- Write a SQL query to retrieve the top 5 customers who have spent the most in the last year from an `orders` table. The `orders` table contains the following columns: `order_id`, `customer_id`, `order_date`, and `order_total`. Assume the current date is `2024-08-14`. You should account for customers who may have multiple orders. The query should return the customer ID and the total amount spent in
  descending order.

  ```
  SELECT top 5 * FROM
  (SELECT order_total, customer_id
  FROM orders
  where YEAR(order_date) = YEAR('2024-08-14') -1)
   ORDER BY order_total , customer_id desc;
  ```

- Describe how you would prioritize and plan the following support tickets:
  - A customer reports a system outage affecting multiple users.
  - A customer requests a password reset.
  - A customer reports a minor UI bug.
  - A customer requests help with a new feature.

  I will prioritize the system outage or system failure promptly to prevent potential data breaches and ensure customer satisfaction. Second will be password reset since it is important for user's security and access and these needs to be addressed as soon as possible. Third will be the minor UI bug, even if it is minor it may cause confusion and or series of issues on the customer's side. Last but important as well is to help users with new system features, this can be addressed by providing the user with published 'how to' documents. Documents must be up to date so users can retrieve it whenever they need it. It is our priority to provide support to users but this can be handled by L1 and only reach for L2 if the issue persists.

- Explain how you would diagnose a situation where a database query that previously ran in under a second is now taking several minutes to complete.

First I will check my internet if the connection is fine since connection to the database relies on the internet or vpn. Second I will check communication sites, emails, announcements if there are network activities, patching, deployment on the systems we are using, as some activities require killing of sessions and turning-on downtime on the servers we use. Third will be, check if there are characters accidentally inserted/removed after running the query the first time and causing the prolonged run of the query the second time.

- How would you approach handling a recurring error in a production environment that affects multiple users but has not been reproducible in the testing environment?

If I cannot reproduce the issue in the lower environment the only way to resolve it is connect with the dev. First, check all data and identify where the process stopped working properly. Second, connect with the user if the same issue is persisting. Third, connect with the dev to do code check or debug and providing them all my findings and data as reference. This may be caused by code change/ patching and L3 will be able to help L2 for these kind of issues.

- Describe how you would use JIRA to track a support ticket from the time it's created to the time it’s resolved. Include statuses you would use, how you ensure proper communication and documentation at each stage.

JIRA has its KANBAN feature where we can create classifications of status of an incident ticket. For example, NEW INCIDENT, WORK IN PROGRESS, BACKLOG, NEEDS INVESTIGATION, NEEDS CHANGE, etc. and if the incident is already closed, then that incident shouldn’t be in the KANBAN anymore it needs to be closed or deleted. In every incident ticket there is a comment/ communication

part at the bottom where we can communicate with anyone in your team or other teams as well. We can also attach the incident link thru TEAMS, EMAIL and other comms options to share it to others faster.

- What steps would you take to optimize a slow-running SQL query? Provide at least three different approaches you could try.

1. I will change the response time on the query of the sql app. To make it faster.
2. Make my query specific as much as possible so that the query wouldn't have to load a bunch of records that isn't what I am looking for.
3. Use joins , cases, and other functions properly and make sure that the similar columns on the tables are properly mapped to avoid redundancy of records returned therefore will optimize running of the query.

- How would you use Postman to test an API endpoint that requires both a specific header and a JSON body? Include the steps you would take to validate the response.

First check the API documentation to understand how it works and what's in it. There
you will also see the endpoints you can use for the response you are looking for.
Provide the API and the endpoint on the query tab. Run the query by clicking SEND.
On the result part of the window you will see the **header** Content-type: **application/JSON** and on the **body** part you will see it is in **JSON.** The body recognizes that the result should be in JSON that's is why it displayed the result in JSON format.
You will see that your query is good by checking the status on the result part of the screen
Response 200 - means that the request is understood and everything is fine.
Response 400 - means a bad request, either used a nonexistent parameter value or non existent endpoint.

You can also validate it by viewing the console for detailed information about your request.

- An user is experiencing intermittent connectivity issues with our application that hosted on AWS. What steps would you take to troubleshoot and resolve the issue?

It is always important to identify the scope of the issue and use the tools to identify the problem.

1.) Are all users affected?
2.) Are all our applications hosted on AWS affected?
3.) Are there any ongoing issues with AWS?
4.) Since it is our application hosted in AWS therefore we must have a monitoring tool or we have logs when the issue started and is it related to the number of users using the app or number of transactions it is handling?
5.) Since it is in the cloud, we are the ones configuring the instances, ACL and the route traffic. Are they in the correct setting?
6.) We can use AWS tools to identify the source of the problem which are the Traceroute to check routing and MTR to check the latency. Are the results fine?
7.) Also check the utilization. Since it is in the cloud we need to monitor proactively the utilization to avoid overloading as this might cause issues.
8.)If everything looks good, yet the issue persists then use the AWS reachability analyzer for you to identify which of the following components are not working fine and where the process stops.

We can compare the configurations when it was last working properly and the configuration when it stopped working. If it differs, then most likely we need to configure it back.

Database Investigation:

- **Scenario:** A user reports that when using the company's web application to search for orders placed within the last month, the results are missing some records that should be included. The user is particularly concerned because they know of specific orders that should appear but do not. Your task is to investigate and resolve the issue.
- **Table Schema:**
    a. `orders` Table:
        - `order_id` (INT, Primary Key)
        - `customer_id` (INT, Foreign Key referencing `customers` table)
        - `order_date` (DATETIME)
        - `order_total` (DECIMAL)
        - `status` (VARCHAR, e.g., 'completed', 'pending', 'cancelled')

    b. `customers` Table:
        - `customer_id` (INT, Primary Key)
        - `customer_name` (VARCHAR)
        - `email` (VARCHAR)
        - `country` (VARCHAR)
- **Task:**
    a. Write a SQL query or queries to investigate the issue and explain the possible reasons why records might be missing. Document the steps you would take to identify the root cause, and propose a solution to fix it. (You can make assumptions for information not given)

    For example the user provided the details below that should be in the table but cannot be found.

    Order_id: 12354564

    Customer_id: 1487

    Order_date: 2024/08/19

Customer_name: Andi

Records existing in the tables

orders

ORDER_ID	CUSTOMER_ID	ORDER_DATE	ORDER_TOTAL	STATUS
6468764	8645	2024-08-23	14.50	completed
8796135	1487	2024-08-25	15.90	canceled
1234566	1486	2024-08-19	22.60	canceled
1234565	1487	2024-08-19	12.99	pending

customers

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	COUNTRY
8645	Gabrielle Jose	gab.08@gmail.com	Philippines
1487	Claren Phen	clrnintl31@gmail.com	Thailand
1486	Andi Mateo	andzzz@gmail.com	Philippines

My step by step query :

I will check the orders table first using two columns to reduce the query runtime.

Select *from orders where order_id = 1234564 and customer_id = 1487;

-No results-

Validate the customer details in the customers table

Select *from customers where customer_id = 1487; -- the customer exist with customer_id but different customer_name. Mismatch found.

Result:

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	COUNTRY
1487	Claren	clrnintl31@gmail.com	Thailand

Select *from customers where customer_name = 'Andi Mateo';

Result: - -we now know the correct customer_id of the user

CUSTOMER_ID	CUSTOMER_NAME	EMAIL	COUNTRY
1486	Andi Mateo	andzzz@gmail.com	Philippines

Then, Recheck orders table with order_id only.

Select *from orders where order_id = 1234564;

-No Results- **--** Order_id itself does not exist in the table

First I will ask users where the order_id came from? like does the order_id they provided really exists or they guessed it or mistyped it? No results in the query have different causes. One, the data that the user provided is wrong or mistyped. Second, the data provided have mismatching details.

Results of the investigation: the order_id provided is not existing. To find her real order, we can use the query below :

Select * from orders where order_id like '%12345%' and customer_id =1486 and order_date = '2024-08-19';

ORDER_ID	CUSTOMER_ID	ORDER_DATE	ORDER_TOTAL	STATUS
1234566	1486	2024-08-19	22.60	canceled

Final step: confirm to the user if the data that we see from our end is rather the data she's actually looking for.

Support Case Study:

- **Scenario:** A critical production bug is reported by a high-priority client. The end user is using a feature in our web application that generates stats dashboard based on specific data inputs. The user reports that when they try to generate the dashboard, the system either times out or returns incomplete data. This feature is crucial for the customer's operations, and they have a major deadline approaching. The initial ticket provides limited information: "Stats dashboard is failing; need urgent fix."
- **Additional Context:**
    - The web application has recently undergone a minor update that involved changes to the analytic module.
    - The customer is accessing the application during peak usage hours, and you have noticed similar but non-critical issues reported by other users in the past week.
    - The dashboard feature interacts with several backend services, including a database and an external API for fetching additional data.
- **Task:** Outline a step-by-step investigation plan, including how you would gather additional details from the customer, diagnose the root cause using tools such as monitoring logs and SQL queries, and manage communication with the customer to keep them informed and manage their expectations. Also, propose both immediate and long-term solutions to mitigate the issue and prevent future occurrences. (You can make assumptions for information not given)

1. Gather the data that answers the following questions:
    - When did the issue actually started?
    - What are the samples of the specific inputs/samples the user tries to generate?
    - What are the specifics of the update implemented on the app?

- Does the issue persist even during off-peak hours?
- Any Network changes happening? As this may cause interruption in database and API connection
- Any database (Oracle/microsoft) Activity?
- Are other servers working properly?

Having the answers here might help identify the cause of the problem.
Is the issue only encountered during peak hours? Or even in off-peak hours? Does the minor update have something to do with the number of users or time? If it does then, that's the issue. If not then, are the non-critical incidents reported by users started after the minor update implemented in the analytic module?If yes then, what are the specific parts of the module that has been updated? As it may be the exact reason why there's an issue.
If the issue was concluded not caused by the update then, what are the specific data inputs the user is actually trying to generate using our app? Is it coming from our database? If the data is coming from our database then are there any on-going Network change? If yes and there is a loss of connection from our database to our online app then it is safe to say that the reason why they're seeing 'Stats dashboard is failing; need urgent fix' error is caused by a Network change or other activities on the application we use.
If the issue persists even the network activity and the recent minor update has been ruled out then let's check our servers if they are receiving and responding properly to our queries. Is the server up and running? Are we able to retrieve data from the server using our database? If Yes, then is our application feature available? Or is it accidentally turned off by a support trainee? Then we found where the problem comes from.

We should initiate a bridge call immediately that includes all the teams from upstream, downstream, dev and customer of our application to identify where the process stopped. We can ask the dev questions related to the implemented update whether it needs to be rolled back or not. And also ask upstream teams whether there are ongoing changes on their end that may affect our application.

We can answer some of the questions by looking at the monitoring tools and our previous logs. Not sure yet what are the tools available in the project but there should be a dashboard tool where all the components of our application are being monitored. We can create tasks for our support resources to monitor it from time to time or we can create automations that will send emails to all the support team members the

status of the application, the servers, and other features. We should have another automation that will try to query something on servers/applications' features to check whether it is responding or not.

Early detection helps to avoid major issues that could potentially cause breach SLA.