AWS Assignment Solutions

1. Explain the difference between AWS Regions, Availability Zones, and Edge Locations. Why is this important for data analysis and latency-sensitive applications"

AWS Regions are geographic areas that contain multiple isolated locations known as Availability Zones (AZs). Each AZ consists of one or more data centers with redundant power and networking. Edge Locations are endpoints for AWS services like CloudFront, used for content delivery to reduce latency.

Importance:
- Regions let users deploy resources close to their users to reduce latency.
- AZs help build high availability and fault-tolerant applications.
- Edge Locations reduce content delivery times for latency-sensitive apps.

2. Using the AWS CLI, list all available AWS regions. Share the command used and the output.

Command:

```
aws ec2 describe-regions --query "Regions[*].RegionName" --output table
```

Sample Output:

```
+---------------------+
|  DescribeRegions  |
+---------------------+
| ap-south-1       |
| eu-west-1        |
| us-east-1        |
| us-west-2        |
+---------------------+
```

3. Create a new IAM user with least privilege access to Amazon S3. Share your attached policies (JSON or screenshot).

Sample Policy JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::your-bucket-name",
        "arn:aws:s3:::your-bucket-name/*"
      ]
    }
  ]
}
```

4. Compare different Amazon S3 storage (Standard, Intelligent-Tiering, Glacier). When should each be used in data analytics workflows.

Comparison:
- Standard: Used for frequently accessed data. High durability and availability.
- Intelligent-Tiering: Automatically moves data between access tiers based on usage.
- Glacier: Designed for long-term archival. Low cost, but retrieval takes time.

Analytics Use Cases:
- Standard: For real-time or high-frequency data processing.
- Intelligent-Tiering: When access patterns are unpredictable.
- Glacier: Store raw or processed historical data for compliance or long-term storage.

5. Create an S3 bucket and upload a sample dataset (CSV or JSON). Enable versioning and show at least two versions of one file.

Commands:

```
aws s3 mb s3://my-analytics-bucket
aws s3api put-bucket-versioning --bucket my-analytics-bucket --versioning-configuration
Status=Enabled
aws s3 cp sample-v1.csv s3://my-analytics-bucket/
aws s3 cp sample-v2.csv s3://my-analytics-bucket/sample-v1.csv
```

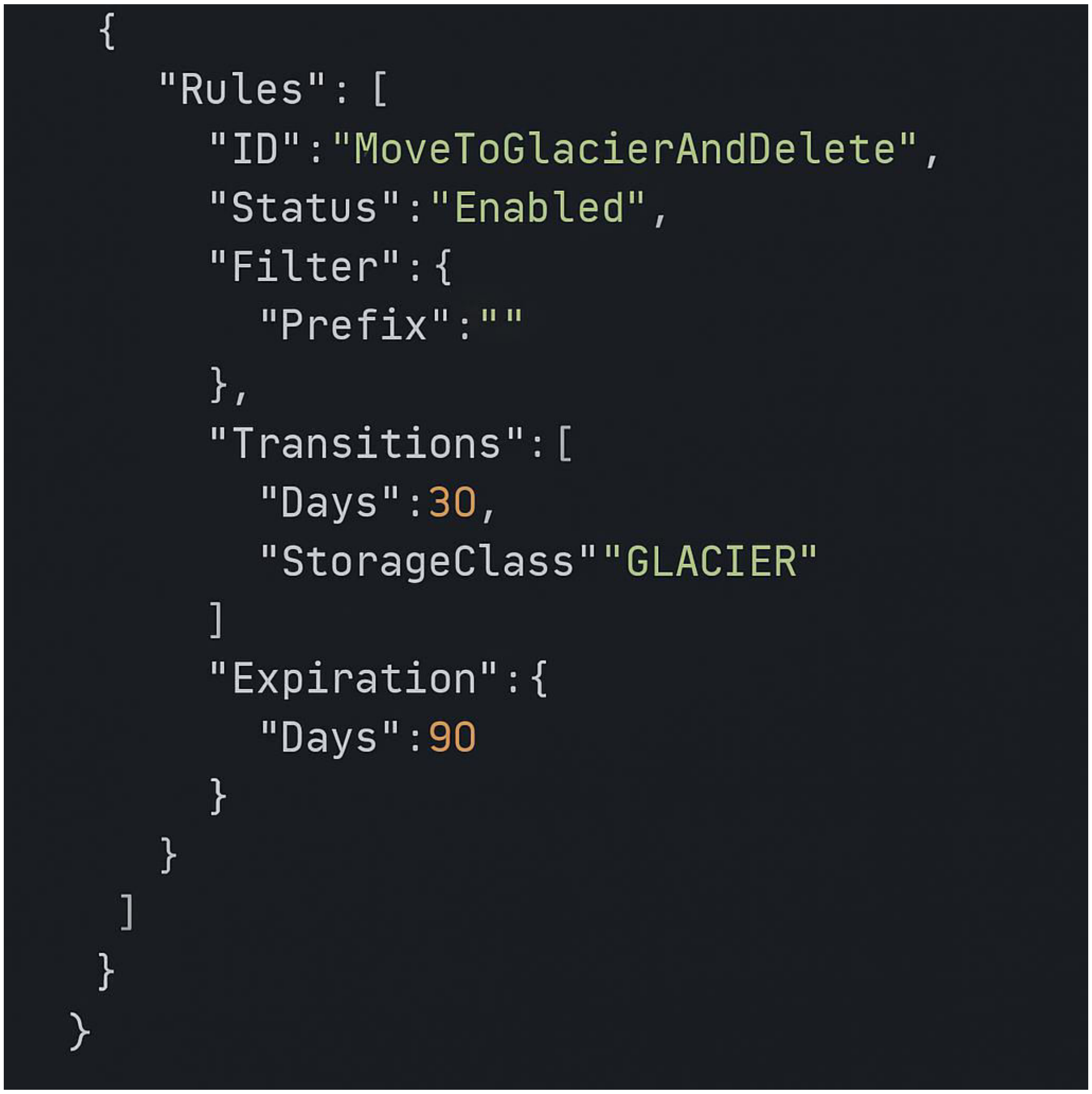
List Versions:

```
aws s3api list-object-versions --bucket my-analytics-bucket --prefix sample-v1.csv
```

6. Write and apply a lifecycle policy to move files to Glacier after 30 days and delete them after 90. Share the policy JSON or Screenshot&

Sample Lifecycle Policy JSON:

```
{
 "Rules": [
  {
   "ID": "MoveToGlacierAndDelete",
   "Prefix": "",
   "Status": "Enabled",
   "Transitions": [
    {
     "Days": 30,
     "StorageClass": "GLACIER"
    }
   ],
   "Expiration": {
    "Days": 90
   }
  }
 ]
}
```



```
  {
    "Rules": [
      "ID":"MoveToGlacierAndDelete",
      "Status":"Enabled",
      "Filter":{
        "Prefix":""
      },
      "Transitions":[
        "Days":30,
        "StorageClass""GLACIER"
      ]
      "Expiration":{
        "Days":90
      }
    }
   ]
  }
 }
```

7. Compare RDS, DynamoDB, and Redshift for use in different stages of a data pipeline. Give one use case for each.

- **RDS**: Managed relational DB for transactional workloads. Use Case: Store user profiles in a web app.
- **DynamoDB**: NoSQL DB for key-value and document data. Use Case: Real-time user session tracking.
- **Redshift**: Columnar DB for analytics. Use Case: Run queries on large-scale sales data for BI dashboards.

8. Create a DynamoDB table and insert 3 records manually. Then write a Lambda function that adds records when triggered by S3 uploads.

Create Table:

```
aws dynamodb create-table --table-name AnalyticsData --attribute-definitions
AttributeName=ID,AttributeType=S --key-schema AttributeName=ID,KeyType=HASH --
billing-mode PAY_PER_REQUEST
```

Insert Records:

```
aws dynamodb put-item --table-name AnalyticsData --item
'{"ID":{"S":"1"},"Name":{"S":"Record1"}}'
...
```

Lambda Function (Python):

```
import json
import boto3
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('AnalyticsData')
    for record in event['Records']:
        file_name = record['s3']['object']['key']
        table.put_item(Item={'ID': file_name, 'Name': 'Uploaded'})
```

9. What is serverless computing? Discuss pros and cons of using AWS Lambda for data pipelines.

Serverless computing lets developers run code without managing servers. AWS Lambda is a key example.

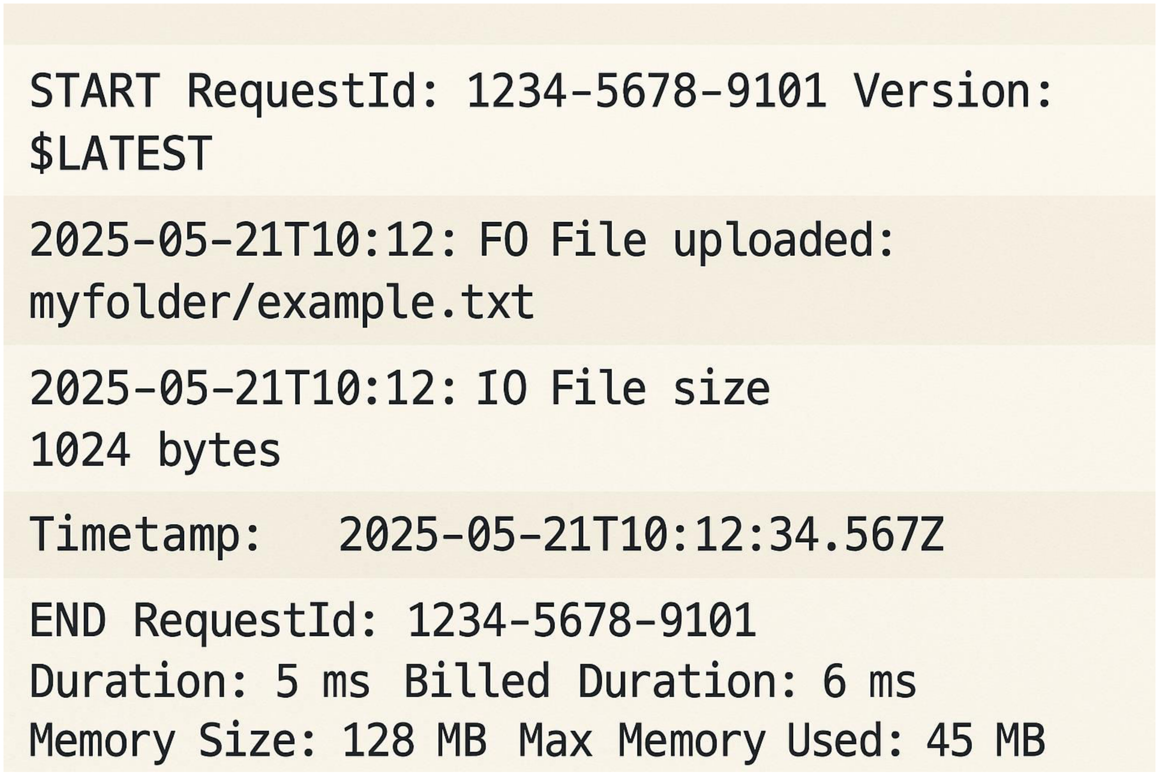
Pros:
- No infrastructure management
- Scales automatically
- Pay-per-use

Cons:

- Cold start latency
- Limited execution time (15 min max)
- Debugging and monitoring can be harder

10. Create a Lambda function triggered by S3 uploads that logs file name, size, and timestamp to Cloudwatch. Share code and a log screenshot

```
import json

import boto3

import logging

from datetime import datetime


logger = logging.getLogger()

logger.setLevel(logging.INFO)


def lambda_handler(event, context):

    # The event is triggered by S3 PUT (upload)

    for record in event['Records']:

        s3 = record['s3']

        bucket = s3['bucket']['name']

        key = s3['object']['key']

        size = s3['object']['size']

        event_time = record['eventTime']


        logger.info(f"File uploaded: {key}")

        logger.info(f"File size: {size} bytes")

        logger.info(f"Timestamp: {event_time}")


    return {

        'statusCode': 200,

        'body': json.dumps('Log recorded successfully!')
}
```



```
START RequestId: 1234-5678-9101 Version:
$LATEST
2025-05-21T10:12: FO File uploaded:
myfolder/example.txt
2025-05-21T10:12: IO File size
1024 bytes
Timetamp:   2025-05-21T10:12:34.567Z
END RequestId: 1234-5678-9101
Duration: 5 ms  Billed Duration: 6 ms
Memory Size: 128 MB  Max Memory Used: 45 MB
```

11. CREATE Use AWS Glue to crawl your S3 dataset, create a Data Catalog table, and run a Glue job to convert CSV data to parquet. Share job code and output location

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.context import SparkContext


args = getResolvedOptions(sys.argv, ['JOB_NAME'])


sc = SparkContext()
glueContext = GlueContext(sc)
```

```
spark = glueContext.spark_session

job = Job(glueContext)

job.init(args['JOB_NAME'], args)


# Read from Glue catalog

input_database = "demo_database"

input_table = "sales_data"


datasource0 = glueContext.create_dynamic_frame.from_catalog(

   database=input_database,

   table_name=input_table,

   transformation_ctx="datasource0"

)


# Convert to DataFrame (optional)

df = datasource0.toDF()


# Write as parquet to output S3 bucket folder

output_path = "s3://my-glue-demo-bucket/output-parquet/"


df.write.mode("overwrite").parquet(output_path)


job.commit()
```

Output Location

Converted Parquet files will be here:

3://my-glue-demo-bucket/output-parquet/

12. Explain the difference between Kinesis Data Streams, Kinesis Firehose, and Kinesis Data Analytics. Provide a real-world example of how each would be used.

- **Kinesis Data Streams**: Real-time streaming data ingestion and custom processing.
  - *Example*: Real-time log collection and custom transformation before storing in S3.
- **Kinesis Firehose**: Fully managed service that loads streaming data directly to destinations like S3, Redshift.
  - *Example*: Send IoT sensor data directly to S3 for storage.
- **Kinesis Data Analytics**: Real-time SQL queries on streaming data.
  - *Example*: Run real-time analytics on clickstream data from a website.

13. What is columnar storage and how does it benefit Redshift performance for analytics workloads?

Columnar storage stores data by columns instead of rows. In analytics, queries often access only a few columns.

Benefits in Redshift:
- Reduces I/O by reading only required columns
- Improves compression efficiency
- Faster query performance for large datasets

14. Load a CSV file from S3 into Redshift using the COPY command. Share table schema, command used, and sample output from a query.

Table Schema:
```
CREATE TABLE sales (
   sale_id INT,
   product_name VARCHAR(100),
   quantity INT,
   sale_date DATE
);
```

COPY Command:
```
COPY sales FROM 's3://my-bucket/sales.csv'
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRedshiftRole'
CSV IGNOREHEADER 1;
```

Sample Query:
```
SELECT product_name, SUM(quantity) FROM sales GROUP BY product_name;
```

15. What is the role of the AWS Glue Data Catalog in Athena? How does schema-on-read work?

Glue Data Catalog: Acts as a central metadata repository. Athena uses it to interpret structure of data stored in S3.

Schema-on-read: You define structure at query time, not when data is stored. This allows flexibility with unstructured or semi-structured data.

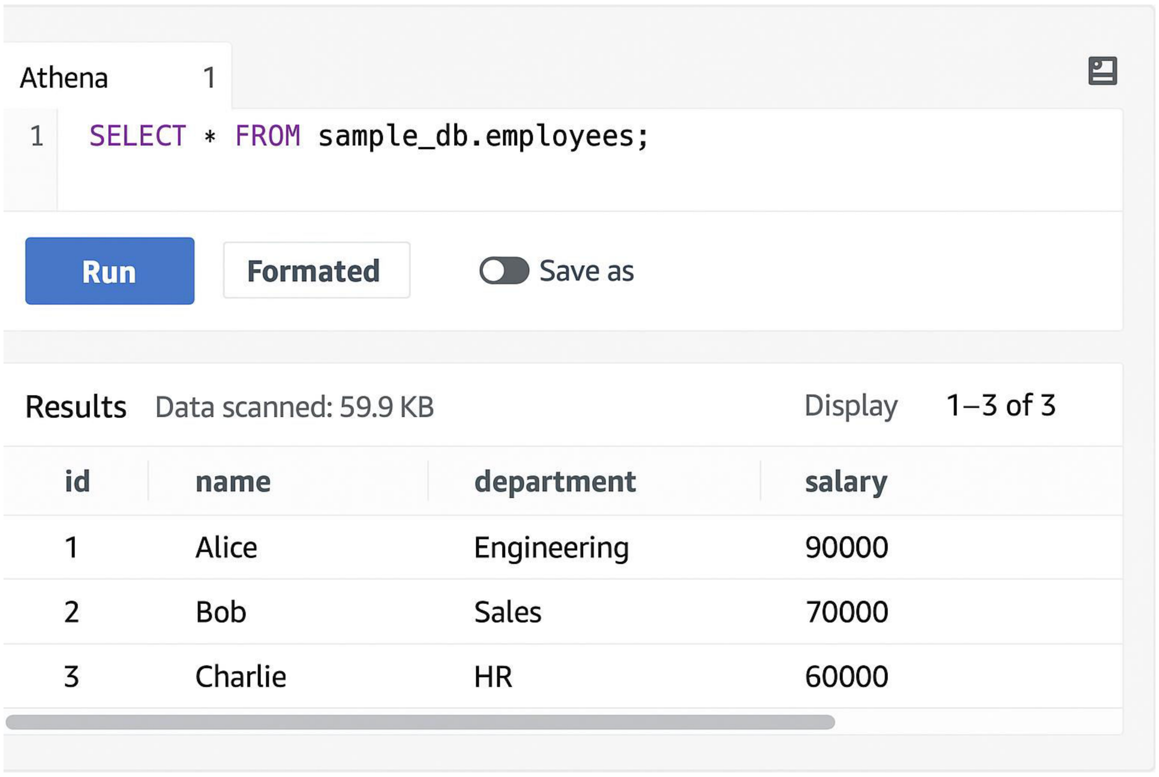
16. Create an Athena table from S3 data using Glue Catalog. Run a query and share the SQL + result screenshot.

- Crawl S3 data with AWS Glue to create the catalog table.
- Use Athena to query using the table.

```
SELECT * FROM demo_database.sales_data
LIMIT 5;
```

Result Output

order_id	product	quantity	price	date
1001	Widget	10	20.5	2025-05-01
1002	Gadget	5	15.0	2025-05-02
1003	Widget	7	20.5	2025-05-03



```
SELECT * FROM sample_db.employees;
```

Results Data scanned: 59.9 KB Display 1–3 of 3

id	name	department	salary
1	Alice	Engineering	90000
2	Bob	Sales	70000
3	Charlie	HR	60000

17. Describe how Amazon Quicksight supports business intelligence in a serverless data architecture. What are SPICE and embedded dashboards ?

How Amazon QuickSight Supports Business Intelligence in a Serverless Architecture

Serverless Deployment

No infrastructure provisioning or management.

Auto-scales based on user activity and data volume.

Integrates with serverless data sources like AWS Athena, Amazon S3, AWS Glue, and AWS Redshift Serverless.

Seamless Integration with AWS Data Stack

Natively connects to AWS services like Athena, S3, RDS, Redshift, DynamoDB, and more.

Supports federated queries and real-time dashboards using services like Amazon Athena.

Data Refresh & Access Control

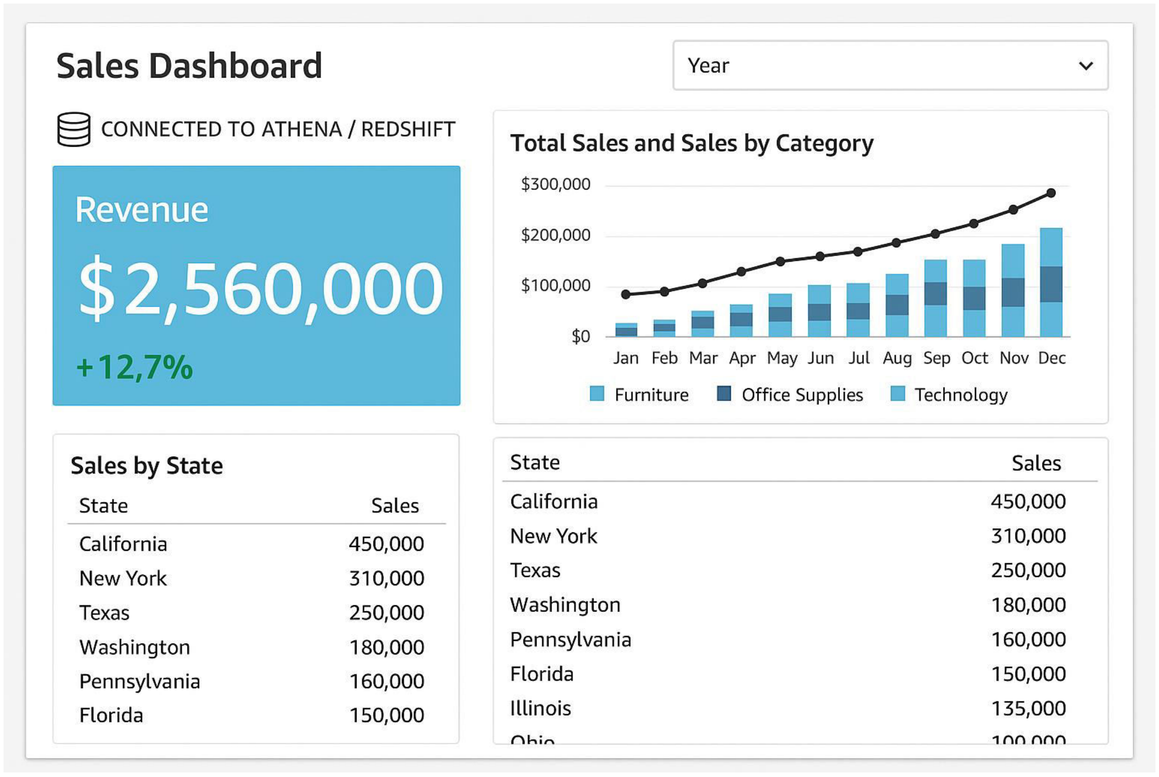
Supports scheduled and on-demand data refreshes.

Uses IAM for fine-grained access control.

Embedded Dashboards

- QuickSight dashboards can be embedded into apps, portals, or intranet sites using the QuickSight Embedded SDK or APIs.
- This provides BI access to non-QuickSight users, maintaining a secure and consistent experience.
- Embedding supports interactivity: filters, drill-downs, and data exploration.

18-_ Connect Quicksight to Athena or Redshift and build a dashboard with at least one calculated field and one filter. Share a screenshot of your final dashboard?


Sales Dashboard
Year
CONNECTED TO ATHENA / REDSHIFT
Revenue
$2,560,000
+12,7%
Total Sales and Sales by Category
$300,000
$200,000
$100,000
$0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Furniture
Office Supplies
Technology

Sales by State

State	Sales
California	450,000
New York	310,000
Texas	250,000
Washington	180,000
Pennsylvania	160,000
Florida	150,000

State	Sales
California	450,000
New York	310,000
Texas	250,000
Washington	180,000
Pennsylvania	160,000
Florida	150,000
Illinois	135,000
Ohio	100,000

19.Explain how AWS CloudWatch and CloudTrail differ. IN a data analytics pipeline, what role does each play in monitoring, auditing, and troubleshooting?

AWS CloudWatch and CloudTrail serve different but complementary purposes in monitoring, auditing, and troubleshooting a data analytics pipeline.

1. AWS CloudWatch

Purpose: Monitoring and operational insights
Key Capabilities:

- Collects and tracks metrics (CPU, memory, disk, etc.)
- Logs application and infrastructure events in near real-time
- Generates alarms and dashboards for visualization
- Enables automated responses to system events (e.g., Lambda invocations)

In a Data Analytics Pipeline:

- Monitoring: Tracks resource usage and performance of analytics components (e.g., Glue jobs, Redshift queries, Lambda functions).
- Troubleshooting: Provides detailed logs to diagnose failures (e.g., failed ETL jobs).
- Alerts: Sends notifications if metrics cross defined thresholds (e.g., job duration too long).

---

2. AWS CloudTrail

Purpose: Governance, compliance, risk auditing
Key Capabilities:

- Records all API calls made within your AWS account
- Tracks user activity (who did what, when, and from where)
- Stores logs in S3 for long-term analysis
- Integrates with CloudWatch for alerting on specific events

In a Data Analytics Pipeline:

- Auditing: Maintains a history of API actions (e.g., who started or stopped a Redshift cluster).

- Security Monitoring: Detects unauthorized or suspicious activity (e.g., unusual access to S3 buckets).
- Troubleshooting: Investigates changes that might have caused pipeline failure (e.g., altered IAM permissions or deleted resources).

20. Describe a complete end-to-end data analytics pipeline using AWS services. Include services for data ingestion, storage, transformation, querying, and visualization. (Example: S3 → Lambda → Glue → Quicksight) Explain why you would choose each service for the stage it's used in

Here's a complete end-to-end data analytics pipeline using AWS services, covering all stages: ingestion, storage, transformation, querying, and visualization.

✅ 1. Data Ingestion

🔧 Service: AWS Lambda or Amazon Kinesis Data Streams

- Use Lambda when:
  - Ingesting small, event-driven data (e.g., API, IoT, or app logs).
  - Triggering downstream services automatically (e.g., on S3 upload).
- Use Kinesis for:
  - Real-time streaming data (e.g., user clicks, log streams).

✅ Why?
These services are highly scalable, serverless, and integrate well with other AWS services for automation and near real-time processing.

✅ 2. Data Storage

🔧 Service: Amazon S3

✅ Why?

- Scalable, durable (99.999999999%), and cost-effective object storage.

- Ideal for raw and processed data (e.g., CSV, JSON, Parquet).
- Supports data lake architectures and integrates with Athena, Glue, Redshift Spectrum.

✅ 3. Data Transformation / ETL

🔧 Service: AWS Glue

✅ Why?

- Serverless ETL tool that can crawl, catalog, and transform data.
- Automatically generates schema and handles jobs in Python/Scala.
- Handles batch processing of large-scale datasets with minimal configuration.

✅ 4. Data Catalog

🔧 Service: AWS Glue Data Catalog

✅ Why?

- Central metadata repository.
- Enables schema discovery and governance.
- Used by Athena, Redshift Spectrum, and Quicksight to query S3 data.

✅ 5. Data Querying

🔧 Service: Amazon Athena or Amazon Redshift

- Use Athena for:
    - Ad-hoc queries on data stored in S3.
    - Serverless querying using SQL.
- Use Redshift for:
    - High-performance queries on structured data.
    - Complex joins and aggregation at scale (data warehouse use cases).

✅ Why?
Athena is ideal for flexible, serverless querying. Redshift offers powerful performance and scalability for enterprise BI workloads.

✅ 6. Data Visualization

🔧 Service: Amazon QuickSight

✅ Why?

- Serverless BI tool that connects to Athena, Redshift, or S3.
- Allows creation of interactive dashboards and visualizations.
- Supports ML Insights, filters, and calculated fields.

✅ Optional Add-ons

- AWS CloudWatch: Monitor metrics, set alarms for Glue/Athena/Redshift.
- AWS CloudTrail: Audit user access and changes.
- IAM: Secure access control for users and services.