Azure Mini Project: End-to-end ETL

1) Why should one use Azure Key Vault when working in the Azure environment? What are the alternatives to using Azure Key Vault? What are the pros and cons of using Azure Key Vault?

Azure Key Vault should be used for the following reasons:

- Centralize application secrets
- Securely store secrets and keys
- Monitor access and use
- Simple administration of application secrets

A few of Azure Key Vault's alternatives are: AWS Secrets Manager, HashiCorp Vault, LastPass.

The key advantages of this technology are:

- Increase security and control over keys and passwords
- Create and import encryption keys in minutes
- Applications have no direct access to keys
- Use FIPS 140-2 Level 2 and Level 3 validated HSMs
- Reduce latency with cloud scale and global redundancy
- Simplify and automate tasks for SSL/TLS certificates

2) How do you achieve the loop functionality within an Azure Data Factory pipeline? Why would you need to use this functionality in a data pipeline?

Loop functionality within Data Factory is achieved via ForEach Activity. This activity is used to iterate over a collection and executes specified activities in a loop. The loop implementation of this activity is like Foreach looping structure in programming languages.

3) What are expressions in Azure Data Factory? How are they helpful when designing a data pipeline (please explain with an example)?

JSON values in the definition can be literal or expressions that are evaluated at runtime. For example:

"name": "value"

Or

"name": "@pipeline().parameters.password"

Expressions can appear anywhere in a JSON string value and always result in another JSON value. If a JSON value is an expression, the body of the expression is extracted by removing the at-sign (@). If a literal string is needed that starts with @, it must be escaped by using @@. The following examples show how expressions are evaluated.

JSON value	Result
"parameters"	The characters 'parameters' are returned.
"parameters[1]"	The characters 'parameters[1]' are returned.
"@@"	A 1 character string that contains '@' is returned.
" @"	A 2 character string that contains ' @' is returned.

4) What are the pros and cons of parametrizing a dataset in Azure Data Factory pipeline's activity?

Parameterization is useful because it allows for a much more flexible ETL solution, which will dramatically reduce the cost of solution maintenance and can save a tremendous amount of time.

Parametrizing a dataset eliminates the need to create new dataset objects for every file needed for an activity.

5) What are the different supported file formats and compression codecs in Azure Data Factory? When will you use a Parquet file over an ORC file? Why would you choose an AVRO file format over a Parquet file format?

Azure Data Factory supports the following file formats:

- Avro format
- Binary format
- Delimited text format
- Excel format
- JSON format
- ORC format
- Parquet format
- XML format

If you have a lot of complex nested columns in your dataset and often only query a subset of the subcolumns, Parquet would be a good choice. Parquet is implemented using the record shredding and assembly algorithm described in the Dremel paper, which allows you to access and retrieve subcolumns without pulling the rest of the nested column.

ORC should be used instead of Parquet as ORC has the best compression rate due to its columnar format.

Avro is better to use over Parquet if the data schema needs to be updated very often and if there is a need for high compatibility for old/new applications.

ORC works best with Hive (since it is made for Hive). Spark provides great support for processing Parquet formats. Avro is often a good choice for Kafka.