Create reports that answer the following questions:
- How long do orders need to be fulfilled?
- How many orders are open at specific times?
- How high is the probability of a cancelled order depending on the age?

1) SELECT min(t.time), avg(t.time), max(t.time) from (select (TIMESTAMPDIFF(SECOND, orderedAt, orderFulfilledAt)) time from ProductOrder where type='order_fulfilled') t;
2) set @time= '2019-08-01 16:02:19.000000';
   SELECT count(*) from ProductOrder where orderedAt <= @time and lastUpdatedAt > @time;
3) select floor(datediff(p.orderedAt, c.birthDate)/365) as age, (count(c.name) * 100)/(select count(*) from ProductOrder p1 where p1.type='order_cancelled') ratio from ProductOrder p
   inner join Customer c on c.aggregateId = p.customer_aggregateId
   where type = 'order_cancelled'
   group by age order by age asc;

Hints and Questions
- How can you test the code that you've written for the data pipeline?
- Can you apply the SOLID principles to your code? How does it look different from the easiest possible approach?
- So far the reports probably are static once generated. Think of how to update the reports in real time as new events come in. How could such a solution look like?

- The code is written in Spring Framework with the help of Spring Boot. There is Unit Test support using Spring Batch Test. To test this system, we would check if the test exited with the "COMPLETED" status. We may further check the accuracy of our batch processes if we have the desired output with us in one way or another. Since the transformation is done in a single step, we can test the transformation of the single record and match it with expected results. When the batch is processed, we would also be able to track the number of records that are processed.
- The SOLID principles are applied to some extent in the code. The classes are designed in such a way that each class/entity has its own responsibility. The model classes represent only the properties of the entities that reflect our business. Using interface-based contracts, classes are assigned specific tasks e.g. classes in the repository package only manage persisting the data of appropriate model entity to appropriate table in the database. Spring is unanimous for Dependency Injection at the runtime, hence this allows for great flexibility. For instance, switching the database is as easy as changing the url in the application.properties file.
- Spring Batch sees these tasks that are developed as jobs. This means that they can be run or scheduled in a flexible manner. However, they do not work with an incoming stream of data. Although a wide variety of technology exists that allow Spring Batch to

utilized with a stream of data, a more realistic choice would be to use Apache Spark Stream. Use of MQ is also suitable to process events as streams.