Powerful & Simple?!

Perform Analytics using MySQL & MongoDB

Rutgers Libraries - NB Data Science Workshop Series

Pratiksha Sharma

Dec 01, 2022

Fall 2022 Hours

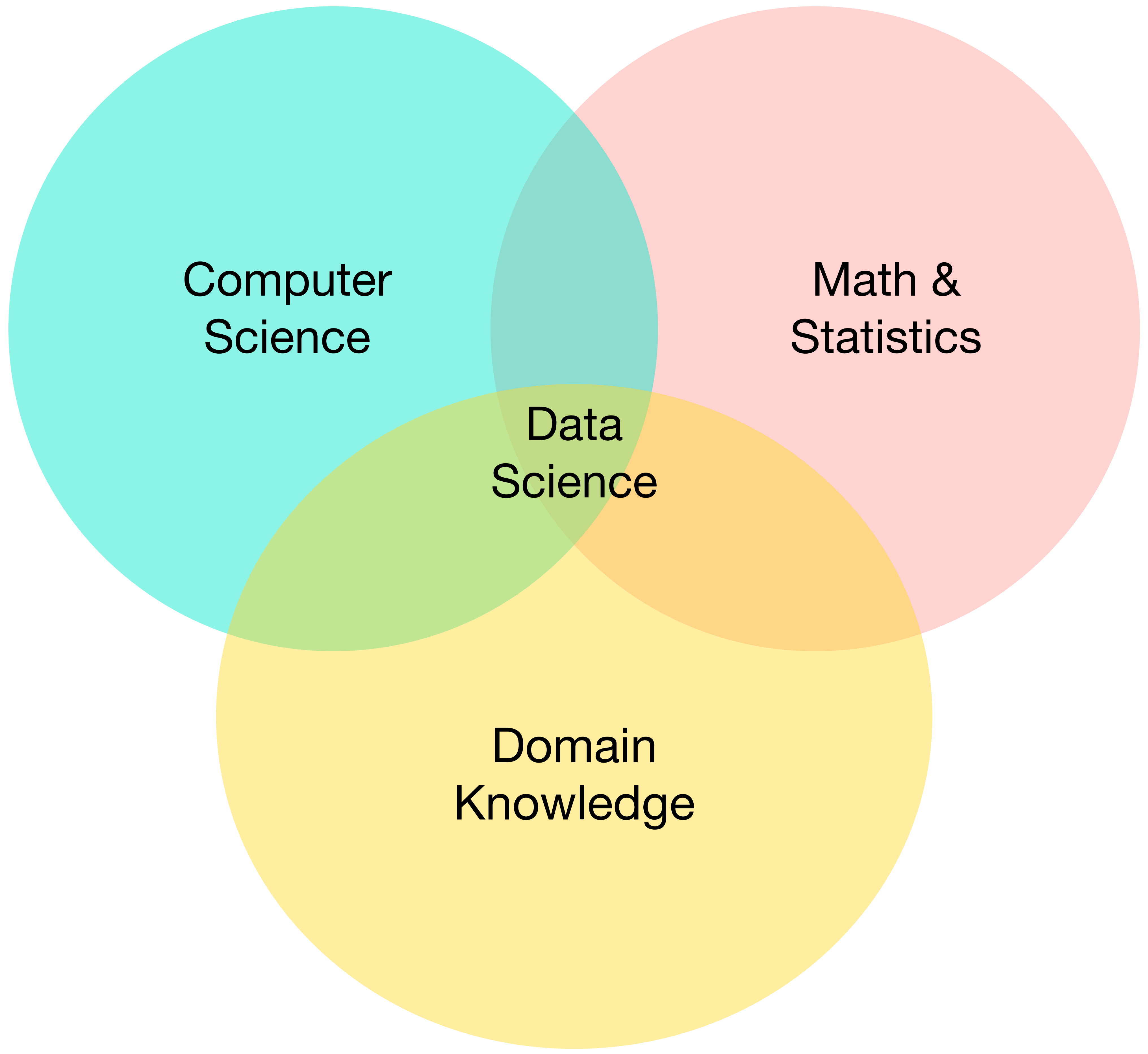
Pratiksha Sharma - Data Science Graduate Specialist

Email: pratiksha.sharma@rutgers.edu
Topics: Data Science, Tableau, Python, SQL & NoSQL Databases
Office Hours (by appointment):
Thursday 12:30 - 01:00 pm (on days when workshop ends at 12:30 pm)
Thursday 01:00 - 01:30 pm (on days when workshop ends at 01:00 pm)
General Consultation: Request an appointment via email
Location: Zoom

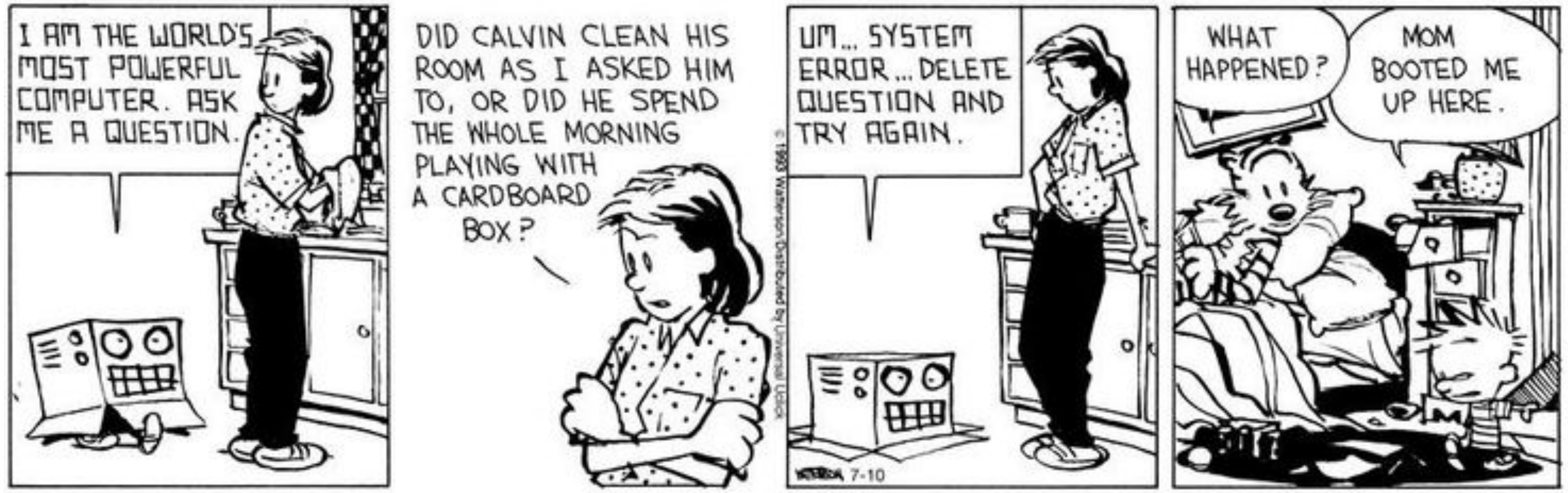
Quick Look: Data Science

What's all the fuss about?

- A combination of Maths & Statistics, Computer Science and Domain Knowledge.
- This workshop is on Programming - concentrates on the **Computer Science!**
- You don't need to be an expert - but data science is a part of everyday life!



Motivation


I AM THE WORLD'S MOST POWERFUL COMPUTER. ASK ME A QUESTION.
DID CALVIN CLEAN HIS ROOM AS I ASKED HIM TO, OR DID HE SPEND THE WHOLE MORNING PLAYING WITH A CARDBOARD BOX?
UM... SYSTEM ERROR... DELETE QUESTION AND TRY AGAIN.
WHAT HAPPENED?
MOM BOOTED ME UP HERE.
WATTERSON 7-10

Source: Bill Waterson | Universal Press Syndicate

Before we begin..

- We will be using MySQL Community Server, MySQL Workbench, MongoDB Community Server & MongoDB Compass.
- The goal for today's workshop is to: 1. Quickly discuss the concept of keys & how storage internally works; 2: Discuss OLTP & OLAP with examples.
- You are encouraged to participate & interact!
- Materials would be provided in the Zoom chat & will be available later on with the workshop materials on: Rutgers Libguides Data Science Workshops

OLTP

- Stands for Online Transaction Processing.
- It consists of executing and securing concurrent transactions such as financial transactions, billing transactions, banking transactions or other tasks that may qualify as a valid transaction.
- For example: Adding, updating or deleting a customer record also qualifies as a transaction. These days tasks like a successful installation of a file may also qualify as a transactional task.
- To work with such data, we have to rely upon relational databases such as MySQL.
- But this data needs to be in a Normalized Form (which we will discuss soon) & the database should be ACID compliant (discussing next).

ACID

- Atomic: Atomicity controls guarantee that all the steps in a transaction are completed successfully as a group. That is, if any steps between the transactions fail, all other steps must also fail or be reverted. The successful completion of a transaction is called commit. The failure of a transaction is called abort.
- Consistent: The transaction preserves the internal consistency of the database. If you execute the transaction all by itself on a database that's initially consistent, then when the transaction finishes executing the database is again consistent.
- Isolated: The transaction executes as if it were running alone, with no other transactions. That is, the effect of running a set of transactions is the same as running them one at a time. This behavior is called serializability and is usually implemented by locking the specific rows in the table.
- Durable: The transaction's results will not be lost in a failure.

Source: Oracle

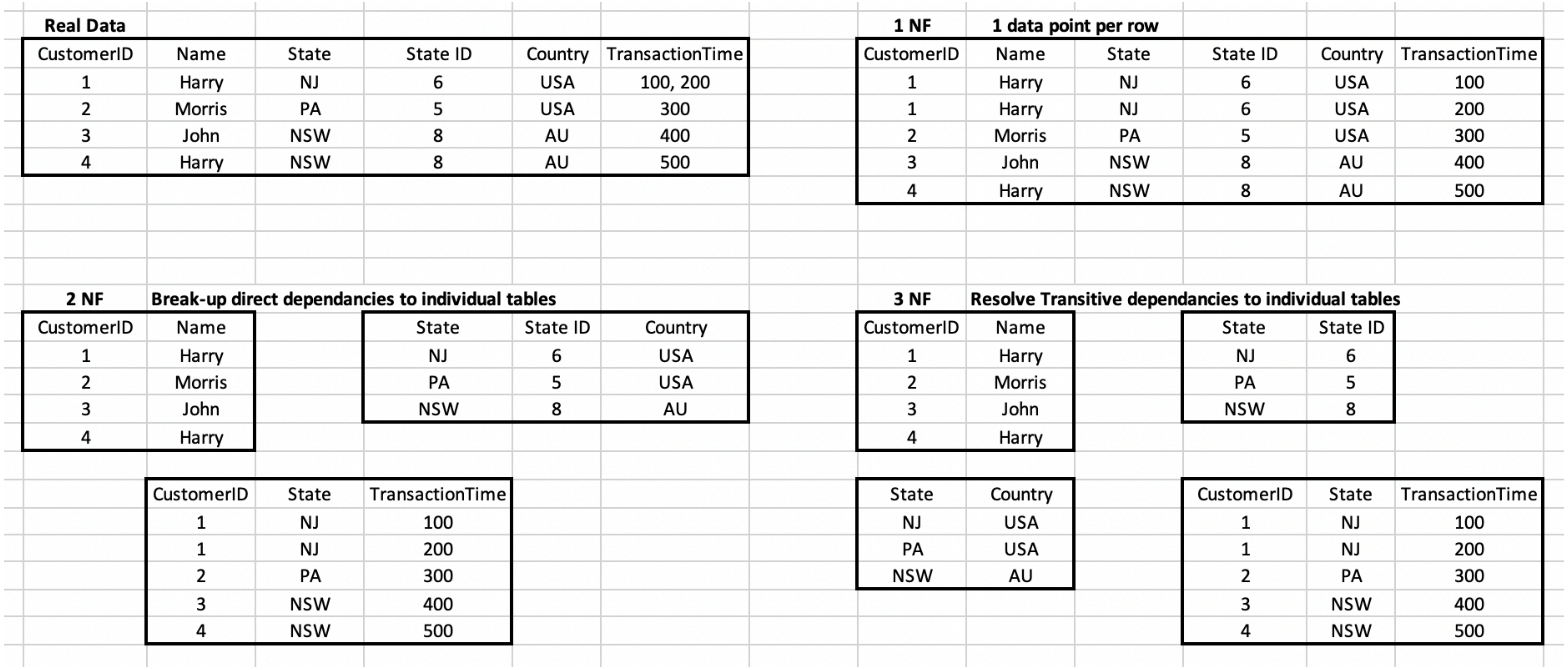
Primary Keys

- Relational Databases such as MySQL shall have a Primary Key or simply put - a column with unique values to identify & index each row in your table. Eg: RUID, NetID
- A common question is whether a Primary Key is mandatory or not - you already know the answer. No. Why? Because it is possible to create a table without explicitly defining a Primary Key Column/Attribute.
- In cases where a single column/attribute cannot uniquely identify each row or record, we can choose a combination of records to perform this task. Such a key is known as a 'composite key'.
- Having a primary key (or a composite key) is rather a best practice & essential if you want to use your database for an OLTP (Online Transaction Processing) task.

Normalization

- While creating a SQL table, you can indicate what your Primary (or Composite) Key is. Or, you can elect a column(s) in an existing table as your Primary (or Composite) Key. This creates a constraint for all records to have a non-null, unique value for the key attribute.
- For the task of OLTP, another requirement is that you have your data in a normalized form. Normalization is the act of breaking down your main data table to resolve dependancies and reduce duplication/redundancy.
- Normalization makes transactions more concise, streamlined and alterations/ updates more systematic. We will take a quick look at Normalization on the next slide.

Normalization



Real Data

CustomerID	Name	State	State ID	Country	TransactionTime
1	Harry	NJ	6	USA	100, 200
2	Morris	PA	5	USA	300
3	John	NSW	8	AU	400
4	Harry	NSW	8	AU	500

1 NF **1 data point per row**

CustomerID	Name	State	State ID	Country	TransactionTime
1	Harry	NJ	6	USA	100
1	Harry	NJ	6	USA	200
2	Morris	PA	5	USA	300
3	John	NSW	8	AU	400
4	Harry	NSW	8	AU	500

2 NF **Break-up direct dependancies to individual tables**

CustomerID	Name
1	Harry
2	Morris
3	John
4	Harry

State	State ID	Country
NJ	6	USA
PA	5	USA
NSW	8	AU

CustomerID	State	TransactionTime
1	NJ	100
1	NJ	200
2	PA	300
3	NSW	400
4	NSW	500

3 NF **Resolve Transitive dependancies to individual tables**

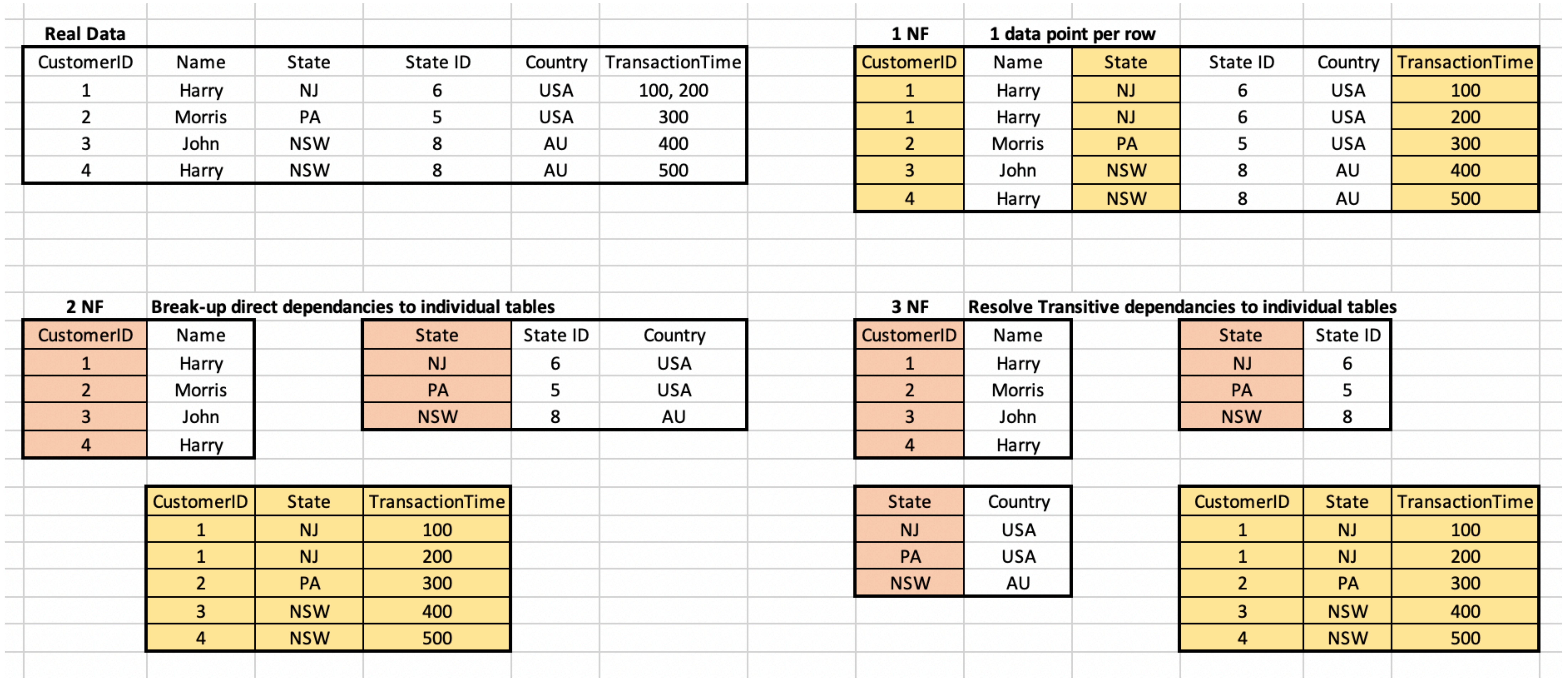
CustomerID	Name
1	Harry
2	Morris
3	John
4	Harry

State	State ID
NJ	6
PA	5
NSW	8

State	Country
NJ	USA
PA	USA
NSW	AU

CustomerID	State	TransactionTime
1	NJ	100
1	NJ	200
2	PA	300
3	NSW	400
4	NSW	500

Normalization: Primary & Composite Keys



Real Data

CustomerID	Name	State	State ID	Country	TransactionTime
1	Harry	NJ	6	USA	100, 200
2	Morris	PA	5	USA	300
3	John	NSW	8	AU	400
4	Harry	NSW	8	AU	500

1 NF **1 data point per row**

CustomerID	Name	State	State ID	Country	TransactionTime
1	Harry	NJ	6	USA	100
1	Harry	NJ	6	USA	200
2	Morris	PA	5	USA	300
3	John	NSW	8	AU	400
4	Harry	NSW	8	AU	500

2 NF **Break-up direct dependancies to individual tables**

CustomerID	Name
1	Harry
2	Morris
3	John
4	Harry

State	State ID	Country
NJ	6	USA
PA	5	USA
NSW	8	AU

CustomerID	State	TransactionTime
1	NJ	100
1	NJ	200
2	PA	300
3	NSW	400
4	NSW	500

3 NF **Resolve Transitive dependancies to individual tables**

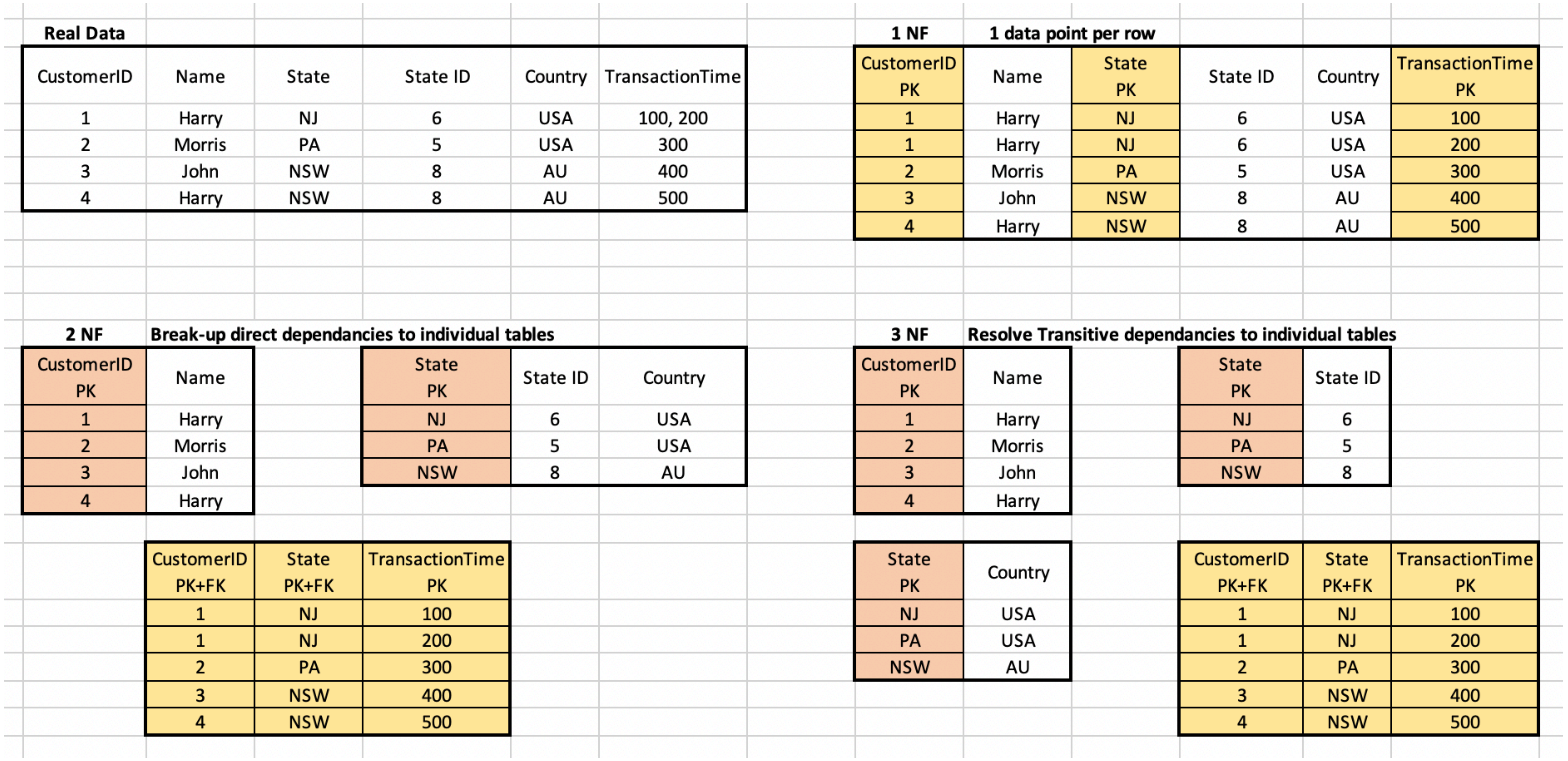
CustomerID	Name
1	Harry
2	Morris
3	John
4	Harry

State	State ID
NJ	6
PA	5
NSW	8

State	Country
NJ	USA
PA	USA
NSW	AU

CustomerID	State	TransactionTime
1	NJ	100
1	NJ	200
2	PA	300
3	NSW	400
4	NSW	500

Normalization: Foreign Keys



Real Data

CustomerID	Name	State	State ID	Country	TransactionTime
1	Harry	NJ	6	USA	100, 200
2	Morris	PA	5	USA	300
3	John	NSW	8	AU	400
4	Harry	NSW	8	AU	500

1 NF **1 data point per row**

CustomerID	Name	State	State ID	Country	TransactionTime
PK		PK			PK
1	Harry	NJ	6	USA	100
1	Harry	NJ	6	USA	200
2	Morris	PA	5	USA	300
3	John	NSW	8	AU	400
4	Harry	NSW	8	AU	500

2 NF **Break-up direct dependancies to individual tables**

CustomerID	Name
PK	
1	Harry
2	Morris
3	John
4	Harry

State	State ID	Country
PK		
NJ	6	USA
PA	5	USA
NSW	8	AU

CustomerID	State	TransactionTime
PK+FK	PK+FK	PK
1	NJ	100
1	NJ	200
2	PA	300
3	NSW	400
4	NSW	500

3 NF **Resolve Transitive dependancies to individual tables**

CustomerID	Name
PK	
1	Harry
2	Morris
3	John
4	Harry

State	State ID
PK	
NJ	6
PA	5
NSW	8

State	Country
PK	
NJ	USA
PA	USA
NSW	AU

CustomerID	State	TransactionTime
PK+FK	PK+FK	PK
1	NJ	100
1	NJ	200
2	PA	300
3	NSW	400
4	NSW	500

Foreign Keys

- This constraint connects two tables together. The data might be broken up into different tables, but primary key-foreign key concept establishes a relation between the tables.
- The constraint bars you from deleting a primary key record if a foreign key referencing that record exists in some other table.
- You have to explicitly define a foreign key either while creating a table in SQL or electing a column after creation.
- SQL will give you an error if you violate the primary key-foreign key constraint. This also means that you cannot have a new value of a foreign key, that isn’t a primary key in the main table.

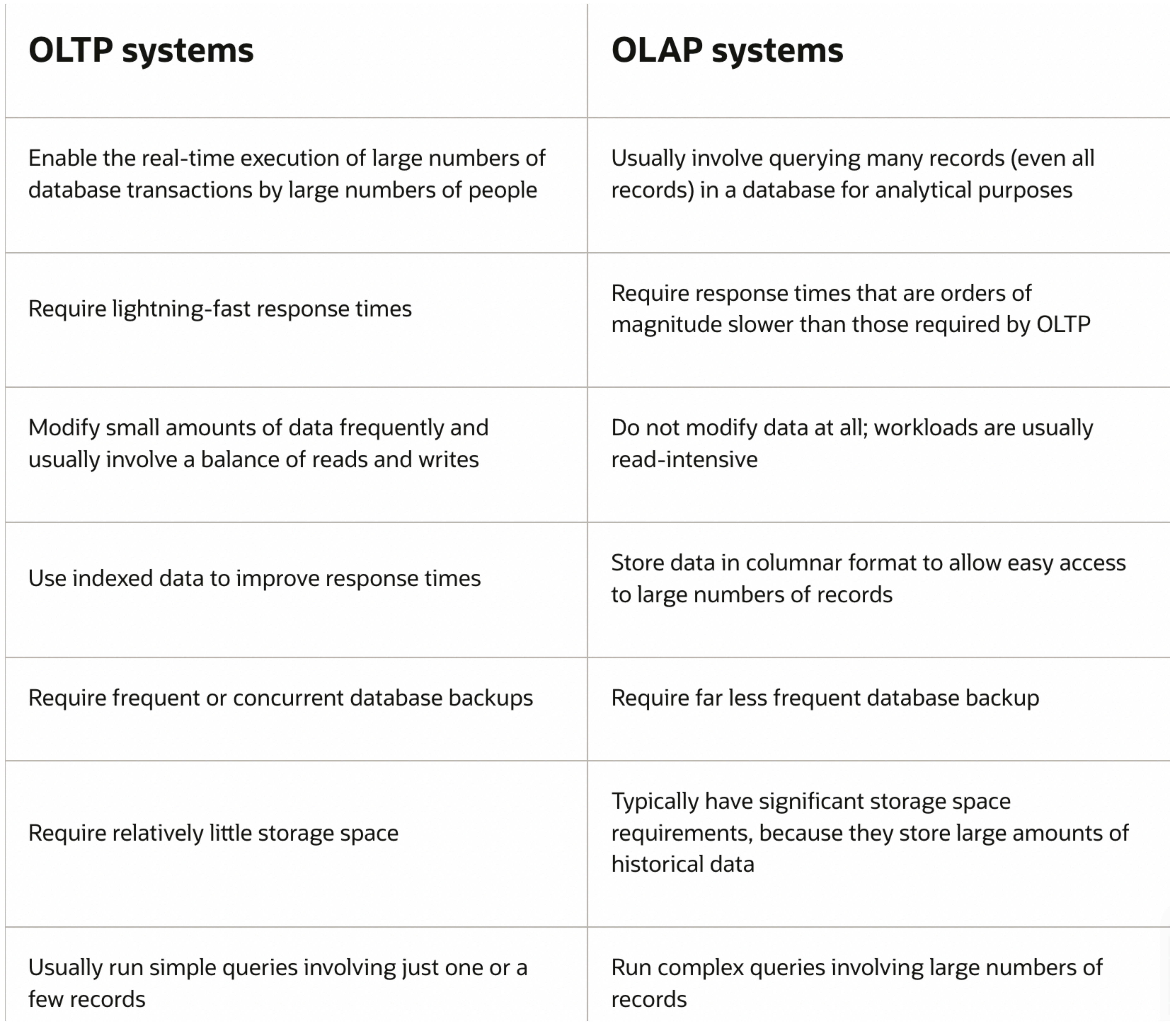
Deletion & Update Anomalies

- As seen in the Normalization example a little while ago, upon normalizing your data, you are reducing duplication & redundancy.
- This makes it easier for you to make updates. Think about having to change the state ID for a state. You do not have to make individual updates for each row or each transaction, but a single update in State_ID table should work.
- Next, deletion becomes more difficult and subject to restraints. Upon normalization, what you are essentially doing is creating a lookup-table.
- While you can delete records from a table where the attributes are Foreign keys, it is not possible to delete a record from a table where you have a primary key attribute that is being referenced by a foreign key attribute in another table.

OLAP

- Stands for Online Analytical Processing.
- Retrieving data for Analytical Processing is extremely expensive when you have to make multiple small operations on different tables. Think about wanting to know the customer name and State ID for a particular transaction from our previous example. You would have to lookup data from 3 different tables.
- That is how OLAP databases came in the picture. There are several benefits of an OLTP design and Normalization, but that can be quite expensive and tedious when you need a lot of data in one place to answer complex, analytical queries.
- Here's another one, just count how many sales transactions in a country - this would be a rather long task to work with if your data looked like how it did in 3NF. So, what's the solution to make this task easier?
- Condense data into a single big set. Think about 1NF or even the real data directly in some cases. Everything is in one place, for aggregate sales transactions in a country, one can just count the number of rows in 1NF for each unique value of country.
- Essentially, with an OLAP database design, you are not concerned with adding, deleting, updating etc. You are rather analyzing historic data.

OLTP vs OLAP



OLTP systems	OLAP systems
Enable the real-time execution of large numbers of database transactions by large numbers of people	Usually involve querying many records (even all records) in a database for analytical purposes
Require lightning-fast response times	Require response times that are orders of magnitude slower than those required by OLTP
Modify small amounts of data frequently and usually involve a balance of reads and writes	Do not modify data at all; workloads are usually read-intensive
Use indexed data to improve response times	Store data in columnar format to allow easy access to large numbers of records
Require frequent or concurrent database backups	Require far less frequent database backup
Require relatively little storage space	Typically have significant storage space requirements, because they store large amounts of historical data
Usually run simple queries involving just one or a few records	Run complex queries involving large numbers of records

Source: Oracle

Takeaway:

Focus on the Industry Requirements!

Upcoming Workshops

https://libcal.rutgers.edu/nblworkshops

- “Powerful & Simple?!" Perform Analytics using MySQL & MongoDB : Dec 01
- A Peek Behind the Curtain: An end-to-end data science project : Dec 08

Feedback Form

https://rutgers.libwizard.com/f/graduate_specialist_feedback