SQL QUERIES

```
SELECT salary, empname
FROM employee
JOIN salary
on employee.empID = salary.empID
ORDER BY Salary DESC LIMIT n-1,1
```

```
select * from employee;
select * from salary;
update salary SET salary = 5000 where EmpID = (select EmpID from employee where
Date_of_birth < '1990-07-16');

select * from salary;
```

APPLY PROMO CODE TEST CASES.

- Check whether he/she is a new user.
- Promo code shall not be applicable for the repeated customer.
- Check billing amount is more than or equal to 1000
- Check with a billing amount less than 1000. Proper error message shall be shown while trying promo code.
- Check it is deducting a maximum 300rs from the billing amount.
- As a user, check the deduction with billing amount 1000
- As a user, check the deduction with the billing amount more than any value to 1000.
- As a user, I shall see 30% discount is applied.
- As a user, max. Cap is 300 shall be mentioned in terms and conditions of the promo code.
- Check terms and conditions of the promo code are configurable from backend or hard coded.
- If they’re configurable from the backend. Check, they’re getting changed.

- As a user, I shall check if the promo code is case sensitive or not.
- As a user, I shall check with NEW30 and new30.
- As a user, I shall not be able to type more than 7 characters in the promo code field.
- As a user, I shall get a proper error message if I’ve typo in writing promo code.
- As a user, I shall get a proper message if i’m a repeated user.
- As a user, I shall see an error message for any other specified user roles.
- Submit button of the promo code shall be grayed out if the field is empty.
- Validity of promo code shall be configurable from backend.
- Check adding/removing/updating/deleting of the promo code is configurable from backend.
- Ensure minimum order limit is configurable from backend.
- Ensure maximum cap amount is accessible from backend(admin panel).
- Check whether promo code is clubbed with other offers as well or not.

FIRST 100 PRIME NUMBERS MANUAL TEST CASES.

- Let “num” be the prime number with the initial value of “3”.
- Check with initializing the status as a true.
- //For num = 1 code shall not work as it is a composite number.
- Number “2” shall be printed as a prime number.
- “for” loops shall be implemented with proper syntax.
- First “for” loop shall be iterated with a value of “i” ranging from 2 to less than or equal to 100. “i” shall be initialized in the “for” loop.
- Second “for” loop shall be iterated with a value of “j” ranging from 2 to less than or equal to the square root of “num”. “j” shall be initialized in the “for” loop and the value of “j” shall be incremented accordingly.
- First loop shall be successful, then only the condition in the second loop shall be checked.
- Considering it as a non-prime number:
    - When num%j is equal to zero, the value of the “status” shall be changed to “false” and a break statement would be executed.
    - If the value of the status is not “true”, nothing shall be printed.
    - Once the above steps are executed “status” shall be changed to “true” and “num” shall be incremented.

- Considering it as a prime number:
    - When num%j is not equal to zero, the value of the “status” shall not get changed and be “true” only.
    - If the value of the status is “true”, the “num”(i.e Prime Number) shall be printed and the value of “i” shall be incremented.
    - Once the above steps are executed “status” shall be changed to “true” and “num” shall be incremented.
- Num shall not be an even number.
- Output shall have 100 numbers, not more than that or less than.
- Output shall not have any non-prime number.