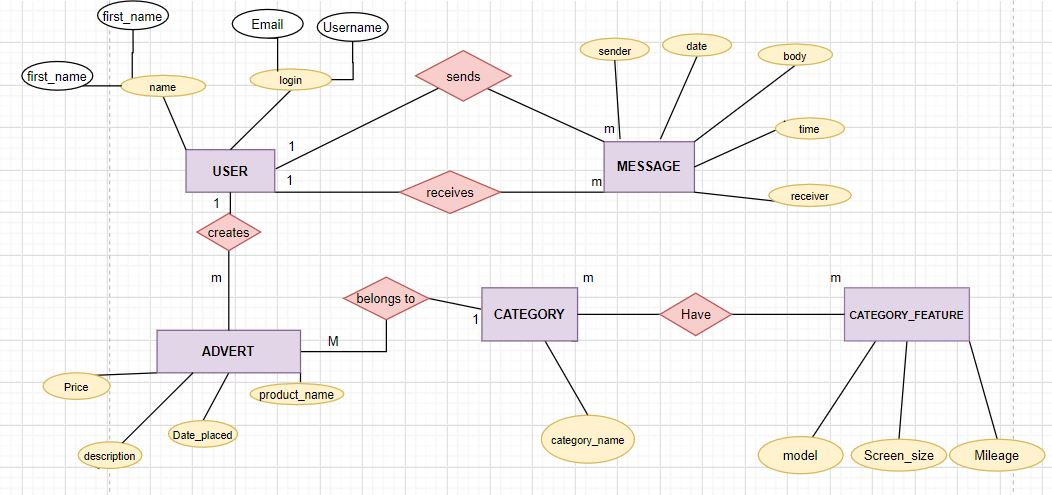
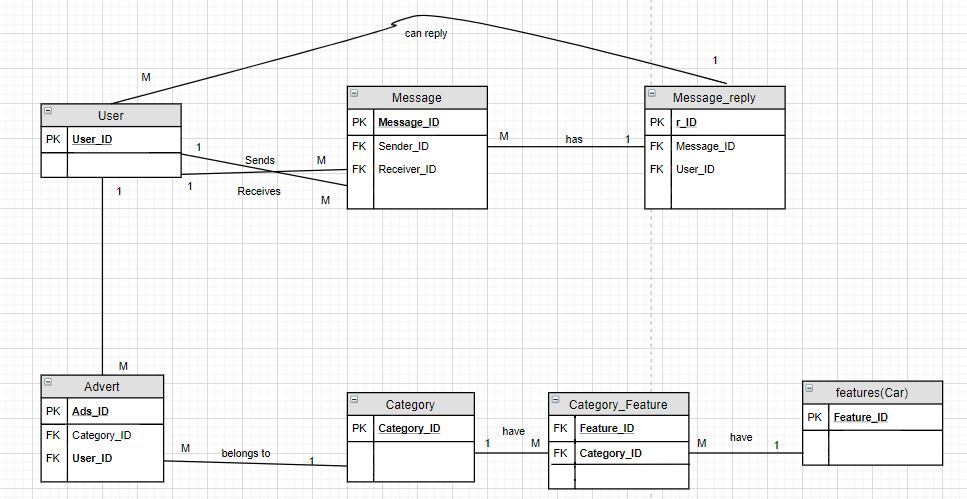
**Question 1**

**1a**

**ER Diagram for a classified advert website**



**1b**



The table below shows the primary key of each table and the reasons behind the choice of key:

|  |  |  |
| --- | --- | --- |
| Table\_Names | Primary\_Key | Reasons |
| User | User\_ID | It uniquely identifies a user (buyer/seller), no user is expected to have the same ID. It is set to auto-increment so that each time each user registers on the site, the user table gets updated immediately. |
| Advert | Ads\_ID | This is a unique identifier of each item for sale. Every item must have its own key for easy reference and it auto-increments so that each time an item is added, it gets a new ID. |
| Category | Category\_ID | This uniquely identifies the classification of each item for sale i.e an item must belong to one classification. |
| Features | Feature\_ID | Every feature is a table on its own so there is a need for a unique identifier for each feature table |
| Category\_Feature | Feature\_ID  Category\_ID | These are composite keys for the purpose of eliminating the many-to-many relationship |

**Normalization**

**User**

|  |  |  |
| --- | --- | --- |
| **User\_ID** | Username | Date\_of\_Birth |
| **1001** | Younggee | 5/11/1990 |
| **1002** | Musty | 6/1/1982 |
| **1003** | Samia | 23/8/2002 |

The User table above shows that the table is in third normal form (3NF) following the rules of normalization:

* The non-key fields are functionally depended on the primary key
* There is no transitive dependency.

**Advert**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ads\_ID** | Seller\_id | Item\_Name | Category\_ID | Date\_placed | Price($) |
| **1** | 1003 | Honda | 1 | 2/3/2020 | 5000 |
| **2** | 1002 | Iphone 7 | 3 | 6/9/2020 | 9000 |

From the illustration above we can see that the Advert (Product) is in Second Normal form (2NF) because all the non key field are fully dependent on Ads\_ID which is the primary key and this satisfies the condition for 2NF except for the **Price($)** field which is transitively dependent on the primary key through the Item\_Name.

The Advert table can further be normalized to achieve 3NF by removing the Item\_Name and Price ($) to eliminate transitivity, I choose to leave it in 2NF because field was not mentioned in my design.

**Category**

|  |  |
| --- | --- |
| **Category\_ID** | Category\_Name |
| **1** | Car |
| **2** | Electronics |

The above table is in 2NF because the non-key field is dependent on the primary key.

**Feature**

|  |  |
| --- | --- |
| **Feature\_ID** | Feature\_name |
| **1** | Model |
| **2** | Processor speed |

The above table is in 2NF because it satisfies the rules of normalization i.e the Feature\_name is dependent on the primary-key.

**Categoy\_Feature**

|  |  |  |
| --- | --- | --- |
| **Category\_ID** | **Feature\_ID** | Feature\_name |
| **1** | **1** | Model |
| **2** | **2** | Processor speed |

The above table is in 2NF because

* The feature field is fully dependent on the composite keys

**Message**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Message\_ID** | Sender\_ID | Receiver\_ID | Message\_Content | Date |
| **1** | 1001 | 1002 | Hello! How re u | 22/11/2020 |
| **2** | 1010 | 1008 | Where re u | 16/6/2020 |

The table above is in 2NF because all non-key fields are dependent on the primary-key i.e they won’t make meaning when they exist alone.

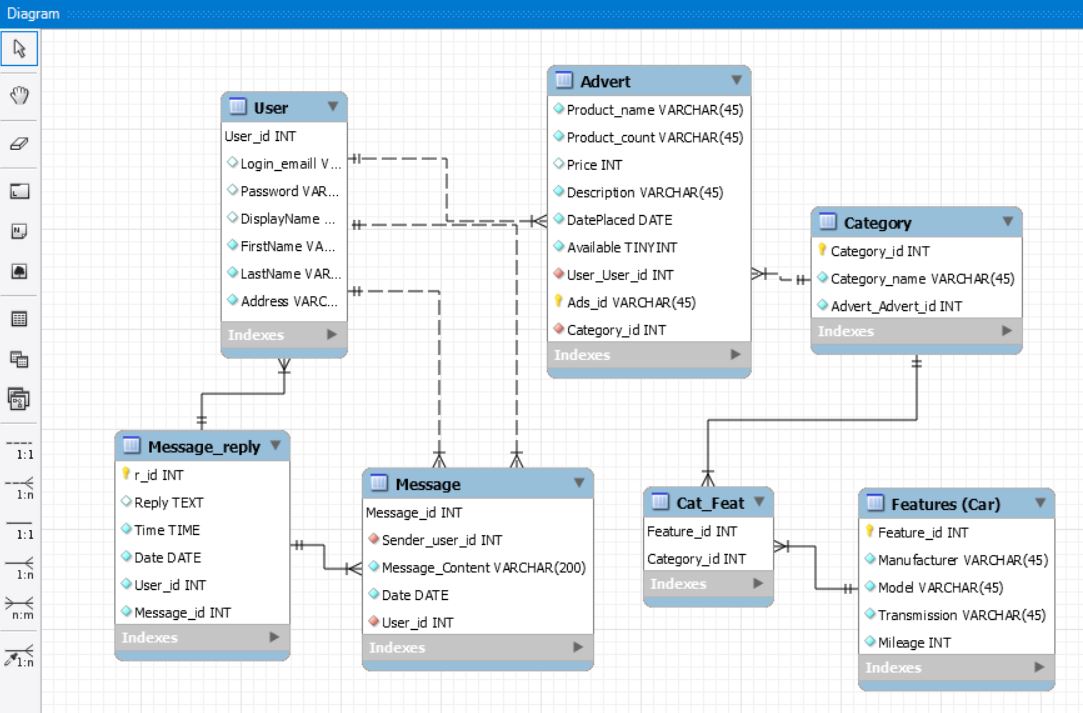
**Message\_reply**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reply\_ID** | Message\_ID | Reply | User\_ID | Date |
| **1** | 1 | I am fine | 1002 | 23/11/2020 |
| **2** | 2 | Am at home | 1008 | 17/6/2020 |

The above table is also in 2NF but can further be decomposed to achieve 3NF. It is kept in 2NF because messages can be redundant since it is exchanged by users from the same User table.

**1c**

Database schema Design



MySQL Workbench was used to design the above database schema for the normalized relations. Crow’s foot notation has been used to show the cardinality of the relationship between tables. Some possible fields are been included on each table for the purpose of illustration, the primary keys are shown, the red diamond sign shows foreign keys and finally, all the relationships are One-to-Many.

**1d**

**Assumptions**

Assuming we are keeping the details of items showing buyer’s name and seller’s name, we could include a transaction table to the database relating it to the user table and the advert table, example of table is below:

**Transaction**

|  |  |  |  |
| --- | --- | --- | --- |
| **Trans\_no(PK)** | Trans\_date | Buyer\_id (FK) | Ads\_id(FK) |
| 112 | 23/1/2020 | 149 | 1 |
| 113 | 11/6/2020 | 453 | 2 |

**Advert**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ads\_id (PK)** | User\_id (FK) | Manufacturer | Model | Cat\_id (FK) | Price($) | Date\_placed |
| 1 | 12 | HP | G650 | 2 | 1700 | 12/1/2020 |
| 2 | 18 | Apple | Iphone 6 | 1 | 600 | 28/2/2020 |

**User**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User\_id (PK)** | First\_name | Surname | Address | Post\_code | Email |
| 12 | Mustapha | Omale | Stirling | FK77QH | musty@gmail.com |
| 149 | Suleiman | Yakubu | Falkirk | FK92HQ | sule@yahoo.com |

**SQL Query** to select the name of a seller and buyer of HP laptop with Ads\_id of 1

SELECT

User.Firstname AS Seller’s\_Firstname,

User.Surname AS Seller’s\_Surname,

User.Firstname AS Buyer’s\_Firstname,

User.Surname AS Buyer’s\_Surname,

FROM ‘User’

JOIN ‘Advert’ ON User.User\_id = Advert.User\_id

JOIN ‘Transaction’ ON Advert.Ads\_id = Transaction.Ads\_id

JOIN ‘User’ ON Transaction.Buyer\_id = User.User\_id

WHERE Advert.Ads\_id = 1

**Output**

|  |  |  |  |
| --- | --- | --- | --- |
| Seller’s\_Firstname | Seller’s\_Surname | Buyer’s\_Firstname | Buyer’s\_Surname |
| Mustapha | Omale | Suleiman | Yakubu |

**1e**

A relationaldatabase arranges data into tables that can be linked or related based on data common to each other, and these data are stored in a structured format called schema. Hence, due to these characteristics, a relational database tends to pose a problem for a classified advert website. For instance, if a user wants to post an item for sale like laptop, some set of descriptive attributes or features related to it will be stored on the database in order to capture the details about the laptop e.g its classification (Electronics), its color, storage capacity etc.

In some cases, this would be very possible to design a relational data for a classified advert website selling only items (laptop) with the same “classification”, which can have similar fields (attributes/features) to describe them. However, if we wish to design a relational database to store items like mobile phone, car and football boots together on the same database, these items have different classifications which have different attributes/features to describe them. Mobile phones have screen sizes and storage capacities, cars have manufacturers and models etc, these creates a problem because it is almost impossible to have a single table containing every possible feature of each category.

Another problem is that, a table can be created for each classification but one thing to consider is that there will be a lot of work to do each time a new classification is inserted into the database, you must create another table alongside to store its features. Also, the queries would all need to be re-written to search the newly constructed tables. The constraint of the relational data model and normal becomes very ambiguous.

**1st Solution:**

Create a table called category to store each item based on its classification. Secondly, create multiple tables to store all the possible features of each classification.

Category Car\_features

|  |  |
| --- | --- |
| Id | Cat\_Name |
| 1 | Car |
| 2 | Electronic |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cat\_id | Mileage | Transmission | Fuel\_type | Colour |
| 1 | 12,000 | Automatic | Diesel | Red |

Items Electronics\_Features

|  |  |  |
| --- | --- | --- |
| Id | Item\_Name | Category |
| 1 | BMW x5 | Car |
| 2 | HP 650 | Electronic |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cat\_id | OS | Screen\_size | Processor\_speed | Colour |
| 2 | Window7 | 14 inch | 2.30ghz | Blue |

The relational tables above have some advantages and disadvantages

Advantages

1. It has abolished the problem of data redundancy and duplications, so query results will be consistent.
2. Using join queries across the database can be done easily
3. Database integrity is good as there will be no empty fields or null values.

Disadvantages

1. Each time a new item with different category is added to the database another feature table must be created and this is a problem when the website grows over time.
2. Additional space is required each time a new table is added to the database by occupying additional memory location for each record.

**2nd Solution:**

One-table approach

A dingle table can be designed to store all items with different classifications and their respective attributes/features together in one database with the assumption of being in First Normal Form (1NF). Some predefined fields for possible features can be created to items associated with while those that are not applicable will be stored as null as seen in **Table 1** below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | Item\_Name | Category | Mileage | Transmission | Model | Color | Price\_($) | Screen\_size |
| 1 | Honda | Car | 13,000 | Fuel | X Sport | Green | 12,000 | **Null** |
| 2 | Hp 17 | Electronic | **Null** | **Null** | Elitebook | Black | 1,000 | 15 inch |

Table 1: Classified advert website for Car and Laptop

Advantages

1. Query performance is improved with low latency since joins are not required – details about each item for sale is saved on a single table. This is an advantage from the customer’s point of view as queries are returned quickly.

2. It requires minimum space to store data records of each item for sale.

Disadvantages

1. Could contain data redundancies/anomalies and duplication, so query results may be inconsistent or with errors.

2. Insertion and deletion of records become problematic and this could introduce data anomalies. Products, such as mobile phones for example, are always being developed to include new features/qualities which may not yet be in the database, and these become difficult to include in the relational database design.

**Comparison**

Comparing the two solutions above, I will choose the 1st solution for classified advert website problem because of the various advantages it has over the one-table approach. However, it will require more time to fully develop the database involving careful schema design and normalization.

**Question 2**

**2a**

MongoDb offers the best solution for a classified advert problem due to its capability of deriving a document-based data model. Each advert with its associated classification and attributes can be stored as a document, since many objects have different classification at the same time the classifications have different features/quality associated with them. However, when other classification is being added to the database, it can be stored as a different document on its own.

For a classified advert, Mongodb help to solve the problem of storing item’s classifications and features since multiple values can be saved as a single field on the database i.e features of a classification can be embedded in the “Category” field.

The requirements of availability and low latency are of greater importance (for improved user experience) for a classified advert application than the absolute integrity which the relational model offers. With an increased demand of agile programming methods, it is suitable to use a database which is easy to alter during rapid growth.

In summary, choosing the NoSQL MongoDB as a supporting database for the classifieds website, provides a host of advantages which solve many of the problems encountered with the relational model and offers additional functional benefits:

1. MongoDB has a very flexible schema. Data can be stored in BSON format which is of a great advantage since data types can vary and only data which is necessary or relevant can be stored.
2. The schema and content and length of documents describing an object can vary from one another (attributes of car and electronics).
3. As no complex joins are required the database provides reduced latency resulting in faster queries for the user, provided the database is appropriately aggregated, indexed and sharded.
4. MongoDB allows indexing on any attribute resulting in quicker queries.
5. Unique object or document keys are automatically assigned to each document which offers functional indexing/querying advantages.
6. It becomes far easier to add and remove (or update) attributes without the strict schema enforced or imposed by the relational database.
7. MongoDB provides easy creation, insertion, read, deletion, update/extend for documents or embedded documents so functionally it becomes easier to work with as a database.
8. MongoDB supports the Map Reduce model on clusters which allows for large scale distributed queries on with the database

MongoDB has become very useful for Big Data applications because of the reasons explained above, and this would serve well as a suitable database to support the classified advertisement website.

**2b**

**Use case 1**

A user may want to check messages related to the ads he/she has posted. It can be accessed through the User’s and message documents, and this is a common activity on the website.

**Use case 2**

A user may want to check the list of all the items they have posted. This is a rare search i.e it is not frequently done, and the data can also be accessed from the user and item documents.

**2d**

The schema of a database describes the data type to be stored. MongoDB is a document-oriented database that stores data as collections in key-value pairs format. The key is a unique identifier in the database while the value is the document (JSON, XML, YAML, etc). MongoDB is called a schemaless database due to the following reasons:

1. It does not require predefined structure of the type of data to be stored in the database (tables, data type).

2. Structured and unstructured data can be stored in the database.

3. Can handle data that numerous characteristics.

4. The database can simply be transformed.

In addition, the capability of MongoDB being a schemaless database creates a more flexible approach of updating and redefining data within documents. However, some restrictions can be imposed on some documents which describes objects for sale in a classified advert database because the maximum size of a document cannot exceed 16MB in MongoDB. This restriction can also help in ensuring that a single cannot use up enormous amount of RAM or consume huge amount of bandwidth during transmission.

Thus, for a classified advert website where items for sale are being posted, a user wish to create an advert about the item which includes pictures, it seems reasonable to limit the number of pictures to be posted with its minimum size describing the item. The advantage of doing so is that it will help to minimize memory consumption but there is a disadvantage of doing so because the this will reduce the quality of the item’s picture and might not contain enough pictures which can convince a user to buy the item.

Other restrictions involve maintaining consistency in the database when posting an item for sale on the classified advert website like – the item must have a name (string), item price (int), category, model, etc must be included.

**Example of this restrictions can be imposed using the following:**

db.createCollection(“item”), {

validator: {

$jsonSchema: {

bsonType: “object”,

required: [“ref number”, “condition”, “item name”, “price”],

properties: {

serial number: {

bsonType: “int”,

minimum: 1,

maximum: 20,

description: “ref number must be an integer and it is required”

},

condition: {

bsonType: “string”,

description: “condition must be a string and it is required”

},

Item name: {

bsonType: “string”,

description: “item name must be a string and it is required”

},

price: {

bsonType: “int”,

minimum: 10,

maximum: 4000,

},

},

},

}

**2e**

Indexing the **Item Name and Price** fields will make searches quicker on the website as these are frequently used while searching for items on the website. This will help the database to save the huge amount of time needed to go through the entire documents.

MongoDB supports indexes. Without indexes, MongoDB will have to investigate all the documents in a collection to find matching documents which corresponds to the query statements. Index is a data structure in a database that exists independently which improves the optimization ability to execute queries and search faster. Indexing helps to improve the performance and efficiency of a database.

However, every field in the database cannot be indexed because indexes also use up disk space and memory storage. Imagine data is being amended on a regular basis, this means updating the all the indexes in the database engine is required, also too many indexes slow down the performance of a database.

**2f**

As time goes on, the classified advert website database will grow which will lead to a high volume of datasets e.g adding more and more items with different categories and features. Therefore, the Mongodb database will require sharding to aid deployments with very large data sets and high throughput operation. A database system containing large data sets or high throughput applications can be a great threat to the size of a single server e.g large rates of query will be able to consume the server’s CPU capacity.

Ideally, a shard key is built upon one or multiple chosen fields – is seems logically to **use category name** as the best field to be used as a shard key because it is sensible to keep all items in a category on the same shard and it appears in all records. I wouldn’t need to introduce another field to be used as a shard key since category name has been chosen and it is present in the database design.

**QUESTION 3**

The best way of choosing the right database for a task, is to analyze its need and what will be its functions. SQL is more preferably for any database task that requires set of schemas and predefined structure, especially if it involves multi-row transactions where consistency must be maintained without giving any opportunity for errors.

For a database experiencing accelerated growth which does not require a well-defined schema, NoSQL is the perfect pick for. It tenders more resilience than a relational database, so it is more appropriate for a classified advert website.

Below are major differences between NoSQL and SQL which are very important to when deciding about the perfect data management system for a classified advert website. These differences include:

**Language**

SQL database uses Structured Query Language for manipulating and defining data. This made SQL to be immensely adaptable and commonly used. Defining a schema is also required for the data structure before working with it. It is compulsory that the data must follow the thesame structure, involving substantial upfront planning.

NoSQL database has a flexible schema which is used for storing structured and unstructured data. This resilience allows an admin to design a document-database without any careful plan or structure definition, add fields at anytime, and change the syntax from one database to another.

**Scalability**

In most cases, SQL databases are vertically scalable. Increasing some components like CPU and RAM can enhance having huge load on a single server. In distinction, NoSQL databases are horizontally scalable i.e it can manage enlarged traffic which is achieved by attaching extra servers to the database. This database can also turn out to be more robust, making it the best preference for enormous emerging data sets.

**The Structure**

Lastly, something to put into consideration when choosing a database is its structure. SQL databases are based on tables making them a best option for applications which involves multi-row transactions e.g accounting systems that was initially constructed using a relational structure. NoSQL databases can be document-based, wide-column stores, graph databases, or key-value pairs.

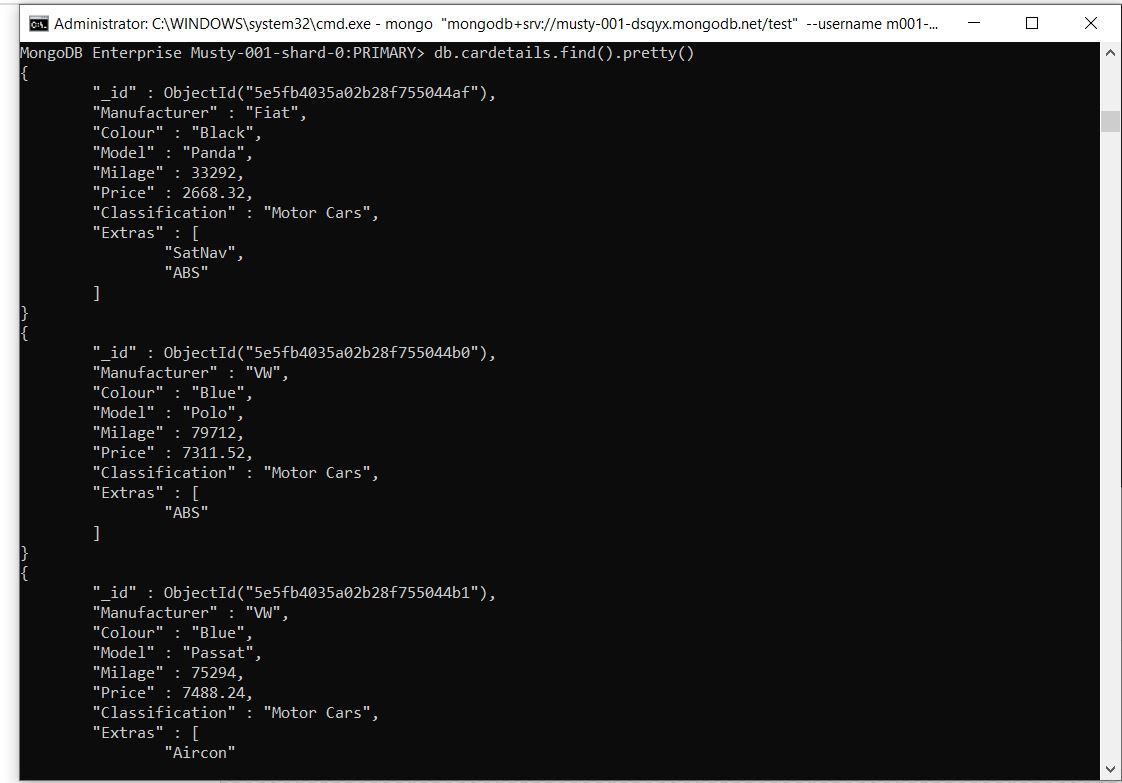
In conclusion, NoSQL is the best choice for a classified advert website which could experience rapid growth e.g adding many other categories to the database. NoSQL has many advantages than a relational database and a recommendation for classified advert website because of its advantages as embedding classification of objects along with their features/qualities can be stored as a document in a collection.

**QUESTION 4**

**Loading the JSON file**

Load(“Assgcars.js”)

db.cardetails.find().pretty()

****

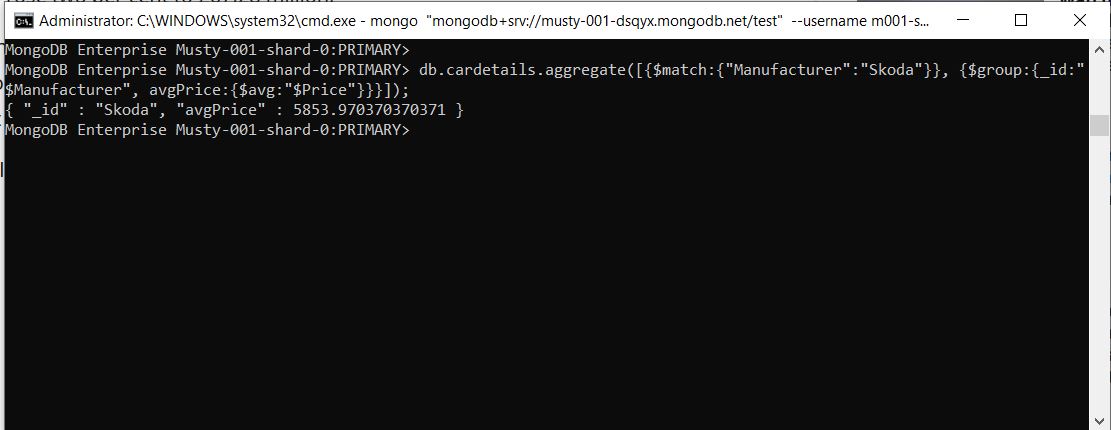
**4a.** Price list of all Skoda cars and Model is Octavia

db.cardetails.find({"Manufacturer":"Skoda","Model":"Octavia"},{Price:1,\_id:0});



**4b.** The average price of all Skoda cars for Sale

db.cardetails.aggregate([{$match:{"Manufacturer":"Skoda"}}, {$group:{\_id:"$Model", avgPrice:{$avg:"$Price"}}}]);



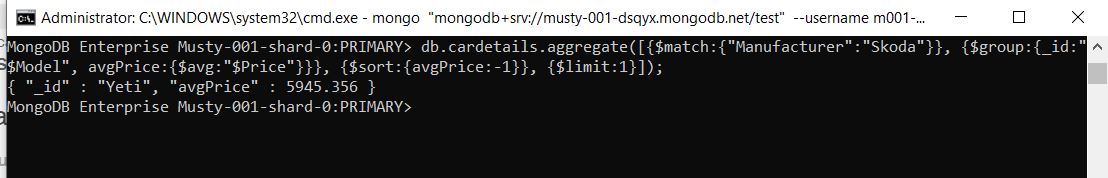
**4c.** The average price of each different model from Skoda cars for sale

db.cardetails.aggregate([{$match:{"Manufacturer":"Skoda"}}, {$group:{\_id:"$Model", avgPrice:{$avg:"$Price"}}}]);



**4d.** Model of Skoda Car with Highest average price

db.cardetails.aggregate([{$match:{"Manufacturer":"Skoda"}}, {$group:{\_id:"$Model", avgPrice:{$avg:"$Price"}}}, {$sort:{avgPrice:-1}}, {$limit:1}]);



**4e.** Extras of all cars

db.cardetails.aggregate({$unwind":"$Extras"}, {$group:{\_id:"$Manufacturer", uniqueValues:{$addToSet:"$Extras"}}}]);

