Q1

Ans: A stream was created from a list of EmployeeQ1 objects. The first employee was retrieved using findFirst() and mapped to a full name string by concatenating firstName + lastName.

Q2

Ans: Used Collectors.groupingBy() with Collectors.counting() to group employees by department and count how many belong to each.

Q3

Ans: The values of the map were streamed using flatMap(). A filter() operation checked if either firstName or lastName contained the search keyword, ignoring case.

Q4

Ans: Used String.format("%4s", id).replace(' ', '0') in a map() operation over the list of IDs to left-pad store IDs with zeros up to 4 digits.

Q5

Ans: Filtered the stream using a negated .equalsIgnoreCase() condition to exclude employees from the specified department.

Q6

Ans: Sorted the employee list using Comparator.comparing() on the firstName field inside a stream.

Q7

Ans: Used .max(Comparator.comparingInt(...)) on the stream to find the employee with the highest empId.

Q8

Ans: Mapped each employee to their full name (firstName + lastName) and joined them using Collectors.joining("|").

Q9

Ans: Skipped the first seven employees using .skip(7) and retrieved the 8th using .findFirst(), then printed full name and department.

Q10

Ans: Used .filter() on the main employee list to check if each employee’s ID is present in the list of target IDs.

Q11

Ans: Used Collectors.groupingBy(Employee::getGender, Collectors.counting()) to count employees grouped by gender.

Q12

Ans: Grouped employees by gender and used Collectors.mapping() with Collectors.joining("-") to build a formatted string like MALE: [John-Mike].

Q13

Ans: Sorted the stream of employees in ascending order using Comparator.comparingDouble(Employee::getSalary).

Q14

Ans: Defined Optional<String> for the email field and wrapped it using Optional.ofNullable(). Used .orElse() to return a default if email was absent.

Q15

Ans: Defined the address field as Optional<Address> and used .orElse() to return a default address if no address was provided.

Q16

Ans: Filtered employees by department "IT" and used Optional.ifPresent() on the salary field to apply and print a salary increase.

Q17

Ans: Sorted the list of addresses using Comparator.comparing(...).thenComparing(...) to sort first by city and then by country.

Q18

Ans: Created a map using Collectors.toMap() where the key is the concatenation of firstName + lastName, and the value is the employee’s address.

Q19

Ans: Used both .findAny() and .findFirst() on the stream to retrieve any or the first employee and printed the result using Optional.ifPresent().

Q20

Ans: Used .anyMatch(), .allMatch(), and .noneMatch() to check if any employee is from HR, all have emails, and none have a null name respectively.