Small Data v/s Big Data

Replication v/s sharding!

Imperative programming v/s declarative programming!

How to do things v/s What to do (how to do depends on the system)

Editing is fast and easy; EDITING is a very expensive ops! (Most systems avoid editing the data)

Search is a nightmare on small data; Search is a LOT more optimised in big data!

Bigger the system SLOWER the search in SMALL Data; Bigger the system FASTER the search in Big Data!

Online Transactional Process (OLTP); Online Analytical Process (OLAP)

Small data -> SQL Server, MySQL, PostgreSQL
Big Data-> Spark, Hadoop, ML/DA, TensorFlow

HYPOTHESIS:
Data = numbers!
Machine size = 1 machine can store 2 numbers!

If I have 3 numbers-> big data!

Big data -> data that is too big for 1 machine!

1,2,3
M1 -> 1,2 —— replication —— M3-> 1,2
M2 -> 3 M4 -> 3

3 nums = 4 machines!!!

Cost of IT starting rising exponentially!

Google -> index the entire internet!

M1-> 1,2 M2-> 2,3 M3-> 3,1
SHARDING!

Storage Account -> Sharding

Hadoop -> sharding
Apache spark-> sharding

BIG DATA IS ALWAYS SHARDING!

Math:
- **Calculate sum of 2 numbers!**

M1 -> 1,2 —— replication —— M3-> 1,2
M2 -> 3 M4 -> 3

Asking sum from M1, m2
M1 = 1+2 = 3
M2 = 3 = 3
SUM = M1+M2 = 3 + 3 = 6
Imperative Programming!
How to do things

Declarative Programming!
What to do- NOT how to do
M1-> 1,2 M2-> 2,3 M3-> 3,1
M1 = 1+2 = 3
M2 = 2+3 = 5
M3 = 3 + 1= 4
M1 + M2 + M3 = 1 + 2 + 3 = 12
No direct programming!
Rely upon other algorithms to MAP the problem to the Machines and REDUCE the result from those machines

MAP-REDUCE
(SPARK)

———

M1 -> 1,2 —— replication —— M3-> 1,2
M2 -> 3 M4 -> 3

New problem statement->
UPDATE 2 to 4
Every ops takes 10ms (ASSUMPTION)

M1-> 1, 4 -> 10ms (2 not found), 10 ms (2 found and replaced)
= 20 ms

M2 -> 10 ms (2 not found and search exited)

Total cluster time = (10 + 10 M1) + (10 M2)= 30ms
Longest cluster time = M1 = 20 ms

M1-> 1,2 M2-> 2,3 M3-> 3,1
M1 -> 10ms (2 not found), 10 ms (2 found and replaced)
= 20 ms
M2-> 10 ms (2 found and replaced)
M3 -> 10 ms (2 not found), 10 ms (2 not found and exit)

Total time = (20 ms M1) + (10 ms M2) + (20 ms M3)
= 50 ms
Longest time= 20 ms

BIGGER the data-> bigger the time!!!

SEARCHING : 2
M1 -> 1,2 —— replication —— M3-> 1,2
M2 -> 3 M4 -> 3

M1 -> 2 not found (10ms) , 2 found (10 ms)
M2 -> 2 not found(10ms), exit search

Total time = 20 ms
Search time = M1 found it= 20 ms

M1-> 1,2 M2-> 2,3 M3-> 3,1
10ms 20ms 30ms
M1-> NF
M2-> FOUND! Exit search!
M3-> NF

Total time = 30ms
Search Time = M2 -> 10ms

Unique values-> data[col_name].value_counts()

Discrete/Continuous->data[col_name].value_counts().count()

Null value check-> data[col_name].isna().sum()

Descriptive stats (mean/median/std/count) -> data.describe()

Correlations -> data.corr()
Correlations are better understood as visualisation
seaborn.heatmap(data.corr(), cmap='cool warm')

Outliers -> extreme values that do NOT fit with rest of the data
Define-?
If any data-point > 3 times the standard deviation, we call it an outlier
Formulae- (data-mean)/standard_deviation

If (data-mean)/standard_deviation > 3 or < -3, we call it an outlier

That we need mean and standard deviation from our dataset

data.describe() -> this should give us mean and standard_dev!

(Data-mean)/standard_deviation = (data-mu)/sigma