In [1]:
```
#S14 T01:base de dades NoSQL
```

In [2]:
```
#Nivell 1
```

In [3]:
```
#Exercici 1
```

In [4]:
```
#Crea una base de dades NoSQL utilitzant MongoDB.
#Afegeix-li algunes dades d'exemple que et permetin comprovar
#que ets capaç de processar-ne la informació de manera bàsica.
```

In [5]:
```
#Abans de crear una base de dades NoSQL procedim a 3 passos:
#1. Instal.lem MongoDB.
#2. Creem un compte d'usuari a Atlas MongoDB.
#3. Instal.lem Mongoshell per interactuar mitjançant comandes.
```

In [6]:
```
#Mitjançant la plataforma Atlas MongoDB implementem una base de dades NoSQL "MyFirst
#Aquesta base de dades conté dades referents a equipaments de la Ciutat de Barcelona
#opendatabcn_llista-equipaments_cultura-js.json
#Desem una captura de la plataforma al github.
```

In [7]:
```
#Exercici 2
```

In [8]:
```
#Connecta la base de dades NoSQL a Python utilitzant per exemple pymongo.
```

In [9]:
```
#Llibreries import
from pymongo import MongoClient
```

In [10]:
```
#Connexió a Atlas Mongodb des de python.

def get_database():
    from pymongo import MongoClient
    import pymongo

    #Preparem la connexió de python a mongodb mitjançant pymongo.
    CONNECTION_STRING = "mongodb://dbusr:dbusr@cluster0-shard-00-01.a3myf.mongodb.ne

    #Connexió amb MongoClient.
    myclient = MongoClient(CONNECTION_STRING)

    #Retornem la base de dades.
    return myclient['myFirstDatabase']

# This is added so that many files can reuse the function get_database()
if __name__ == "__main__":

    #Mostrem la base de dades
    dbname = get_database()
    print(dbname)
```

```
Database(MongoClient(host=['cluster0-shard-00-01.a3myf.mongodb.net:27017'], document
_class=dict, tz_aware=False, connect=True), 'myFirstDatabase')
```

In [11]:
```
#Nivell 2
```

In [12]:
```
#Exercici 1
```

In [13]:
```
#Carrega algunes consultes senzilles a un Pandas Dataframe.
```

In [14]:
```
#Estructura de la base de dades NoSQL
#{
#    _id: #
#    name:"" #varchar
#    district_name:"" #varchar
#    address:"" #varchar
#    address_town:"" #varchar
#    zip_code:"" #varchar
#    geo_X: #double
#    geo_Y: #double
#}
```

In [15]:
```
#Mostrem la base de dades
dbname = get_database()
print('El nom de la database en curs:', dbname.name)
```

```
El nom de la database en curs: myFirstDatabase
```

In [21]:
```
item_A = {
    "_id":"623b3bcda0f6294b4c03ee75",
    "name":"Discoteca mojito club",
    "district_name":"Sants-Montjuïc",
    "address":"C Rosselló",
    "address_town":"Barcelona",
    "zip_code":"08004",
    "geo_X":429966.28604534914,
    "geo_Y":4580403.151166491
}
```

In [22]:
```
item_B = {
    "_id":"623c44246c6af32637a28a4f",
    "name":"Sala Espai Lliure",
    "district_name":"Sants-Montjuïc",
    "address":"Pl Margarida Xirgu",
    "address_town":"Barcelona",
    "zip_code":"08004",
    "geo_X":429538.34523939574,
    "geo_Y":4580249.751575231
}
```

In [23]:
```
item_C = {
   "_id":"623c917f9fb3c7bf7798e293",
    "name":"Bar Marsella",
    "district_name":"Ciutat Vella",
    "address":"C Sant Pau",
    "address_town":"Barcelona",
    "zip_code":"08001",
    "geo_X":430698.3507857741,
    "geo_Y":4581082.246688013
}
```

In [24]:
```
myFD_name = dbname["myFirstDatabase"]
```

In [25]:
```
print(myFD_name)
```

```
Collection(Database(MongoClient(host=['cluster0-shard-00-01.a3myf.mongodb.net:2701
7'], document_class=dict, tz_aware=False, connect=True), 'myFirstDatabase'), 'myFirs
tDatabase')
```

In [26]:
```
# Inserir dades/document NoSQL a la base de dades.
myFD_name.insert_many([item_A, item_B])
```

```
--------------------------------------------------------------------------
```

```
ServerSelectionTimeoutError                Traceback (most recent call last)
<ipython-input-26-f89e56cad847> in <module>
      1 # Inserir dades/document NoSQL a la base de dades.
----> 2 myFD_name.insert_many([item_A, item_B])

~\anaconda3\lib\site-packages\pymongo\collection.py in insert_many(self, documents,
 ordered, bypass_document_validation, session)
    613         blk = _Bulk(self, ordered, bypass_document_validation)
    614         blk.ops = [doc for doc in gen()]
--> 615         blk.execute(write_concern, session=session)
    616         return InsertManyResult(inserted_ids, write_concern.acknowledged)
    617

~\anaconda3\lib\site-packages\pymongo\bulk.py in execute(self, write_concern, sessio
n)
    457                 self.execute_no_results(sock_info, generator)
    458         else:
--> 459             return self.execute_command(generator, write_concern, session)

~\anaconda3\lib\site-packages\pymongo\bulk.py in execute_command(self, generator, wr
ite_concern, session)
    349
    350         client = self.collection.database.client
--> 351         with client._tmp_session(session) as s:
    352             client._retry_with_session(self.is_retryable, retryable_bulk, s,
self)
    353

~\anaconda3\lib\contextlib.py in __enter__(self)
    111         del self.args, self.kwds, self.func
    112         try:
--> 113             return next(self.gen)
    114         except StopIteration:
    115             raise RuntimeError("generator didn't yield") from None

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _tmp_session(self, session,
close)
   1654             return
   1655
-> 1656         s = self._ensure_session(session)
   1657         if s:
   1658             try:

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _ensure_session(self, sessi
on)
   1641             # Don't make implicit sessions causally consistent. Applications
   1642             # should always opt-in.
-> 1643             return self.__start_session(True, causal_consistency=False)
   1644         except (ConfigurationError, InvalidOperation):
   1645             # Sessions not supported.

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in __start_session(self, impli
cit, **kwargs)
   1592     def __start_session(self, implicit, **kwargs):
   1593         # Raises ConfigurationError if sessions are not supported.
-> 1594         server_session = self._get_server_session()
   1595         opts = client_session.SessionOptions(**kwargs)
   1596         return client_session.ClientSession(self, server_session, opts, impl
icit)

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _get_server_session(self)
   1627     def _get_server_session(self):
   1628         """Internal: start or resume a _ServerSession."""
-> 1629         return self._topology.get_server_session()
   1630
   1631     def _return_server_session(self, server_session, lock):

~\anaconda3\lib\site-packages\pymongo\topology.py in get_server_session(self)
    532             # Sessions are always supported in load balanced mode.
```

```
    533             if not self._settings.load_balanced:
--> 534                 session_timeout = self._check_session_support()
    535             else:
    536                 # Sessions never time out in load balanced mode.

~\anaconda3\lib\site-packages\pymongo\topology.py in _check_session_support(self)
    518                     )
    519             elif not self._description.readable_servers:
--> 520                 self._select_servers_loop(
    521                     readable_server_selector, self._settings.server_selectio
n_timeout, None
    522                 )

~\anaconda3\lib\site-packages\pymongo\topology.py in _select_servers_loop(self, sele
ctor, timeout, address)
    221             # No suitable servers.
    222             if timeout == 0 or now > end_time:
--> 223                 raise ServerSelectionTimeoutError(
    224                     "%s, Timeout: %ss, Topology Description: %r"
    225                     % (self._error_message(selector), timeout, self.descript
ion)

ServerSelectionTimeoutError: cluster0-shard-00-01.a3myf.mongodb.net:27017: connectio
n closed, Timeout: 30s, Topology Description: <TopologyDescription id: 623c8fd100b61
3bdc25d937f, topology_type: Unknown, servers: [<ServerDescription ('cluster0-shard-0
0-01.a3myf.mongodb.net', 27017) server_type: Unknown, rtt: None, error=AutoReconnect
('cluster0-shard-00-01.a3myf.mongodb.net:27017: connection closed')>]>
```

In [37]:
```
# Inserir dades/document NoSQL a la base de dades.
myFD_name.insert_one(item_C)
```

```
---------------------------------------------------------------------------
ServerSelectionTimeoutError                Traceback (most recent call last)
<ipython-input-37-5b0e1adc033c> in <module>
      1 # Inserir dades/document NoSQL a la base de dades.
      2 collection_name = dbname["myFirstDatabase"]
----> 3 collection_name.insert_one(item)

~\anaconda3\lib\site-packages\pymongo\collection.py in insert_one(self, document, by
pass_document_validation, session)
    540         write_concern = self._write_concern_for(session)
    541         return InsertOneResult(
--> 542             self._insert_one(
    543                 document,
    544                 ordered=True,

~\anaconda3\lib\site-packages\pymongo\collection.py in _insert_one(self, doc, ordere
d, check_keys, write_concern, op_id, bypass_doc_val, session)
    492             _check_write_command_response(result)
    493
--> 494         self.__database.client._retryable_write(acknowledged, _insert_comman
d, session)
    495
    496         if not isinstance(doc, RawBSONDocument):

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _retryable_write(self, retr
yable, func, session)
   1383     def _retryable_write(self, retryable, func, session):
   1384         """Internal retryable write helper."""
-> 1385         with self._tmp_session(session) as s:
   1386             return self._retry_with_session(retryable, func, s, None)
   1387

~\anaconda3\lib\contextlib.py in __enter__(self)
    111         del self.args, self.kwds, self.func
    112         try:
--> 113             return next(self.gen)
    114         except StopIteration:
    115             raise RuntimeError("generator didn't yield") from None
```

```
~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _tmp_session(self, session, close)
   1654             return
   1655
-> 1656         s = self._ensure_session(session)
   1657         if s:
   1658             try:

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _ensure_session(self, session)
   1641             # Don't make implicit sessions causally consistent. Applications
   1642             # should always opt-in.
-> 1643             return self.__start_session(True, causal_consistency=False)
   1644         except (ConfigurationError, InvalidOperation):
   1645             # Sessions not supported.

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in __start_session(self, implicit, **kwargs)
   1592     def __start_session(self, implicit, **kwargs):
   1593         # Raises ConfigurationError if sessions are not supported.
-> 1594         server_session = self._get_server_session()
   1595         opts = client_session.SessionOptions(**kwargs)
   1596         return client_session.ClientSession(self, server_session, opts, implicit)

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _get_server_session(self)
   1627     def _get_server_session(self):
   1628         """Internal: start or resume a _ServerSession."""
-> 1629         return self._topology.get_server_session()
   1630
   1631     def _return_server_session(self, server_session, lock):

~\anaconda3\lib\site-packages\pymongo\topology.py in get_server_session(self)
    532             # Sessions are always supported in load balanced mode.
    533             if not self._settings.load_balanced:
--> 534                 session_timeout = self._check_session_support()
    535             else:
    536                 # Sessions never time out in load balanced mode.

~\anaconda3\lib\site-packages\pymongo\topology.py in _check_session_support(self)
    518                     )
    519             elif not self._description.readable_servers:
--> 520                 self._select_servers_loop(
    521                     readable_server_selector, self._settings.server_selection_timeout, None
    522                 )

~\anaconda3\lib\site-packages\pymongo\topology.py in _select_servers_loop(self, selector, timeout, address)
    221             # No suitable servers.
    222             if timeout == 0 or now > end_time:
--> 223                 raise ServerSelectionTimeoutError(
    224                     "%s, Timeout: %ss, Topology Description: %r"
    225                     % (self._error_message(selector), timeout, self.description)

ServerSelectionTimeoutError: cluster0-shard-00-01.a3myf.mongodb.net:27017: connection closed, Timeout: 30s, Topology Description: <TopologyDescription id: 623b06a4152a22837b728a7e, topology_type: Unknown, servers: [<ServerDescription ('cluster0-shard-00-01.a3myf.mongodb.net', 27017) server_type: Unknown, rtt: None, error=AutoReconnect('cluster0-shard-00-01.a3myf.mongodb.net:27017: connection closed')>]>
```

In [69]:

```
#Mostrem els registres existents a la base de dades NoSql.
for item in myFD_name.find():
    print(item)
```

```
---------------------------------------------------------------------------
ServerSelectionTimeoutError                Traceback (most recent call last)
```

```
<ipython-input-69-19cc851bbcef> in <module>
      1 #Mostrem els registres existents a la base de dades NoSql.
----> 2 for item in myFD_name.find():
      3     print(item)

~\anaconda3\lib\site-packages\pymongo\cursor.py in next(self)
   1193         if self.__empty:
   1194             raise StopIteration
-> 1195         if len(self.__data) or self._refresh():
   1196             return self.__data.popleft()
   1197         else:

~\anaconda3\lib\site-packages\pymongo\cursor.py in _refresh(self)
   1086
   1087         if not self.__session:
-> 1088             self.__session = self.__collection.database.client._ensure_sessi
on()
   1089
   1090         if self.__id is None:  # Query

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _ensure_session(self, sessi
on)
   1641             # Don't make implicit sessions causally consistent. Applications
   1642             # should always opt-in.
-> 1643             return self.__start_session(True, causal_consistency=False)
   1644         except (ConfigurationError, InvalidOperation):
   1645             # Sessions not supported.

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in __start_session(self, impli
cit, **kwargs)
   1592     def __start_session(self, implicit, **kwargs):
   1593         # Raises ConfigurationError if sessions are not supported.
-> 1594         server_session = self._get_server_session()
   1595         opts = client_session.SessionOptions(**kwargs)
   1596         return client_session.ClientSession(self, server_session, opts, impl
icit)

~\anaconda3\lib\site-packages\pymongo\mongo_client.py in _get_server_session(self)
   1627     def _get_server_session(self):
   1628         """Internal: start or resume a _ServerSession."""
-> 1629         return self._topology.get_server_session()
   1630
   1631     def _return_server_session(self, server_session, lock):

~\anaconda3\lib\site-packages\pymongo\topology.py in get_server_session(self)
    532             # Sessions are always supported in load balanced mode.
    533             if not self._settings.load_balanced:
--> 534                 session_timeout = self._check_session_support()
    535             else:
    536                 # Sessions never time out in load balanced mode.

~\anaconda3\lib\site-packages\pymongo\topology.py in _check_session_support(self)
    518                     )
    519             elif not self._description.readable_servers:
--> 520                 self._select_servers_loop(
    521                     readable_server_selector, self._settings.server_selectio
n_timeout, None
    522                 )

~\anaconda3\lib\site-packages\pymongo\topology.py in _select_servers_loop(self, sele
ctor, timeout, address)
    221             # No suitable servers.
    222             if timeout == 0 or now > end_time:
--> 223                 raise ServerSelectionTimeoutError(
    224                     "%s, Timeout: %ss, Topology Description: %r"
    225                     % (self._error_message(selector), timeout, self.descript
ion)

ServerSelectionTimeoutError: cluster0-shard-00-01.a3myf.mongodb.net:27017: connectio
```

```
n closed, Timeout: 30s, Topology Description: <TopologyDescription id: 623c8cc3152a2
2837b728a88, topology_type: Unknown, servers: [<ServerDescription ('cluster0-shard-0
0-01.a3myf.mongodb.net', 27017) server_type: Unknown, rtt: None, error=AutoReconnect
('cluster0-shard-00-01.a3myf.mongodb.net:27017: connection closed')>]>
```

In [ ]:
```
#Nivell 3
```

In [ ]:
```
#Exercici 1
```

In [ ]:
```
#Genera un resum estadístic de la informació que conté la base de dades.
```