# Logic used for Assignment – Databases

Table of Contents

[Logic used for Assignment – Databases 1](#_Toc67946735)

[Task-01: Data Loading – Logic 2](#_Toc67946736)

[Task-2: Queries – Logic 3](#_Toc67946737)

[Query-01 3](#_Toc67946738)

[Query-02 3](#_Toc67946739)

[Query-03 3](#_Toc67946740)

[Query-04 3](#_Toc67946741)

[Query-05 4](#_Toc67946742)

[APPENDIX – Sequence Diagram (Method Call Flow) 5](#_Toc67946743)

[APPENDIX – AWS Screenshots 6](#_Toc67946744)

## Task-01: Data Loading – Logic

DriverManager from Java SQL package is used to establish connection with the Upgrad AWS RDS MySQL Database URL, using provided credentials.

MongoDB Java driver package is used to create client instance for establishing connection to AWS MongoDB NoSQL database, created in AWS Cloud using shared AWS account.

[Refer Appendix for AWS screenshots for VPC, EC2, Security group configurations & mongo db installation for reference.]

Created mongoDB “pgcdata” with collections named as “products”.

Upon establishing connection with AWS RDS MySQL DB & AWS EC2 MongoDB, part of initialization steps, all the documents under “products” collection were deleted. Though this operation isn’t mentioned as an explicit requirement part of the assignment, this is done to ensure that query results remain consistent upon multiple executions of the assignment application.

Same cleanup operation shall be done as last operation of this assignment application. But since it is possible for the assignment evaluators to verify the mongoDB collection from console, currently this cleanup operation at end of application execution is disabled/commented.

Considering the limited scope of applications evaluation rubrics, following assumptions were made –

1. MySQL table will contain all the fields of type VARCHAR / Strings.
2. List of records shall not be huge in size, yet restricted fetch size to 10, considering given MySQL dataset tables were having only 5 records.
3. Havent employed any other 3rd party libraries for ResultSet to JSON to MongoDB Import based data migration. Used simple methods only.

Created static stub methods to query individual tables & refactored common methods for better code reuse. Below methods are created for same purpose, which are called from main() method of Driver class –

1. importRDSVarcharTablesToMongoDb() - Method to import multiple RDS SQL table with only VARCHAR (String) fields to Mongo DB, uses below method.
2. insertVarcharTableToMongoDb() - Method to import records fetched by given sql query on a RDS SQL table with only VARCHAR (String) fields by adding a new category field referring to product category to Mongo DB. This method currently fetches records by paging 10 records at once. This shall be set as per needs. Uses below method.
3. getDocumentList() - Method to construct Mongo Document object for each records. This method uses to fetch individual fields of each records as present part of returned upon querying SQL table with only VARCHAR/String fields.

Refer [Appendix – Sequence Diagram](#_APPENDIX_–_Sequence) for method call flow.

## Task-2: Queries – Logic

Query-01: Show the complete product inventory with common attributes. (Print the data only for the common attributes which are there for all three product categories)

**Logic:**

1. Build Query to find all documents
2. add (optionally) projection to include only the common fields & get MongoCursor.
3. Iterate cursor & Call PrintHelper.printSingleCommonAttributes() to display the common attributes on the Screen.

Additionally, usage of Projections here seems to be optional, as the PrintHelper.printSingleCommonAttributes() method is hardcoded to print the common attributes only, irrespective of whether other non-common / non-null columns are present or not part of the fetched result mongo cursor.

Query-02: Get the information about the first five products under the category ‘Mobiles’.

**Logic:**

1. Build Query to find all documents with category as "Mobile"
2. Limit records to 5
3. get MongoCursor on returned documents
4. Iterate cursor & call respective PrintHelper.printAllAttributes() method to display output as required.

Query-03: Get the products ordered by category without their auto-generated ‘\_id’ column in MongoDB.

**Logic:**

1. Build Query to find all documents
2. Use Projections to exclude ID field &
3. Sort documents by $Category field in descending order
4. Get MongoCursor on returned documents
5. Iterate cursor & call respective PrintHelper.printAllAttributes() method to display output as required.

Query-04: Get the product count for every category.

**Logic:**

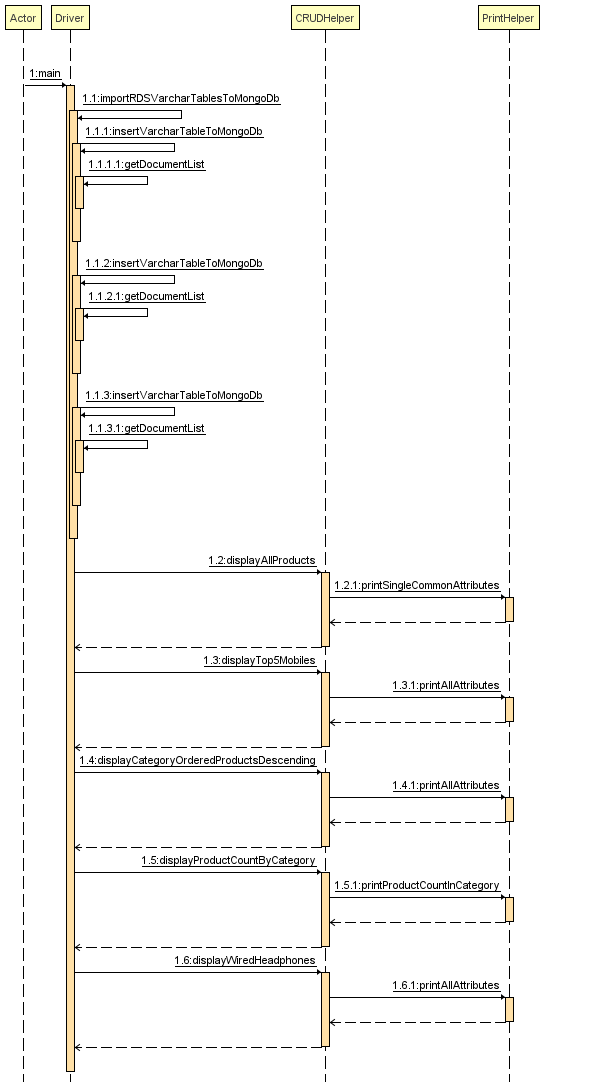
1. Build Query to aggregate documents, by grouping documents by "$Category"
2. Accumulate the number of records using Accumulator.sum to keep adding 1 for every record grouped by "$Category" field
3. Sort the list & get MongoCursor on returned documents
4. Iterate cursor & call respective PrintHelper.printProductCountInCategory() method to display output as required.

Query-05: Get the information about all the wired headphones.

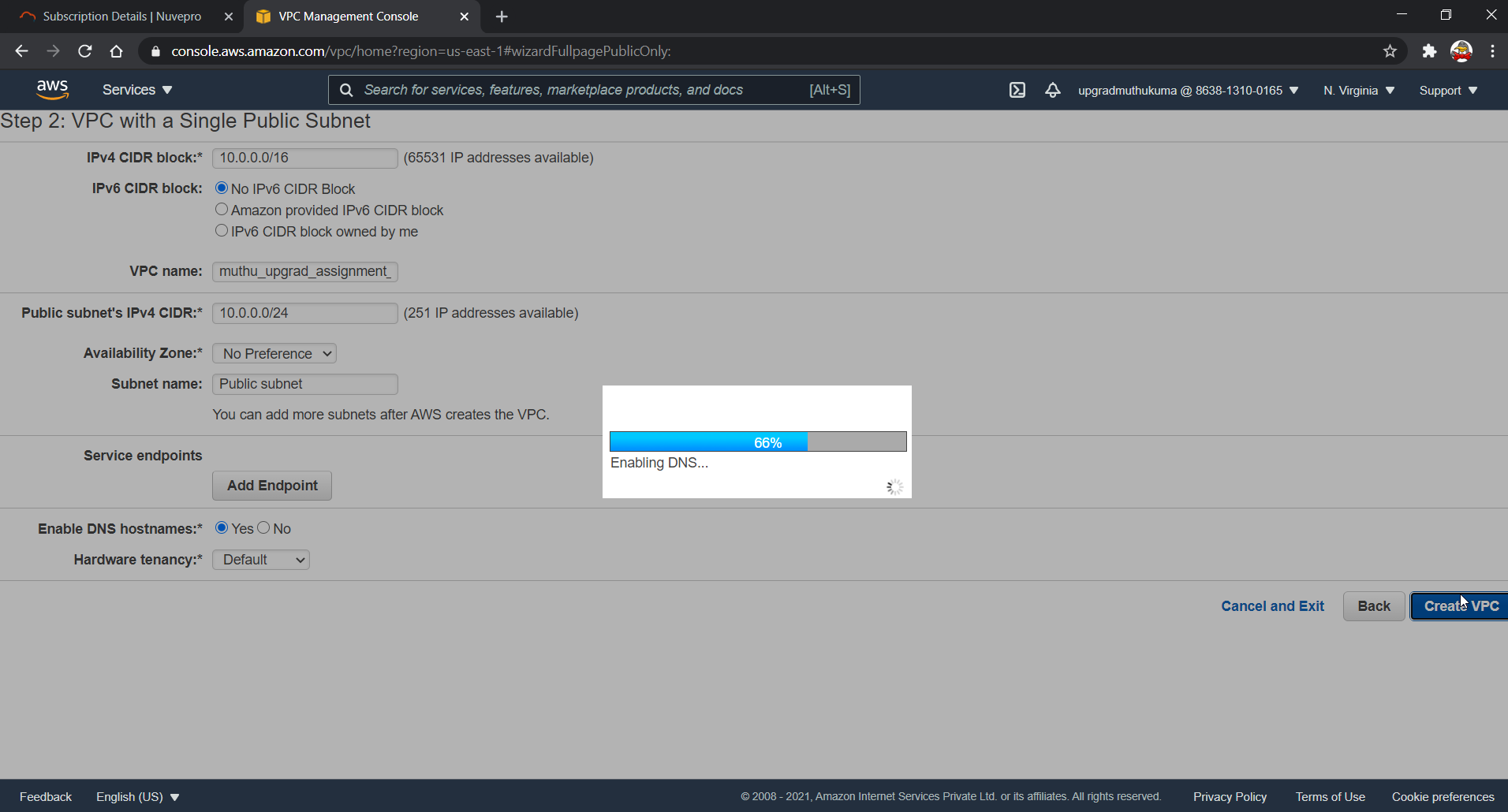
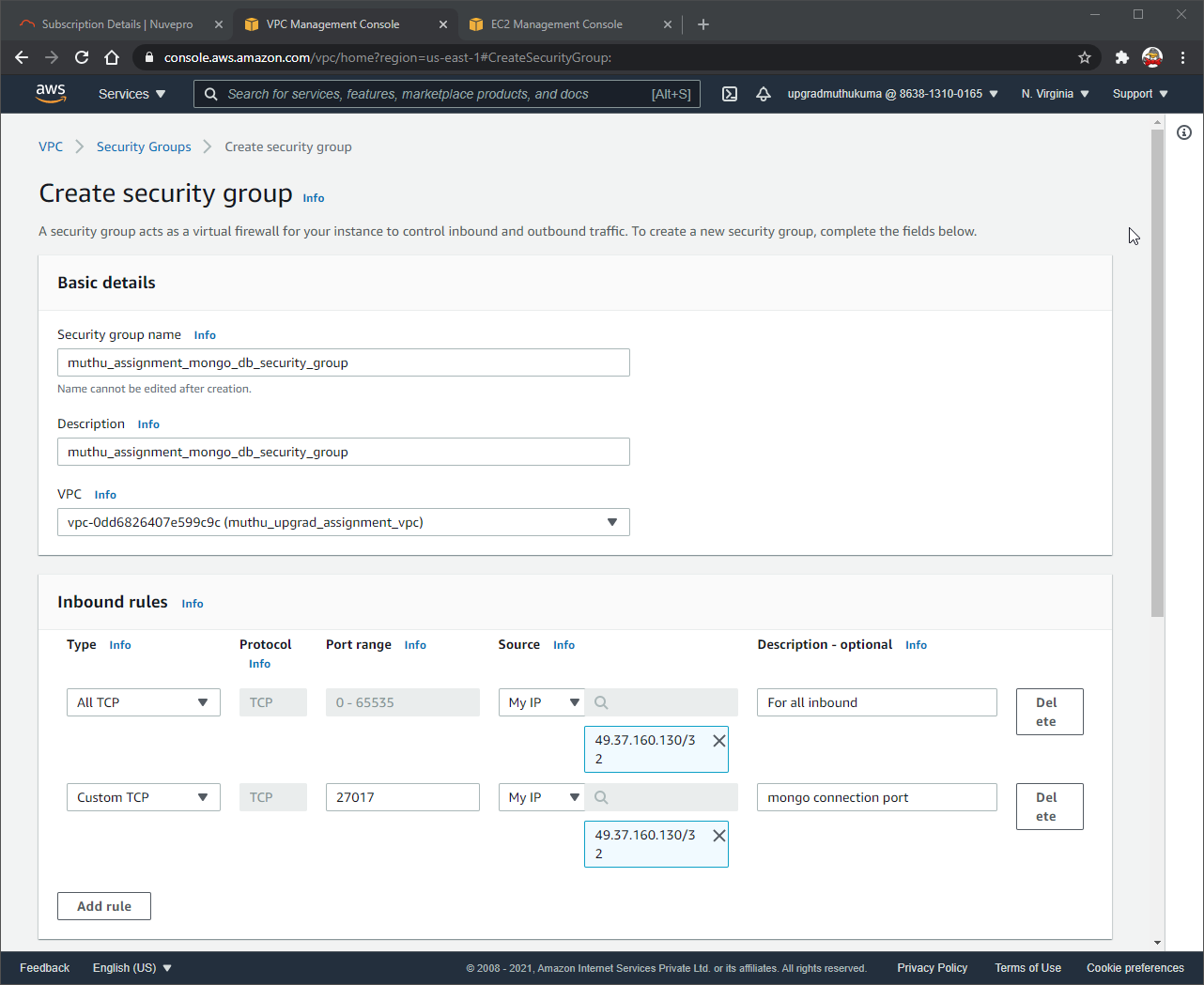
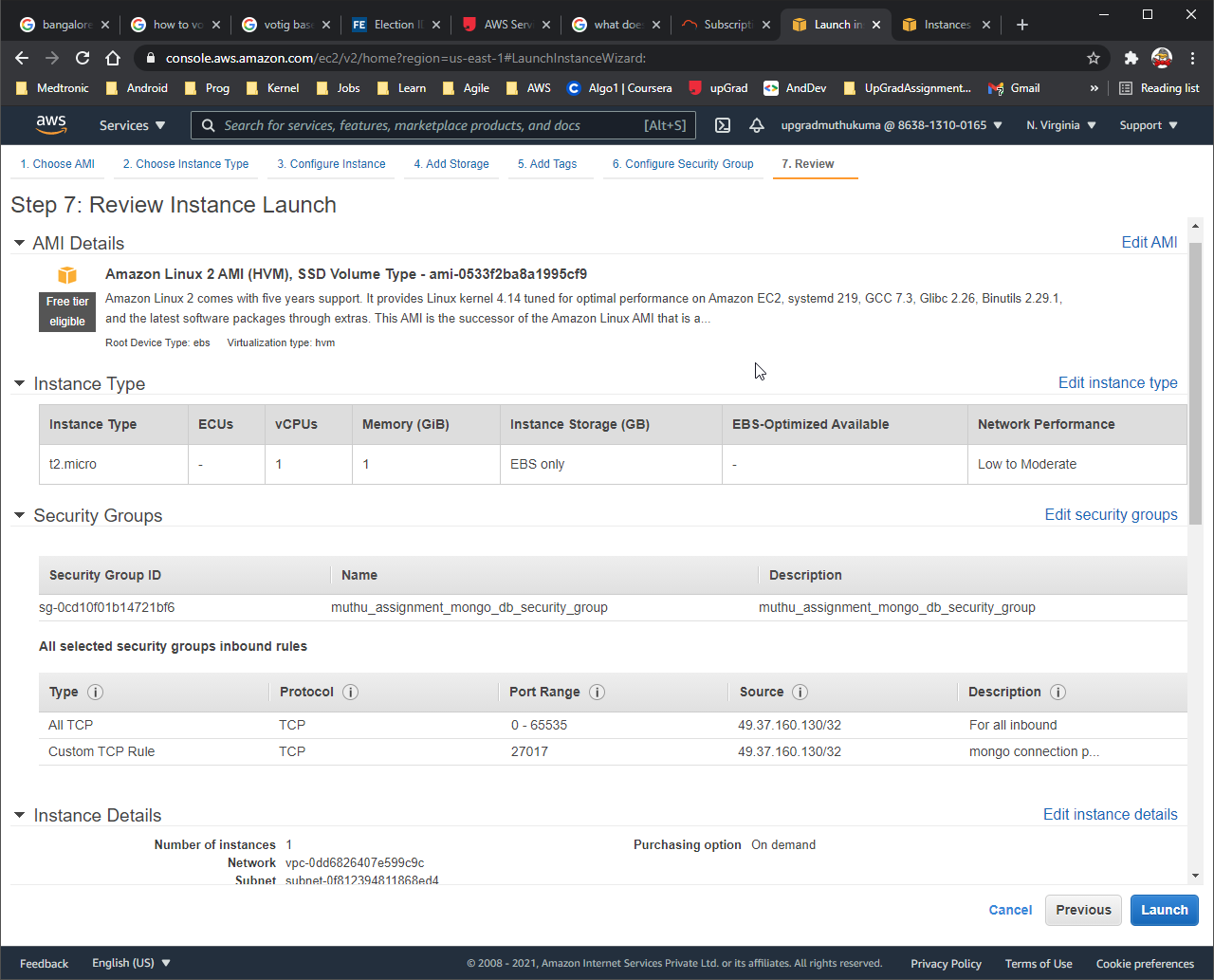
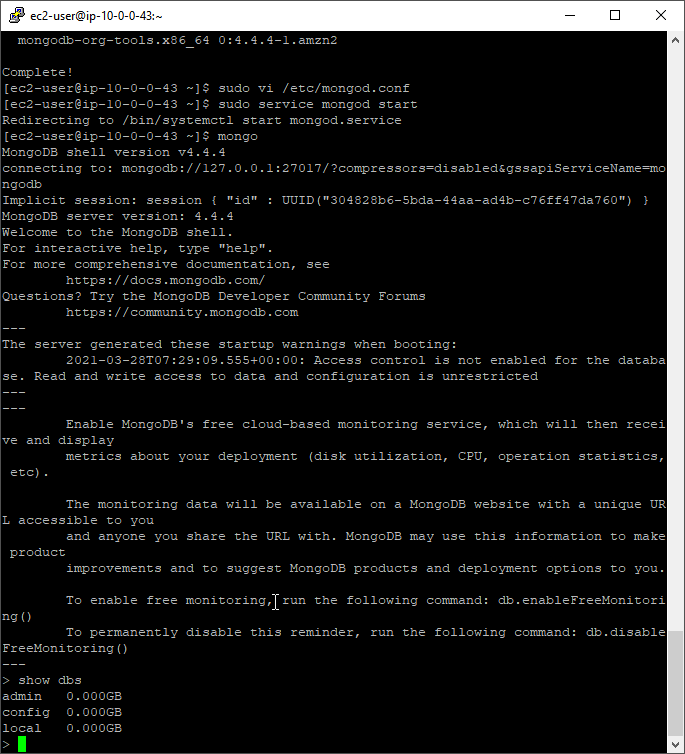
**Logic:**

1. Create BasicDBObject to define criterias, such that “Category” key contains value “Headphone” & “ConnectorType” key contains value “Wired”
2. Build query to find all documents matching criteria set by BasicDBObject
3. Get MongoCursor on returned documents.
4. Iterate cursor & call respective PrintHelper.printAllAttributes() method to display output as required.

## APPENDIX – Sequence Diagram (Method Call Flow)



## APPENDIX – AWS Screenshots

1. VPC Creation  
   
2. Configuring Security Group  
   
3. Creating EC2  
   
4. Connecting to EC2 from putty & setting up mongo DB  
   
5. Running EC2 instance with mongo DB installed.

