Question 2a)

- MySQL
    - MySQL is a pure relational database, and it is relatively easy to set up and manage. It can also handle high concurrency of reading and writing, and It is also known for its reliability. The default engine for the MySQL – InooDB is also ACID Compliance. Main weakness is that it has some limitations with complex data types. It has also been acquired by Oracle; therefore, the development is not community driven and has slowed down its development process in the past few years.
    - MySQL can be used when dealing with high volume websites. This is because it can support large concurrency of reading and writing process. Other than that, MySQL should be considered when dealing applications that stores large number of sensitive datasets as it has built-in user management capabilities.
    - Applications that stores data that are not relational (JSON) should not use MySQL. Other than that, applications that only requires to run on a device, with low concurrency, should not consider using MySQL as it would be an overkill.
- PostgreSQL
    - PostgreSQL is an object-relational database management system, so it serves as a bridge between OOP and relational programming. This allows the user to define objects and table inheritance, which translates to more complicated data structure. It can prevent data corruption and preserves the integrity of data at the transactional level (ACID Compliance). Since PostgreSQL is an open source project, it relies heavily on external tooling to do a lot of basic stuff properly. It requires subject matter expert to put together a reliable Postgres deployment. Also, it may not be as portable as compared to SQLite DB.
    - PostgreSQL can be used when dealing with large datasets with a wide range of datatypes. This is because it natively supports a rich variety of data types, including JSON and XML. It also allows you to define original data types and set up custom functions. When dealing with applications that emphasized on accuracy such as banking/finance industry, PostgreSQL is a suitable pick as it is fully ACID compliant which makes it an ideal choice for OLTP.
    - PostgreSQL has lots of features built-in for handling large datasets and supports high loads. Therefore, it is not suitable for applications that only requires to read-only from a small dataset as it is an overkill. For applications that requires high concurrency of writing along with large data replications, PostgreSQL's architecture is not that efficient comparing with MySQL. Interestingly, that is also the reason why Uber decided to migrate from PostgreSQL to MySQL.
- SQLite
    - SQLite is a compact, fast, and highly reliable database engine. It is a serverless solution which makes it simple to set up and no configurations is required. However, this also means that you cannot access it over the network. It has also no built-in user management which might be an issue when it comes to data privacy.
    - SQLite can be used when transferring data from one system to another, as it is in a well-defined, compact, cross-platform format. Other than that, it is often used as a temporary database in python programs for data analysis. This is because we need to

preprocess (sort, casting types etc.) the given data and it is often easier and quicker to load the data into an in-memory database and use queries to help with it. Personally, I also usually use SQLite for Python scripting purpose, just because it is so easy to set up and has very good integration with the Python language.

- However, SQLite is usually not suitable for very large datasets as it has a limited size of approximately 281 terabytes only. Since it only allows one writer at any instant in time, applications that require high concurrency should not consider SQLite as a suitable database.

Question 2b)

- Elasticsearch
  - Elasticsearch is a full-text, distributed NoSQL Database. It allows for a real-time searching and analyzing of your data. It is also highly reliable as it can redirect data to other locations so that your information is protected. It is also able to conduct any task with a RESTful API, via JSON through HTTP. One of the main weakness is that it has a high learning curve as it is not as simple compare to its competitors. It also does not have multi-language support in terms of handing request and response data (only JSON).
  - Applications that require a full text search should use Elasticsearch. For example, chatbots that deals with customers, we can leverage on Elasticsearch's in-built fuzzy matching. Other than that, we can also use Elasticsearch when searching for an item on an e-commerce website. For example, in our company, we use Elasticsearch to display job and candidate listing which helps to speed up searching of the data.
  - Applications that emphasizes on accuracy should not use Elasticsearch, this is because most of the task is conducted via RESTful API. If the applications do not handle HTTP error well, data loss will occur. Other than that, applications that do not have subject matter expert, it requires a long time to set up, and maintain this database. In addition to that, the documents provided by Elasticsearch is also not very concise which leads to an even higher learning curve.
- MongoDB
  - MongoDB is a widely used document-based database. It is very easy to scale out, replicate, and have high availability. It supports automatic failover process that takes place when the primary node is not accessible. One of the main limitations of MongoDB is that it has a high memory usage. It stores key names for each value pair, and without the functionality of joins, there is data redundancy. There is also a maximum document size of 16mb.
  - MongoDB is suitable as a centralized database. This is because its document-based model will be a great fit to provide a single unified view of the data from different data sources. Applications that stores complex data structures should also use MongoDB as it allows embedding of documents to describe nested structures and tolerate variations in data in generations of documents.
  - Applications that do not have large storage and can tolerate should not use a MongoDB. This is because it has lots of data redundancy and the storage size can be larger than SQL databases. If the application deals with flattened data and has a rigid structure,

MongoDB is not a suitable choice as its advantage in dealing with nested and flexible structure are not utilized.

- Neo4j
    - Neo4j is a graph database. It has a high performance when it comes to querying deeply connected data with complex relationships. It is also ACID compliant and is very flexible when it comes to managing the ontology of your graph. The main weakness of Neo4j is that it is designed as a master-slave architecture, therefore only 1 the master node is handling all the writes.
    - Applications that utilized recommendation engines should use Neo4j. Since Neo4j is built on entities and relations between them, it is able to correlate data and detect new interest of a person. Other than that, applications such as social network that emphasizes on people connection should also use Neo4j. It enhances the overall performance and helps the user to understand their data better.
    - Neo4j is not suitable for applications that do not exploit relationships. For example, applications that is build for customer transaction with analytics is probably not a good fit for Neo4j. Other than that, applications that requires a high concurrency of writing should not use Neo4j, as it is only able to scale vertically but not horizontally.

Question 2c)

	ETL (Extract Transform Load)	ELT (Extract Load Transform)
Differences	- It extracts the data from different data sources, then stores it in a staging area where it is cleaned and transformed, and finally stored in a data warehouse.	- It extracts the data from different sources, then stores the data in a data lake (no staging area is required), transformation is only done on the data you need.
Benefit	- It improves the speed and efficiency of data analysis as most of transformation process has been done beforehand. - Allows the removal/encryption of sensitive data before loading it into a data warehouse	- It provides high flexibility and ease of storing new unstructured data. - It has a higher data availability as compared to ETL as the data is stored immediately, and the users can decide which data they need to transform and analyze.

Weakness	- Since data is transformed before it is loaded, Users are expected to understand their business needs well to understand which data is useful. - Data availability is slower compare to ELT process as transformation is done beforehand	- This process does not allow the removing/encryption of sensitive data which might violate data compliance standards. - More system resources are required as we are storing raw data

Popular ETL Tools

- Airflow
  - Strength:
    - Allows you to programmatically author, schedule, and monitor workflows
    - Has very good built in user interface which allows user to visualize pipelines running in production
  - Weakness:
    - Does not allow versioning of workflows
    - Does not support parametrized workflows, as airflow DAGs are supposed to run on fixed schedules and not receive inputs.
  - Application that requires lots of automation of workflow/pipelines should utilize Airflow. It also provides a standard way to develop, deploy and monitor these pipelines.
- Kafka
  - Strength
    - High throughput – capable of handling high-velocity and high-volume data
    - Fault tolerant – If a leader fails, a replica can take over as the new leader
  - Weakness
    - Does not come with built in set of management and monitoring tools, which result in difficult ability to visualize what is happening
    - Each subscriber can only match to exact topic name, that means it does not support wildcard topic selection.
  - Kafka is used for stream processing, and real-time analysis. This is because it has features that guarantees consistency and availability of data.

- Programming Language - (Python, SQL)
    - Python
        - Strength:
            - Able to handle large datasets (Pandas, Dask)
            - Well integrated with most of the cloud services provided
            - Lots of open source library that helps with the ETL process such as Mara, Prefect and Petl
        - Weakness:
            - Slow in speed, this is because python executes with the help of an interpreter instead of the compiler
            - It is a weakly typed language, which allows lots of hidden errors
        - Python should be used as the main ETL language. This is because it allows the data consumer (data scientist/analyst) to understand what is happening in the transformation process.
    - SQL
        - Strength:
            - Powerful query language, with simple commands that can be learned quickly
            - Can handle large transformation of data in a single query
        - Weakness
            - Cannot handle complex transformation logic
            - Becomes hard to maintain when the size of the query increases
        - SQL should be used when doing simple cleaning for a large dataset. Other than that, simple aggregation should also be done using SQL as it has built-in functions such as window functions, group functions that help speeds up this process.