





# Database Fundamentals

Session 1



# Session Outline

Introduction to Databases

Relational Databases

Keys

Normalisation

NoSQL

Other Types of Database

Recap



# Learning Objectives



- Explain the concepts and uses of a **relational database management system**
- Identify the different types of **key** in a RDBMS
- Understand the principles of **normalisation** on a relational database

# Introduction to Databases



**A Database Management  
System (DBMS) provides...**

**...an efficient, reliable, convenient and safe  
multi-user storage and access to massive  
amounts of persistent data.**

# A Database is ...

- Massive
- Persistent
- Safe
- Multi-user
- Convenient
- Efficient
- Reliable





# Key Concepts

→ Data Model

→ Schema vs Data

→ Data Definition Language (DDL)

→ Data Manipulation Language (DML)



# Key People

→ DBMS Implementer

→ Database Designer

→ Database Application Developer

→ Database Administrator

multiverse

# Relational Databases

## **Relational Database Management System (RDBMS):**

**A database where data is organised into  
tables with defined relationships between  
them**



Efficiency

Redundancy

Update Issues

Deletion

Standardisation

Increased speed and storage



Efficiency  
Redundancy  
Update Issues  
Deletion  
Standardisation

Less redundant and duplicated  
data



Efficiency  
Redundancy  
Update Issues  
Deletion  
Standardisation

Fewer problems updating data



Efficiency  
Redundancy  
Update Issues  
Deletion  
Standardisation

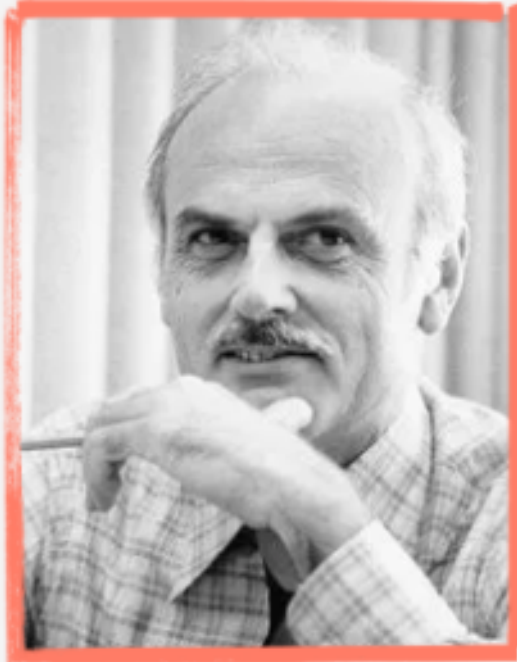
Less chance of deleting  
important data

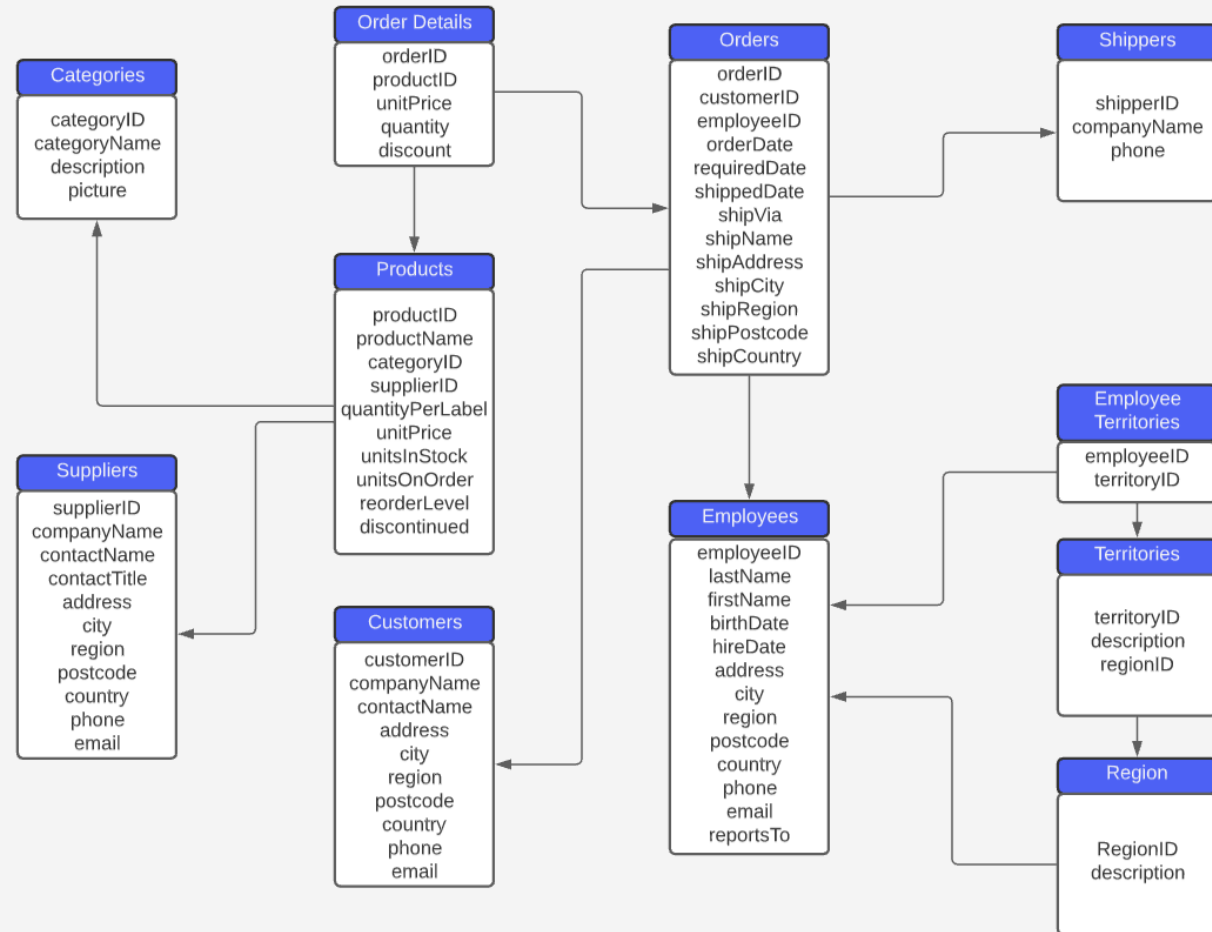





Efficiency  
Redundancy  
Update Issues  
Deletion  
Standardisation

Design follows consistent  
principles









SHIPPERID	COMPANYNAME	PHONE
234	Big Shippers	01302858888
235	DeliverForce	01302823444
236	ShipUK	01709282327
237	WeShip	01302339188

shippers\_table

## The Alternative...



COACH ID	COACH NAME	NO. COHORTS	ASSISTANT	APPRENTICE	COHORT	AGE	LINE
1	Steph	5	Ashray	Adam	Standard	20	Eva
1	Steph	5	Ashray	Natasha	Outliers	19	Bori
2	Ben	2	Ashray	Kingsley	Patch	23	Mila
3	Tony	3	John	Grace	Sprite	21	Mila
1	Steph	5	Ashray	Greta	Outliers	22	Isabe
4	Bruce	5	John	Alison	Movers	20	Henn



## Difficult to insert new data

COACH ID	COACH NAME	NO. COHORTS	ASSISTANT	APPRENTICE	COHORT	AGE	LINE
1	Steph	5	Ashray	Adam	Standard	20	Eva
1	Steph	5	Ashray	Natasha	Outliers	19	Bori
2	Ben	2	Ashray	Kingsley	Patch	23	Mila
3	Tony	3	John	Grace	Sprite	21	Mila
1	Steph	5	Ashray	Greta	Outliers	22	Isabe
4	Bruce	5	John	Alison	Movers	20	Henn

Cannot modify existing data

COACH ID	COACH NAME	NO. COHORTS	ASSISTANT	APPRENTICE	COHORT	AGE	LINE
1	Steph	5	Ashray	Adam	Standard	20	Eva
1	Steph	5	Ashray	Natasha	Outliers	19	Bori
2	Ben	2	Ashray	Kingsley	Patch	23	Mila
3	Tony	3	John	Grace	Sprite	21	Mila
1	Steph	5	Ashray	Greta	Outliers	22	Isabe
4	Bruce	5	John	Alison	Movers	20	Henn



## Cannot delete information

COACH ID	COACH NAME	NO. COHORTS	ASSISTANT	APPRENTICE	COHORT	AGE	LINE
1	Steph	5	Ashray	Adam	Standard	20	Eva
1	Steph	5	Ashray	Natasha	Outliers	19	Bori
2	Ben	2	Ashray	Kingsley	Patch	23	Mila
3	Tony	3	John	Grace	Sprite	21	Mila
1	Steph	5	Ashray	Greta	Outliers	22	Isabe
4	Bruce	5	John	Alison	Movers	20	Henn

multiverse

Keys

# Primary Key

A unique identifier for each **row in a table**

COACH ID	COACH NAME	NO. COHORTS	ASSISTANT
1	Steph	5	Ashray
2	Ben	2	Ashray

coach\_table

# Primary keys...

...cannot be NULL

...must be unique

...should rarely be changed

...given a new value when a new record is created

# Foreign Key

A field in one table that is a primary key in **another table** . These keys enable relationships between tables and allow them to be joined

APP_ID	NAME	COACH_ID
16	Adam	1
23	Natasha	1
20	Kingsley	2

apprentice\_table

COACH_ID	NAME	ASSISTANT
1	Steph	Ashray
2	Ben	Ashray

coach\_table

# Composite Key

Two or more columns together acting as a **primary key**

APPRENTICE_ID	MODULE_ID	GRADE	COACH_ID
1	1	99	1
1	2	75	1
2	1	80	2
2	2	78	2



assignment\_table



## Activity

In groups discuss how this [table](#) can be redesigned into something more useable.

- How many tables would you create?
- What are the primary/foreign keys?
- What information would be placed in each?



coach_table	program_table	cohort_table	apprentice_table
<ul style="list-style-type: none"><li>▪ coach_id</li><li>▪ coach_name</li><li>▪ no. cohorts</li><li>▪ apprenticeship</li></ul>	<ul style="list-style-type: none"><li>▪ apprenticeship</li><li>▪ TA</li></ul>	<ul style="list-style-type: none"><li>▪ coach_id</li></ul>	<ul style="list-style-type: none"><li>▪ app_name</li><li>▪ age</li><li>▪ LM</li><li>▪ apprenticeship</li><li>▪ company_name</li><li>▪ cohort</li></ul>





multiverse

# Normalisation

**Normalisation is the process of structuring a database to reduce data redundancy and improve data integrity.**



ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING	RATING
101	Aero	Nestle	Coop/Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	One Sixty	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad



FIRST NORMAL FORM	SECOND NORMAL FORM	THIRD NORMAL FORM
<ul style="list-style-type: none"><li>→ Each cell only contains one data point</li><li>→ Each column contains only one data subject</li><li>→ Columns should each have a unique name</li><li>→ Identification should not rely on the way the data is sorted</li></ul>	<ul style="list-style-type: none"><li>→ Compliant with 1NF</li><li>→ Each table contains relevant data</li><li>→ There are no partial dependencies</li></ul>	<ul style="list-style-type: none"><li>→ Compliant with 2NF</li><li>→ There are no transitive dependencies</li></ul>

# First Normal Form

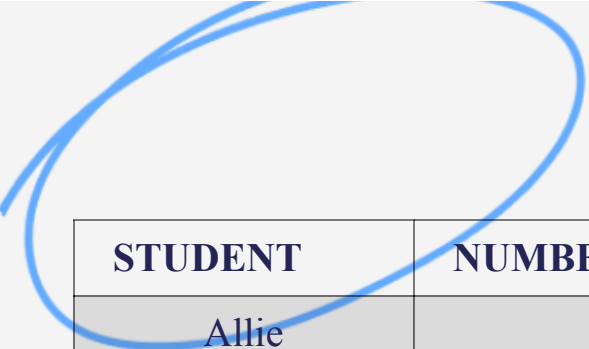
Each cell only contains one data point

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING	RATING
101	Aero	Nestle	Coop/Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	One Sixty	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad

# First Normal Form

Each cell only contains one data point

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING	RATING
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	One Sixty	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad



STUDENT	NUMBER
Allie	07551502613
Bernard	07723871451, 07464998651
Charlotte	07818771127



STUDENT	NUMBER 1	NUMBER 2
Allie	07551502613	
Bernard	07723871451	07464998651
Charlotte	07818771127	

STUDENT	NUMBER
Allie	07551502613
Bernard	07723871451
Bernard	07464998651
Charlotte	07818771127

# First Normal Form

Each column contains only one data type

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING	RATING
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	One Sixty	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad





STUDENT	MISC
Allie	Spaghetti
Bernard	Toyota
Charlotte	Blue

STUDENT	FAV FOOD
Allie	Spaghetti
Bernard	
Charlotte	



# First Normal Form

Each column contains only one data type

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING	RATING
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	1.60	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad

# First Normal Form

Columns should each have a unique name

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING	RATING
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	1.6	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad

# First Normal Form



Columns should each have a unique name

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING_NUM	RATING_DESC
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	1.6	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad

# First Normal Form

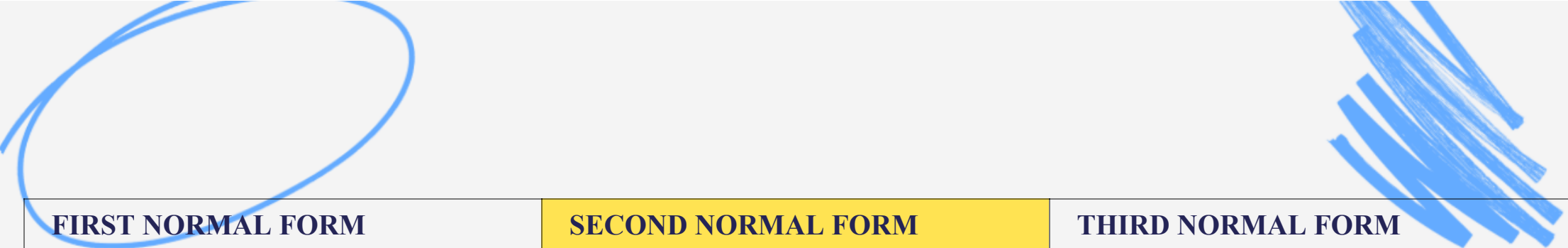
Identification should not rely on the way data is sorted

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING_NUM	RATING_DESC
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	1.6	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad



STUDENT	FAVOURITE SOUP
Allie	Tomato
	Mushroom
Charlotte	Pea

STUDENT	FAVOURITE SOUP
Allie	Tomato
Allie	Mushroom
Charlotte	Pea



FIRST NORMAL FORM	SECOND NORMAL FORM	THIRD NORMAL FORM
<ul style="list-style-type: none"><li>→ Each cell only contains one data point</li><li>→ Each column contains only one data subject</li><li>→ Columns should each have a unique name</li><li>→ Identification should not rely on the way the data is sorted</li></ul>	<ul style="list-style-type: none"><li>→ Compliant with 1NF</li><li>→ Each table contains relevant data</li><li>→ There are no partial dependencies</li></ul>	<ul style="list-style-type: none"><li>→ Compliant with 2NF</li><li>→ There are no transitive dependencies</li></ul>

# Second Normal Form

Each table contains relevant data

ID	BRAND	COMPANY	SUPERMARKET	COUNTRY	PRICE	RATING_NUM	RATING_DESC
101	Aero	Nestle	Coop	UK	1.70	10	Excellent
101	Aero	Nestle	Tesco	UK	1.70	10	Excellent
101	Aero	Nestle	Sainsburys	UK	1.6	10	Excellent
102	Bounty	Mars	Walmart	USA	1.30	2	Bad
102	Bounty	Mars	Tesco	UK	1.20	2	Bad
102	Bounty	Mars	Sainsburys	UK	1.10	2	Bad



# Second Normal Form

Each table contains relevant data

SUPERMARKET	PRICE	COUNTRY
Coop	1.70	UK
Tesco	1.60	UK
Sainsburys	1.60	UK
Walmart	1.30	USA
Tesco	1.20	UK
Sainsburys	1.10	UK

price\_table

ID	BRAND	COMPANY	RATING_NUM	RATING
101	Aero	Nestle	10	Excellent
102	Bounty	Mars	2	Bad

chocolate\_table

# Second Normal Form



SUPERMARKET	PRICE	COUNTRY
Coop	1.70	UK
Tesco	1.60	UK
Sainsburys	1.60	UK
Walmart	1.30	USA
Tesco	1.20	UK
Sainsburys	1.10	UK

price\_table

ID	BRAND	COMPANY	RATING_NUM	RATING
101	Aero	Nestle	10	Excellent
102	Bounty	Mars	2	Bad

chocolate\_table



# Second Normal Form

Compliant with 1NF

SUPERMARKET	PRICE	COUNTRY
Coop	1.70	UK
Tesco	1.60	UK
Sainsburys	1.60	UK
Walmart	1.30	USA
Tesco	1.20	UK
Sainsburys	1.10	UK

price\_table

ID	BRAND	COMPANY	RATING_NUM	RATING
101	Aero	Nestle	10	Excellent
102	Bounty	Mars	2	Bad

chocolate\_table

# Second Normal Form

Compliant with 1NF

SUPERMARKET	PRICE	COUNTRY
Coop	1.70	UK
Tesco	1.60	UK
Sainsburys	1.60	UK
Walmart	1.30	USA
Tesco	1.20	UK
Sainsburys	1.10	UK

price\_table

# Second Normal Form

Compliant with 1NF

CHOCOLATE_ID	SUPERMARKET	PRICE	COUNTRY
101	Coop	1.70	UK
101	Tesco	1.60	UK
101	Sainsburys	1.60	UK
102	Walmart	1.30	USA
102	Tesco	1.20	UK
102	Sainsburys	1.10	UK

price\_table

# Second Normal Form

There are no partial dependencies

CHOCOLATE_ID	SUPERMARKET	PRICE	COUNTRY
101	Coop	1.70	UK
101	Tesco	1.60	UK
101	Sainsburys	1.60	UK
102	Walmart	1.30	USA
102	Tesco	1.20	UK
102	Sainsburys	1.10	UK

price\_table

**Partial dependencies occur when a table with a composite key has a field which is only dependent on part of it**

# Second Normal Form

There are no partial dependencies

CHOCOLATE_ID	SUPERMARKET	PRICE	COUNTRY
101	Coop	1.70	UK
101	Tesco	1.60	UK
101	Sainsburys	1.60	UK
102	Walmart	1.30	USA
102	Tesco	1.20	UK
102	Sainsburys	1.10	UK

price\_table



# Second Normal Form

# Second Normal Form

There are no partial dependencies

CHOCOLATE_ID	SUPERMARKET	PRICE
101	Coop	1.70
101	Tesco	1.60
101	Sainsburys	1.60
102	Walmart	1.30
102	Tesco	1.20
102	Sainsburys	1.10

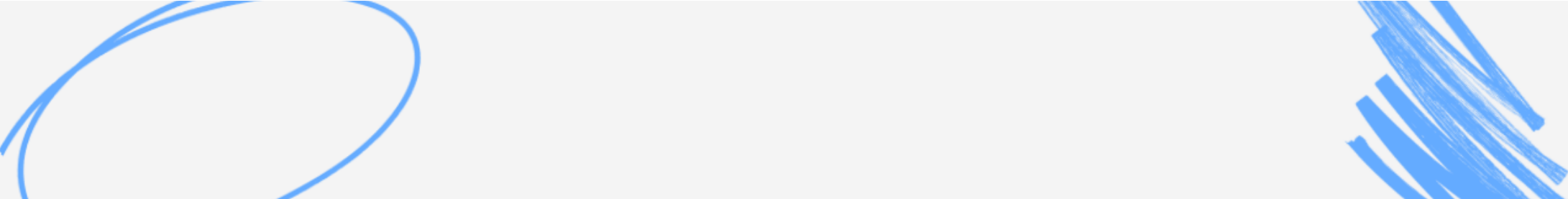
price\_table

# Second Normal Form

There are no partial dependencies

SUPERMARKET	COUNTRY
Tesco	UK
Walmart	USA

supermarket\_table



FIRST NORMAL FORM	SECOND NORMAL FORM	THIRD NORMAL FORM
<ul style="list-style-type: none"><li>→ Each cell only contains one data point</li><li>→ Each column contains only one data subject</li><li>→ Columns should each have a unique name</li><li>→ Identification should not rely on the way the data is sorted</li></ul>	<ul style="list-style-type: none"><li>→ Compliant with 1NF</li><li>→ Each table contains relevant data</li><li>→ There are no partial dependencies</li></ul>	<ul style="list-style-type: none"><li>→ Compliant with 2NF</li><li>→ There are no transitive dependencies</li></ul>

# Third Normal Form

There are no transitive dependencies

ID	BRAND	COMPANY	RATING_NUM	RATING_DESC
101	Aero	Nestle	10	Excellent
102	Bounty	Mars	2	Bad

chocolate\_table

**Transitive dependencies occur when  
a field can be inferred from another  
field that **not the primary key****

# Third Normal Form

There are no transitive dependencies

ID	BRAND	COMPANY	RATING_NUM
101	Aero	Nestle	10
102	Bounty	Mars	2

chocolate\_table

# Third Normal Form

There are no transitive dependencies

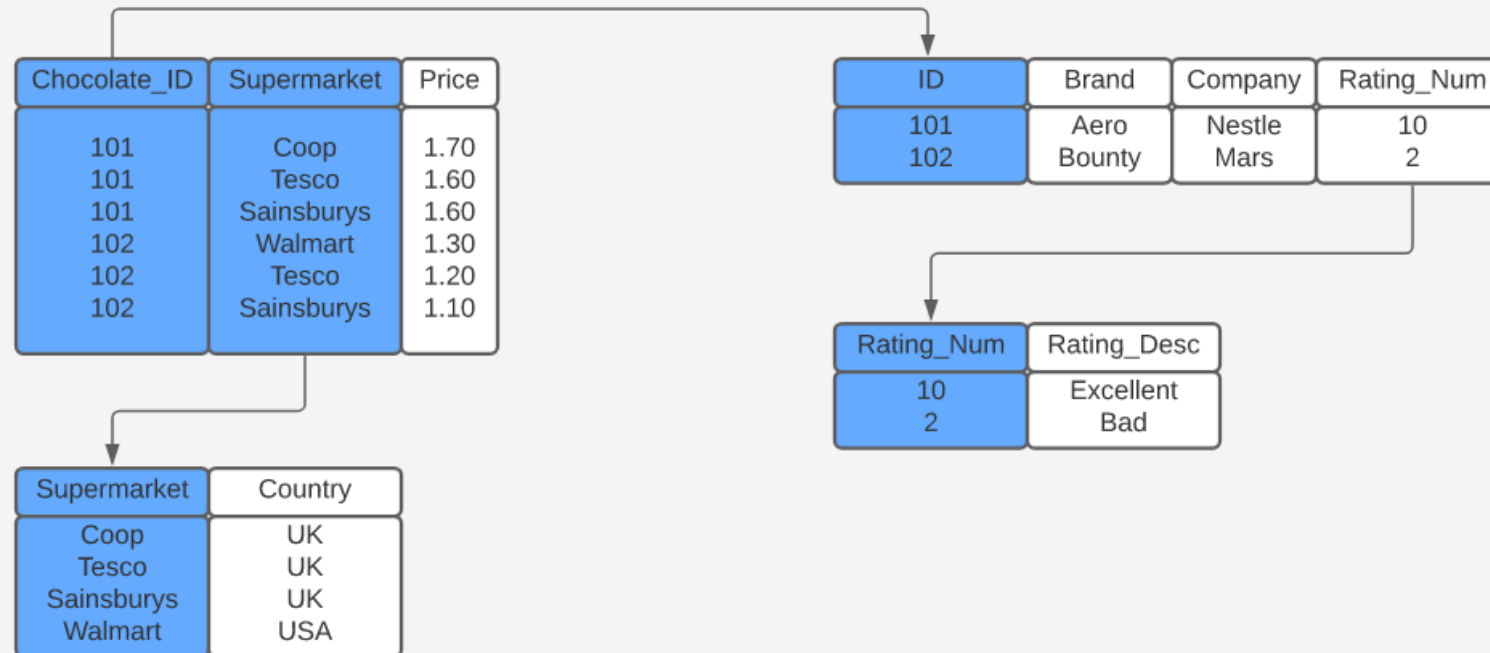
RATING_NUM	RATING_DESC
10	Excellent
2	Bad

rating\_table



# Third Normal Form

Compliant with 2NF



# Partial vs Transitive Dependency



Partial dependency occurs when a table has a **composite key** and a field is dependent on **one part of it**

ITEM_ID	VENDOR	PRICE	CITY
01	Tesco	1.70	London

City partially dependent on composite key

Transitive Dependency occurs when a field can be inferred from another field that is **not the primary key**

COUNTY	POPULATION	CATEGORY
Essex	1432000	large

Category can be inferred from population



Understanding how databases are designed, and why they are designed this way makes the SQL - especially joins - a lot easier to understand!



Understanding how databases are designed, and why they are designed this way makes the SQL - especially joins - a lot easier to understand!

The key is to consider the relationships and then apply normal form



Understanding how databases are designed, and why they are designed this way makes the SQL - especially joins - a lot easier to understand!

The key is to consider the relationships and then apply normal form

You most likely will never need to design a full on database - that's a very specific career progression. However, understanding the principles that went into the design of the databases you work with will help your understanding of them.



## Activity

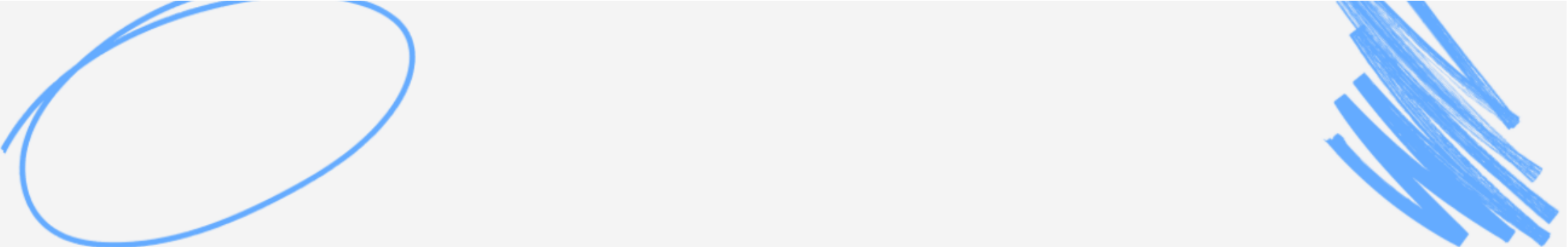
Convert this [table](#) step by step so it is in Third Normal Form

Also discuss:

What are the main benefits of Data Normalisation?

How does querying change because of First Normal Form?





TRANSACTION_TABLE	PRODUCT_TABLE	STORE_TABLE	COUNTY_TABLE
<ul style="list-style-type: none"><li>▪ id</li><li>▪ date</li><li>▪ item_no</li><li>▪ store_id</li></ul>	<ul style="list-style-type: none"><li>▪ item_no</li><li>▪ item_description</li><li>▪ case_cost</li><li>▪ proof</li></ul>	<ul style="list-style-type: none"><li>▪ store_id</li><li>▪ store</li><li>▪ store_address</li><li>▪ county</li></ul>	<ul style="list-style-type: none"><li>▪ county</li><li>▪ population</li></ul>

multiverse

NoSQL

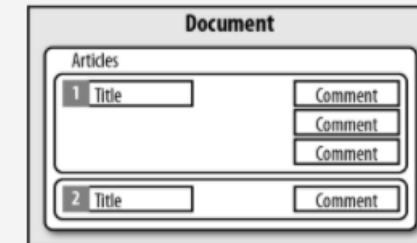
**NoSQL ("Not only SQL") is an alternative to traditional relational databases that can accomodate a wide variety of data models.**

# Document Orientated Database

A common NoSQL database where all instances of an object is stored in one document as opposed to spread across multiple tables. To access the data you reference the internal structure.

Documents are organised into collections (similar to RDBMS tables)

E.g. XML, JSON



## JSON

```
{  
  contact{  
    "firstname":"Bob",  
    "lastname":"Smith",  
    "address":"5 Oak St",  
    "number":"07464998651"  
  }  
}
```

## XML

```
<contact>  
  <firstname>Bob</firstname>  
  <lastname>Smith</lastname>  
  <adress>5 Oak St</address>  
  <number>07464998651</hobby>  
</contact>
```

# Key Value Database

The simplest type of NoSQL database, where data (structured or unstructured) is mapped to a key and stored in one location.

Data is extracted by referencing the key.

Wrapping several keys in a JSON format creates a document

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

# Columnar Database

Data is stored in columns instead of rows. SQL can be used to extract data quickly as it will go down the columns for information instead of scanning each row.

A column orientated database applies the row key to each item in a column, allowing it to precisely retrieve information from a select group of columns.

Row-oriented (1)

	name	age	sex	zipcode
1	thomas	18	male	1416
2	martin	33	male	1645
3	bob	25	male	1613

Column-oriented (2)

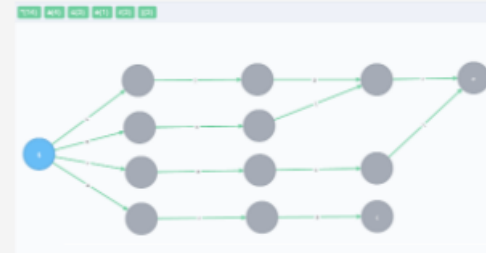
	name	age	sex	zipcode
1	thomas	1 18	1 male	1 1416
2	martin	2 33	2 male	2 1645
3	bob	3 25	3 male	3 1613

# Graph Database

Information is stored in nodes with the edges representing the relationships between them.

Each node is a datapoint (e.g. a customer, product, group ,etc) and edges define the relationships (e.g. person (node1) is a member of this group (node2)).

Querying is fast as relationships between nodes have already been defined.



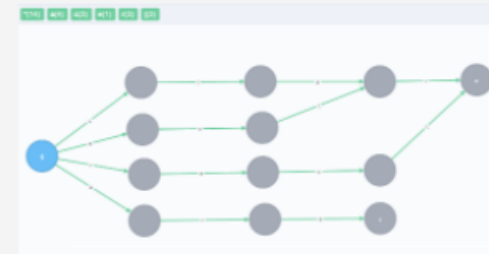


# Graph Database

A node can contain any type of data (structured or unstructured) including tables

As these types of databases do not use indexing, data retrieval is fast as the query follows the edges to obtain the connected information

Facebook is an example of this where a user (node) is connected to other users and groups by edges



# Key Features



# Key Features

**No Fixed Schema** - They do not have to follow historical rules making them more flexible



# Key Features

**No Fixed Schema** - They do not have to follow historical rules making them more flexible

**No Joins** - NoSQL databases tend to store data in one large table, taking advantage of cheaper storage and faster processing where redundant data is no longer such a big issue



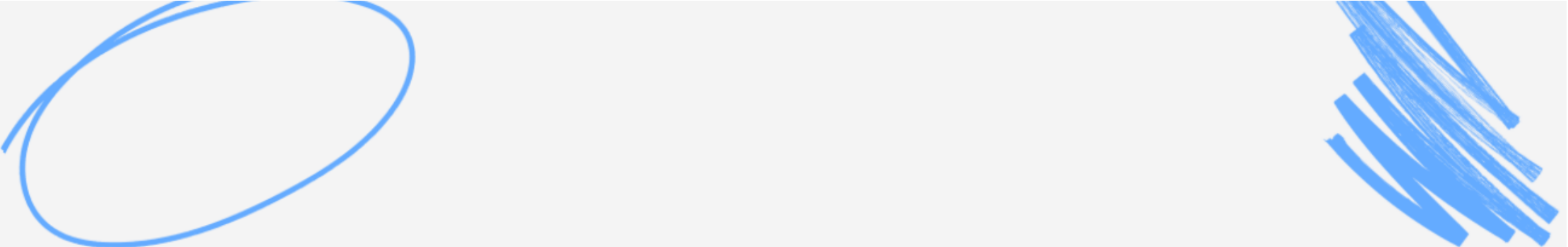
# Key Features

**No Fixed Schema** - They do not have to follow historical rules making them more flexible

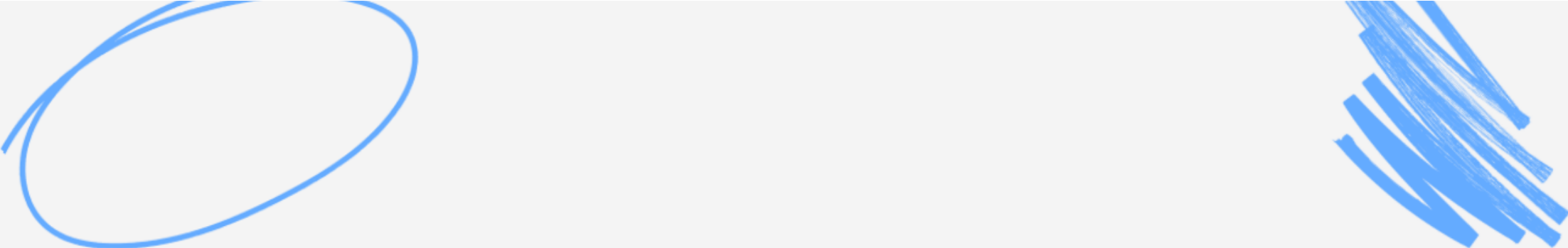
**No Joins** - NoSQL databases tend to store data in one large table, taking advantage of cheaper storage and faster processing where redundant data is no longer such a big issue

**Size and Scale** - No defined limits allows them to scale according to the resources available

# Other Types of Database

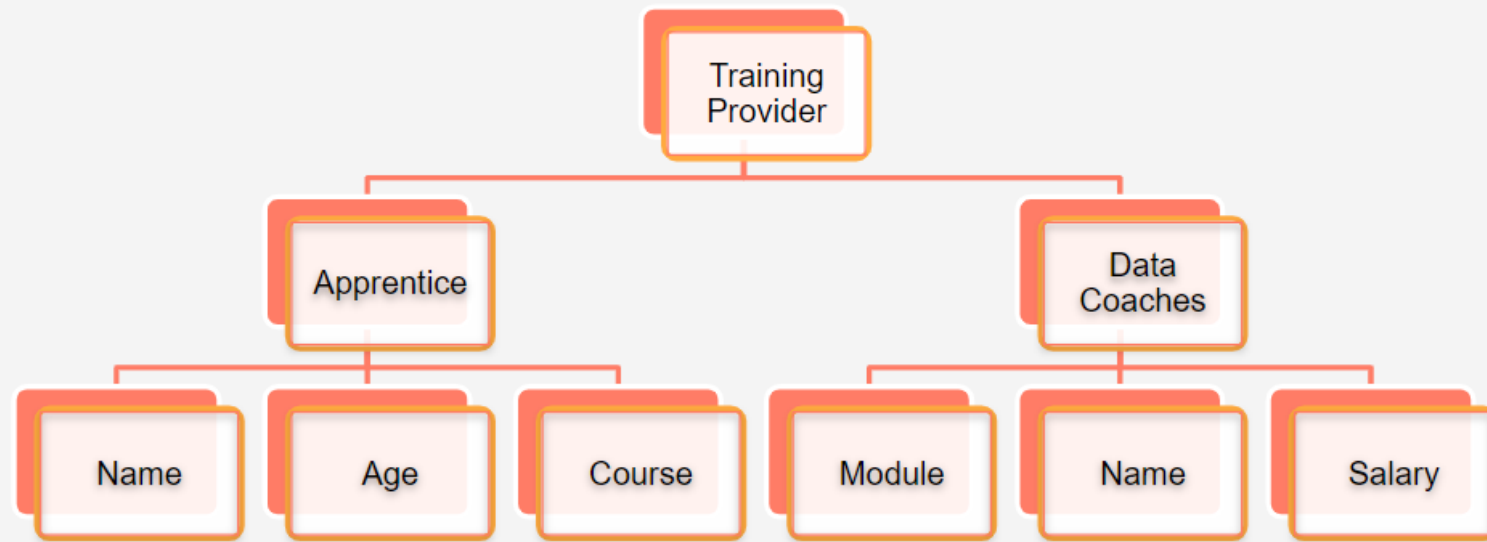


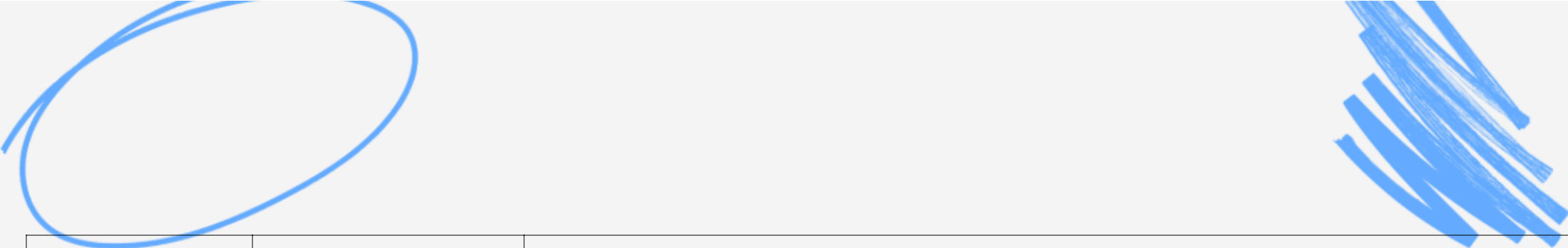
TYPE OF DB	EXAMPLES	DESCRIPTION
Relational	MsSQL, Postgres, mySQL	This is the most common form of database. It contains tables that can be joined to other tables through their relations. Normalisation is a key part of relational databases.



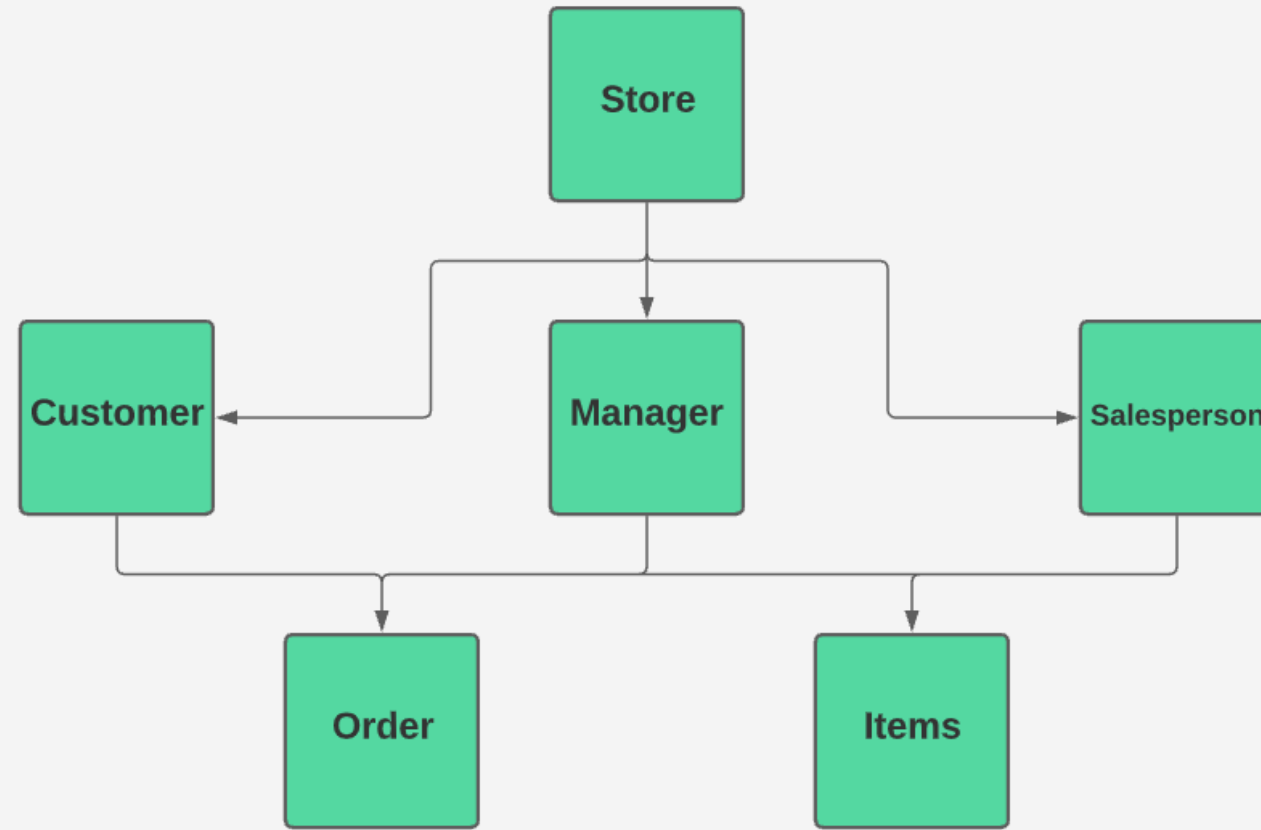
TYPE OF DB	EXAMPLES	DESCRIPTION
Hierarchical	IBM Information Management System (IMS), Windows Registry	Data is stored in a parent-children relationship nodes, in a tree like structure. The data is stored as a collection of fields where each field contains only one value. The records are linked to each other via a parent-children relationship. In a hierarchical database model, each child record has only one parent. A parent can have multiple children. To retrieve a field's data, we need to traverse through each tree until the record is found.

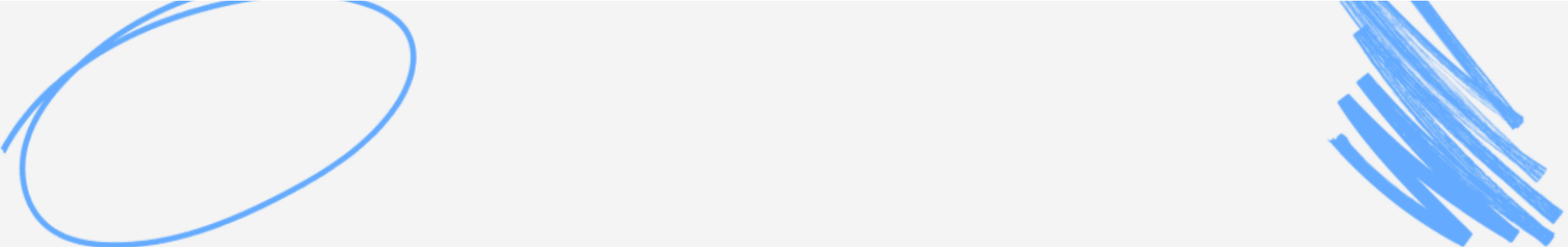




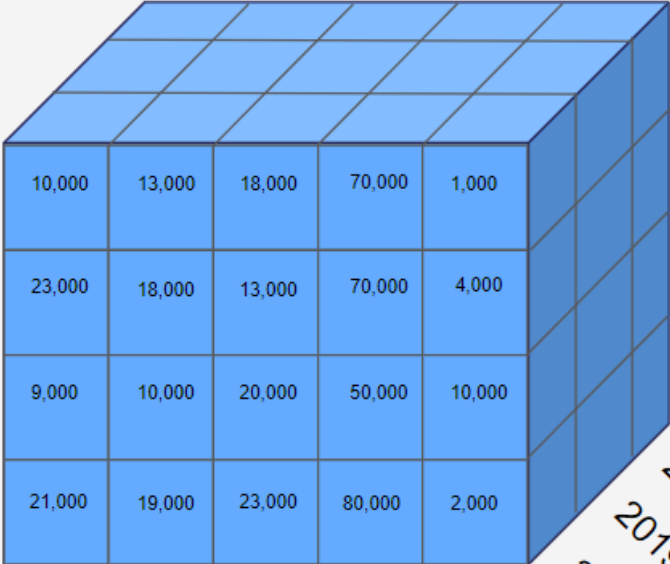


TYPE OF DB	EXAMPLES	DESCRIPTION
Network	Data Store (IDS), IDMS (Integrated Database Management System), Raima Database Manager, TurboIMAGE	Network database management systems (Network DBMSs) use a network structure to create relationship between entities. Network databases are mainly used on large digital computers. Network databases are hierarchical databases but unlike hierarchical databases where one node can have one parent only, a network node can have relationship with multiple entities. A network database looks more like a cobweb or interconnected network of records.

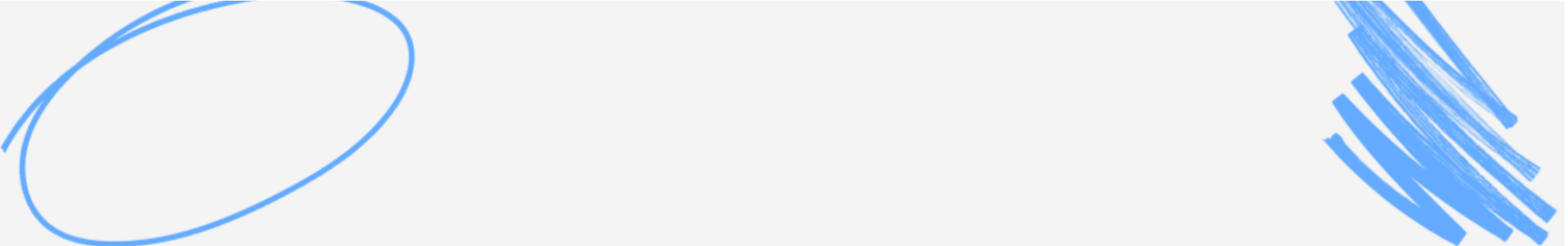




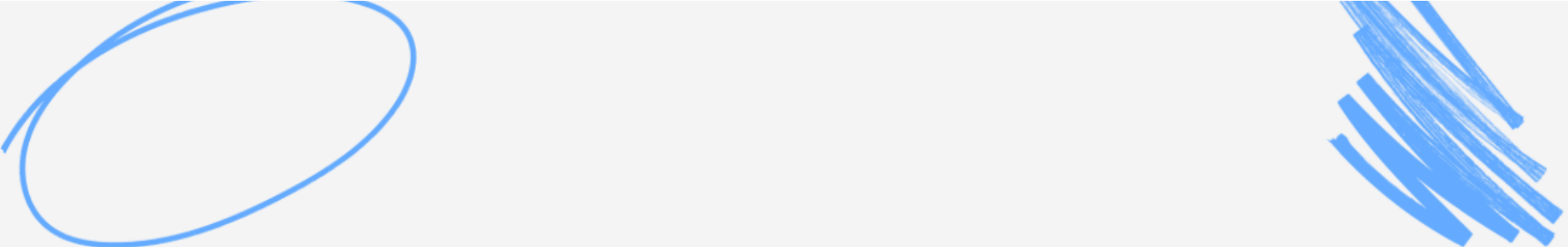
TYPE OF DB	EXAMPLES	DESCRIPTION
Multidimensional	Microsoft Analysis Services, Hyperion Essbase, Cognos PowerCube	Multi-dimensional databases (MDBs) use the concept of a data cube (or hypercube) to represent the dimensions of data available to users (though physically they are stored as compressed multidimensional arrays with offset positioning). An MDB with three dimensions looks like a cube, whilst an MDB with four or more dimensions is called a hypercube, and becomes more difficult to visualise. They are designed to assist with decision support systems, and to optimise online analytical processing (OLAP) and data warehouse applications.



	Bread	Milk	Cheese	Toilet paper	Broccoli
North	10,000	13,000	18,000	70,000	1,000
South	23,000	18,000	13,000	70,000	4,000
East	9,000	10,000	20,000	50,000	10,000
West	21,000	19,000	23,000	80,000	2,000



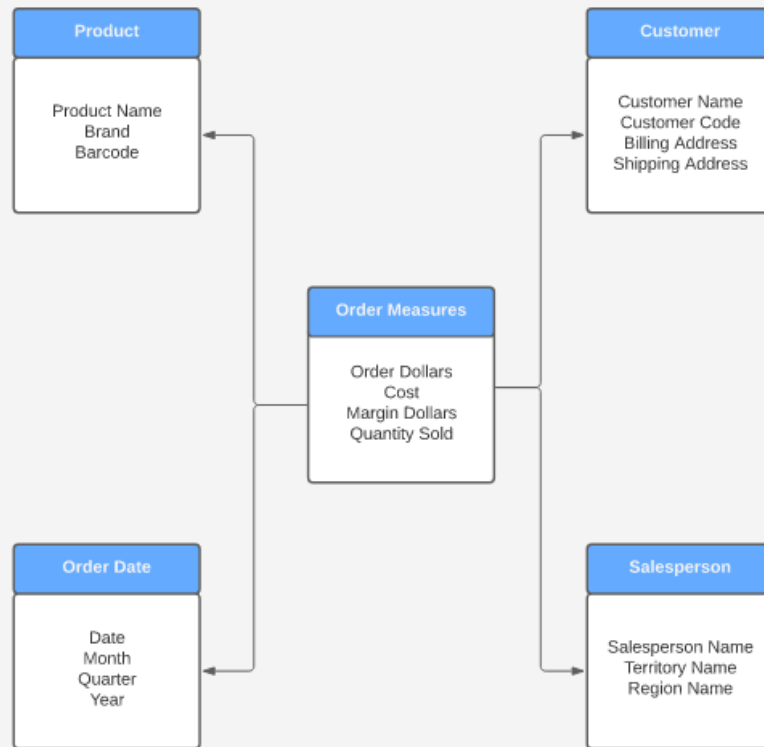
TYPE OF DB	EXAMPLES	DESCRIPTION
Object-oriented	TORNADO, Gemstone, ObjectStore, GBase	Object-oriented databases use small, recyclable separated chunks of data called objects. The objects themselves are stored in the object-oriented database. Each object contains two things: the object itself, and the metadata that explains that object, it's purpose, what it is and where it fits in



TYPE OF DB	EXAMPLES	DESCRIPTION
NoSQL		The 'No' stands for 'Not Only' SQL - these database seek to improve on standard SQL based DBMS by enhancing the forms of analytics available and changing the way that data is stored (no longer relational.) These databases are ideal for storing unstructured data.

'Data Warehousing' is a practice in **data management** whereby data is copied from various operational systems into a **persistant data store** in a **consistent format** to be used for **analysis, decision making and reporting**.





# Key Features

# Key Features

Optimised for low number of complex queries

# Key Features

Optimised for low number of complex queries

Contains historical data

# Key Features

Optimised for low number of complex queries

Contains historical data

Are taken off line when updates are required

# Benefits

Increased availability of data

Benefits

Increased availability of data

Superior quality of data

Benefits



Increased availability of data

Superior quality of data

Collaboration opportunities

Benefits

Increased availability of data  
Superior quality of data  
Collaboration opportunities  
Greater insights and improvements

## Benefits

multiverse

# Recap

# Learning Objectives



- Explain the concepts and uses of a **relational database management system**
- Identify the different types of **key** in a RDBMS
- Understand the principles of **normalisation** on a relational database



## ASSIGNMENT

### DATABASE DESIGN

Use a work-related dataset to design your own relational database. You should describe the dataset, follow the normalisation steps and create an Entity Relationship Diagram (ERD). You can run a Kahoot! quiz if you want.

Word Count	Max 1500 words
Deadline	3 weeks
Deliverables	Word Document, PowerPoint, Excel File, PDF, Lucid Chart

# Complete **Session Attendance** **Log** and Update Your OTJ