1. Write a SQL query that will provide you with the 10th-highest employee salary from an Employee table.

```
SELECT DISTINCT salary

FROM Employee

ORDER BY salary DESC

LIMIT 1 OFFSET 9;
```

Explanation:

- `ORDER BY salary DESC` sorts salaries in descending order.
- `DISTINCT` ensures unique salaries.
- `LIMIT 1 OFFSET 9` fetches the 10th highest salary (as OFFSET starts from 0).

2. How do you read the last five records from a database using a SQL query?

```
SELECT * FROM Employee

ORDER BY id DESC

LIMIT 5;
```

Alternative (for databases supporting `ROW_NUMBER()`)

```
SELECT * FROM (

    SELECT *, ROW_NUMBER() OVER (ORDER BY id DESC) AS row_num

    FROM Employee

) temp

WHERE row_num <= 5;
```

Explanation:

- `ORDER BY id DESC` sorts records in descending order.
- `LIMIT 5` fetches the last 5 records.

3. Write an SQL query to find the names of employees starting with 'A'.

```
SELECT * FROM Employee

WHERE name LIKE 'A%';
```

Explanation:

- `'A%'` ensures that names start with 'A'.
- `%` is a wildcard that matches any sequence of characters.

4. What is a join in SQL? What are the types of joins?

A **JOIN** in SQL is used to combine rows from two or more tables based on a related column.

Types of Joins:

1. **INNER JOIN** - Returns only matching records from both tables.
2. **LEFT JOIN (LEFT OUTER JOIN)** - Returns all records from the left table and matching records from the right table.
3. **RIGHT JOIN (RIGHT OUTER JOIN)** - Returns all records from the right table and matching records from the left table.
4. **FULL JOIN (FULL OUTER JOIN)** - Returns all records from both tables, with NULLs where there is no match.
5. **CROSS JOIN** - Returns the Cartesian product of both tables (every combination of rows).
6. **SELF JOIN** - A table joins itself based on a related column.

5. Difference between JOIN and UNION in SQL

Feature	JOIN	UNION
Purpose	Combines columns from multiple tables based on a condition.	Combines rows from multiple queries with the same structure.
Number of Tables	Works with multiple tables.	Works with multiple `SELECT` statements.
Duplicate Rows	Can return duplicate rows.	Removes duplicates unless `UNION ALL` is used.
Example	`SELECT A.id, B.name FROM TableA A JOIN TableB B ON A.id = B.id;`	`SELECT name FROM TableA UNION SELECT name FROM TableB;`

6. What is the difference between primary key and unique constraints?

Feature	Primary Key	Unique Constraint
Uniqueness	Ensures unique values.	Ensures unique values.
NULL Values	Cannot contain NULL values.	Can contain NULL values (one per column).
Number Allowed	Only one per table.	Multiple unique constraints per table.

Feature	Primary Key	Unique Constraint
Purpose	Uniquely identifies each row in a table.	Ensures column uniqueness but does not enforce a primary identifier.

7. Why do we use Commit and Rollback commands?

- **COMMIT**: Saves all changes made in the current transaction.
- **ROLLBACK**: Reverts all changes made in the current transaction.

Example:

```
BEGIN TRANSACTION;

UPDATE Employee SET salary = salary + 5000 WHERE department = 'IT';

-- If satisfied

COMMIT;

-- If an error occurs

ROLLBACK;
```

Use Cases:

- Ensures data consistency.
- Helps in error recovery during transactions.

8. How can you fetch common records from two tables?

Using **INTERSECT** (if supported):

```
SELECT * FROM TableA

INTERSECT
```

```
SELECT * FROM TableB;
```

Using **JOIN** (for wider support):

```
SELECT A.* FROM TableA A

INNER JOIN TableB B

ON A.id = B.id;
```

Explanation:

- `INTERSECT` returns only common records.
- `INNER JOIN` fetches matching records based on a condition.

9. What is the need for a MERGE statement?

The **MERGE** statement is used to perform `INSERT`, `UPDATE`, and `DELETE` operations in a single statement based on conditions.

Use Cases:

- Synchronizing tables.
- Upserting (insert if not exists, update if exists).
- Efficient bulk operations.

Example:

```
MERGE INTO Employee AS Target

USING NewEmployee AS Source

ON Target.id = Source.id

WHEN MATCHED THEN

    UPDATE SET Target.salary = Source.salary

WHEN NOT MATCHED THEN

    INSERT (id, name, salary) VALUES (Source.id, Source.name, Source.salary);
```

10. What is the difference between DELETE and TRUNCATE statements?

Feature	DELETE	TRUNCATE
Purpose	Removes specific rows based on condition.	Removes all rows from a table.
WHERE Clause	Supports WHERE clause.	Does not support WHERE clause.
Logging	Logs individual row deletions.	Minimal logging (faster).
Rollback	Can be rolled back.	Cannot be rolled back (in most DBMS).
Auto-increment Reset	Does not reset identity column.	Resets identity column.

Example:

```
DELETE FROM Employee WHERE department = 'HR';

TRUNCATE TABLE Employee;
```

11. What is a Unique key?

A **Unique key** is a constraint that ensures all values in a column or combination of columns are unique.

Characteristics:

- Prevents duplicate values.
- Allows one `NULL` value (unlike primary key).
- Multiple unique keys can exist in a table.

Example:

```
CREATE TABLE Employee (
    id INT PRIMARY KEY,
    email VARCHAR(100) UNIQUE
);
```

Here, `email` must be unique but can have one `NULL` value.

1. Fetch Employee First Name in Uppercase with Alias

```
SELECT UPPER(EmpFname) AS EmpName
FROM EmployeeInfo;
```

Explanation:

- `UPPER(EmpFname)` converts the first name to uppercase.
- `AS EmpName` assigns an alias to the column.

2. Count Number of Employees in 'HR' Department

```
SELECT COUNT(*) AS EmployeeCount
FROM EmployeeInfo
WHERE Department = 'HR';
```

Explanation:

- `COUNT(*)` counts all employees in the 'HR' department.

3. Get the Current Date

```
SELECT CURRENT_DATE;  -- Works in PostgreSQL, MySQL
```

```
-- OR

SELECT GETDATE();  -- Works in SQL Server

-- OR

SELECT SYSDATE FROM DUAL;  -- Works in Oracle
```

Explanation:

- Retrieves the system's current date.

4. Retrieve First Four Characters of `EmpLname`

```
SELECT LEFT(EmpLname, 4) AS FirstFourChars

FROM EmployeeInfo;
```

Explanation:

- `LEFT(EmpLname, 4)` extracts the first 4 characters.

5. Fetch Place Name (String Before Brackets) from Address Column

```
SELECT SUBSTRING_INDEX(Address, '(', 1) AS PlaceName

FROM EmployeeInfo;
```

Explanation:

- `SUBSTRING_INDEX(Address, '(', 1)` fetches the string before `(`.

6. Create a New Table with Data & Structure from Another Table

```
CREATE TABLE NewEmployeeInfo AS

SELECT * FROM EmployeeInfo;
```

Explanation:

- This creates `NewEmployeeInfo` with data and structure from `EmployeeInfo`.

7. Find Employees with Salary Between 50,000 and 100,000

```
SELECT * FROM EmployeeInfo

WHERE Salary BETWEEN 50000 AND 100000;
```

Explanation:

- `BETWEEN` selects salaries within the range (inclusive).

8. Find Employees Whose Names Begin with 'S'

```
SELECT * FROM EmployeeInfo

WHERE EmpFname LIKE 'S%';
```

Explanation:

- `%` wildcard allows any sequence of characters after `S`.

9. Fetch Top N Records

```
SELECT * FROM EmployeeInfo

ORDER BY Salary DESC

LIMIT N;  -- For MySQL, PostgreSQL
```

For SQL Server:

```
SELECT TOP N * FROM EmployeeInfo ORDER BY Salary DESC;
```

For Oracle:

```
SELECT * FROM EmployeeInfo ORDER BY Salary DESC FETCH FIRST N ROWS ONLY;
```

Explanation:

- Retrieves the top N highest-paid employees.

10. Retrieve Full Name by Concatenating First & Last Name

```
SELECT CONCAT(EmpFname, ' ', EmpLname) AS FullName
FROM EmployeeInfo;
```

Alternative for SQL Server:

```
SELECT EmpFname + ' ' + EmpLname AS FullName
FROM EmployeeInfo;
```

Explanation:

- `CONCAT()` merges first and last names with a space.

1. What is Caching?

Caching is the process of storing frequently accessed data in a temporary storage location to reduce access time and improve performance. It helps in minimizing database queries, network latency, and computational overhead.

Types of Caching:

- **Application Caching** (e.g., in-memory caching using Redis)
- **Database Caching** (e.g., query result caching)

- **Content Caching** (e.g., CDN for web pages)
- **Browser Caching** (e.g., storing static files locally)

2. What is Azure Synapse?

Azure Synapse Analytics is an enterprise-grade analytics service by Microsoft that integrates big data and data warehousing solutions. It enables large-scale data processing, real-time analytics, and seamless integration with Power BI and Azure Machine Learning.

Key Features:

- **Synapse SQL** (SQL-based data querying)
- **Spark Pools** (Big data processing using Apache Spark)
- **Pipelines** (ETL workflows similar to ADF)
- **Data Explorer** (Real-time log and telemetry analysis)

3. What are the Various Types of Clusters in Azure Databricks?

Azure Databricks provides three main types of clusters:

1. **All-Purpose Clusters:**
    1. Used for interactive analysis and collaborative data science.
    2. Shared among multiple users.
2. **Job Clusters:**
    0. Created for a specific job/task and terminated once completed.
    1. More cost-effective than all-purpose clusters.
3. **High-Concurrency Clusters:**
    0. Designed for concurrent query execution with multiple users.
    1. Supports **Photon Engine** for faster performance.

Additional Cluster Types Based on Scaling:

- **Standard Cluster** (Manually configured)
- **Autoscaling Cluster** (Dynamically adjusts based on workload)

Would you like more details or explanations on any of these topics? 🚀

Here are the refined questions and answers tailored to your **Azure Data Engineering** profile:

1. What is a JOIN in SQL? What are the types of JOINs?

A **JOIN** in SQL is used to combine rows from two or more tables based on a related column.

Types of JOINs:

- **INNER JOIN** – Returns only matching rows between tables.
- **LEFT JOIN (OUTER JOIN)** – Returns all rows from the left table and matching rows from the right.
- **RIGHT JOIN (OUTER JOIN)** – Returns all rows from the right table and matching rows from the left.
- **FULL JOIN (OUTER JOIN)** – Returns all rows when there is a match in either table.
- **CROSS JOIN** – Returns a Cartesian product of both tables.
- **SELF JOIN** – Joins a table to itself.

2. Difference between JOIN and UNION in SQL

- **JOIN** combines columns from multiple tables based on a condition (`ON`).
- **UNION** stacks the results of two queries vertically (must have the same number of columns).
- **JOIN** retains all matching values, while **UNION** removes duplicates (unless `UNION ALL` is used).

3. Difference between Primary Key and Unique Constraint

- **Primary Key:**
  - Uniquely identifies a record in a table.
  - Only **one** per table.
  - Cannot have `NULL` values.
- **Unique Constraint:**
  - Ensures unique values in a column.
  - Multiple unique constraints can exist in a table.
  - Can have `NULL` values.

4. Why do we use COMMIT and ROLLBACK commands?

- **COMMIT:** Saves all changes permanently in the database.
- **ROLLBACK:** Reverts changes made in the current transaction.

5. How can you fetch common records from two tables?

```
SELECT * FROM TableA

INTERSECT

SELECT * FROM TableB;
```

- `INTERSECT` returns only the common rows between both tables.

6. What is the use of the MERGE statement in SQL?

The `MERGE` statement combines **INSERT, UPDATE, and DELETE** operations into a single query based on conditions.
Example:

```
MERGE INTO TargetTable AS T
```

```
USING SourceTable AS S
ON T.ID = S.ID
WHEN MATCHED THEN
    UPDATE SET T.Name = S.Name
WHEN NOT MATCHED THEN
    INSERT (ID, Name) VALUES (S.ID, S.Name);
```

- Helps in handling **slowly changing dimensions (SCD)** in data warehousing.

7. Difference between DELETE and TRUNCATE statements

- **DELETE:** Removes specific rows (can use `WHERE` clause), logs transactions, can be rolled back.
- **TRUNCATE:** Removes all rows, faster, does not log individual row deletions, cannot be rolled back.

8. What is a Surrogate Key?

A **Surrogate Key** is a unique identifier (usually an auto-incremented ID) that has no business meaning but is used as a **Primary Key**.

Example:

```
CREATE TABLE Employees (
    EmpID INT IDENTITY(1,1) PRIMARY KEY,
    Name VARCHAR(100)
);
```

9. What is a Foreign Key?

A **Foreign Key** is a column that establishes a relationship between two tables by referring to the **Primary Key** in another table.

Example:

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    EmpID INT FOREIGN KEY REFERENCES Employees(EmpID)
);
```

Azure Data Engineering & Big Data-Specific Questions

10. What are Data Quality Rule Checks and Data Shuffling?

- **Data Quality Rule Checks:** Ensure data integrity by validating structure, format, completeness, and consistency.
- **Data Shuffling:** Redistributes data across partitions in **Apache Spark** for parallel processing, improving performance in **Azure Databricks**.

11. Relational vs Non-Relational Databases

Feature	Relational (SQL)	Non-Relational (NoSQL)
Data Model	Structured (tables)	Unstructured/Semi-structured (JSON, key-value, documents)
Scalability	Vertical scaling	Horizontal scaling
Examples	SQL Server, MySQL, PostgreSQL	MongoDB, Cassandra, Cosmos DB

Feature	Relational (SQL)	Non-Relational (NoSQL)
Use Cases	Transactions, structured data	Big data, flexible schema

12. What is Azure Synapse Analytics?

Azure Synapse is a **cloud-based analytics service** that integrates **big data and data warehousing**.

🔷 Features:

- **Synapse SQL** (T-SQL queries over structured data)
- **Apache Spark Pools** (Big data processing)
- **Data Pipelines** (Similar to Azure Data Factory)
- **Integration with Power BI**

13. Difference between Structured and Unstructured Data

- **Structured Data:** Organized in tables, has a predefined schema (e.g., SQL databases).
- **Unstructured Data:** No fixed format, includes images, videos, logs (e.g., Azure Data Lake, Blob Storage).

14. Which ETL Tools Have You Worked With?

- **Azure Data Factory** (ADF)
- **SSIS (SQL Server Integration Services)**
- **Databricks (PySpark for ETL Pipelines)**
- **Talend, Informatica (limited experience if applicable)**

15. What Data Visualization Tools Have You Used?

✅ **Power BI**
✅ **Tableau**
✅ **Google Data Studio**
✅ **Grafana**
✅ **Salesforce Einstein Analytics**

16. Difference Between PySpark and Spark

Feature	PySpark	Spark (Scala/Java)
Language	Python	Scala/Java
Ease of Use	Easier for data analysts	More efficient for large-scale processing
Performance	Slightly slower due to Python overhead	Faster with native JVM execution
Libraries	Pandas, NumPy	Better integration with Hadoop & Java libraries

17. What is Data Modeling?

Data Modeling is the process of **designing a structured representation of data** in a database.

Types of Data Models:

- **Conceptual Model:** High-level business structure.
- **Logical Model:** Defines tables, columns, and relationships.
- **Physical Model:** Actual database schema (indexing, partitions).

Example: **Star Schema vs Snowflake Schema** in Data Warehousing.

Would you like me to refine or expand on any topics? 🚀