Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔷 What is Amazon Redshift?

* Amazon Redshift is a fully managed cloud data warehouse service offered by Amazon Web Services (AWS).

* Designed for analytical workloads involving large-scale data (terabytes to petabytes).

* Supports OLAP (Online Analytical Processing) — optimized for read-heavy queries and reporting.

* Built on PostgreSQL, but heavily optimized for performance and scalability.

🧱 Core Architecture

* Uses a Massively Parallel Processing (MPP) architecture.

* Cluster-based with:

  * Leader Node – Receives queries, creates execution plans.

  * Compute Nodes – Store data and perform actual computation.

* Data is columnar stored, which improves query speed and compression.

🧩 Key Components

* Tables – Store structured data.

* Sort Keys – Optimize query performance by physically sorting data.

* Distribution Keys – Decide how data is spread across nodes.

* Node Types – Includes Dense Storage (DS), Dense Compute (DC), and RA3 nodes (decoupled compute & storage).

🚀 Key Features

* Fully managed: No hardware or OS management.

* Columnar Storage: Ideal for analytics and compression.

* Scalable: Easily scale from gigabytes to petabytes.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

* **High Performance:**
  * Parallel query execution
  * Data caching
  * Query optimization
* Concurrency Scaling: Automatically adds temporary capacity for spikes.
* Redshift Spectrum: Query S3 data directly using SQL.
* Data Sharing: Share data across Redshift clusters in real-time.
* Backup & Restore: Automated snapshots stored in S3.
* Security: Supports encryption (SSL, KMS), IAM, VPC, audit logging.

🧠 Use Cases

* Business Intelligence (BI) and analytics.
* Data Warehousing from multiple sources (e.g., RDS, S3, DynamoDB).
* Data Lake integration (via Redshift Spectrum).
* Reporting and dashboarding (Power BI, Tableau, QuickSight).
* Machine Learning integration (with Amazon SageMaker).

⚙️ Data Loading & Unloading

* Use `COPY` command to load data from:
  * Amazon S3, DynamoDB, EMR, Kinesis, etc.
* Use `UNLOAD` to export data to S3 in compressed formats.
* Supports structured (CSV, TSV, JSON) and semi-structured data (Parquet, ORC).

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

💵 Pricing Model

* Based on:
  * Node type and count (e.g., DC2, RA3)
  * Storage usage
  * On-demand or Reserved Instances
  * Redshift Serverless: Pay only for query runtime (no provisioning needed)

🛡️ Security & Compliance

* VPC support: For private networking.
* IAM roles: For access control.
* KMS integration: For encryption at rest.
* SSL: For encryption in transit.
* Audit logs: Through CloudTrail and system tables.

🆚 Comparison with Other Data Warehouses

Feature	Redshift	BigQuery	Snowflake
Storage	RA3 (Managed)	Serverless	Cloud-agnostic
Execution	MPP	Serverless	MPP
Scaling	Manual or Auto	Auto	Auto
Integration	AWS-native	GCP-native	Multi-cloud
Pricing	Instance-based + usage	Usage-based	Usage-based

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

📊 Tools & Integration

* Compatible with:

  * SQL clients

  * BI tools like Power BI, Tableau, QuickSight

  * ETL tools like AWS Glue, Talend, Informatica

  * Machine Learning via SageMaker or external engines

✅ Summary

* Amazon Redshift is ideal for businesses looking to analyze large volumes of data quickly and cost-effectively.

* Supports complex SQL queries, integrates with the AWS ecosystem, and offers high performance through parallelism and optimization.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

Amazon Redshift Interview Questions and Answers in bullet-point format, ideal for preparing for technical interviews in data engineering, cloud, and analytics roles:

🔶 1. What is Amazon Redshift?

- **Amazon Redshift** is a fully managed, petabyte-scale **data warehouse service** in the AWS cloud.
- It is used for **OLAP (Online Analytical Processing)** and supports standard **SQL queries**.
- Built on **PostgreSQL**, but optimized for large-scale read-heavy workloads.

🔶 2. What are the key components of Redshift architecture?

- **Leader Node**:
  - Handles query parsing, planning, and result aggregation.
  - Coordinates queries across compute nodes.
- **Compute Nodes**:
  - Execute queries and store data.
  - Can scale horizontally (in RA3 or Dense Compute clusters).
- **Node Slices**:
  - Compute nodes are divided into slices for parallel processing.
- **Clients**:
  - Connect via JDBC/ODBC/Redshift Query Editor.

🔶 3. What is a Distribution Style?

Used to define how data is distributed across nodes:

- **KEY** – Based on the value of one column (helps with joins).
- **EVEN** – Data is spread evenly across nodes.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

- **ALL** – Copies the entire table to each node (used for small dimension tables).

🔶 4. What is a Sort Key in Redshift?

- Determines how data is **physically sorted** on disk.
- Helps **optimize query performance** with range filtering, joining, and ordering.
- Types:
  - **Single/Compound Sort Key**
  - **Interleaved Sort Key** (good for queries on multiple columns)

🔶 5. What is VACUUM in Redshift?

- Reclaims space and **resorts data** after DELETE/Update operations.
- Types of VACUUM:
  - **FULL** – Re-sorts and reclaims space.
  - **DELETE ONLY** – Reclaims space without sorting.
  - **SORT ONLY** – Re-sorts rows only.
  - **REINDEX** – Rebuilds interleaved sort keys.

🔶 6. What is ANALYZE in Redshift?

- Updates table **statistics**, enabling the optimizer to choose efficient query plans.
- Should be run after INSERT, DELETE, UPDATE.

🔶 7. How does Redshift store data?

- Uses **columnar storage**, which improves compression and performance for analytical queries.
- Uses **Zone Maps** and **encoding** to reduce disk I/O.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔶 8. What is Redshift Spectrum?

- Allows querying **data in S3 directly** using Redshift SQL.
- Uses **external tables** defined via Glue Data Catalog or Athena.
- Ideal for a **data lake + data warehouse** hybrid architecture.

🔶 9. What are RA3 instances in Redshift?

- Decouples **compute and storage** using managed Redshift storage.
- Enables **scaling compute independently** from storage.
- Charges based on usage (Pay-as-you-go).

🔶 10. What is Concurrency Scaling in Redshift?

- Automatically adds **transient clusters** to handle increased workloads.
- Improves **performance during peak** hours without paying for idle capacity.

🔶 11. How do you load data into Redshift?

- Use `COPY` command to load data from:
  - S3
  - DynamoDB
  - EMR
  - Remote hosts via SSH
- Supports CSV, JSON, Parquet, ORC, Avro formats.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔶 12. How do you unload data from Redshift to S3?

- Use UNLOAD command:

```
UNLOAD ('SELECT * FROM tablename')
TO 's3://mybucket/output/'
CREDENTIALS 'aws_access_key_id=...;aws_secret_access_key=...'
DELIMITER ','
ADDQUOTES;
```

🔶 13. What are Redshift snapshots?

- **Automated and manual backups** of your cluster.
- Stored in S3.
- Can be used for disaster recovery and restoring clusters.

🔶 14. How does Redshift handle security?

- **Encryption at rest** via AWS KMS or HSM.
- **Encryption in transit** via SSL.
- **VPC** for network isolation.
- Integration with **IAM**, **KMS**, **Audit Logging**, and **CloudTrail**.

🔶 15. How is Redshift different from RDS?

Feature	Redshift	RDS
Use case	OLAP	OLTP
Storage	Columnar	Row-based
Scaling	Petabyte-scale	Limited to instance size
Performance	Optimized for analytics	Optimized for transactions

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔶 16. Redshift vs Snowflake vs BigQuery?

Feature	Redshift	Snowflake	BigQuery
Deployment	AWS only	Multi-cloud	GCP only
Storage	RA3 (Decoupled)	Fully decoupled	Fully serverless
Pricing	Instance-based	Usage-based	Usage-based
Performance Tuning	Manual (VACUUM, ANALYZE)	Auto	Auto
Ecosystem	AWS native	Multi-cloud	GCP native

🔶 17. Common Redshift Interview Scenarios

- How to optimize a slow query?
- Explain your Redshift table design strategy.
- How do you secure data in Redshift?
- Difference between interleaved and compound sort keys?
- When do you use Redshift Spectrum?

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

How to optimize a slow query in amazon redshift

Optimizing a slow query in Amazon Redshift requires a systematic approach — examining query structure, table design, data distribution, and cluster performance. Here's a detailed step-by-step guide to help you:

✅ How to Optimize a Slow Query in Amazon Redshift

🔹 **1. Use `EXPLAIN` to Analyze the Query Plan**

* Run:

EXPLAIN SELECT * FROM your_query;

* Look for:

  * Sequential scans instead of index-like scans

  * Data redistribution (DS_DIST) between nodes

  * Nested loops on large tables

* Tip: Use `EXPLAIN ANALYZE` (in newer versions) to view actual runtime.

🔹 **2. Check Table Distribution Style**

* Improper distribution leads to data shuffling between nodes.

* Use:

  * KEY distribution for large table joins on a common column.

  * ALL for small dimension tables.

  * EVEN as default when no good distribution key exists.

-- Check current dist style

SELECT "table", diststyle, distkey FROM SVV_TABLE_INFO WHERE "table" = 'your_table';

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

- 3. **Use Sort Keys Appropriately**

* Sort keys speed up filtering, joining, and ordering.

* Ideal if the query uses:

  * Range filters (e.g., `WHERE date BETWEEN ...`)

  * Joins or `ORDER BY` on sorted columns.

* Use Interleaved Sort Key if queries filter on multiple non-prefix columns.

- **4. Avoid SELECT ***

* Only select required columns.

* Reduces I/O and memory usage, especially on wide tables.

SELECT col1, col2 FROM your_table WHERE ...

- **5. Use Compression (Encoding)**

* Apply proper compression (encoding) to reduce disk I/O.

* Check encoding with:

SELECT * FROM SVV_COLUMNS WHERE table_name = 'your_table';

* Use `ANALYZE COMPRESSION` to recommend better encodings.

- **6. Run ANALYZE and VACUUM**

* Keeps statistics and data blocks optimized.

ANALYZE your_table;

VACUUM your_table;

* Use `VACUUM FULL` or `VACUUM SORT ONLY` if the table has heavy updates/deletes.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔹 7. Rewrite the Query (Logical Optimization)

* Replace subqueries with joins or CTEs.

* Break complex queries into intermediate steps using `WITH` clauses.

* Use UNION ALL instead of UNION if duplicates aren't a concern.

🔹 8. Check for Missing Stats or Skewed Data

* Check:

  * `SVV_TABLE_INFO` → Look at `unsorted` %, `stats_off`, and `skew_sortkey`.

* Consider repartitioning or redefining sort/dist keys if skew is high.

🔹 9. Use Result Caching (Redshift Serverless)

* For Redshift Serverless, repeated identical queries may benefit from result caching.

* Ensure the same session and parameters are used.

🔹 10. Monitor with Performance Tools

* Use Amazon Redshift Console → Query Monitoring tab.

* Use system views:

```
SELECT * FROM STL_WLM_QUERY WHERE service_class > 5;

SELECT * FROM SVL_QLOG ORDER BY starttime DESC LIMIT 10;
```

* Look for high CPU, block time, or memory usage.

🧪 Pro Tips

* Materialize complex subqueries using `CREATE TEMP TABLE` for reuse.

* Use `DISTKEY` and `SORTKEY` columns in your WHERE clauses.

* Consider using Redshift Spectrum or external tables for rarely accessed or large external data.

* Enable Concurrency Scaling if query queues are the bottleneck.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

Explain your Redshift table design strategy.

When designing tables in Amazon Redshift, the goal is to maximize query performance, minimize I/O, and ensure scalability as data grows.

Here's a structured approach:

- 1. Understand the Workload

* Identify whether the table is used for:
  * OLAP (reporting, analytics)
  * Data warehouse staging
  * Real-time dashboarding
* Understand query patterns: joins, filters, aggregations, update frequency.

- 2. Choose the Right Distribution Style

To minimize data movement (shuffling) during joins:

* KEY Distribution
  * Use when the table is frequently joined on a specific column (e.g., `customer_id`).
  * Ensures matching keys are stored on the same node.

* ALL Distribution
  * Use for small dimension tables (lookup tables).
  * Replicates the entire table on all nodes, minimizing join cost.
* EVEN Distribution
  * Default style when there’s no suitable distribution key.
  * Spreads data uniformly but may cause broadcast joins.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

✅ Tip: Always consider distribution compatibility between fact and dimension tables.

🔹 3. Define Appropriate Sort Keys

To optimize query filtering, joining, and ordering:

* Compound Sort Key
  * Good when queries filter or join on leading columns.
  * Optimal for range queries (e.g., dates).

* Interleaved Sort Key
  * Useful when filtering on multiple, non-leading columns.
  * Slightly higher overhead during data load and VACUUM.

✅ Example: Use `sortkey(created_date)` for reporting on daily loads.

🔹 4. Use Compression (Encoding)

* Apply column compression to save storage and reduce disk I/O.
* Run:

  ANALYZE COMPRESSION tablename;

  * Use `ENCODE` options during table creation.
* Redshift applies default encodings, but manual tuning improves performance.

🔹 5. Choose Appropriate Data Types

* Use smallest suitable data types:
  * `SMALLINT` instead of `INT`, `VARCHAR(50)` instead of `VARCHAR(256)`.
* Avoid unnecessary `VARCHAR(MAX)` — affects memory and performance.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

- 6. Partition with Date Columns (Optional)

* For large fact tables, sort by date/time for partition elimination.

* Facilitates time-based purging and archiving.

- 7. Denormalize When Needed

* Redshift is columnar and favors star or flattened schemas.

* Avoid excessive normalization that causes multiple joins.

* Use wide tables with repeated dimensions for better performance.

- 8. Control Table Size and Updates

* Minimize `UPDATE` and `DELETE`; Redshift is optimized for `INSERT`-only workloads.

* Use staging tables for data transformation.

* Periodically VACUUM and ANALYZE:

```
VACUUM FULL tablename;
ANALYZE tablename;
```


- 9. Leverage Table Types

* Temporary Tables – For intermediate processing in ETL.

* UNLOGGED Tables – Faster loads; use when durability isn't critical.

* External Tables – Use Redshift Spectrum to query data in S3.

🧠 Example Table Design for a Sales Fact Table

```
CREATE TABLE sales_fact (
  sale_id BIGINT,
  customer_id INT,
```

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com


```
  product_id INT,
  sale_date DATE,
  amount DECIMAL(10,2)
)
DISTKEY (customer_id)
SORTKEY (sale_date)
ENCODE az64;
```

✅ Summary

Design Factor	Strategy
Distribution Style	KEY for joins, ALL for dimensions
Sort Key	Compound for date, Interleaved for multiple filters
Compression	Use `ANALYZE COMPRESSION`
Data Types	Use minimal and appropriate types
Table Structure	Denormalize when needed
Maintenance	Schedule ANALYZE, VACUUM
Storage Optimization	Use columnar benefits with proper encoding

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

How do you secure data in Amazon Redshift?

Amazon Redshift offers multiple layers of security to protect your data at rest, in transit, and during access. Below is a comprehensive breakdown:

- 1. Encryption at Rest

* Redshift supports AES-256 encryption for data at rest.

* Encryption applies to:

  * Disk storage

  * Backups

  * Snapshots

  * Redshift-managed storage (RA3)

* You can choose:

  * AWS Key Management Service (KMS) – AWS-managed or customer-managed keys.

  * Hardware Security Modules (HSM) – For advanced compliance requirements.

✅ Example:

Enable encryption when creating the cluster using KMS key.

- 2. Encryption in Transit

* SSL (Secure Sockets Layer) is used to encrypt data in transit between:

  * SQL clients ↔ Redshift cluster

  * Redshift ↔ S3 or other services

* You can enforce SSL by setting:

  * `require_ssl = true` in connection strings.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

✅ Example:

jdbc:redshift://cluster-endpoint:5439/dev?ssl=true

- 3. VPC and Network Security

* Deploy Redshift clusters inside an Amazon VPC for network isolation.

* Control access using:

  * Security Groups (acts like a firewall)

  * NACLs (Network ACLs) – Layered access control

* Use VPC Peering or PrivateLink for secure connections with other AWS services.

- 4. Identity and Access Management (IAM)

* Use IAM roles to:

  * Grant Redshift access to S3, Glue, etc.

  * Control what users and applications can access Redshift resources.

* Apply fine-grained access using IAM policies.

✅ Example:

json

```
{
  "Effect": "Allow",
  "Action": "redshift:GetClusterCredentials",
  "Resource": "*"
}
```

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔹 5. Database-Level User Access

* Use Redshift database users and groups for internal access control.

* Apply GRANT/REVOKE statements to control:

  * Table-level access

  * Column-level access

  * Schema or function access

✅ Example:

GRANT SELECT ON orders TO analyst_user;

REVOKE INSERT ON orders FROM analyst_user;

🔹 6. Row-Level and Column-Level Security

* Use view-based security or column masking logic.

* For sensitive data, restrict columns via views:

CREATE VIEW customer_masked AS

SELECT customer_id, email, NULL AS ssn FROM customer;

🔹 7. Audit Logging and Monitoring

* Enable Redshift Audit Logging to track:

  * Connection attempts

  * User activity

  * DDL and DML operations

* Logs are stored in S3 buckets and can be analyzed using CloudWatch, Athena, or Redshift itself.

* Use CloudTrail to monitor API-level access.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔹 8. Redshift Role-Based Access Control (RBAC) – Newer Feature

* Manage privileges using roles, similar to enterprise DBMSs.

* Easier to manage group-level privileges across schemas and objects.

✅ Example:

CREATE ROLE read_only;

GRANT USAGE ON SCHEMA sales TO read_only;

GRANT SELECT ON ALL TABLES IN SCHEMA sales TO read_only;

🔹 9. Multi-Factor Authentication (MFA)

* Enforce MFA via AWS IAM federation (e.g., SAML/SSO, Okta).

* Combine with GetClusterCredentials for secure passwordless authentication.

🔹 10. Best Practices

* Rotate credentials and access keys regularly.

* Use temporary IAM credentials instead of hardcoded passwords.

* Restrict network access by IP using security groups.

* Periodically review access privileges and audit logs.

✅ Summary: Redshift Security Layers

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

Layer	Mechanism
Data at Rest	KMS, HSM, AES-256 encryption
Data in Transit	SSL
Network Security	VPC, Security Groups, PrivateLink
Identity Access	IAM roles & policies
Database Access	GRANT/REVOKE, Roles, Views
Monitoring	Audit Logs, CloudTrail
MFA & Federation	IAM + SAML + Okta (optional)

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

Difference between interleaved and compound sort keys in Amazon Redshift

Feature	Compound Sort Key	Interleaved Sort Key
Definition	Sorts data based on **left-to-right column order**	Sorts data by giving **equal weight to all columns**
Best For	Queries that filter on the **first (leading) column(s)**	Queries that filter on **multiple columns independently**
Sort Priority	High priority to **first column**, lower to next	All columns are given **equal priority**
Performance	Very fast if queries use **prefix columns**	Better if queries filter on **non-prefix columns**
Storage Overhead	Lower overhead	Higher storage and maintenance overhead
VACUUM Cost	Lower; only affects the prefix order	Higher; **requires full re-indexing** of all columns
Query Optimization	Optimized for range scans and time-series data	Optimized for **ad-hoc filtering** across many fields
Use Case Examples	- `WHERE sale_date BETWEEN ...` - `ORDER BY region, product`	- `WHERE region = 'US'` - `WHERE product = 'Laptop'`
Default Sort Type	**Yes** (default when not specified)	No (must be explicitly declared)
Syntax	`SORTKEY(col1, col2)`	`INTERLEAVED SORTKEY(col1, col2)`

When to Use Which?

✅ Use Compound Sort Key When:

- You filter most queries on a leading column (e.g., date ranges).
- Your workload is time-series, log, or ETL-based.
- You want lower storage overhead and fast bulk loads.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

✅ Use Interleaved Sort Key When:

- You filter queries on different columns independently.
- You have ad-hoc, exploratory, or unpredictable query patterns.
- You need balanced performance across multiple filter conditions.

🧪 Example:

Compound:

```
CREATE TABLE sales (
 sale_id INT,
 customer_id INT,
 sale_date DATE
)
SORTKEY (sale_date, customer_id);
```


- Best for: WHERE sale_date BETWEEN ...

Interleaved:

```
CREATE TABLE sales (
 sale_id INT,
 customer_id INT,
 region TEXT
)
INTERLEAVED SORTKEY (customer_id, region);
```

- Best for: WHERE customer_id = 123 OR WHERE region = 'West'
-

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

When do you use Amazon Redshift Spectrum ?

Amazon Redshift Spectrum is used when you want to query data directly from Amazon S3 without loading it into Redshift tables.

✅ Use Cases for Redshift Spectrum

- 1. Querying External Data in S3

* When your data lives in Amazon S3 in open file formats like Parquet, ORC, CSV, or JSON.

* Ideal for data lake architectures or hybrid DW+DL systems.

✅ Example:

```
SELECT * FROM spectrum_schema.external_table
WHERE event_date BETWEEN '2024-01-01' AND '2024-01-31';
```

- 2. Handling Massive Data Volumes

* When datasets are too large to load into Redshift (terabytes/petabytes).

* Spectrum scales automatically without provisioning clusters.

- 3. Cost Optimization

* Reduces storage costs: Keep cold or infrequently accessed data in S3.

* You only pay per query for the amount of data scanned.

- 4. Data Lake + Warehouse Integration

* Use Redshift for frequently accessed data, and Spectrum for rarely accessed historical data in S3.

* Enables seamless JOINs between Redshift and S3 datasets.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

🔹 5. Schema-on-Read Access

* Allows querying data without defining strict schemas upfront.

* Useful when working with semi-structured or evolving data formats.

🔹 6. Multiple Query Engines Sharing Same Data

* Share data across Redshift, Athena, and EMR using AWS Glue Data Catalog.

* Encourages single source of truth for raw data.

🔹 7. Ad-hoc Data Exploration

* When data scientists or analysts want to quickly query external files without ingesting.

🛠️ How It Works

1. External tables are defined using:

   * `CREATE EXTERNAL SCHEMA` and `CREATE EXTERNAL TABLE`

2. Uses AWS Glue Data Catalog for metadata

3. Queries are executed on data in S3 without importing into Redshift

🚫 When NOT to Use Redshift Spectrum

* For frequent, repetitive queries — better to load into Redshift.

* When data latency is critical — Spectrum has slight delay compared to local tables.

* For heavy write/update operations — Spectrum supports only read-only access.

Praveen Madupu
Mb: +91 98661 30093
Sr SQL Server DBA
praveensqldba12@gmail.com

✅ Summary

Use Redshift Spectrum When...

You have large datasets in S3 and want to query directly.

You want to extend Redshift with a data lake approach.

You need to reduce costs for infrequent, cold data.

You want to join Redshift and S3 data in a single query.