Real-Interim-April-Previously-Asked

- 11. What is a clause in SQL
  - 🔹 Common SQL Clauses (with simple explanations):
  - Example Query Using Multiple Clauses:
  - What Each Clause Does:
- 12. Write a subquery for displaying the max salary?
- 13. What is window functions in SQL?
- 14. What is the difference between EXISTS and IN
  - Basic Difference
  - Using IN:
  - Using EXISTS:
- 15. What are the SQL Data types?
  - What are SQL Data Types?
  - 1. Numeric Data Types
  - 2. Character/String Data Types
  - 3. Date and Time Data Types
  - 4. Boolean Data Type
  - 5. Binary Data Types
- 16. What is DISTINCT keyword?
- 17. What is Primary and Foreign Key?
- 18. What is the difference between JOIN and UNION?
  - JOIN Example
  - UNION Example

- Python Questions
  - 1. Difference between list and tuple?
  - 2. Difference between dict and list?
  - 3. What is a lambda function? Write a code
  - 4. Difference between extend and append? Explain using Code. Show the output.
  - 5. What is the difference between list and arrays?
  - 6. What is python decorators
  - 7. What do you understand by iterator in python?
  - 8. What is a zip function?
  - 9. Handling errors in python with try except block
  - 10. What are modules and how to import with examples

        - Examples:
            - Import a whole module:
        - Import specific functions or variables:
- Azure Questions
    - 1. What do you understand by storage account in azure? And its type
    - 2. What is azure blobs storage
    - 3. How blobs differ from azure files?
    - 4. What is ADF?
    - 5. What are the different ways to execute pipelines in ADF?
    - 6. Is there any way to monitor a pipeline?
    - 7. Explain the role of azure monitor and azure log analytics in monitor and managing azure resources
    - 8. What is scaling in azure?
    - 9. What is an azure function
    - 10. What is the difference between azure SQL DB and Synapse analytics
    - 11. What is Integration Runtime(IR) and its types?
    - 12. What is ARM Templates?
    - 13. What is azure monitor
    - 14. What is azure synapse analytics?

SQL Questions

1. Display second highest salary

```
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (
    SELECT MAX(Salary) FROM Employees
);
```

- The inner query gets the highest salary.
- The outer query finds the maximum salary that is less than the highest, which is the second highest.

2. What is the difference between Where and Having

- WHERE filters rows before grouping, and it can't use aggregate functions.
- HAVING filters groups after aggregation and can use functions like SUM() or AVG()

Example-WHERE

```
SELECT * FROM Sales
WHERE SalesAmount > 1000;
```

Example-HAVING

```
SELECT Region, SUM(SalesAmount) AS TotalSales
FROM Sales
GROUP BY Region
HAVING SUM(SalesAmount) > 2000;
```

3. What are the joins you used frequently in SQL, and explain different types of Joins

In SQL, joins are used to combine rows from two or more tables based on a related column between them.

In SQL, **joins** are used to combine rows from two or more tables based on a related column between them. Here's a **viva-friendly explanation** of the most commonly used joins and their differences:

1. INNER JOIN

- **Returns:** Only the rows that have matching values in both tables.
- **Use Case:** Most common join — used when you only want data that exists in both tables.

```
SELECT *
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

2. LEFT JOIN (or LEFT OUTER JOIN)

- **Returns:** All rows from the **left table**, and matching rows from the right table. If no match, returns `NULL` for right table columns.
- **Use Case:** When you want all records from one table, even if there's no match in the other.

```
SELECT *
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

3. RIGHT JOIN (or RIGHT OUTER JOIN)

- **Returns:** All rows from the **right table**, and matching rows from the left table. If no match, returns `NULL` for left table columns.
- **Use Case:** Less common, but useful when the right table is the main focus.

```
SELECT *
FROM Orders
RIGHT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

4. FULL JOIN (or FULL OUTER JOIN)

- **Returns:** All rows from both tables. If there's no match, returns `NULL` for missing side.
- **Use Case:** When you want to see all data, matched or unmatched.

```
SELECT *
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

5. CROSS JOIN

- **Returns:** Cartesian product — every row from the first table is combined with every row from the second.
- **Use Case:** Rarely used, but useful for generating combinations.

```
SELECT *
FROM Products
CROSS JOIN Categories;
```

Summary Table

Join Type	Returns	Use Case
INNER JOIN	Matching rows from both tables	Most common, matched data only
LEFT JOIN	All rows from left + matched from right	Keep all left-side data
RIGHT JOIN	All rows from right + matched from left	Keep all right-side data
FULL JOIN	All rows from both tables	See everything, matched or not
CROSS JOIN	All combinations of rows	Generate combinations

4. Nested loop in SQL?

A nested loop in SQL can be simulated using a correlated subquery, where the inner query runs for each row of the outer query — similar to a loop inside a loop.

Example

```
SELECT e.EmployeeID, e.Name,
       (SELECT COUNT(*)
        FROM Orders o
        WHERE o.EmployeeID = e.EmployeeID) AS OrderCount
FROM Employees e;
```

For each employee, the inner query counts orders related to that employee

5. Hash join and merge join?

Hash Join

- The database creates a hash table from the smaller table (usually the one on the left).
- Then it scans the larger table and matches rows using the hash.

```
SELECT *
FROM Orders
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

If Customers is small, the engine may use a hash join.

Merge Join

- Both tables are sorted on the join key.
- The engine merges them like merging two sorted lists.

```
SELECT *
FROM Sales
JOIN Targets ON Sales.RegionID = Targets.RegionID;
```

If both tables are already sorted by RegionID, a merge join is efficient.

6. What are index?

- An index is a database object that improves query performance by allowing faster data retrieval.
- It works like a book index — instead of scanning every row, the database uses the index to jump directly to the needed data.

7. What is the difference between cluster and non cluster index

- A clustered index sorts and stores the actual data rows in order, so there can only be one.

- A non-clustered index stores a separate structure with pointers to the data, and you can have many of them.
- Clustered is best for sorting and range queries, while non-clustered is great for quick lookups

Clustered Index = The actual book shelf

- Books are **physically arranged** in alphabetical order.
- If you want a book by title, you go directly to the shelf and find it — **fast and direct**.
- Only **one way** to arrange books physically — so only **one clustered index** per table.

Non-Clustered Index = The index page at the front of the library

- It lists book titles and tells you **which shelf** to go to.
- You use it to **look up** a book quickly, then go to the shelf.
- You can have **many index pages** for different things — title, author, genre — so **multiple non-clustered indexes** are allowed.

Feature	Clustered Index	Non-Clustered Index
Data Arrangement	Data is **physically sorted**	Data is **not sorted**, just pointed to
Number Allowed	Only **one** per table	Can have **many**
Speed	Fast for **range queries** and sorting	Fast for **specific lookups**
Analogy	Actual **book shelf**	**Index page** pointing to books

8. Difference between Drop, truncate and delete

1. DELETE

- What it does: Removes specific rows from a table.

```
DELETE FROM Employees WHERE Department = 'HR';
```

2. TRUNCATE

- What it does: Removes all rows from a table — faster than DELETE.

```sql
TRUNCATE TABLE Employees;
```

3. DROP

- What it does: Deletes the entire table (structure + data).

```sql
DROP TABLE Employees;
```

9. Write a delete command?

```sql
DELETE FROM Employees WHERE Department = 'HR';
```

10. What is view in sql? Write a query for create update and delete view

A view is a virtual table based on the result of a SELECT query. It doesn't store data itself but shows data from one or more tables. You can use it like a regular table in your queries.

1. Create View

```sql
CREATE VIEW EmployeeView AS
SELECT EmployeeID, Name, Department
FROM Employees
WHERE Status = 'Active';
```

➡ This view shows only active employees.

2. Update View

You cannot directly update a view's structure. Instead, you drop and recreate it:

```sql
DROP VIEW EmployeeView;
```

```sql
CREATE VIEW EmployeeView AS
SELECT EmployeeID, Name, Department, JoiningDate
FROM Employees
WHERE Status = 'Active';
```

3. Delete a view

```sql
DROP VIEW EmployeeView;
```

11. What is a clause in SQL

A clause in SQL is a part of a query that performs a specific function, like filtering (`WHERE`), grouping (`GROUP BY`), or sorting (`ORDER BY`). Clauses are combined to form complete SQL statements.

- **Common SQL Clauses (with simple explanations):**

Clause	Purpose
SELECT	Specifies the columns to retrieve
FROM	Specifies the table(s) to retrieve data from
WHERE	Filters rows based on a condition
GROUP BY	Groups rows that have the same values in specified columns
HAVING	Filters groups (used with `GROUP BY`)
ORDER BY	Sorts the result set by one or more columns
JOIN	Combines rows from two or more tables based on a related column
LIMIT / TOP	Limits the number of rows returned (depends on SQL dialect)

Example Query Using Multiple Clauses:

```sql
SELECT Department, COUNT(*) AS EmployeeCount
FROM Employees
WHERE Status = 'Active'
GROUP BY Department
```

```
HAVING COUNT(*) > 5
ORDER BY EmployeeCount DESC;
```

What Each Clause Does:

- `SELECT` : Choose columns to display
- `FROM` : Choose the table
- `WHERE` : Filter rows before grouping
- `GROUP BY` : Group rows by department
- `HAVING` : Filter groups with more than 5 employees
- `ORDER BY` : Sort the result

12. Write a subquery for displaying the max salary?

```
SELECT Salary
FROM Employees
WHERE Salary = (
    SELECT MAX(Salary)
    FROM Employees
);
```

- The inner query (SELECT MAX(Salary)) finds the highest salary.
- The outer query returns the row(s) where the salary matches that maximum.

13. What is window functions in SQL?

A window function performs a calculation across a set of rows that are related to the current row — without collapsing the result into a single value like GROUP BY does.

Example

```
SELECT EmployeeID, Name, Salary,
       RANK() OVER (ORDER BY Salary DESC) AS SalaryRank
```

```
FROM Employees;
```

➡ This assigns a rank to each employee based on salary, without grouping the data.

- Window functions
  - RANK()
    - Assigns rank with gaps
  - DENSE_RANK()
    - Assigns rank without gaps
  - ROW_NUMBER()
    - Gives a unique row number
  - SUM()
    - Running total
  - AVG()
    - Moving average
  - LEAD() / LAG()
    - Access next/previous row values

14. What is the difference between EXISTS and IN

Basic Difference

Feature	`IN`	`EXISTS`
Purpose	Checks if a value exists in a list or subquery	Checks if a subquery returns any rows
Returns	True if the value is found in the list	True if the subquery returns at least one row
Performance	Better for small datasets	Better for large datasets or correlated subqueries

Using `IN`:

```sql
SELECT name FROM students
WHERE id IN (SELECT student_id FROM marks WHERE score > 90);
```

- This checks if the student's `id` is **in the list** of `student_id` from the `marks` table.

Using `EXISTS`:

```sql
SELECT name FROM students s
WHERE EXISTS (
  SELECT 1 FROM marks m
  WHERE m.student_id = s.id AND m.score > 90
);
```

- This checks if **there exists** a row in `marks` where the `student_id` matches and score is above 90.

1. **When to use `IN`?**
   - Use when you have a **static list** or a **simple subquery**.
   - Example: `WHERE country IN ('India', 'USA', 'UK')`
2. **When to use `EXISTS`?**
   - Use when you need to **check existence** of related data.
   - Especially useful in **correlated subqueries**.

15. What are the SQL Data types?

What are SQL Data Types?

SQL data types define the **type of data** that can be stored in a column of a table. They help the database understand how to **store, retrieve, and process** the data.

1. Numeric Data Types

Used to store numbers.

Data Type	Description
`INT`	Integer numbers (e.g., 1, 100, -50)

Data Type	Description
`FLOAT`	Floating-point numbers (e.g., 3.14)
`DECIMAL(p,s)`	Exact numbers with precision
`SMALLINT`	Smaller range of integers
`BIGINT`	Larger range of integers

2. Character/String Data Types

Used to store text.

Data Type	Description
`CHAR(n)`	Fixed-length string (e.g., CHAR(10))
`VARCHAR(n)`	Variable-length string
`TEXT`	Large text data

3. Date and Time Data Types

Used to store date and time values.

Data Type	Description
`DATE`	Stores date (YYYY-MM-DD)
`TIME`	Stores time (HH:MM:SS)
`DATETIME`	Stores date and time
`TIMESTAMP`	Stores date and time with time zone

4. Boolean Data Type

Stores true/false values.

Data Type	Description
`BOOLEAN`	Stores TRUE or FALSE

5. Binary Data Types

Used to store binary data like images or files.

Data Type	Description
`BINARY`	Fixed-length binary data
`VARBINARY`	Variable-length binary data
`BLOB`	Binary Large Object (e.g., images)

16. What is DISTINCT keyword?

The DISTINCT keyword in SQL is used to remove duplicate values from the result set of a query. When we apply DISTINCT to a column or a combination of columns, it ensures that only unique rows are returned for that selection.

17. What is Primary and Foreign Key?

- A Primary Key is a column or a set of columns in a table that uniquely identifies each row in that table.
- It cannot have NULL values and must be unique for every record. For example, a student_id in a Students table can be a primary key, because each student will have a unique ID.
- A Foreign Key is a column in one table that refers to the Primary Key of another table. It is used to establish a relationship between two tables. For instance, if we have a Marks table with a student_id column, that student_id can be a foreign key that refers to the student_id in the Students table.

18. What is the difference between JOIN and UNION?

- The JOIN and UNION clauses are both used to combine data from multiple tables, but they work differently:
- JOIN is used to combine columns from two or more tables based on a related column between them, usually a foreign key. It merges rows horizontally.
    - Example: INNER JOIN, LEFT JOIN, etc.

- UNION is used to combine rows from two or more result sets that have the same number and type of columns. It merges rows vertically.
    - It removes duplicate rows by default unless we use UNION ALL.

JOIN Example

Tables:

Students

student_id	name
1	Alice
2	Bob

Marks

student_id	score
1	85
2	90

Query (JOIN):

```
SELECT Students.name, Marks.score
FROM Students
JOIN Marks ON Students.student_id = Marks.student_id;
```

Result:

name	score
Alice	85
Bob	90

This joins the two tables based on `student_id` and combines the **columns**.

UNION Example

Table 1:

```
SELECT name FROM Students
```

Table 2:

```
SELECT name FROM Teachers
```

Query (UNION):

```
SELECT name FROM Students
UNION
SELECT name FROM Teachers;
```

Result (Example):

name
Alice
Bob
David

This combines the **rows** from both tables into one list of names, removing duplicates.

Python Questions

1. Difference between list and tuple?

The main difference between a **list** and a **tuple** is that:

- A **list** is **mutable**, meaning we can **change**, **add**, or **remove** elements after it is created.
- A **tuple** is **immutable**, so once it's created, its elements **cannot be changed**.
  Other differences include:
- **Syntax:**
  - List: uses square brackets `[ ]`
  - Tuple: uses parentheses `( )`

- **Use cases:**
  - Lists are used when data may need to be modified.
  - Tuples are used when data must be **constant and protected**.
  Example:

```
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
```

If you try to change `my_tuple[0] = 10`, it will give an error, but doing that in a list is allowed.

2. Difference between dict and list?

- A **list** is an **ordered collection** of elements accessed by their **index** (position), starting from 0.
- A **dictionary (dict)** is an **unordered collection** of **key-value pairs**, where each value is accessed by a unique **key** instead of a numeric index

```
my_list = [10, 20, 30]
my_dict = {'a': 10, 'b': 20, 'c': 30}

print(my_list[1])    # Output: 20
print(my_dict['b'])  # Output: 20
```

3. What is a lambda function? Write a code

A **lambda function** is a small, anonymous function defined using the `lambda` keyword. It can have any number of input parameters but only **one expression**, which is evaluated and returned. Lambda functions are often used for short, simple operations where defining a full function is unnecessary.

```
# A lambda function that adds 10 to the input
add_ten = lambda x: x + 10
```

```
print(add_ten(5))  # Output: 15
```

4. Difference between extend and append? Explain using Code. Show the output.

- `append()` adds **a single element** to the end of the list. If you append another list, it adds that entire list as **one element** (nested list).
- `extend()` adds **each element** of an iterable (like a list) to the list, extending the list by multiple elements.

Example

```
my_list = [1, 2, 3]

# Using append
my_list.append([4, 5])
print("After append:", my_list)

# Reset list
my_list = [1, 2, 3]

# Using extend
my_list.extend([4, 5])
print("After extend:", my_list)
```

Output

```
After append: [1, 2, 3, [4, 5]]
After extend: [1, 2, 3, 4, 5]
```

5. What is the difference between list and arrays?

- **List** is a built-in Python data structure that can hold **elements of different data types** (e.g., integers, strings, objects) together.
- **Array** (from the `array` module or libraries like NumPy) holds elements of the **same data type**, which makes arrays more **memory-efficient** and faster for numerical operations.

6. What is python decorators

A **decorator** in Python is a special function that **modifies or enhances** the behavior of another function or method **without changing its code**. It “wraps” a function, adding extra functionality before or after the original function runs.

Example

```
def my_decorator(func):
    def wrapper():
        print("Before the function runs")
        func()
        print("After the function runs")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")

say_hello()
```

Output

```
Before the function runs
Hello!
After the function runs
```

7. What do you understand by iterator in python?

An **iterator** in Python is an object that allows you to **traverse through all the elements** of a collection (like a list, tuple, or dictionary) one at a time. It implements two main methods:

- `__iter__()` which returns the iterator object itself.
- `__next__()` which returns the next item from the collection. When there are no more items, it raises a `StopIteration` exception.

Example

```
my_list = [1, 2, 3]
it = iter(my_list)  # Get iterator object

print(next(it))  # Output: 1
print(next(it))  # Output: 2
print(next(it))  # Output: 3
# next(it) now raises StopIteration
```

8. What is a zip function?

- The `zip()` function takes two or more lists (or other collections) and puts their items together in pairs (or groups), matching items by their position.
- For example, it pairs the first item of each list, then the second item of each list, and so on.
- It stops when the shortest list runs out of items.
  Example

```
numbers = [1, 2, 3]
letters = ['a', 'b', 'c']

result = zip(numbers, letters)

print(list(result))
```

Output

```
[(1, 'a'), (2, 'b'), (3, 'c')]
```

9. Handling errors in python with try except block

- In Python, errors (also called exceptions) can be handled using a **`try-except`** block. You put the code that might cause an error inside the `try` block, and if an error occurs, the program jumps to the `except` block where you can handle the error gracefully instead of crashing.

Example

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print("Result is", result)
except ZeroDivisionError:
    print("Error: You cannot divide by zero!")
except ValueError:
    print("Error: Please enter a valid number!")
```

- If the user enters zero, it will catch the `ZeroDivisionError`.
- If the user enters something that's not a number, it will catch the `ValueError`.
- This way, the program continues running without crashing.

10. What are modules and how to import with examples

A **module** in Python is a file containing Python code — like functions, classes, or variables — that you can reuse in other Python programs. Modules help organize code and make it easier to manage and reuse.

Examples:

Import a whole module:

```
import math
print(math.sqrt(16))  # Output: 4.0
```

Import specific functions or variables:

```
from math import pi, sqrt
print(pi)            # Output: 3.141592653589793
print(sqrt(25))      # Output: 5.0
```

Import with an alias:

```
import math as m
print(m.sqrt(9))     # Output: 3.0
```

Azure Questions

1. What do you understand by storage account in azure? And its type

- An **Azure Storage Account** is a container in Microsoft Azure that holds all your data objects like blobs, files, queues, tables, and disks.
- **Standard General-purpose v2:**
  - Stores blobs (files like images, videos), files, queues, and tables.
  - Good for most uses.
  - Offers different ways to keep your data safe by making copies (called redundancy).
- **Premium Block Blobs:**
  - For storing blobs (like big files) that need fast access and many transactions.
  - Best if you work with lots of small or frequently changing files.
- **Premium File Shares:**
  - Special for fast and reliable file shares.
  - Supports common file sharing protocols used in businesses.
- **Premium Page Blobs:**
  - Used mainly for things like virtual machine disks and databases.
  - Designed for fast read/write operations on blocks of data.

2. What is azure blobs storage

Azure Blob Storage is a service in Azure that lets you store **large amounts of unstructured data** like images, videos, documents, backups, and logs. “Blob” stands for **Binary Large Object**.

It’s used when you need to save files that don’t have a fixed format or structure, and you want to access them from anywhere over the internet.

3. How blobs differ from azure files?

Feature	Azure Blob Storage	Azure Files
Purpose	Store large unstructured data like images, videos, backups (objects)	Provide fully managed file shares that work like a network drive
Access method	Accessed via HTTP/HTTPS with REST APIs	Accessed via SMB or NFS protocols (like regular file shares)
Use case	Storing files for apps, streaming, backups, big data	Sharing files between multiple VMs or users like a shared folder
Structure	Object storage (blobs)	File system storage with folders and files
Mountable?	No	Yes, can be mounted like a network drive

4. What is ADF?

Azure Data Factory (ADF) is a cloud-based service that helps you **move, transform, and manage data** from different sources to different destinations. It lets you create data pipelines to automate data workflows.

5. What are the different ways to execute pipelines in ADF?

- **Manual Trigger:**
  Run the pipeline manually from the portal or command line.
- **Scheduled Trigger:**
  Run pipelines on a fixed schedule (like every day at 1 AM).
- **Event-based Trigger:**
  Run pipelines when an event happens, such as a new file arriving in Blob Storage.
- **Tumbling Window Trigger:**
  Runs pipelines in fixed, non-overlapping time intervals (called windows), like every hour or every day.

6. Is there any way to monitor a pipeline?

1. **ADF Monitoring Dashboard:**
   You can see the status of your pipelines, activities, and triggers in the Azure portal under the **Monitor** tab.
2. **Activity Runs and Pipeline Runs:**
   View detailed logs for each pipeline run and individual activity runs, including success, failure, duration, and error messages.
3. **Alerts and Notifications:**
   You can set up alerts to notify you by email or other methods when pipelines fail or succeed.

7. Explain the role of azure monitor and azure log analytics in monitor and managing azure resources

- **Azure Monitor:**
  It is a full **monitoring service** in Azure that collects and analyzes **performance and health data** from your Azure resources, applications, and infrastructure. It helps you **detect issues, diagnose problems, and get alerts** so you can keep your services running smoothly.
- **Azure Log Analytics:**
  It is a **tool within Azure Monitor** that lets you collect, search, and analyze **log data** from

your Azure resources. You write queries to explore logs and create reports or dashboards to understand what's happening in your environment.

8. What is scaling in azure?

Scaling in Azure means **changing the amount of resources** (like CPU, memory, or instances) your application or service uses to handle more or less work.

9. What is an azure function

- An **Azure Function** is a **serverless compute service** that lets you run small pieces of code (called *functions*) without managing servers. You only pay for the time your code runs.
- Runs code in response to events (like HTTP requests, timer schedules, or messages).
- Supports many programming languages like C#, Python, JavaScript, etc.
- Great for building lightweight, event-driven applications or microservices.

10. What is the difference between azure SQL DB and Synapse analytics

Aspect	Azure SQL Database	Azure Synapse Analytics
What it is	A cloud database to store and manage data for apps	A big data tool for analyzing huge amounts of data
Use	Used for day-to-day app data and fast transactions	Used for running complex reports and analyzing big data
Data size	Good for small to medium data	Designed for very large data sets
Speed	Fast for normal database queries	Handles very big and complex queries
Extra tools	Basic database features	Works with big data tools like Spark and Data Lake

- Use **Azure SQL Database** when you want to store app data and run regular queries quickly.
- Use **Azure Synapse** when you need to analyze lots of data and run complex reports.

11. What is Integration Runtime(IR) and its types?

Integration Runtime (IR) is the **compute infrastructure** used by Azure Data Factory (ADF) to **move data and run activities** like data transformation and copying between different data stores.

Types of Integration Runtime:

1. **Azure Integration Runtime:**
   - Runs in the cloud.
   - Used for data movement, dispatching activities, and data flow execution in the cloud.
   - Handles copying data between cloud data stores.
2. **Self-hosted Integration Runtime:**
   - Installed on your own on-premises or private environment (like your local machine or VM).
   - Used to securely move data between on-premises and cloud or between private networks.
3. **Azure-SSIS Integration Runtime:**
   - Runs SSIS (SQL Server Integration Services) packages in Azure.
   - Helps migrate existing SSIS workflows to the cloud without changing much.

12. What is ARM Templates?

ARM Template (Azure Resource Manager Template) is a **JSON file** that defines the **infrastructure and configuration** for your Azure resources in a declarative way.

Key points:

- It describes what resources (like VMs, databases, storage accounts) you want to create or update.
- Helps automate deployment and makes it repeatable and consistent.
- You don’t write code to create resources step-by-step; instead, you declare the desired state in the template.
- Supports parameters, variables, and outputs for flexible and reusable deployments.

ARM Templates help you **automate and manage Azure resources** by defining them in a simple JSON file you can deploy anytime

13. What is azure monitor

Azure Monitor is a service in Azure that helps you **collect, analyze, and act on data from your Azure resources** and applications.

- Tracks metrics (like CPU usage, memory) and logs (events and errors).
- Helps detect problems and alerts you when something goes wrong.
- Provides dashboards and tools to understand how your applications and resources are performing.
- Supports integration with other tools for advanced monitoring and automation.

14. What is azure synapse analytics?

Azure Synapse Analytics is a cloud service that helps you **store, manage, and analyze large amounts of data** all in one place.

Azure Synapse is a powerful tool for **big data analytics and reporting** in the cloud.Azure Synapse is a powerful tool for **big data analytics and reporting** in the cloud.

Key points:

- Combines big data and data warehousing.
- Lets you run fast queries on huge data sets using SQL or Spark.
- Integrates with data lakes, machine learning, and business intelligence tools.
- Helps businesses get insights from their data quickly.