|  | **Java Questions** | **Answers** | |
| --- | --- | --- | --- |
| 1. | Have the following code snippet. Identify and correct at least 4 issues with it:  import org.springframework.stereotype.Service;  @Service  public class BillingService {  private BillingRepository billingRepository;  public void createBillingReport(BillingReport billingReport) {  billingRepository.save(billingReport);  }  public BillingReport getBillingReportById(Long id) {  return billingRepository.findById(id).orElse(null);  }  public void updateBillingReport(BillingReport billingReport) {  BillingReport existingReport = billingRepository.findById(billingReport.getId()).orElse(null);  if (existingReport != null) {  existingReport.setTotalAmount(billingReport.getTotalAmount());  billingRepository.save(existingReport);  }  }  public void deleteBillingReport(Long id) {  billingRepository.deleteById(id)  }  } | import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  @Service  public class BillingService {  private final BillingRepository billingRepository;  @Autowired  public BillingService(BillingRepository billingRepository) {  this.billingRepository = billingRepository;  }  public void createBillingReport(BillingReport billingReport) {  billingRepository.save(billingReport);  }  public BillingReport getBillingReportById(Long id) {  return billingRepository.findById(id).orElseThrow(() -> new IllegalArgumentException("Billing Report not found with id: " + billingReport.getId();  }  public void updateBillingReport(BillingReport billingReport) {  BillingReport existingReport = billingRepository.getBillingReportById(billingReport.getId());  existingReport.setTotalAmount(billingReport.getTotalAmount());  billingRepository.save(existingReport);  }  public void deleteBillingReport(Long id) {  billingRepository.deleteById(id);  }  } | |
|  | **Testing Questions** | **Answers** | |
| 1. | A simple Accounts Payable system which provides a report on net amount due to Supplier with below functionalities:   * Groups Invoices due on the same date * Offset maximum Credit Notes against these Invoices to reduce the amount due. * Calculate and display the net amount to be paid to the Supplier on the due date.   **Task**:   1. Prepare a test plan for the new functionality 2. Point out missing information in the requirements (if any). | 1. [Test plan](https://github.com/denis-novikov) is located in github and is accessible by link 2. How system behaves if Credit Notes are bigger than total amount due | |
| 2. | Test a report that shows all the invoices received between two dates.  The From and To Dates are selected via a date picker (yyyy/mm/dd) in the UI.  The invoices table stores the date and time that the invoices are received as a timestamp.    1. What would be a typical testing method to test the date parameters  2. Can you give some examples of common developer mistakes that can be made in the given scenario and how the testing method in (1) would help to catch the error. | * Boundary values:   Test the minimum possible dates  Test the maximum possible dates  Test dates very close to each other and within same day   * Equivalent partitioning:   Test within valid dates range.  Test within dates range in the future   * Edge cases:   Leap year  DST(Daylight Saving Time)  Time zones   * Invalid date range:   Date “To” is earlier than “From”   1. Developer can incorrectly handle boundary dates, excluding start or end date invoices. With testing boundary values we can easily handle it. | |
|  | **Database Questions** | | **Answers** |
| 1. | Which SQL statement lists the buyer names in the buyer table that are not contained in the invoice table?   1. select b.name   from buyer b  left join invoice i on b.id = i.buyer\_id  where i.buyer\_id is null;   1. select distinct b.name   from buyer b  join invoice i on b.id = i.buyer\_id;     1. select b.name   from buyer b  join invoice i on b.id = i.buyer\_id;     1. select distinct b.name   from buyer b  left join invoice i on b.id = i.buyer\_id; | | A |
| 2. | Which SQL statement returns the department number with the maximum salary given to any employee?   1. select department\_id, max(salary) from employees; 2. select max(salary) from employees group by department\_id ; 3. select department\_id, max(salary) from employees group by department\_id; 4. select max(salary) from employees; | | A |
| 3. | What does the below query do:  update inv  set inv.status\_id =  (case  when inv.buyer\_id = 1 then 'In Progress'  when inv.buyer\_id = 2 then 'New'  else 'Rejected'  end)  from invoices inv;   1. Updates no invoices 2. Updates all invoices for buyer 1 to 'In Progress', buyer 2's invoices to 'New' and any other buyer's invoices to 'Rejected' 3. Updates all invoices to 'Rejected' 4. Throws exception | | B |
| 4. | Which statement below is correct to insert ‘Baker’ as the lastname in the persons table?   1. Insert into persons ('Baker') into lastname; 2. Insert into persons values ('Baker'); 3. Insert into persons (lastname) values ('Baker'); 4. Insert ('Baker') into persons (lastname); | | C |
| 5. | What is the output of the below query?  *MS SQL:* select substring('123456789', charindex('b', 'abcabcabc'), 4);  *Oracle:* select substr('123456789', instr('abcabcabc', 'b'), 4) from dual;   1. 6789 2. 2345 3. 1234 4. 456789 | | B |
| 6. | A table my\_numbers has 6 number values: 1, 2, null, 1, 1, null.  Predict the output of the below query:  select count(num) from my\_numbers;   1. 6 2. 4 3. 3 4. Throws exception because count function does not work with null value | | B |

**Cucumber**

* Go to <https://github.com/kpleshkova/qatest/blob/main/purchase_orders.xls> and download xls file.
* Create cucumber test scenario with such steps (steps names could differ but should be at least 3 steps):

1. File with name "purchase\_orders.xls" exists

2. Verify that header have next headers

Buyer    | Buyer    | Buyer | External ID    | Sales Order

3. Verify that there is more than 10 lines in file

**Cypress**

Write a Cypress test in GitHub (or send code in text editor) to perform the following actions:

1. Open the page: <https://www.globalsqa.com/angularJs-protractor/BankingProject/#/login>

2. Log in using the customer 'Harry Potter'

3. Deposit 100, 10 and 5. Check always that the confirmation message contains 'Deposit Successful'

4. Verify that the Balance is 115

5. Check that the sum of the amounts in the Transaction screen is equal to 115