Quiz 1

1.
Which of the following is NOT correct on data extraction?

○	Many HTML documents are auto-generated these days, making it relatively easier to precisely target relevant patterns in a document
○	XML and JSON are already structured and can convey hierarchical structure of the data well
◉	CSV data is always normalised, making it an ideal candidate for database import tasks
○	Processing text from a PDF document requires understanding its layout as well as the content

2.
Select the NOT correct statements about Impedance Mismatch?

◉	It is a translation layer between the objects in the application code and the database model of tables/row/columns
○	It is one of the problems mostly related to relational databases
○	It is also called "Object-Relational Mismatch”
○	Refers to the problem of a mismatch between application data model (your business objects) and data model for storage (in relational tables)

3.
Which statement is NOT true about NoSQL databases

○	NoSQL databases can have no schema
○	NoSQL databases are suitable for embedded document-like data model
○	MongoDB is an example of NoSQL database
◉	Performance-wise, NoSQL databases can only scale vertically

4.

Which of the following can be considered as metadata?

○	The description of an object and its type
○	The time an object is created or modified
○	Information about the origin of the object
◉	All of the above mentioned options

5.

Consider following Python code which using SQLAlchemy to construct DB schema, please choose the correct code which can generate the same table (If you have no idea about SQLAlchemy, google it):

```
class Contact(db.Model):
    __tablename__ = 'contacts'
    id = db.Column(db.Integer, primary_key=True)
    first_name = db.Column(db.String(100))
    last_name = db.Column(db.String(100))
    phone_number = db.Column(db.String(32))
    address = db.Column(db.String(100))
    post_code = db.Column(db.Integer)

    def __repr__(self):
        return '<Contact {0} {1}: {2}>'.format(self.first_name,
                                               self.last_name,
                                               self.phone_number,
                                               self.address,
                                               self.post_code)
```

○

```
CREATE TABLE CONTACTS(
    ID INT PRIMARY KEY        NOT NULL,
    FIRST_NAME     CHAR(100)  NOT NULL,
    LAST_NAME      CHAR(100)  NOT NULL,
    PHONE_NUMBER   CHAR(32)   NOT NULL,
    ADRESS         CHAR(100)  NOT NULL,
    POST_CODE      INT,
);
```

◉

```
CREATE TABLE CONTACTS(
    ID INT PRIMARY KEY        NOT NULL,
    FIRST_NAME     CHAR(100),
    LAST_NAME      CHAR(100),
    PHONE_NUMBER   CHAR(32),
    ADRESS         CHAR(100),
    POST_CODE      INT,
);
```

```
CREATE TABLE CONTACTS(
    ID INT PRIMARY KEY        NOT NULL,
    FIRST_NAME     CHAR(100),
    LAST_NAME      CHAR(100),
    PHONE_NUMBER   CHAR(32),
    ADRESS         CHAR(100),
    POST_CODE      INT NOT NULL,
);
```

```
CREATE TABLE CONTACTS(
    ID INT PRIMARY KEY        NOT NULL,
    FIRST_NAME     CHAR(100),
    LAST_NAME      CHAR(100),
    PHONE_NUMBER   CHAR(32),
    ADRESS         CHAR(100),
    POST_CODE      INT(32),
);
```

```
CREATE TABLE CONTACTS(
    ID INT PRIMARY KEY,
    FIRST_NAME     CHAR(100),
    LAST_NAME      CHAR(100),
    PHONE_NUMBER   CHAR(32),
    ADRESS         CHAR(100),
    POST_CODE      INT,
);
```

6.

What's the relational pattern for following code:

```
from sqlalchemy import Table, Column, Integer, ForeignKey
from sqlalchemy.orm import relationship
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

association_table = Table('association', Base.metadata,
    Column('left_id', Integer, ForeignKey('left.id')),
    Column('right_id', Integer, ForeignKey('right.id'))
)

class Parent(Base):
    __tablename__ = 'left'
    id = Column(Integer, primary_key=True)
    children = relationship("Child",
                            secondary=association_table,
                            backref="parents")

class Child(Base):
    __tablename__ = 'right'
    id = Column(Integer, primary_key=True)
```

○	Many-To-Many
○	One-To-One
◉	One-To-Many
○	Many-To-One