peter Lim
10	Andy	M	18S12	Peter Lim

Inserting data	A new student cannot be inserted unless a civics class and a civics tutor have been assigned.
Updating data	If Peter Lim decide to leave the college, all the records in the Student table would need to be updated. If we miss any record, it will lead to inconsistent data.
Deleting data	If all the records of the Student table are deleted, information on civics class and civics tutor will be lost.

Data Dependencies

- Hence, we need to normalise data.
- To do that, first, we need to understand data dependencies, specifically functional and transitive dependencies.

RegNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Agnes	F	18S12	Peter Lim
4	Aisha	F	18S12	Peter Lim
5	Ajay	M	18S12	Peter Lim
6	Alex	M	18S12	Peter Lim
7	Alice	F	18S12	Peter Lim
8	Amy	F	18S12	Peter Lim
9	Andrew	M	18S12	Peter Lim
10	Andy	M	18S12	Peter Lim

Functional Dependency

- Attribute Y is **functionally dependent** on attribute X (usually the primary key), if for every valid instance of X, the value of X uniquely determines the value of Y (X → Y).

- For example, suppose we have a `Student` table with the following attributes: `MatricNo`, `Name`, `Gender`, `CivicsClass` and `CivicsTutor`.

MatricNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Agnes	F	18S12	Peter Lim
4	Aisha	F	18S12	Peter Lim
5	Ajay	M	18S12	Peter Lim
6	Alex	M	18S12	Peter Lim
7	Alice	F	18S12	Peter Lim
8	Amy	F	18S12	Peter Lim
9	Andrew	M	18S12	Peter Lim
10	Andy	M	18S12	Peter Lim

`MatricNo` is a unique number assigned to every student in the college.
`MatricNo` uniquely identifies the `Name` because if we know the `MatricNo`, we can know the `Name` associated with it.
Therefore, we can say `Name` is functionally dependent on `MatricNo` (`MatricNo` → `Name`).

Transistive Dependency

- A functional dependency is said to be **transitive** if it is indirectly formed by two functional dependencies. Z is transitively dependent on X if Y is functionally dependent on X but X is not functionally dependent on Y, and Z is functionally dependent on Y. In other words, X → Z is a transitive dependency if the following hold true:
  - X → Y
  - Y does not → X
  - Y → Z

- For example, `CivicsClass` is functionally dependent on `MatricNo` (`MatricNo` → `CivicsClass`) but `MatricNo` is not functionally dependent on `CivicsClass`. On the other hand, `CivicsTutor` is functionally dependent on `CivicsClass` (`CivicsClass` → `CivicsTutor`). Therefore, `CivicsTutor` is transitively dependent on `MatricNo` (`MatricNo` → `CivicsTutor`).

Normalization

- Normalisation is the process of organising the tables in a database to reduce data redundancy and prevent inconsistent data.

- There are at least three normal forms associated with normalisation: first normal form (1NF), second normal form (2NF), and third normal form (3NF).

1NF

- ***First Normal Form***
- For a table to be in 1NF, all columns must be atomic.

- There can be no multi-valued columns – i.e., columns that would hold a collection such as an array or another table. In another words, the information in each column cannot be broken down further.

- Entities (objects of interest, e.g. person, item, place) do not contain repeated groups of attributes (e.g. TelephoneNo 1, TelephoneNo 2, etc.).

1NF

It is assumed that every civics class only has one civics tutor, and each CCA has only one teacher in-charge (IC). Why is this table not in 1NF?

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAInfo
1	Adam	M	18S12	Peter Lim	TR1	Tennis Teacher IC = Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir Teacher IC = Adeline Wong, Student Council Teacher IC = David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby Teacher IC = Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton Teacher IC = Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir Teacher IC = Adeline Wong, Chess Club Teacher IC = Edison Poh

This table is not in 1NF because the `CCAInfo` column contains multiple values.

1NF

In order for the table to be in 1NF:

- split `CCAInfo` into two single-value columns `CCAName` and `CCATeacherIC`.

- Notice that the students with `MatricNo` 2 and `MatricNo` 5 have multiple CCAs.
- We keep this information intact by splitting their records into multiple records, each corresponding to a different CCA.

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAName	CCATeacherIC
1	Adam	M	18S12	Peter Lim	TR1	Tennis	Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir	Adeline Wong
2	Adrian	M	18S12	Peter Lim	TR1	Student Council	David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby	Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton	Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir	Adeline Wong
5	Bee Lay	F	18A10	Pauline Lee	TR2	Chess Club	Edison Poh

The values for `CCAName` and `CCATeacherIC` are now atomic for each row.

The primary key for the above table will be the composite key formed by `MatricNo` and `CCAName`.

2NF

- ***Second Normal Form***
- For a table to be in 2NF, it must satisfy two conditions:

  1) The table should be in 1NF

  2) Every non-key attribute must be fully dependent on the entire primary key. This means no attribute can depend on part of the primary key only.

2NF

`Name, Gender, CivicsClass, CivicsTutor` and `HomeRoom` is dependent on only part of the primary key, `MatricNo`.

`CCATeacherIC` is dependent only on `CCAName`.

Hence, this table is not in 2NF since all attributes are dependent only on part of the primary key, and not solely dependent on the entire primary key.

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAName	CCATeacherIC
1	Adam	M	18S12	Peter Lim	TR1	Tennis	Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir	Adeline Wong
2	Adrian	M	18S12	Peter Lim	TR1	Student Council	David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby	Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton	Lilian Lim
5	Bee Lay	F	18A10	Pauline Lee	TR2	Choir	Adeline Wong
5	Bee Lay	F	18A10	Pauline Lee	TR2	Chess Club	Edison Poh

2NF

To change it to 2NF, we decompose the table into three smaller tables:

Student

Matric No	Name	Gender	CivicsClass	CivicsTutor	HomeRoom
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline Lee	TR2

StudentCCA

MatricNo	CCAName
1	Tennis
2	Choir
2	Student Council
3	Rugby
4	Badminton
5	Choir
5	Chess Club

CCAInfo

CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
Student Council	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Chess Club	Edison Poh

The primary key for table `Student` will be `MatricNo`.
`MatricNo` and `CCAName` will form a composite key for table `StudentCCA`.
`CCAName` will be the primary key for table `CCAInfo`.

3NF

- ***Third Normal Form***
- For a table to be in 3NF, it must satisfy two conditions:

  1) The table should be in 2NF

  2) The table should not have transitive dependencies

- Looking at table Student, `CivicsTutor` and `HomeRoom` are dependent on `CivicsClass` and `CivicsClass` is dependent on `MatricNo`. Therefore, `CivicsTutor` and `HomeRoom` are transitively dependent on `MatricNo`.

Student

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline Lee	TR2

3NF

To remove the transitive dependency, we decompose the Student table as follows:

Student

MatricNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Adam	M	18A10
4	Bala	M	18A10
5	Bee Lay	F	18A10

Civics

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	TR1
18A10	Pauline Lee	TR2

`MatricNo` remains the primary key for table Student and `CivicsClass` will be the primary key of the newly formed table Civics.

3NF

- The final design after normalisation is represented below:

Student

MatricNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Adam	M	18A10
4	Bala	M	18A10
5	Bee Lay	F	18A10

Civics

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	TR1
18A10	Pauline Lee	TR2

StudentCCA

MatricNo	CCAName
1	Tennis
2	Choir
2	Student Council
3	Rugby
4	Badminton
5	Choir
5	Chess Club

Student (MatricNo, Name, Gender, CivicsClass)

Civics (CivicsClass, CivicsTutor, HomeRoom)

StudentCCA (MatricNo, CCAName)

CCAInfo (CCAName, CCATeacherIC)

CCAInfo

CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
Student Council	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Chess Club	Edison Poh

The primary key for each table is indicated by underlining one or more attributes. Foreigh keys are indicated by using a dashed underline.