what operation has the most impact on performance? Propose an optimization for each query (ex. adding an index, changing the order of items in the SQL) that you think will improve the performance of the query.

1. **select** title, description, name  
   **from** film   
   **join** **language** **on** film.language\_id = language.language\_id   
   limit **300**

One way to improve performance would be to ensure both tables have an index created for the Language ID column, since this is how they are being joined.

1. **select** customer\_id   
   **from** rental   
   **join** inventory **on** rental.inventory\_id = inventory.inventory\_id  
   **join** film **on** inventory.film\_id = film.film\_id  
   **where** film.title = **'MONTEZUMA COMMAND'**

Creating indexes for the join columns, along with the Film Title column, since that is the primary column being filtered will allow for the greatest performance. Adding a DISTINCT constraint to the select Customer\_ID will also prevent duplicate entries from slowing the query.

1. **select** first\_name, last\_name, **sum**(amount)  
   **from** customer   
   **join** payment **on** customer.customer\_id = payment.customer\_id  
   **group** **by** customer.customer\_id, first\_name, last\_name

Creating an index on the primary group by column of Customer\_ID allows for an aggregation column to be used to remove an inefficiency with listing the different entries per customer. Displaying these as one total per customer will greatly improve the speed of the query.

1. **select** **distinct** actor.\*  
   **from** actor   
   **join** film\_actor **on** actor.actor\_id = film\_actor.actor\_id  
   **join** film **on** film\_actor.film\_id = film.film\_id  
   **where** film.rental\_rate = **.99**

Removing the wildcard from the select statement will greatly improve the speed of the query. Along with creating indexes for the join and filtering columns to further increase speeds.

1. **select** first\_name, last\_name, address, city, country, postal\_code, phone   
   **from** customer   
   **join** address **on** customer.address\_id = address.address\_id   
   **join** city **on** address.city\_id = city.city\_id   
   **join** country **on** city.country\_id = country.country\_id   
   **where** customer\_id **in** (**select** customer\_id **from** rental **where** return\_date **is** **null**)

Using a distinct constraint to remove duplicate customer rows, along with adding indexes for the join and filtering columns will greatly increase the speed ot his query. Another method to improve the speed is to replace the nested select statement with a join statement as these operations are faster and can be further optimized.

Using the distributed database described by the diagram below, specify the minimum types of operations the database must support to perform the following tasks. These operations include remote requests, remote transactions, distributed transactions, and distributed requests.

1. At Site C
   * Get a list of all products

Since the PRODUCT table exists separately at two different sites, the DBMS must be able to support distributed requests. Transactions are not necessary at all since this is a read-only operation.

1. At Site C
   * Print a customer invoice

Since the CUSTOMER and INVOICE tables exist at two separate locations, the DBMS must support distributed requests. Transactions are not necessary at all since this is a read-only operation.

1. At Site A
   * Create a new order for a customer. This involves creating the invoice and lines as well as updating the product and customer.

Since the PRODUCT table exists separately at two different sites, and the CUSTOMER and INVOICE tables exist at two separate locations, the DBMS must be able to support distributed requests. Additionally, since this involves inserting a new row into some of the tables, the DBMS will also require support for distributed transactions.

1. At Site B
   * Get a list of all customers

Since the CUSTOMER table exists at one location and this list can be access with one request, the DBMS will need to support remote requests. Transactions are not necessary since this is a read-only operation.

1. At Site B
   * Get a list of all the products.

Since the CUSTOMER and INVOICE tables exist at two separate locations, the DBMS must support distributed requests. Transactions are not necessary at all since this is a read-only operation.