**PART 1 - MongoDB Code**  
**Tasks**

**-- 1. Create a database called A3\_LastnameFirstname**

use A3\_BishopNoriegaManuel

**-- 2. Create a collection called "games"**

I could use db.createCollection() but since collections are created implicitly the  
first time they are referenced, I'm just going to use the command to add documents   
to collection "games" and that will create the collection.

**-- 3. Add 3 to 5 games to the collection, meeting the specs**

db.games.insertMany([  
{title:"quake2",genre:"fps",rating:89,  
reviews:[{date:"2001-01-01",rating:9.5,text:"nice game!..."}]} ,  
 {title:"quake3",genre:"fps",rating:95,  
reviews:[{date:"2003-01-01",rating:10,text:"the best game I've ever played!!!"},  
{date:"2003-01-01",rating:7,text:"meh!"}]},  
{title:"sim city 2000",genre:"strategy",rating:92,  
reviews:[{date:"2003-01-01",rating:9.3,text:"Amazing! I could spend the whole day developing my city!"},  
{date:"2003-01-01",rating:3,text:"Boooooooooooooooooooooring.... :\ "}]},  
 {title:"Genshin Impact",genre:"rpg",rating:97,  
reviews:[{date:"2003-01-01",rating:10,text:"well, my daugther loves it... so I gave it a try and got hooked :D"},  
{date:"2023-01-01",rating:8.7,text:"It's not bad, but there's room for improvement"}]}])  
finding by title

**-- 4. Finds one of these games by title**

db.games.find({title:"quake3"})

**-- 5. Update the rating of 1 of these games based on it's title.**

db.games.updateOne({title:"quake3","reviews.rating":10},{$set:{"reviews.$.rating":9.5}})

**-- 6. Find all games who's genre is Strategy**

db.games.find({genre:"strategy"})

**-- 7. Find all games that have a rating above 70.**

db.games.find({"rating":{$gt:70}})

-------------------------------------------------------------------------------  
**PART 2 - (Free type)**

Considering the current requirements, I strongly suggest to use a NoSQL database to build this analytics application. The main reasons are:

- Given that collected data would refer to a wide variety of factor, from many different sources, trying to put it all together in a well defined and organized structure as Relational DB's require could take important amounts of time, time we don't have. NoSQL databases can work without this rigid structure and still make data accessible in a timely fashion.  
- Data collected and stored for sure will be really huge and despite relational databases could deal with it, queries retrieving data could become really slow, again ralenting performance and jeopardizing our main goal.  
- It must be emphasized, given the characteristics of data we'll be handling that we need a flexible solution able to be adapted in order to keep working despite the different data shapes we may found in the future.  
- We don't need to keep consistency between different data pieces, we mainly want to get them ready to be analyzed as fast as possible.

SUMMARIZING. NoSQL databases solutions were precisely designed for that purpose, gathering and analysis of Big Data. They perfrom at high speeds. Scalable and easy to update schemas. Compatible with all major programming languages and can take advantage of the cloud to implement storage. It is not by chance that companies such as Google, Facebook, Amazon or Shopify rely on them to handle their data.

-------------------------------------------------------------------------------

**PART 3 - SQL Code**

**Tasks**  
**-- 1. Create a database called A3\_LastnameFirstname**

create database if not exists A3\_BishopNoriegaManuel;  
use A3\_BishopNoriegaManuel;

**-- 2. Create 2 tables: EMPLOYEE (ID, name, email), CUSTOMER (ID, account, name). There are no relationships.**

drop table if exists employee;  
create table employee(  
id int primary key auto\_increment,  
name varchar(50),  
email varchar(50));

drop table if exists customer;  
create table if not exists customer(  
id int primary key auto\_increment,  
account varchar(25),  
name varchar(50));

**-- 3. Create an SQL user for an admin (username: admin, password: admin),**  
**--    a user for a manager (username: manager, password: Manager!@#$)**  
**--    a user for an assistant (username: assistant, password: Assistant!@#$)**

create user 'admin'@'localhost' identified by 'admin';  
create user 'manager'@'localhost' identified by 'Manager!@#$';  
create user 'assistant'@'localhost' identified by 'Assistant!@#$';

**-- 4. Give your users the following privileges**  
**--    - Admin has access to the entire database. Admin cannot grant privileges**  
**--    - Manager can select, update, and insert on the employee table**  
**--    - Assistant can lookup information from both tables, but cannot do anything else**

grant all privileges on A3\_BishopNoriegaManuel.\* to 'admin'@'localhost';  
grant select, update, insert on A3\_BishopNoriegaManuel.employee to 'manager'@'localhost';  
grant select on A3\_BishopNoriegaManuel.\* to 'assistant'@'localhost';

**-- 5. Write a command that will export your entire database into an SQL file located in c:\temp\A3\_LastnameFirstname\_export.sql**

mysqldump -uroot -p A3\_BishopNoriegaManuel>c:\temp\A3\_BishopNoriegaManuel\_export.sql

**6. Write the command to restore the database into a new database named A3\_Names\_Restored**

mysql -uroot -p A3\_BishopNoriegaManuel\_Restored