Tools Used:

1)Eclipse 2)MySQL database ( I used version 5.7.19 ) 3)MySQL Workbench ( For loading OrderLines.csv file to the MySQL database ) 4)MySQL Java connector ( Dependency )

Summary of the set up:

1.Create the Table rowaggregateexercise in the MySQL database named mra. cssreate table rowaggregateexercise (order\_id INT NOT NULL, product\_id INT NOT NULL, sku float NOT NULL, price float NOT NULL, quantity INT NOT NULL, discountprice float NOT NULL, title VARCHAR(120) NOT NULL, id INT NOT NULL AUTO INCREMENT, PRIMARY KEY (id) );

2.Import the required CSV file to the MySQL database using MySQL Workbench [ When importing you will find lot of duplicate values in the given data-set. So I created a ID column which is autoincremented for every record and also serves as an primary key]

3.Once the required data is populated, Used Eclipse and created a project named Multithreaded\_Row\_Aggregation. Add the necessary dependency to the build path. The required dependency is MySQL java connector ( jar file )

4.I created a package named Multithreaded\_Row\_Aggregation. The Multithreaded\_Row\_Aggregation\_Main class inside this package contains the driver program to get the connection to the MySQL database, create MultipleThreads for performing some specific task. The host, username and password of the Database can be taken as command line arguements from the user. If you wish to vary the number of threads, change the integer value parameter in the line ExecutorService executor = Executors.newFixedThreadPool(200) to other value.

5. MySQL\_Singleton class is thread-safe singleton class required for getting the connection instance to connect to the required MySQL database using the credintials.

6.The Query\_Logic class contains the actual logic for finding the count for each order price.

The source code details of these packages are explained in detail as in-line comments.

For Running the application:

1.Before running the application. Make sure your MySQL server service is up and running. For windows, you can view in the Services app.

2.Give the command line arguements in Eclipse in the following order <hostname> <username> <password>. In Eclipse you can set the command line arguements by going to Run menu -> Click Run Configurations -> Select Multithreaded\_Row\_Aggregation\_Main class -> Click on Arguments tab and enter the your parameters as mentioned in that order. You can run the Multithreaded\_Row\_Aggregation\_Main class as Java application ( RightClick on this class and select Run as Java application ). You can view the price and its required count in the Standard output

You can also run the application by exporting as Jar file with main class Multithreaded\_Row\_Aggregation\_Main. The command for running the Jar file is "java -jar <FileName>.jar <hostname> <username> <password>

Best design practices: 1. Thread safe singleton pattern for getting the Database Object 2. Thread safe ConcurrentHashMap with AtomicInteger datatype for performing the order-price count increment (atomicity) 3. ThreadPool for resuing the Threads that completed its tasks 4. Modular approach having the business logic and database connection separated from the driver program