Present in detail the NIST Definition of Cloud Computing.

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics**,** three service models, and four deployment models."

The five essential characteristics are:

- **On-demand self-service** a client asks a service when needed and pay for its usage.
- **Broad network access** resources are available through the net
- **Rapid elasticity** scale in (add more resources), scale down (reduce resources), scale out (add another virtual machine) almost in real time
- **Measure service** cloud dynamically optimizes and monitors the usage of resources
- **Resource pooling** putting together resources and dividing them among several clients, with location independence.

The three service models are:

- **IaaS** the user receives real resources, real infrastructure on demand [CPU, memory, computing resources]. User controls operating systems, storage and install application; similar to having a physical server. → need to execute a job without owing a proper machine for execution.
- **PaaS** the user installs and creates applications developed using programming languages, libraries, services and tools supported by the provider. Platform is ready to be used for web applications, so user just need to configure and use them. → without installing the whole software with all the libraries and tools, leaving technical parts to experts. [LAMP]
- **SaaS** the user uses the application executed on the cloud infrastructure [Dropbox]. Users can control only limited number of configurations, only useful tools.

The deployment models are:

- Private cloud: exclusive use of a single organization.
- Community cloud: exclusive use of a community of users, like organizations with common goals.
- Public cloud: any user can access. On premise for cloud provider.
- Hybrid cloud: combination of two or more cloud infrastructure that remain separated.

Big Data Platform-as-a-Service has been defined to address the challenges users have to face in managing Big Data Platforms. Discuss in detail these challenges.

Big Data is the growing technical ability to collect process and extract new and predictive knowledge from great volume, velocity and variety of data. Since the adoption is convenient for companies, but the costs of setting up Big Data infrastructure may be prohibitive for some organizations, in particular small and medium-size, especially without a strong technical expertise, BDPaaS has been defined [Amazon S3].

This permits companies to store, manage and analyze Big Data in the cloud without having a team of experts and without deviating from the core business of the organization. The 3 different models offered are BDIaaS, BDPaaS, BDSaaS. In particular, BDPaaS provides a storage of data (data layer), the integration of data

(plumbing) and processes data (crunching); the analysis must be done by the company itself (otherwise, is possible to have it with BDSaaS).

The main 3 challenges are:

- **Lack of skills and clarity on big data technology**: driven by the last technological release
- **Lack of general architecture and standard processes**: having tools available without a general architecture and standard approach causes problems with adopters, especially the one that are not able to build a data science department in their business.
- **Ineffective governance models**: not managing our data and analytics at best, so we are unable to correctly maximize the value out of our big data.

Provide an overview of the Big Data Analytics-as-a-Service methodology, focusing on the role of procedural models.

??????

Consider a scenario where an e-commerce application selling Christmas items is migrated to the cloud. Upon describing the migration process, discuss in details the benefits the cloud can bring to the application.

Migration to the cloud can be divided in 5 levels: application, code, design, architecture, usage. Each level provides and carries different requirements, expectations. In migration we have 7 steps:

1. Cloud Migration Assessment (design migration, taking the system and redesigned it with cloud in mind)
2. Isolate the dependencies (identify dependencies between components and how the cloud can support them, to keep them connected)
3. Map the messaging & the environment (the system and the structure we are using are completely different, so we need to adapt the solution to the cloud)
4. Re-architect and implement the lost functionalities (moving to the cloud we might lose some functionalities, so need re-architect or rethink components and solutions)
5. Leverage cloud functionalities & features (otherwise we are not taking all the advantages we are paying for)
6. Test the migration
7. Iterate and optimize

The cloud offers a lot of advantages, for example reduces the complexity of systems, is scalable and the managing is left to cloud provider.

- **On-demand resourcing**: instead of a long purchasing process, we have immediate access to resources when needed
- **Scalability**: is possible to scale in/out/down, so we can request to increasing/decreasing immediately resources

- **Economy of scale**: the more you buy, the cheaper it becomes
- **Flexibility and elasticity**: it is difficult to manage spikes in on premise, in cloud is adaptable depending on needs, no need to establish in advance
- **Growth**: instead of lots of changes, we have immediate support with no limits to growth
- **Utility based metering**: instead of keeping servers 24h/24, we pay-per-use
- **Shared infrastructure**: different tenants share the same resources
- **High availability**: instead of requiring higher costs, replicating resources and services are distributed in several zones and geographical regions
- **Security**: instead of no assurance and certification, the infrastructure is certified and compliant

Besides the advantages, what is important to take into account are the costs: in fact, we do not have only the cost of moving, but also the costs in terms of business: modelling servers, storage, databases, data transfer, elasticity and also growth patterns.

As regards the grow patterns, they can be distinguished in 3 classes: constant growth [number of users increase every day/month], seasonal growth [in one specific period we have increase and then decrease], lifecycle growth [observe temporary growth during launch, higher spikes only few weeks/months and then stabilize].

In particular, in our case we need to take consider the seasonal growth of Christmas items: we expect to see a pick in Christmas period and a shrinkage in the rest of the year. Is not profitable for the e-commerce to buy several physical machines [servers] to cope with the great amount of data concentrated only in that period and then leave these resources unexploited during the rest of the year, it would be a waste of money and resources; instead, it would be more effective to deploy a cloud that, as we said before, is able to manage growth of data, is flexible and scalable and is based on a utility-based metering, there is no need to have servers 24/7/365.

Discuss the criteria for selecting a Big Data Technology.

Selecting the appropriate Big Data technology that suits your business requirements requires to take into consideration the following key aspects:

- There is no single big data technology. BD has many different use cases and skills deficits
- Need to make sure that the planning is long-term
- Balance between bottom-up (tech-led) and top-down (business-led) planning

In particular, the 3 main factors are the technical perspective, the social perspective and the cost/policy perspective. There are many factors that drive a decision toward the selection of a Big Data technology stack. We have to evaluate the capacity of managing the 5 Vs and so many other characteristics.

- Availability and fault tolerance
- Flexibility and scalability
- Performance
- Data security
- Computational complexity
- Distributed storage capacity
- (Handling huge data volumes, variety, velocity)

- Ease of installation and maintenance (graphical interface)
- User interface and reporting (how much the system is complex to use)
- Documentation and support (how can I manage a problem using a documentation available)

- Costs (open source vs commercial)
- Sustainability of the solution (cost associated with system maintenance)
- Policy and regulation (GDPR, law conflicts, restriction of use)

Consider a scenario where a web developer wants to migrate his/her activities to the cloud. Describe which cloud service model the web developer will select and why. Discuss in details the main characteristics of the cloud that make such migration suitable for the web developer.

+ (Definizione cloud)

Cloudonomics: cost savings and economic aspects introduced by the cloud and related trade-offs

The cloud offers a lot of advantages, for example reduces the complexity of systems, is scalable and the managing is left to cloud provider. In particular, the characteristic of cloud that could be interesting for the web developer are:

- **On-demand resourcing**: instead of a long purchasing process, we have immediate access to resources when needed
- **Scalability**: is possible to scale in/out/down, so we can request to increasing/decreasing immediately resources
- **Flexibility and elasticity**: it is difficult to manage spikes in on premise, in cloud is adaptable depending on needs, no need to establish in advance
- **Growth**: instead of lots of changes, we have immediate support with no limits to growth
- **Utility based metering**: instead of keeping servers 24h/24, we pay-per-use
- **High availability**: instead of requiring higher costs, replicating resources and services are distributed in several zones and geographical regions
- **Security**: instead of no assurance and certification, the infrastructure is certified and compliant

Besides the advantages, what is important to take into account are the costs: in fact, we do not have only the cost of moving, but also the costs in terms of business: modelling servers, storage, databases, data transfer, elasticity and also growth patterns.

As regards the grow patterns, they can be distinguished in 3 classes: constant growth [number of users increase every day/month], seasonal growth [in one specific period we have increase and then decrease], lifecycle growth [observe temporary growth during launch, higher spikes only few weeks/months and then stabilize].

The cloud service model I think the web developer would select is PaaS, a platform ready to use provided with the application and OSes needed by the user. As, for example LAMP, the PaaS would provide what is needed for carry out web developer's duties: an environment with the infrastructure, the platform, libraries and tools ready-to-use, where the user does not need to take care of the technology behind it and just work on it.

Illustrate the problem of transforming a monolith into microservices. Discuss in detail the approach based on extract services.

We can that system are often realized as monoliths, software with one single code base, one single technology stack and one build and deployment unit. The problem is that this kind of organization has lots of limitations and becomes more and more complex over time. The main solution is the one that transforms a monolith into microservices, where the functional system is decomposed into manageable and independently deployable components. Each of these microservices run specific functionalities and represents a small application with its own architecture. This makes it easier to develop, maintain and understand and reduces barriers of adopting new technologies.

To do so we can apply 3 strategies: stop digging, split frontend and backend or extract services.

The third one is the one that actually reduces the size of the monolith, transforming the modules into standalone services. To shrink the monolith, we need to go through two phases: in the first one we need to select modules to convert prioritizing the ones that have requirements of resources significantly different from the rest of the monolith and then the ones computationally expensive; in the second phase we extract the module defining a clear interphase between the modules and the monolith [bidirectional API]. We need to provide a shared memory, a shared blackboard, a shared data layer between services and monolith. Then the module selected is transformed into a service.

Discuss the different models of Big Data Platform-as-a-Service (BDaaS).

There are three different BDaaS models. These closely align with the 3 models of cloud infrastructure: IaaS, PaaS, and SaaS.

1. Big Data Infrastructure as a Service (IaaS) – Basic data services from a cloud service provider. It provides storage and minimal integration. You provide the rest.
2. Big Data Platform as a Service (PaaS) – Offerings of an all-round Big Data stack like those provided by Amazon S3, EMR or RedShift. It provides storage, plumbing and crunching. You do the analytics.
3. Big Data Software as a Service (SaaS) – A complete Big Data stack within a single tool. Provides storage, plumbing, crunching and analytics: includes everything from data storage to data visualization, from data to insight, just need to configure execution.

To choose the right BDaaS provider, we also need to look at the following factors:

- **Core B** Uses minimal platform like Hadoop with YARN and HDFS and other services like Hive. We need some core component that are not directly linked to requirements of high performance computing. Carrying out analytic with not regular shape, but not performance intensive for sure. This service has gained popularity among companies which use this for any irregular workloads or as part of their larger infrastructure. [PaaS]
- **Performance Db** Including optimizing infrastructure. One path of vertical integration for BDaaS is downwards to include an optimized infrastructure. The idea is increasing performance. Support businesses that are already employing a cluster-computing framework like Hadoop to further optimize their infrastructure as well as cluster performance. To cope with requirements that are rapidly

increasing and expanding. The benefit of increasing infrastructure is to focus our business on our asset, without deviate from it. [IaaS + PaaS]

- **Feature BD** Need to refer to BDaaS solution that provides you with some specific features that go beyond the Hadoop features. Beyond standard architecture. If your business is in need of additional feature; focus on productivity. May not be within the scope of Hadoop. Have web API interfaces, database adapters that offer a layer of abstraction from the underlying details. Integrate third party vendors within our solution. [PaaS + SaaS]
- **Integrated BD** Fully vertically integrated BDaaS that combines the performance and feature benefits of the previous two BDaaS. This is an appealing approach, provides maximum performance, but could be difficult to achieve. [IaaS + PaaS + SaaS]
- Low or zero startup costs: finding provider with free trial period. Can start seeing results before you even commit dollar.
- Scalable: growth in scale is in the very nature of BDaaS project. The solution should be easy and affordable to scale, especially in terms of storage and processing resources.
- Industry footprint: is a good idea to choose a Big Data provider that already has an experience in your industry. This is doubly important if you are also using them for consultancy and project planning requirement.
- Real-time analysis and feedback: able to reduce them, provide immediate analysis. This helps business to take remedial action instantly instead of working off of historical data.
- Managed or Self-Service: provide a mix of both managed as well as self-service models based on the company's needs. It is common to find a host of technical staff working in the background to provide the client with services needed.

Describe the problem of data confidentiality when storing data in the cloud. Briefly illustrate the approaches based on the adoption of encryption and of fragmentation for providing confidentiality of outsourced data.

The cloud allows users and organizations to rely on external providers for storing, processing and accessing their data. But users may lose control over their own data, this creates new security and privacy issues. We need to take into account of the possibility that the provider can be curious, lazy or malicious. Issues to be addressed are: data protection, query execution (searches over the data), private access, data integrity and correctness, access control enforcement, data publication and utility. To achieve that we can apply encryption, fragmentation or them both.

Encrypting means choosing an encryption key, encrypt data on client side before sending the data to the cloud provider. This will protect data to the eyes of who does not know the encryption key. Data confidentiality is provided by wrapping a layer of encryption around sensitive data at tuple level (typically). Cloud provider cannot decrypt data nor process/access it. There approaches (fine-grained access) that have been proposed are:

- **Keyword-based searches directly on the encrypted data**: need to know in advance the keywords that we want to search.
- **Homomorphic encryption**: support the execution of operation directly on the encrypted data. Problems: quite expensive in computational time to encrypt data; the size of data increase a lot because of the encryption.

- **Encryptions schemas**: to maintain distinction between values, can perform searches (check whether two values are the same or different). Each column can be encrypted with different encryption schema, depending on the conditions to be evaluated on it.
- **Onion encryption**: different onion layers each of which supports the execution of a specific SQL operation, so we encrypt data several times using different algorithms.

Fragmentation consists in splitting the table in different fragments, that store subset of attributes and which can be reconstructed by applying a join on them. Often what is sensitive is the association between values of different attributes, rather than values themselves [name, salary]. The fragmentation can have two approaches based on the direction of the flow of information:

- **Server-Client strategy**: server (cloud provider) evaluates C_s and returns to client, who receives the result and joins it with F_o and evaluates C_o and C_{so} on the joined relation. Only 1 flow of information.
- **Client-Server strategy**: client evaluates C_o and send the result to server, which joins input with F_s and evaluate C_s and returns to client, who joins result from server with F_o and evaluate CSO. We have flows of information.

Describe the problem of integrity of stored data and of the results of computations. Illustrate the difference between deterministic and probabilistic integrity verification solutions, also presenting an example of each of them.

Integrity in general means that we need to have some kind of technique to determine whether someone accessed and improperly modified data. Data owner and users need mechanisms that prove integrity for query results:

- **Correctness** = computed on genuine data, not modified
- **Completeness** = computed on the whole data collection, not a fraction
- **Freshness** = computed on the most recent version of the data, new data

In particular, is possible to distinguish 2 different approaches:

- **Deterministic**: uses authenticated data structured [signature chains] or encryption-based solutions [verifiable homomorphic encryption schema]. It is built on top of database, has a specific and identifiable data structure.
  One example is the Merkle Hash Tree, a tree created by the data owner and stored at the sever side. Is a binary tree in which each leaf corresponds to a tuple and contains the result of the hash function over that tuple and the concatenation of its children. The root node has hash function that depends on all the leaves and all the contend of the dataset. Since the data owner signs the root node, everybody can verify the signature: if someone changes the table without being authorized, the root would have a different value with respect to the one publicly released. Besides, if we want to start from one of th leaves and reach the root, the sever returns a set of nodes necessary to the client for re-computing the path to the root that is called verification object. This technique can be also used for verification in case of rank queries because we are sure that there are no objects between two subsequent *t* that is not returned to us.
- **Probabilistic**: exploits insertion of fake tuples in query results, replication of tuples in query results, pre-computed tokens. Not provide 100% guarantee that data is integrate when you get them from query, but more flexible: is independent from the data collection used, from the kind of searches

performed and from the idea of adding structure on top of data. It is based on the injection of additional tuples/files into the dataset, where the client knows exactly which are the fake ones, while the cloud provider does not: because of this information advantage, we can be sure that the cloud provider did or did not operate in a proper manner. Some examples of the techniques are:

- markers =fake tuples injected
- twins = p-percentage of duplicates of subset of tuples
- salts and buckets =in case of one-to-many relationships, we flatten frequency slipping in buckets the "many" side and adding copies in the "one" side in order to adjust the connections between attributes
- semi-join = we first extract all the attributes removing duplicate, we find out which values are present in both the relations and then we compare them with the result of that values only provided by the 2 cloud provider (compute join only on the column of interest)
- MapReduce = in a parallel and distributed scenario, we give to multiple machines data on which we applied fragmentation, indexing and used the techniques above. If who received the original version of tuples and who received the copy retrieve the same result, we know they are both working in a proper manner, otherwise one of the e two is not.

Clearly illustrate the problems of confidentiality and integrity protection when moving data to the cloud. Briefly describe how fragmentation can be used to provide confidentiality guarantees.

The cloud allows users and organizations to rely on external providers for storing, processing and accessing their data. But users may lose control over their own data, this creates new security and privacy issues. We need to take into account of the possibility that the provider can be curious, lazy or malicious.

Let's focus of two issues: confidentiality and integrity. We talk about confidentiality problem when actions that the cloud provider can perform on data may expose users' identities; while integrity is the concept that guarantees that we will find exactly the same data we stored and that nobody (not authorized) can modify it, (so queries would return the complete and correct list of subjects having the characteristics we are looking for).

In particular **data confidentiality** can be obtained by applying some protection techniques, like fragmentation, encryption, indexing, and bloom filter (hashing). Each of these has its own pros and cons relatively to the kind of data we are protecting and the searches we have planned to perform.

To guarantee **data integrity**, we cannot expect the cloud provider to take care of the access control, because data owner should always maintain the control of data security. What we can do is to establish a sort of authorization enforcement using encryption policies: a user can access if his/her private key (based in his/her attributes) is enabled to decrypt a specific data item [a Professor can only see marks of his exam], or we can have the situation in which keys are distributed to subjects such that each user can only access the files written with the key they know. Furthermore, it is possible to rely on tokens, public tools that allow users to derive multiple encryption keys and can be stored on a remote sever.

Discuss the main goal of an access control system. Illustrate the working of Discretionary Access Control (DAC), discussing discretionary policies, the access matrix model and provide and example of DAC.

The access control evaluates access requests to the resources by the authenticated users and, based on some access rules, it determines whether the requests must be granted or denied. It may control only direct access or also inference, information flow and non-inference controls.

Correctness of access control rests on:

- Proper user identification/authentication: none should be able to acquire the privileges of someone else.
- Correctness of the authorizations against which access is evaluated (which must be protected from improper modifications).

The access control can be verified by policies, models (rules) or mechanisms. In particular, the policies are DAC, MAC, RBAC, Credential-based, ABAC.
The Discretionary Access Control (DAC) is the policy that enforces access control on the basis of the identity of requestors and explicit rules of the mechanisms establishing which user can make which action on which object. Besides, it is called "discretionary" because is possible for a user to pass the rights to another user.

The system is based on the presence of S subjects, O objects (resources, files) and A access matrix. The access matrix is the table in which we organize the privileges/operations that each user can perform over each objects. The users are the rows of the matrix, the columns are the different objects and in the cells we define the operations that can be made. Since the matrix tends to be very large and risks to waste a lot of space (empty cells), some alternative approaches have been presented:

- Authorization table: keeps track of the non-null triples where the columns are user, privilege, object. We have one row for each different combination, so for example if user Ann has multiple privileges for one file, we will find several times the same user along the table, one time for each combination of operation+file.
- Access control list: starting from the objects [file 1] we connect a list of subjects and their privileges on that specific object.
- Capability list: starting from subjects [Ann] we connect a list of objects with the privileges the user has on that specific object.

The DAC weakness is that it constraints only direct access, without any control of flow of information; this makes this approach vulnerable to Trojan Horses: a user not authorized to read an object, could make an authorized subject add this privilege to him behind his back.

	File 1	File 2	Program 1
Ann	read, write		execute
Bob		own, read	
Cate	read, own		write

Clearly illustrate the anonymity problem providing an example of disclosure that causes re-identification. Describe the difference between anonymity and de-identification. Illustrate the goal of k-anonymity and the basic microdata protection techniques for its enforcement.

The anonymity problem is the necessity to protect personal data that may reconstruct the presence of a subjects in a set and that may disclose some confidential attributes of a person. Attributes can be classified as:

- **Identifiers**: single attribute that uniquely identify a microdata respondent [SSN, university ID number].
- **Quasi-identifiers**: set of attributes that in combination with external information or other attributes can identify all or some of the respondents to whom information refers, or reduce the uncertainty over their identities [DoB, ZIP and Gender, preferences]. Not really a key, in general any kind of information which can be used to reduce uncertainty about identity of subjects, even if they are not unique.
- **Confidential**: attributes of the microdata table that contain sensitive information [disease].
- **Non-confidential**: attributes that the respondents do not consider sensitive and whose release does not cause disclosure.

In a microdata table is difficult to find identifiers, which are removed to prevent re-identification. On the other hand, we have quasi-identifier: if we find values of attributes, for example ZIP code, sex and date of birth, and we know that a specific person is represented in that table, based on these external knowledge is easy to re-identify that person. But the problem itself consists in the other pieces of information I can find in that table: in that table we may find confidential attributes, like disease, that in case of re-identification, can be assigned to the subject, exposing him/her.

While applying anonymity we are no more able to use any kind of information to reconstruct the identity of the specific subject, de-identification does only remove explicit identifiers, that as we said before, is not sufficient because does not protect against re-identification through quasi-identifiers and external knowledge.

k-anonymity is a technique used to protect data according to a given threshold from identity disclosure. To enforce this level of protection we adopt two techniques on real data: suppression and generalization.

The released data should be indistinguishably related to no less that a certain number of respondents: there should be at least k real person that could match to that specific tuple in the table; and also each subject in the table should be related to at least k tuples in the table we are releasing. The concept of indistinguishable tuples means that each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents.

If you what to protect your data against attribute disclosure, k-anonymity is not enough.

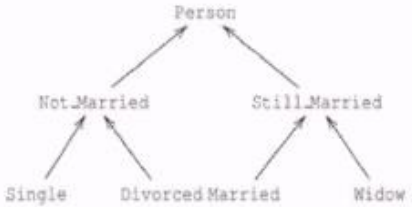
Suppression consists in blanking out some tuples, cells or attribute, depending on the situation. Typically, suppression is applied when generalization is too much to obtain a certain level of anonymity, because tends to lead to higher information loss, that would make data become less useless.

Generalization can operate on value or on domain: we map a hierarchy between a domain and its generalization values, what we obtain is a progressive grouping of values. For example, the attribute “race” have values “black”, “white” and “asian”, that in the first step of generalization may become “person”.

Depending in the preference criteria [minimize absolute distance], we can decide to apply the two techniques and in which terms.

Given the following original relation T_O, compute a 3-minimal generalization of T_O, assuming max_sup=1. To this aim, consider the generalization hierarchy below for attribute *Marital Status* and a generalization hierarchy for attribute *DoB* that removes, in the order, day, month, and the digits of the year of birth from the least to the most significant (e.g., 12/10/1974 → **/10/1974 → **/**/1974 → **/**/197* → **/**/19**).

Marital Status	DoB	Disease
Single	12/10/1974	Flu
Single	08/09/1977	Measles
Single	23/11/1974	HIV
Single	09/04/1974	Yellow fever
Married	16/02/1982	Measles
Married	22/08/1982	Flu
Married	01/06/1982	Malaria
Widow	15/04/1955	Flu
Divorced	13/11/1977	Measles
Divorced	27/12/1977	Measles



	Marital status	DoB	Disease
A	Not married	**/**/74	Flu
B	Not married	**/**/77	Measles
A	Not married	**/**/74	HIV
A	Not married	**/**/74	Yellow fever
C	Still married	**/**/82	Measles
C	Still married	**/**/82	Flu
C	Still married	**/**/82	Malaria
	~~Still married~~	~~**/**/55~~	~~Flu~~
B	Not married	**/**/77	Measles
B	Not married	**/**/77	Measles

Discuss the intuition behind differential privacy for protection respondents in data publication. Illustrate the formal definition of differential privacy. Which are the advantages and disadvantages of differential privacy with respect to k-anonymity?

The differential privacy aims at preventing adversaries from being able to detect the presence or absence of a given individual in a dataset, preventing a single individual to make the difference in the result. The idea behind is that adding an individual should not change much the risk of that individual.

The privacy of a subject is protected whether the result does not depend on a specific information. To test it, we compare results computed over the real world scenario (dataset D) and on the scenario where omitted data of X subject. The difference must be at most equal to the **privacy budget** ε, that represents the noise (from 0 and 1) added to the computation, the trade-off between privacy and accuracy. The larger is ε the higher the distance between the two scenarios: this implies that we would have less noise (so higher utility but less privacy). The smaller is ε the lower is the distance: this implies that we guarantee more privacy but less utility (becomes flatter).

In case we have outliers, one single individual may have high influence, this implies that the global sensistivy is high, thus we must add more noise.

ε can be computed with a sequential composition (sequence of computations over the database with overlapping results, where we sum privacy parameters [#females #diabetes]) or in a parallel composition (sequence of computations of disjoint subset of the database, we look for the max of the sum [#hair color #handedness]).

Differential privacy guarantees also the group privacy but does not work well when we have a high number of outliers. K-anonymity does not guarantee a complete protection but is nice in capturing and with real-world requirements. Besides, differential privacy guarantees a better protection with respect to k-anonymity, but is not easy to enforce.

Describe the difference between the release of macrodata and microdata tables. Illustrate the protection techniques that can be used to protect the release of a count table.

Given the following count table, provide a protected version of the same assuming a threshold of 10 respondents.

	Age				
County	≤ 20	$21-50$	$51-80$	>80	**Total**
C1	15	7	25	3	50
C2	8	12	35	2	57
C3	25	30	50	1	106
C4	2	44	32	3	81
Total	50	93	142	9	294

Releasing macrodata means releasing the result of aggregation or statistical computation performed over the data, while microdata is represented by a table with a row for each subject and attributes defined by the columns.

Macrodata, because of the way it is arranged, can leak pieces of information, especially sensitive. It is classified into two groups (types of tables) depending on the content:

A. Count/Frequency. Each cell contains the number (result of count query) or the percentage (frequency) of respondents that have the same value over all attributes in the table.
B. Magnitude data. Each cell contains an aggregate (sum, mean…) value of a quantity of interest over all attributes in the table.

The advantage of the microdata table is that provides more precise data, is flexible, not precomputed, so provides higher utility because we can apply every type of computation on it. Thus, to protect the anonymity of the respondents, data holders often remove or encrypt explicit identifiers such as names, addresses, phone numbers.

For the count tables, the typical protection techniques are:

- Sampling: we select only a sample of the whole population;
- Special rules: restriction defined on the level of details available in a table [geographical details]. The rules are defined by company or institution, sometimes depend on laws;

- Threshold rules: we fix a threshold to the number of respondents in a cell, to be less than some specific number. If this does not happen, we can apply cell suppression [remove cells], rounding [randomly or in a controlled manner] or confidentially editing [swapping on microdata table].

Country	≤20	21-50	51-80	>80	Total
C1	20	10	30	10	50
C2	0	20	30	0	57
C3	30	30	50	0	106
C4	10	40	30	0	81
Total	50	93	142	9	294

Country	≤50	>50	Total
C1	22	28	50
C2	20	37	57
C3	55	51	106
C4	46	35	81
Total	143	151	294

Country	≤20	21-50	>50	Total
C1 & C2	23	19	65	107
C3 & C4	27	74	86	187
Total	143	93	151	294