**¿Cómo iniciar?**

**R. Instalación y prueba de las siguientes plataformas para Prácticas.**

|  |  |
| --- | --- |
| Plataforma | Descripción |
| 1. AWS Free Tier | Proporciona acceso gratuito a muchos servicios durante 12 meses. |
| 1. Google Colab | Para escribir documentación y realizar prácticas en Python. |
| 1. Visual Studio Code + AWS CLI | Configuración de AWS CLI y SDK de AWS para Python. |
| 1. GitHub | Documentación y colaboración en proyectos. |
| 1. MySQL | Sistema de gestión de bases de datos relacional de código abierto, ideal para almacenar, consultar y gestionar grandes volúmenes de datos. |

Instalación de plataformas:

1. ***AWS Free Tier:***

**1.1 Crea una cuenta en AWS (si no tienes una):**

* Ir a [aws.amazon.com](https://aws.amazon.com).
* Hacer clic en **Crear una cuenta de AWS**.
* Completar la información solicitada (correo, contraseña, detalles de pago).
* **Nota:** Se requiere tarjeta de crédito para verificar tu identidad, pero no se te cobrará mientras uses los servicios del Free Tier.

**1.2. Accede a la Consola de AWS:**

* Iniciar sesión en [AWS Management Console](https://aws.amazon.com/console/).

**1.3. Identifica los servicios Free Tier:**

* Visitar [AWS Free Tier](https://aws.amazon.com/free) para ver la lista completa de servicios gratuitos.
* Algunos ejemplos:
  + **EC2:** 750 horas/mes de instancias t2. micro.
  + **S3:** 5 GB de almacenamiento estándar.
  + **RDS:** 750 horas/mes de base de datos db.t2. micro.

**1.4. Lanza un servicio gratuito:**

* Desde la consola, seleccionar el servicio que deseas (como EC2 o S3).
* Asegurarse de elegir opciones **Free Tier Eligible**.

**1.5. Monitorear el uso:**

* Usar **AWS Billing Dashboard** para verificar que no excedas los límites gratuitos.
* Configurar alertas de presupuesto.

1. ***Acceder a Google Colab:***

**2.1** Comenzar con:

* Ir a https://colab.research.google.com/.
* Iniciar sesión con la cuenta de **Google** (Gmail).

**2.2. Crear un Nuevo Proyecto (Notebook):**

* Hacer clic en **"Nuevo cuaderno"** o selecciona uno existente desde **"Archivos recientes"**.
* Se abrirá un entorno de Python similar a Jupyter Notebook.

**2.3. Guardar los Proyectos:**

* Google Colab guarda automáticamente tus cuadernos en Google **Drive**.
* Se puede guardar en local usando **Archivo → Descargar como. ipynb o .py**.

**2.4. Subir Archivos desde tu PC:**

* Usar el botón **"Subir"** en la sección de archivos para cargar scripts o datasets.

**2.5. Instalar Librerías (si es necesario):**

pip install nombre librería

Ejemplo:

pip install numpy

**2.6. Usar GPU o TPU Gratis (Opcional):**

* Ir a **Entorno de ejecución → Cambiar tipo de entorno de ejecución**.
* Seleccionar **GPU** o **TPU**.

1. ***Visual Studio Code + AWS CLI***

**Paso 3.1: Descargar VS Code**

* Ir a <https://code.visualstudio.com/>.
* Hacer clic en **Download for Windows/Mac/Linux** (dependiendo de tu sistema operativo).

**Paso 3.2: Instalar VS Code**

* Ejecutar el instalador descargado.
* Marcar las opciones:
  + **Agregar a PATH (recomendado)**.
  + **Crear ícono en el escritorio (opcional)**.
* Finaliza la instalación.

**Paso 3.3: Verifica la Instalación**

* Abrir VS Code y verificar que se ejecute correctamente.

**3.4 Instalar AWS CLI:**

**Paso 1: Descargar AWS CLI**

* Windows: https://awscli.amazonaws.com/AWSCLIV2.msi.
* Mac:

brew install awscli

* Linux:

sudo apt install awscli

**Paso 3.5: Instalar AWS CLI**

* Ejecutar el instalador y seguir los pasos.

**Paso 3.6: Verificar la Instalación**

aws --version

Debe mostrar:

aws-cli/2.x.x Python/3.x.x

**3.7 Configurar AWS CLI:**

aws configure

Se pedirá:

* **AWS Access Key ID**
* **AWS Secret Access Key**
* **Region (ej. us-east-1)**
* **Formato de salida (json, text, table)**

**3.8. Extensión de AWS para VS Code (Opcional):**

* En VS Code, ir a **Extensiones (Ctrl+Shift+X)**.
* Buscar e instalar **AWS Toolkit**.
* Se permitirá administrar servicios de AWS directamente desde VS Code.

1. ***GitHub***

**4.1. Crear una Cuenta en GitHub (si no tiene una):**

* Ir a <https://github.com/>.
* Hacer clic en **Sign Up**.
* Crear una cuenta con tu correo electrónico y contraseña.

**4.2. Instalar Git (Para Usar GitHub en Local):**

**Paso 4.2.1: Descargar Git**

* Ve a <https://git-scm.com/downloads>.
* Descarga la versión para tu sistema operativo (Windows, Mac o Linux).

**Paso 4.2.2: Instalar Git**

* Ejecuta el instalador:
  + **Marca "Add to PATH"** durante la instalación.
  + Acepta las configuraciones por defecto (puedes dejar todo como está).

**Paso 4.2.3: Verificar la Instalación**

git --version

Debe mostrar:

git version 2.x.x

**4.3. Configurar Git en Local:**

git config --global user.name "Su Nombre"

git config --global user. email "suemail@gmail.com"

Esto vincula sus commits con tu nombre y correo.

**4.4. Conectar Git con GitHub:**

**Paso 4.4.1: Generar una Clave SSH (Recomendado):**

ssh-keygen -t rsa -b 4096 -C "suemail@gmail.com"

Presionar **Enter** para guardar la clave.

**Paso 4.4.2: Agregar la Clave a GitHub:**

* Copiar la clave pública:

cat ~/.ssh/id\_rsa.pub

* Ir a **GitHub → Configuración → SSH y GPG Keys → New SSH Key**.
* Pegar la clave y guarda.

**4.5. Clonar un Repositorio:**

git clone git@github.com:usuario/repositorio.git

**4.6. Crear y Subir un Proyecto Nuevo:**

git init

git add

git commit -m "Primer commit"

git branch -M main

git remote add origin git@github.com: usuario/repositorio.git

git push -u origin main

1. ***MySQL***

**Pasos para instalar MySQL en Windows:**

1. **Descargar el instalador**
   * Visitar la página oficial de MySQL: <https://dev.mysql.com/downloads/installer/>.
   * Descargar el instalador adecuado para tu sistema (generalmente el "MySQL Installer for Windows").
2. **Ejecutar el instalador**
   * Hacer doble clic en el archivo descargado para iniciar el instalador.
3. **Seleccionar la configuración**
   * Eligir el tipo de instalación:
     + **Developer Default**: Incluye herramientas de desarrollo como Workbench y Shell.
     + **Custom**: Permite personalizar la instalación.
   * Para la mayoría de los casos, seleccionar **Developer Default**.
4. **Descargar los componentes necesarios**
   * El instalador descargará los componentes seleccionados. Asegúrate de estar conectado a Internet.
5. **Configurar MySQL Server**
   * Definir el tipo de configuración:
     + **Standalone MySQL Server** o como parte de un clúster.
   * Seleccionar el puerto (por defecto, 3306) y confirma.
6. **Establecer contraseña de root**
   * Configurar una contraseña para el usuario "root".
   * (Opcional) Crear usuarios adicionales si se desea.
7. **Finalizar la instalación**
   * Completar los pasos de configuración y verifica que el servicio de MySQL Server se ejecute correctamente.
8. **Instalar MySQL Workbench (opcional)**
   * MySQL Workbench es una herramienta gráfica para administrar y trabajar con bases de datos MySQL. Se incluye en el instalador por defecto.
9. **Verificar la instalación**
   * Abrir la terminal o línea de comandos, escribe mysql -u root -p, e ingresa la contraseña configurada.
   * Si todo está correcto, tendrá acceso al entorno de MySQL.

Módulo 1**: *Introducción a AWS y la Computación en la Nube***

***Duración Total:*** *10 horas*

**Objetivo del Módulo:**

Entender los conceptos básicos de computación en la nube y conocer los principales servicios de AWS, así como realizar configuraciones iniciales de seguridad.

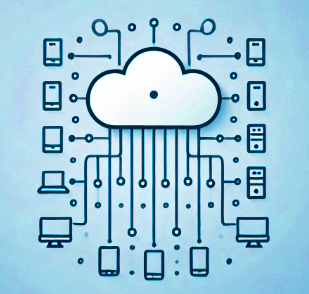
**Distribución de Horas:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sección | Duración (Horas) | Teoría (%) | Práctica (%) |
| 1. Conceptos básicos de la nube | 2 horas | 60% | 40% |
| 2. Introducción a AWS y sus servicios | 4 horas | 50% | 50% |
| 3. Configuración inicial y seguridad | 4 horas | 40% | 60% |
| Total | **10 horas** | **50%** | **50%** |

**Contenido Detallado:**

**1. Conceptos básicos de la nube**

* **Teoría:**
  + 1.1 ¿Qué es la nube?  
    *La nube es un conjunto de servidores remotos que almacenan y gestionan datos, aplicaciones y servicios a través de internet. Permite acceso desde cualquier lugar con conexión.*



[*https://es.educaplay.com/recursos-educativos/21885222-busqueda\_de\_palabras\_nube\_aws.html*](https://es.educaplay.com/recursos-educativos/21885222-busqueda_de_palabras_nube_aws.html)

* + 1.2 Tipos de nube: Pública, Privada, Híbrida.

Tipos de nube:

1. ***Nube Pública***: Servicios disponibles para el público (AWS, Google Cloud).
2. ***Nube Privada***: Infraestructura exclusiva para una organización.
3. ***Nube Híbrida***: Combina nubes públicas y privadas, permitiendo mover datos entre ambas.

*Algunos ejemplos*:

1. **Bucket:** Contenedor en Amazon S3 donde se almacenan objetos y datos.
2. **Fargate:** Servicio que permite ejecutar contenedores sin necesidad de gestionar servidores.
3. **Glue:** Servicio de integración de datos que facilita la preparación y carga de datos para análisis.
4. **IAM:** Servicio que permite gestionar el acceso y permisos de usuarios y recursos en AWS.
5. **QuickSight:** Servicio de inteligencia de negocios que permite crear visualizaciones y análisis de datos.
6. **RedShift:** Servicio de almacenamiento de datos que permite realizar análisis complejos sobre grandes volúmenes de datos.
7. **SNS:** Servicio de mensajería que permite enviar notificaciones a múltiples destinatarios.
8. **Terraform:** Herramienta de infraestructura como código que permite definir y provisionar recursos en AWS.
9. **Yard:** Espacio de trabajo en AWS donde se gestionan recursos y se visualizan arquitecturas.

<https://es.educaplay.com/recursos-educativos/21885308-adivina_la_nube_aws.html>

* + 1.3 Beneficios de la computación en la nube.

**Beneficios de la computación en la nube:**

1. **Escalabilidad:** Aumenta o reduce recursos según la demanda.
2. **Costo-Eficiencia:** Paga solo por lo que usas.
3. **Accesibilidad:** Accede desde cualquier dispositivo conectado.
4. **Seguridad:** Proveedores de nube implementan altos estándares de seguridad.

<https://es.educaplay.com/recursos-educativos/21885369-juego_de_parejas_beneficios_de_la_nube.html>

* + 1.4 Modelos de servicio: IaaS, PaaS, SaaS.

**Modelos de servicio:**

1. **IaaS (Infraestructura como Servicio):** Acceso a infraestructura (servidores, almacenamiento). Amazon EC2.
2. **PaaS (Plataforma como Servicio):** Plataforma para desarrollo de aplicaciones. Google App Engine.
3. **SaaS (Software como Servicio):** Aplicaciones completas en línea. Google Drive, Dropbox.

<https://es.educaplay.com/recursos-educativos/21885381-juego_de_completar_los_espacios_modelos_de_servicio.html>

[actividad1SopaDeLetras.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/actividad1SopaDeLetras.png)

* **Práctica:**
  + 1.5 Discusión grupal sobre ventajas y desventajas de la nube.

¿Qué modelo es más seguro?

**Nube Privada** es el modelo más seguro porque la infraestructura es exclusiva para una organización, lo que permite mayor control sobre la seguridad, el acceso y los datos.

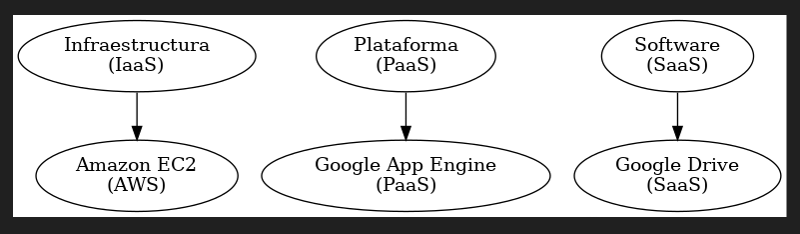
¿Cuál ofrece más flexibilidad?

**Nube Híbrida** ofrece la mayor flexibilidad. Permite mantener datos sensibles en una nube privada y usar la nube pública para tareas menos críticas o de alta demanda, optimizando costos y recursos.

1.6 ¿Qué modelo prefieren las empresas pequeñas y por qué?

**Razones:**

* 1. Depende del tipo de empresa, tipo de dato, tipo de información
  2. Depende del presupuesto
  3. Escalabilidad
* Es más económica (pagan solo por lo que usan).
* No requiere grandes inversiones en infraestructura.
* Fácil de escalar a medida que crecen.
  + Realización de un diagrama de los modelos de servicio.



**2. Introducción a AWS y sus servicios.**

**2.1 ¿Qué es AWS? Historia y crecimiento:**

* **Teoría:**
  + Fundación de AWS y su crecimiento.

AWS fue lanzado por Amazon en **2006**. Comenzó con **EC2 y S3**, ofreciendo almacenamiento y cómputo bajo demanda. Desde entonces, ha crecido hasta convertirse en el líder del mercado cloud, con más de **200** servicios y presencia global.

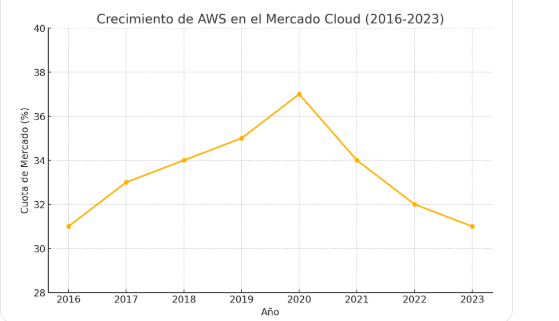
* + Importancia de AWS en el mercado global.

AWS tiene más del **30%** del mercado de la nube. Empresas como **Netflix, NASA y Airbnb** dependen de AWS por su escalabilidad, flexibilidad y seguridad, lo que impulsa la innovación y reduce costos.

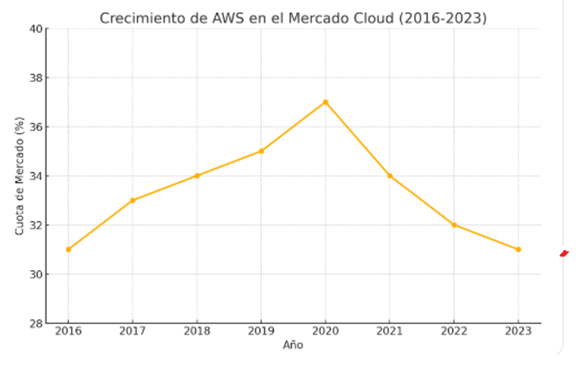
<https://es.educaplay.com/recursos-educativos/21888352-historia_de_aws.html>

[actividad2RuletaDePalabras.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/actividad2RuletaDePalabras.png)

* **Ejemplo Resuelto:**
  + Explicación de qué es AWS con un gráfico de crecimiento.



En 2020 hubo un repunte, ya que la pandemia provoco que muchos negocios/empresas tuvieran que actualizar su forma de trabajar



Considero que el crecimiento de AWS se podría mantener

**2. Principales servicios**

* **Teoría:**
  + Descripción breve de EC2, S3, RDS y Lambda.

Si el enfoque es en "automatización"

Utilizar "automatización" si quieres resaltar cómo AWS Lambda permite automatizar procesos o tareas sin intervención manual.

Es ideal para contextos en los que el objetivo principal es reducir trabajo repetitivo, como:

ETL automatizados: Procesar datos en tiempo real o por lotes.

Procesos de mantenimiento: Monitoreo y alertas automáticas.

Flujos de trabajo desencadenados por eventos: Como ejecutar una función cuando se sube un archivo a S3.

Ejemplo: AWS Lambda permite la automatización eficiente de tareas específicas desencadenadas por eventos en la nube, como la gestión de datos o la ejecución de procesos de backend.

Si el enfoque es en "serverless"

Utilizar "serverless" si quieres destacar que AWS Lambda es una arquitectura sin servidores, donde no necesitas gestionar infraestructura subyacente.

Este término es ideal para explicar ventajas relacionadas con la escalabilidad automática y el pago por uso, como:

Crear APIs rápidas y escalables.

Ejecutar funciones en respuesta a solicitudes de usuarios sin preocuparte por servidores.

Desarrollar aplicaciones modernas con baja carga de infraestructura.

Ejemplo : AWS Lambda es un servicio serverless que permite ejecutar código en la nube sin gestionar servidores, escalando automáticamente en función de la demanda.

<https://es.educaplay.com/recursos-educativos/21888697-palabras_aws.html>

[actividad3JuegoDeParejas.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/actividad3JuegoDeParejas.png)

* **Práctica:**
  + Acceso a documentación de AWS para cada servicio.

| **Servicio** | **Enlace a la Documentación Oficial** |
| --- | --- |

|  |  |
| --- | --- |
| **EC2** | [Amazon EC2 Documentation](https://docs.aws.amazon.com/ec2/) |

|  |  |
| --- | --- |
| **S3** | [Amazon S3 Documentation](https://docs.aws.amazon.com/s3/) |

|  |  |
| --- | --- |
| **RDS** | [Amazon RDS Documentation](https://docs.aws.amazon.com/rds/) |

|  |  |
| --- | --- |
| **Lambda** | [AWS Lambda Documentation](https://docs.aws.amazon.com/lambda/) |

* **Ejemplo Propuesto:**
  + Crear un bucket S3 básico.

**Pasos para Crear un Bucket S3 Básico:**

**1. Iniciar sesión en la consola de AWS**

* Ve a <https://aws.amazon.com/console/>.
* Ingresa con tus credenciales (usuario y contraseña).

**2. Acceder al Servicio S3**

* Desde el **panel principal (Dashboard)**, en la barra de búsqueda superior, escribe **S3** y selecciona **Simple Storage Service (S3)**.
* Haz clic en **Create Bucket** (Crear Bucket).

**3. Configurar el Bucket**

1. **Nombre del Bucket:**
   * Escribe un nombre único (globalmente). Ejemplo: mi-primer-bucket-123. ([mybucketacademiadataengineer](https://us-east-1.console.aws.amazon.com/s3/buckets/mybucketacademiadataengineer))
   * **Nota:** El nombre debe ser en minúsculas y sin espacios.
2. **Región:**
   * Selecciona la región donde deseas crear el bucket (ejemplo: us-east-1 o us-west-2).
3. **Configuración de Objetos Públicos:**
   * Deja la opción **"Block all public access"** marcada (recomendado para buckets privados).

**4. Opciones Adicionales (Opcional):**

* **Etiquetas (Tags):** Agrega etiquetas si deseas organizar mejor tus buckets.
* **Versioning:** Habilítalo si quieres guardar versiones antiguas de los archivos.
* **Encryption:** Puedes habilitar el cifrado para mayor seguridad.

**5. Crear el Bucket**

* Revisa la configuración.
* Haz clic en **Create Bucket**.

**6. Verificación**

* Regresa a la lista de buckets. Deberías ver tu nuevo bucket creado.

**Prueba (Subir un archivo):**

1. Haz clic en tu bucket recién creado.
2. Presiona el botón **Upload (Cargar)**.
3. Selecciona un archivo desde tu computadora y haz clic en **Upload**.

AWS garantiza un tiempo de respuesta máximo de 15 minutos en sus servicios

**3. Arquitectura general de AWS.**

* **Teoría (0.8 horas):**
  + Explicación de la arquitectura de regiones y zonas de disponibilidad.

AWS organiza su infraestructura global en **Regiones** y **Zonas de Disponibilidad (AZs)** para ofrecer alta disponibilidad, tolerancia a fallos y baja latencia.

**1. Regiones:**

**Definición:** Son ubicaciones físicas globales (ej. Norteamérica, Europa, Asia).

**Características:**

* + Cada región es independiente y aislada de otras.
  + Contiene múltiples Zonas de Disponibilidad.
  + Los usuarios eligen la región según sus necesidades (cercanía, cumplimiento legal).
  + Ejemplo: us-east-1 (Virginia del Norte), eu-west-1 (Irlanda).

**2. Zonas de Disponibilidad (AZs):**

**Definición:** Son centros de datos distribuidos dentro de una región.

**Características:**

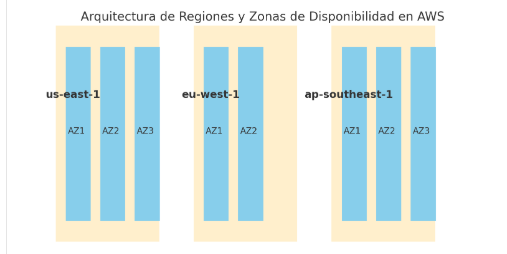
* + Cada región tiene al menos **dos o más AZs**.
  + Están separadas físicamente para evitar fallos simultáneos.
  + Se interconectan mediante redes de alta velocidad y baja latencia.
  + Las aplicaciones pueden replicarse en múltiples AZs para mayor disponibilidad.

**Beneficios de Esta Arquitectura:**

**Alta disponibilidad** y **tolerancia a fallos**.

**Escalabilidad** y **baja latencia** al estar cerca de los usuarios.

**Recuperación ante desastres** al replicar servicios en distintas AZs.



* **Práctica:**
  + Realización de un diagrama de arquitectura.

*Un buen punto de partida es representar una aplicación web simple que utilice múltiples servicios de AWS para garantizar alta disponibilidad, almacenamiento y procesamiento eficiente.*

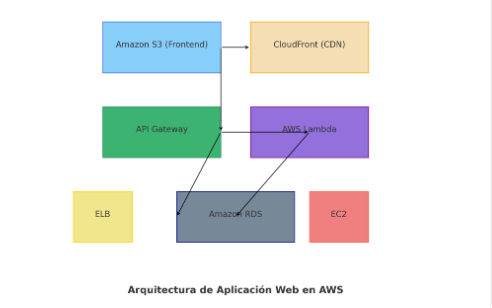
***Arquitectura: Aplicación Web Escalable con Alta Disponibilidad***

***Componentes Clave:***

1. ***Frontend:***
   * *React o Angular (puede ser una app web estática).*
   * *Se aloja en* ***Amazon S3*** *con habilitación de distribución global usando* ***CloudFront*** *(CDN).*
2. ***Backend:***
   * *API desarrollado en* ***AWS Lambda*** *para manejar solicitudes sin servidores.*
   * *API Gateway gestiona las peticiones HTTP hacia Lambda.*
3. ***Base de Datos:***
   * ***Amazon RDS (MySQL/PostgreSQL)*** *para datos transaccionales.*
   * ***Amazon DynamoDB*** *para almacenar datos de acceso rápido (NoSQL).*
4. ***Autenticación y Seguridad:***
   * ***Amazon Cognito*** *para la gestión de usuarios y autenticación.*
   * ***IAM*** *para la gestión de permisos.*
5. ***Alta Disponibilidad:***
   * *La arquitectura está replicada en múltiples Zonas de Disponibilidad (AZs).*
   * ***Elastic Load Balancer (ELB)*** *distribuye tráfico entre instancias de EC2.*

***Ejemplo Resuelto (Pasos):***

1. ***Crear un bucket S3*** *y habilitar la opción de hosting estático.*
2. ***Subir archivos HTML/JS/CSS*** *al bucket S3.*
3. ***Configurar CloudFront*** *para distribuir el contenido con baja latencia.*
4. ***Crear una función Lambda*** *conectada a API Gateway que responda a solicitudes HTTP.*
5. ***Desplegar una instancia RDS*** *para bases de datos.*
6. ***Configurar Auto Scaling*** *en EC2 y conectar con ELB para distribuir tráfico.*
7. ***Diagrama de aplicación Web en AWS:***



Bucket: [aplicacionwebescalableconaltadisponibilidad](https://us-east-1.console.aws.amazon.com/s3/buckets/aplicacionwebescalableconaltadisponibilidad?region=us-east-1&bucketType=general)

PROYECTO DE DATA ENGINEER

Monitoreo y Análisis de Sentimiento de Marca

Objetivo: Crear un sistema que monitoree las menciones de una marca específica en redes sociales y analice el sentimiento de las publicaciones para ayudar en estrategias de marketing y relaciones públicas.

Tecnologías a Utilizar:

* APIs de X/Meta: Para recolectar menciones y publicaciones relacionadas con la marca.
* AWS Lambda: Para la ejecución de funciones que procesen los datos recolectados y realicen análisis de sentimiento preliminar.
* AWS Glue: Para la integración y transformación de datos a un formato adecuado para análisis.
* Amazon Aurora: Para almacenar resultados del análisis y servir como base de datos para consultas y reportes.
* Amazon SageMaker: Para construir y entrenar modelos de machine learning.

DIAGRAMA

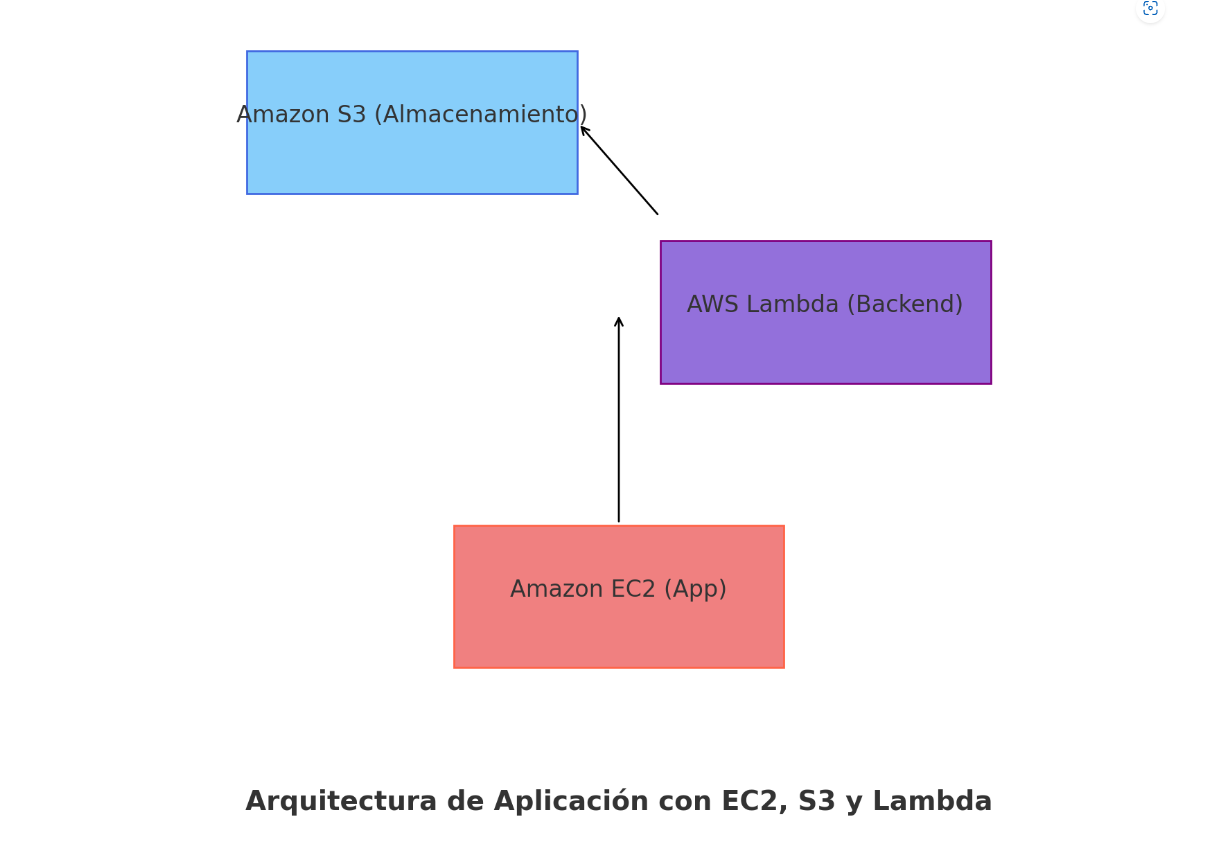
1. ***Frontend:***

* *HTML, CSS y JavaScript/Python*
* *Streamlit*

1. ***Backend:***
   * *APIs X/Meta (Buscar si son gratuitas)*
   * *AWS Lambda*
   * *FastAPI*
   * *Python*
2. ***Base de Datos:***
   * *MySQL*
3. ***Autenticación y Seguridad:***
   * *IAM*
   * *S3*
   * *EC2*
   * *Bucket*
4. ***Alta Disponibilidad:***
   * *AZs*

Bucket: [monitoreoanalisissentimientomarca](https://us-east-1.console.aws.amazon.com/s3/buckets/monitoreoanalisissentimientomarca?region=us-east-1&bucketType=general)

* **Ejemplo Resuelto:**
  + Crear un diagrama de una aplicación que usa EC2, S3 y Lambda.



Se interpreta de forma serverless el lambda

<https://es.educaplay.com/recursos-educativos/21888814-preguntas_de_refuerzo_de_aws.html>

**4. Práctica en consola de AWS (1 hora)**

* **Práctica:**
  + **Creación de una cuenta de AWS.**

Abrir un navegador Web

Ingresar a la página de AWS (<https://signin.aws.amazon.com/signup?request_type=register>)

Ingresar a la opción "cree una cuenta AWS"

Ingresar correo electrónico y nombre de la cuenta

Verificar la dirección de correo electrónico

Ingresar datos de tarjeta de crédito

Continuar instrucciones

{

Enlistar los pasos para llegar al objetivo:

* + Navegación por la consola.

Ordenar los pasos que a continuación se proporcionan:

* Navegar entre regiones:
* Usar la barra de búsqueda superior y escribe el nombre del servicio (ej. EC2, S3).
* Iniciar sesión:
* Arriba a la derecha, hacer clic en la lista desplegable de regiones.
* Hacer clic en tu nombre de usuario (arriba a la derecha).
* Elegir la región donde quieres operar.
* Gestionar recursos:
* Seleccionar un servicio:
* Ir a <https://aws.amazon.com/console/>.
* Hacer clic en **Sign in to the Console**.
* Ingresar tu correo y contraseña.
* Seleccionar **Sign Out**.
* Crear recursos:
* Dentro del servicio, buscar el botón **Create** o **Launch** (ej. "Launch Instance" en EC2).
* O seleccionar desde el panel **Services**.
* Seleccionar el recurso de la lista.
* Cerrar sesión.
* Usar las pestañas **Actions** o **Modify** para editar, iniciar o detener.

Espacio para la solución:

1. Iniciar sesión:

* Ir a <https://aws.amazon.com/console/>
* Hacer clic en **Sign in to the Console**.
* Ingresar tu correo y contraseña.

2. Navegar entre regiones:

* Arriba a la derecha, hacer clic en la lista desplegable de regiones.
* Elegir la región donde quieres operar.

3. Seleccionar un servicio:

* Usar la barra de búsqueda superior y escribe el nombre del servicio (ej. EC2, S3).
* O seleccionar desde el panel Services.

4. Crear recursos:

* Dentro del servicio, buscar el botón Create o Launch (ej. "Launch Instance" en EC2)

5. Gestionar recursos:

* Seleccionar el recurso de la lista.
* Usar las pestañas Actions o Modify para editar, iniciar o detener

6. Cerrar sesión:

* Hacer clic en tu nombre de usuario (arriba a la derecha).
* Seleccionar Sign Out.
  + Lanzamiento de una instancia EC2.
* **Ejemplo Resuelto:**
  + Lanzar y conectar una instancia EC2 desde la consola.

**1. Iniciar sesión en AWS Management Console**

* Abre tu navegador web y ve a [AWS Management Console](https://aws.amazon.com/console/).
* Ingresa con tus credenciales de acceso (usuario y contraseña de AWS).

**2. Acceder a EC2**

* En la barra de búsqueda de la consola de AWS, escribe "EC2" y selecciona **EC2** para ir a la consola de administración de instancias.

Haz clic en **"Launch Instance"** (Lanzar Instancia) en la parte superior derecha.

**Seleccionar una Amazon Machine Image (AMI)**

* Elige una **Amazon Machine Image (AMI)**, que es una plantilla preconfigurada de sistema operativo. Por ejemplo, selecciona **Amazon Linux 2 AMI** o **Ubuntu** (dependiendo de tu preferencia).
* Haz clic en **Select** (Seleccionar) al lado de la AMI que desees.

**Seleccionar un tipo de instancia**

* Elige el tipo de instancia según tus necesidades. Para pruebas simples, puedes seleccionar una **t3.micro** (que es elegible para el nivel gratuito de AWS).
* Haz clic en **Next: Configure Instance Details** (Siguiente: Configurar detalles de la instancia).

**Configurar los detalles de la instancia**

* Si no necesitas configuraciones personalizadas, puedes dejar los valores predeterminados.
* Opcionalmente, puedes cambiar el número de instancias, la red o la subred, si es necesario.
* Haz clic en **Next: Add Storage** (Siguiente: Añadir almacenamiento).

**Configurar el almacenamiento**

* Ajusta el tamaño del disco de almacenamiento si es necesario.
* Haz clic en **Next: Add Tags** (Siguiente: Añadir etiquetas).

**Añadir etiquetas (opcional)**

* Puedes añadir etiquetas para organizar y gestionar las instancias, por ejemplo, agregando una etiqueta con el nombre de la instancia.
* Haz clic en **Next: Configure Security Group** (Siguiente: Configurar grupo de seguridad).

**Configurar el grupo de seguridad**

* Aquí se definen las reglas de firewall para permitir el tráfico hacia y desde tu instancia EC2.
* Si deseas acceder a la instancia a través de SSH (por ejemplo, si usas Linux o Mac), agrega una regla para permitir el tráfico SSH en el puerto 22.
  + Selecciona **SSH** en el tipo y en el origen, elige **Anywhere** (0.0.0.0/0) si deseas acceso global o restringe la IP a una red específica para mayor seguridad.
* Haz clic en **Review and Launch** (Revisar y lanzar).

**10. Revisar y lanzar la instancia**

* Revisa todas las configuraciones y, si todo está bien, haz clic en **Launch** (Lanzar).

**11. Seleccionar o crear un par de llaves**

* Cuando se te pida elegir un par de llaves, selecciona uno ya existente o crea uno nuevo.
  + Si eliges crear uno nuevo, dale un nombre al par de llaves y haz clic en **Download Key Pair** (Descargar par de llaves). **Guarda este archivo.pem en un lugar seguro**, ya que lo necesitarás para conectarte a la instancia.
* Haz clic en **Launch Instances** (Lanzar instancias).

**12. Acceder a la instancia EC2**

* Una vez que la instancia esté en estado **running**, selecciona la instancia desde el panel de EC2.
* Copia la **IP pública** de la instancia.

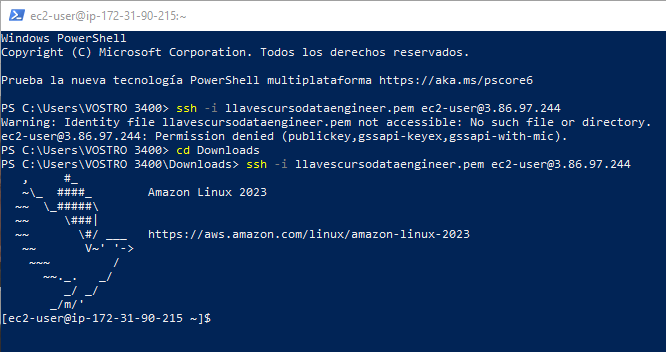
**13. Conectar a la instancia mediante SSH**

* Si usas Linux o Mac, abre una terminal. Si usas Windows, puedes usar herramientas como **PuTTY** o **Windows PowerShell**.
* **Linux/Mac**: En la terminal, usa el siguiente comando (reemplaza path\_to\_key.pem con la ubicación del archivo. pem que descargaste y your\_public\_ip con la IP pública de tu instancia EC2):

Comando para Linux: chmod 400 path\_to\_key.pem

Comando para Windows (Se debe estar posicionado en la carpeta donde se encuentra el archivo .pem): ssh -i path\_to\_key.pem ec2-user@your\_public\_ip

* **Windows (PuTTY)**:
  + Convierte el archivo. pem a. ppk usando **PuTTYgen**.
  + En PuTTY, en el campo **Host Name**, ingresa la IP pública de tu instancia.
  + En el menú de la izquierda, bajo **Connection > SSH > Auth**, selecciona tu archivo. ppk.
  + Haz clic en **Open** para conectar.



* **Ejemplo Propuesto:**
  + Configuración básica de seguridad en una instancia EC2.

..\Archivos complementarios\Archivo link 1.docx

**3. Configuración y seguridad básica en AWS.**

**1. Teoría:**

AWS ofrece una infraestructura segura y herramientas para proteger tus recursos en la nube. Las configuraciones de seguridad básica incluyen:

1. **IAM (Identity and Access Management)**
   * Crea usuarios, roles y grupos con permisos mínimos necesarios (Principio de mínimo privilegio).
   * Usa MFA (Autenticación Multifactorial) para mayor seguridad.
2. **Security Groups y NACLs (Network ACLs)**
   * **Security Groups**: Controlan el tráfico entrante y saliente a nivel de instancia (permiten/deniegan puertos).
   * **NACLs**: Reglas a nivel de subred, permiten o deniegan tráfico específico (se aplican en orden numérico).
3. **S3 (Simple Storage Service) - Seguridad de Buckets**
   * Restringe el acceso a los buckets usando políticas y bloquea el acceso público.
   * Usa el cifrado (AES-256 o KMS).
4. **Cifrado y Logs**
   * Habilita el cifrado de datos en reposo (EBS, RDS) y en tránsito (SSL/TLS).
   * Usa CloudTrail y CloudWatch para auditoría y monitoreo.

<https://es.educaplay.com/recursos-educativos/21896948-configuracion_y_seguridad_basica_en_aws.html>

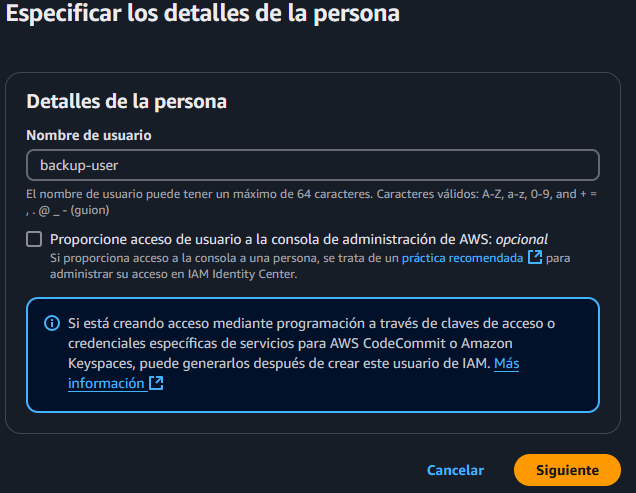
**2. Ejemplo Resuelto**

**🔑 Objetivo:**

Crear un usuario IAM con acceso limitado a S3 y habilitar MFA.

**Pasos:**

1. **Crear Usuario IAM**
   * Ve a IAM > Usuarios/Persona > Crear usuario.
   * Nombre: backup-user



Se deja de esta manera ya que no se hará uso de las demás configuraciones

* + Habilito acceso programático (API, CLI).

1. **Adjuntar Permiso (Política personalizada)**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "s3:ListBucket",

"Resource": ["arn:aws:s3:::nombre\_bucket"]

},

{

"Effect": "Allow",

"Action": "s3:GetObject",

"Resource": ["arn:aws:s3:::backup-bucket/\*"]

}

]

}

1. **Configurar MFA**
   * Ve a Credenciales de Seguridad > Asignar dispositivo MFA.
   * Usa una app (Google Authenticator).

**3. Ejemplo Propuesto**

**🔒 Objetivo:**

Configura un grupo de seguridad (Security Group) que permita tráfico HTTP (puerto 80) y SSH (puerto 22) solo desde tu IP.

**Pasos:**

1. Ve a EC2 > Grupos de Seguridad > Crear grupo.
2. Agrega las siguientes reglas:
   * **SSH (22)**: Tu IP (Ejemplo: 203.0.113.25/32).
   * **HTTP (80)**: Cualquier dirección (0.0.0.0/0).
3. Asigna este Security Group a una instancia EC2.

**Solución (CLI):**

aws ec2 create-security-group --group-name web-sg --description "Grupo para tráfico web y SSH"

aws ec2 authorize-security-group-ingress --group-name web-sg --protocol tcp --port 80 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-name web-sg --protocol tcp --port 22 --cidr 203.0.113.25/32

|  |  |
| --- | --- |
| ***Componente*** | ***Nombre, descripción*** |
| Cloud Watch (Amazon CloudFront) |  |
| Elastic Load Balancer (ELB). |  |
| Amazon RDS |  |
| Auto Scaling Groups |  |
| Acceso programático | AGREGAR LO QUE SE ENTENDIO DEL TEMA |
| AWS KMS (Key Management Service) |  |
| S3 Versioning |  |
| AWS CloudTrail |  |
| AWS Config |  |
| Security Groups |  |
| Network ACLs |  |
| Web Application Firewall (WAF) |  |

***Revisado 13/01/2025***

***¡¡¡ Buen trabajo !!!***

***Apuntes 14/01/2025***

***Tabla comparativa de Rol de IAM vs Acceso programático (CLI/SDK)***

|  |  |  |
| --- | --- | --- |
| **Aspecto** | **Rol de IAM** | **Acceso Programático (CLI/SDK)** |
| **Definición** | Un rol de IAM es una entidad que asume permisos temporales para realizar tareas específicas en AWS. | Permite a usuarios o aplicaciones acceder a AWS mediante claves de acceso desde CLI o SDKs. |
| **Uso Principal** | Para delegar permisos a servicios de AWS, aplicaciones dentro de AWS o cuentas externas. | Para que usuarios, scripts o aplicaciones externas interactúen con AWS desde fuera de AWS. |
| **Credenciales** | Credenciales temporales (administradas automáticamente por AWS). | Access Key ID y Secret Access Key (que deben administrarse manualmente). |
| **Seguridad** | Más seguro, ya que no requiere el manejo directo de claves. | Menos seguro si las claves no se manejan adecuadamente. |
| **Escenario Común** | - EC2 asume un rol para acceder a S3.   - Lambda usa un rol para invocar otro servicio. | - Configurar scripts para automatizar tareas.   - Conectar aplicaciones locales a AWS. |
| **Duración de Permisos** | Temporal (dependiendo de la sesión). | Permanente, hasta que las claves sean deshabilitadas o eliminadas. |
| **Gestión de Permisos** | Asociado a políticas específicas del rol. | Asociado a políticas vinculadas al usuario al que pertenece la clave. |
| **Configuración** | - Crear un rol en la consola de IAM.   - Asociar políticas necesarias.   - Configurar confianza con otras entidades. | - Crear un usuario con acceso programático.   - Descargar las claves y configurarlas en CLI o SDK. |
| **Escalabilidad** | Ideal para aplicaciones que necesitan permisos temporales y dinámicos. | Adecuado para accesos específicos y tareas manuales o automatizadas externas. |
| **Rotación de Credenciales** | No requiere intervención manual, AWS las maneja automáticamente. | Requiere rotación manual de las claves para cumplir con mejores prácticas. |

***Como crear un usuario IAM***

* **Crear un usuario IAM:**
  + Ve a la consola de IAM.
  + Crea un nuevo usuario con acceso programático.
  + Asigna permisos mediante políticas (ej. "AmazonS3FullAccess").
* **Configurar las credenciales:**
  + Descarga el archivo de credenciales (.csv) generado al crear el usuario.
  + Instala AWS CLI en tu máquina.

[usuarioBackup-user2.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/usuarioBackup-user2.png)

## Módulo 2: ***Programación en Python para la Ingeniería de Datos***

***Duración Total:*** *20 horas*

**Objetivo del Módulo:**

Capacitar a los participantes en el uso de Python como herramienta fundamental para la ingeniería de datos, proporcionando conocimientos sólidos sobre sus fundamentos, bibliotecas especializadas como NumPy y Pandas, y SQL para el manejo, análisis y transformación de grandes volúmenes de datos. Al finalizar, los participantes serán capaces de desarrollar soluciones prácticas que optimicen procesos de manipulación y análisis de datos en entornos reales.

**Distribución de Horas:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tema | Duración (hrs) | Teoría (%) | Práctica (%) | Descripción breve |
| 1. Fundamentos de Python | 6 | 40% | 60% | Variables, estructuras de control, funciones, etc. |
| 1. Bibliotecas: NumPy | 5 | 30% | 70% | Operaciones con arrays, manipulación de matrices. |
| 1. Bibliotecas: Pandas | 5 | 30% | 70% | Manipulación y análisis de datos tabulares. |
| 1. SQL básico | 2 | 20% | 80% | Consultas básicas: SELECT, WHERE, JOIN. |
| 1. SQL avanzado | 2 | 20% | 80% | Subconsultas, funciones de agregación, optimización. |

**Distribución general: Teoría total:** **30%** (6hrs) **70%** (14hrs) **Práctica total:**

***1. Fundamentos de Python.***

1.1Python es un lenguaje de programación interpretado (lenguaje que no necesita ser compilado antes de su ejecución, por lo tanto, es más rápida su ejecución), de alto nivel (lenguaje más cercano al del ser humano) y multipropósito (lenguaje que se puede usar para realizar varias tareas, como el desarrollo web, el procesamiento de datos, el machine learning, entre otras).

Es conocido por su sintaxis sencilla (orden gramatical, formato en el que se escriben las instrucciones) y legibilidad, lo que lo hace ideal para principiantes y profesionales.

* 1. Por qué usarlo.
* **Versatilidad:** Python se puede utilizar para desarrollo web, análisis de datos, inteligencia artificial, automatización y más.
* **Comunidad:** Es uno de los lenguajes más populares, con una comunidad grande y activa que genera librerías y framework constantemente.
* **Productividad:** Permite escribir menos código para realizar tareas complejas.
* **Portabilidad:** Python es multiplataforma, es decir, puede ejecutarse en Windows, macOS y Linux sin cambios significativos en el código.

<https://es.educaplay.com/recursos-educativos/21897109-caracteristicas_python.html>

[actividad1CaracteristicasPuthon.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/14012025/actividad1CaracteristicasPuthon.png)

* 1. Instalación y configuración.

**Pasos:**

1. **Descargar Python:**
   * Visitar [python.org](https://www.python.org/downloads/) y descarga la última versión estable.
2. **Seleccionar Plataforma:**
   * Elegir el instalador según tu sistema operativo (Windows, macOS, Linux).
3. **Ejecutar Instalador (Windows/macOS):**
   * Abrir el archivo descargado y marca la opción "Add Python to PATH".
4. **Seguir el Asistente:**
   * Hacer clic en "Next" y finaliza la instalación.
5. **Verificar Instalación:**
   * Abrir la terminal (cmd, PowerShell o terminal de Linux) y ejecuta:

python --version

1. **Instalar PIP (si no viene instalado):**
   * Ejecuta:

python -m ensurepip --upgrade

1. **Configurar Entorno Virtual (Opcional):**
   * Crea un entorno virtual:

python -m venv mi\_entorno

1. **Activar Entorno Virtual:**
2. source mi\_entorno/bin/activate # macOS/Linux

mi\_entorno\Scripts\activate # Windows

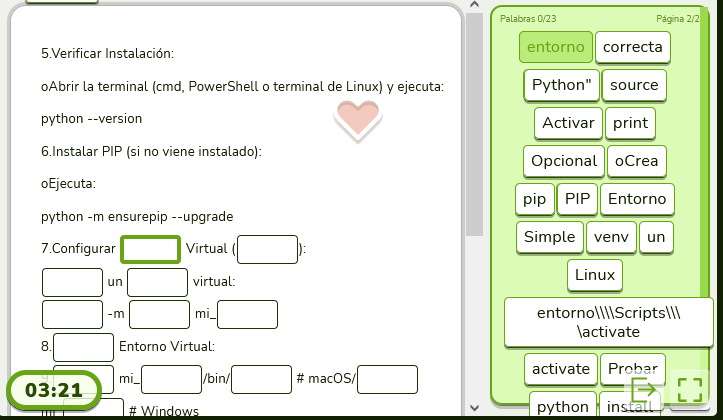
1. **Actualizar PIP:**

pip install --upgrade pip

1. **Probar con un Script Simple:**

print ("Instalación correcta de Python")

<https://es.educaplay.com/recursos-educativos/21897408-pasos_para_instalar_python.html>



Se quedo trabada la pantalla y ya no dejo continuar la actividad

**1.4. Variables y Tipos de Datos**

* Las variables almacenan datos en la memoria y se pueden modificar durante la ejecución del programa. Python soporta diferentes tipos de datos como enteros, flotantes, cadenas y booleanos.
* Definición de variables.

son **nombres simbólicos** que se utilizan para **almacenar valores**. Actúan como **contenedores** que guardan datos que pueden cambiar durante la ejecución del programa. Se definen asignándoles un valor con el operador = y no requieren declarar un tipo específico.

* Tipos de datos básicos: int, float, str, bool.
* Conversión entre tipos de datos.

La conversión de variables en Python se hace usando **funciones de conversión de tipos (casting)**. Estas funciones permiten transformar datos de un tipo a otro.

**Ejemplos de conversión:**

* **A entero:** int()
* **A flotante:** float ()
* **A cadena (string):** str ()
* **A booleano:** bool ()

<https://es.educaplay.com/recursos-educativos/21902816-variables_python_y_algunos_ejemplos.html>

[actividad3DictadoVariablesPythonEjemplos.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/14012025/actividad3DictadoVariablesPythonEjemplos.png)

**Ejemplos:**

x = "10" # String

y = int(x) # Convierte a entero (10)

a = 5.7 # Float

b = str(a) # Convierte a string ("5.7")

c = 0 # Entero

d = bool(c) # Convierte a booleano (False)

<https://es.educaplay.com/recursos-educativos/21899798-repaso_aws_y_python.html>

[actividad4RepasoAWSPython.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/14012025/actividad4RepasoAWSPython.png)

* **Estructuras de Control:**

Son instrucciones que **dirigen el flujo de ejecución** del programa, permitiendo tomar decisiones o repetir bloques de código.

**Ejemplo (if-else):**

edad = 18

if edad >= 18:

print ("Eres mayor de edad")

else:

print ("Eres menor de edad")

* **Funciones:**

Son **bloques de código reutilizables** que realizan una tarea específica. Se definen una vez y se pueden ejecutar cuando se necesiten.

**Ejemplo:**

def saludar(nombre):

return f"Hola, {nombre}!"

***print(saludar("Hanna")) # Hola, Hanna!***

***Para instalar entorno virtual y se ejecuten los programas en python***

pip install pipenv

mkdir carpeta-ejemplos/

pipenvshell

pipenv install numpy

***2. Bibliotecas: NumPy***

**NumPy** es una biblioteca de Python para **cálculos numéricos rápidos**. Permite trabajar con **arrays multidimensionales** y proporciona funciones para realizar **operaciones matemáticas** de manera eficiente. Es más rápido que las listas nativas de Python.

**Características clave:**

* **ndarray**: Estructura de datos principal (array n-dimensional).
* **Vectorización**: Operaciones sin bucles, más rápidas y eficientes.
* **Funciones matemáticas y estadísticas** integradas.

**Práctica:**

**Crear un array:**

import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr)

**Operaciones básicas:**

a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

suma = a + b # [5 7 9]

producto = a \* 2 # [2 4 6]

**Manipulación de matrices:**

import numpy as np

matriz = np. array ([[1, 2], [3, 4]])

print(matriz)

transpuesta = matriz.T # [[1, 3], [2, 4]]

print(transpuesta)

<https://es.educaplay.com/recursos-educativos/21899915-los_egos_de_las_tecnologias.html>

[actividad5EgosTecnologias.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/14012025/actividad5EgosTecnologias.png)

***3. Biblioteca Pandas***

Pandas es una biblioteca de Python utilizada para la manipulación y análisis de datos. Facilita el trabajo con estructuras de datos tabulares (similares a hojas de cálculo) y es fundamental para proyectos de ciencia de datos, análisis financiero, y procesamiento de grandes volúmenes de información.

**Instalación de Pandas**

pip install pandas

**Importar Pandas**

import pandas as pd

**2. Estructuras de Datos en Pandas**

**2.1 Series**

Una Serie es una estructura unidimensional (similar a una lista o array).

**Ejemplo:**

import pandas as pd

data = [10, 20, 30, 40]

serie = pd.Series(data, index=['A', 'B', 'C', 'D'])

print(serie)

**Resultado:**

A 10

B 20

C 30

D 40

**2.2 DataFrames**

Un DataFrame es una estructura bidimensional (similar a una tabla de Excel).

**Ejemplo:**

import pandas as pd

data = {

'Nombre': ['Ana', 'Juan', 'Luis'],

'Edad': [23, 30, 45],

'Ciudad': ['Madrid', 'Barcelona', 'Valencia']

}

df = pd.DataFrame(data)

print(df)

**Resultado:**

Nombre Edad Ciudad

0 Ana 23 Madrid

1 Juan 30 Barcelona

2 Luis 45 Valencia

**3. Manipulación de Datos**

**3.1 Carga de Datos**

import pandas as pd

df = pd. read\_csv('datos.csv')

df = pd. read\_excel('datos.xlsx')

df = pd. read\_json('datos.json')

* 1. **Exploración de Datos**

print(df.head()) # Primeras 5 filas

print(df.info()) # Información general

print(df.describe()) # Estadísticas descriptivas

* 1. **Filtrado de Datos**

filtro = df[df['Edad'] > 30]

print(filtro)

<https://es.educaplay.com/recursos-educativos/21911156-conceptos_python.html>

[actividad6ConceptosPython.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/14012025/actividad6ConceptosPython.png)

* 1. **Modificación de Columnas**

df['Salario'] = [3000, 4000, 5000]

df['Edad'] = df['Edad'] + 1

**4. Análisis de Datos**

* 1. **Agrupación de Datos**

agrupado = df. groupby('Ciudad')['Salario'].mean()

print(agrupado)

* 1. **Ordenar Datos**

ordenado = df.sort\_values(by='Edad', ascending=False)

print(ordenado)

* 1. **Valores Nulos**

df.fillna(0, inplace=True)

**5. Ejercicios Resueltos**

**Ejercicio 1: Crear un DataFrame con datos de empleados y calcular la media de sus salarios.**

import pandas as pd

data = {

'Empleado': ['Carlos', 'Marta', 'Luis'],

'Salario': [2500, 3000, 2800]

}

empleados\_df = pd.DataFrame(data)

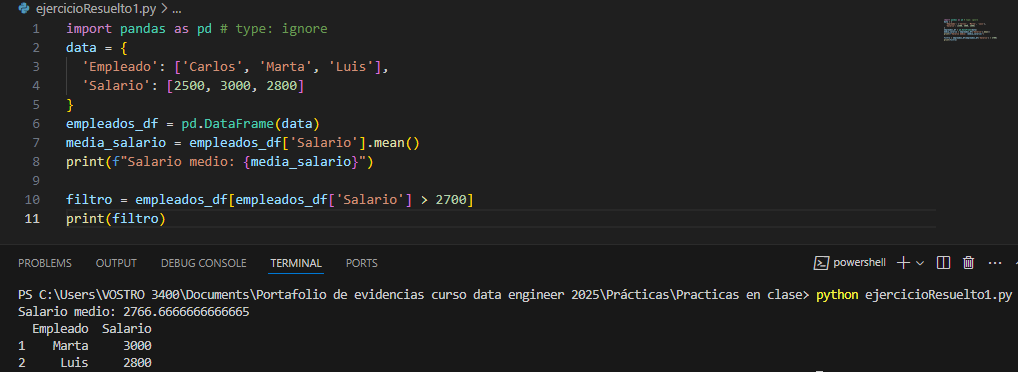
media\_salario = empleados\_df['Salario'].mean()

print(f"Salario medio: {media\_salario}")

**Ejercicio 2: Filtrar empleados con salario mayor a 2700.**

filtro = empleados\_df[empleados\_df['Salario'] > 2700]

print(filtro)

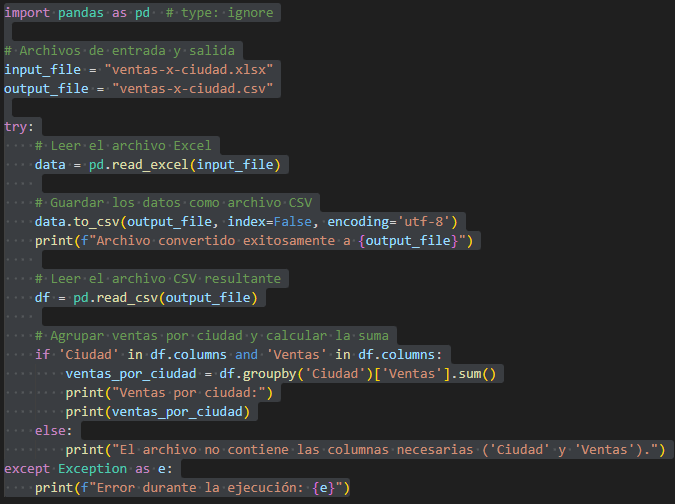
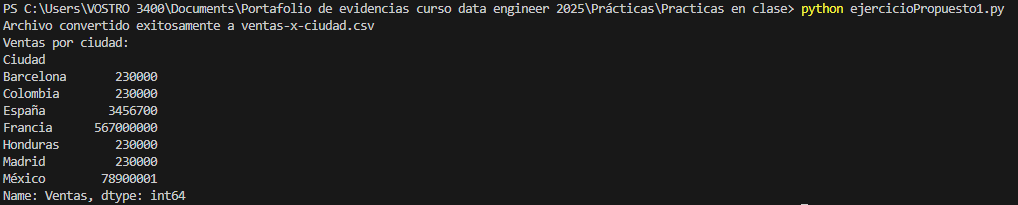


**6. Ejercicios Propuestos (Pasos para Resolverlos)**

**Ejercicio 1: Cargar un archivo CSV con datos de ventas y calcular la suma de ventas por ciudad.**

**Pasos:**

1. Convertir archivo xlsx a csv:  
    import pandas as pd # type: ignore  
    input\_file = "ventas-x-ciudad.xlsx"  
    output\_file = "ventas-x-ciudad.csv"  
    try:  
    # Leer el archivo Excel  
    data = pd.read\_excel(input\_file)  
     
    # Guardar los datos como archivo CSV  
    data.to\_csv(output\_file, index=False, encoding='utf-8')  
    print(f"Archivo convertido exitosamente a {output\_file}")  
    except Exception as e:  
    print(f"Error durante la conversión: {e}")
2. Cargar el archivo CSV: df = pd.read\_csv('ventas.csv')
3. Agrupar por ciudad: ventas\_por\_ciudad = df.groupby('Ciudad')['Ventas'].sum()
4. Imprimir el resultado: print(ventas\_por\_ciudad)

**Ejercicio 2: Crear un DataFrame con datos de alumnos y calcular el promedio de calificaciones.**

**Pasos:**

1. Crear el DataFrame con columnas: Nombre, Materia, Calificacion
2. Agrupar por Nombre y calcular la media de Calificacion
3. Imprimir el resultado

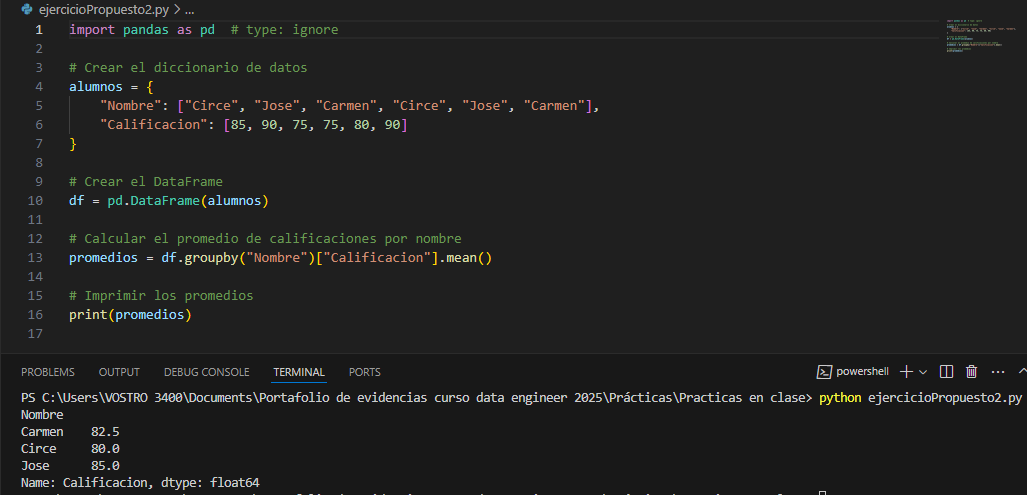
**Formato esperado:**

Nombre Promedio

Ana 8.7

Juan 7.5

Luis 9.0



Ejercicio resuelto

<https://es.educaplay.com/recursos-educativos/21911262-analisis_de_datos.html>

[actividad7AnalisisDatos.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias+Actividades/14012025/actividad7AnalisisDatos.png)

Repositorio de prácticas en clase: https://github.com/CirceDelLop/AcademiaDataEngineer/tree/main/Pr%C3%A1cticas/Practicas%20en%20clase

***4. SQL Básico: Consultas SELECT, WHERE y JOIN***

**1. SELECT**

La instrucción SELECT se utiliza para consultar datos de una o más tablas.

* **Sintaxis:**
* SELECT columna1, columna2, ...
* FROM nombre\_tabla;
* **Ejemplo:**
* SELECT nombre, edad FROM empleados;

Devuelve las columnas nombre y edad de la tabla empleados.

**2. WHERE**

La cláusula WHERE filtra registros que cumplen una condición.

* **Sintaxis:**
* SELECT columnas
* FROM tabla
* WHERE condicion;
* **Ejemplo:**
* SELECT \* FROM empleados WHERE edad > 30;

Devuelve todos los empleados con edad mayor a 30.

Consultas para realizar:

En tabla empleados agregar las columnas:

Ventas

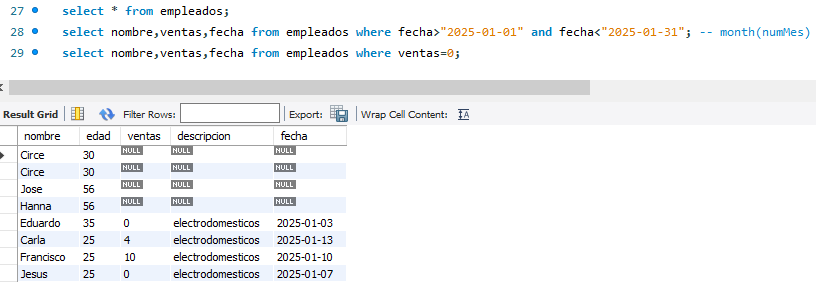
Descripcion

Fecha

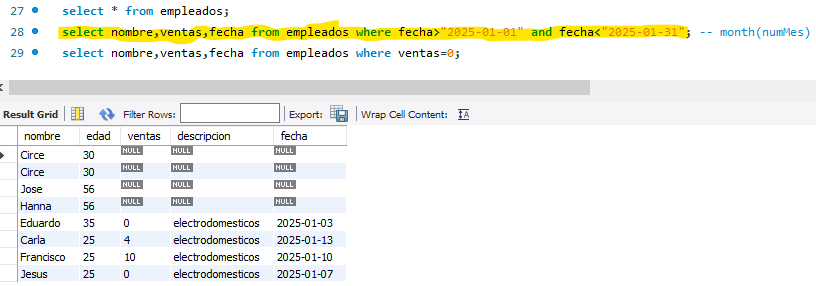
Agregar dos valores

Mostrar las ventas del mes de enero de 2025

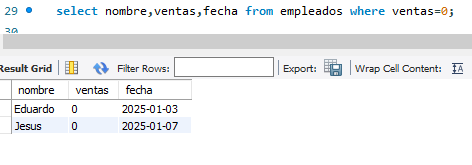
Mostrar los días de enero en que no hubo ventas



Consulta para mostrar todos los registros de la tabla empleados



Consulta para mostrar los registros que en la columna fecha tengan el mes de enero de 2025 de la tabla empledos



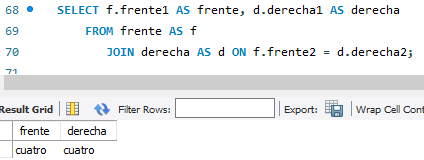
Consulta para mostrar los registros con valor 0 en la columna ventas de la tabla empleados

**3. JOIN**

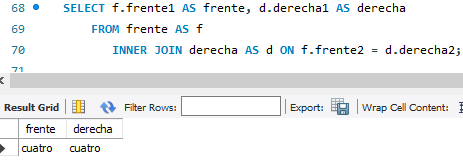
La instrucción JOIN combina filas de dos o más tablas basadas en una columna común.

* **Tipos comunes de JOIN:**
  + **INNER JOIN:** Devuelve filas que tienen coincidencias en ambas tablas.
  + **LEFT JOIN:** Devuelve todas las filas de la tabla izquierda y las coincidentes de la derecha.
  + **RIGHT JOIN:** Devuelve todas las filas de la tabla derecha y las coincidentes de la izquierda.
* **Sintaxis:**
* SELECT columnas
* FROM tabla1
* JOIN tabla2
* ON tabla1.columna = tabla2.columna;
* **Ejemplo:**
* SELECT empleados.nombre, departamentos.nombre
* FROM empleados
* INNER JOIN departamentos
* ON empleados.departamento\_id = departamentos.id;

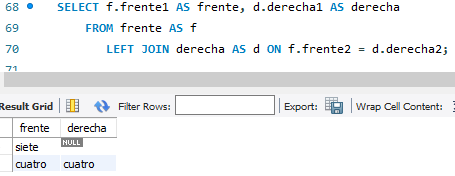
Devuelve los nombres de empleados y los nombres de sus departamentos.



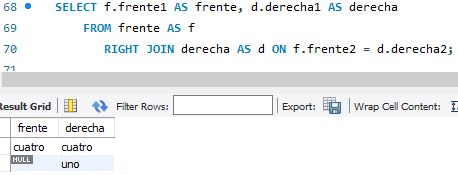
**JOIN**



**INNER JOIN**

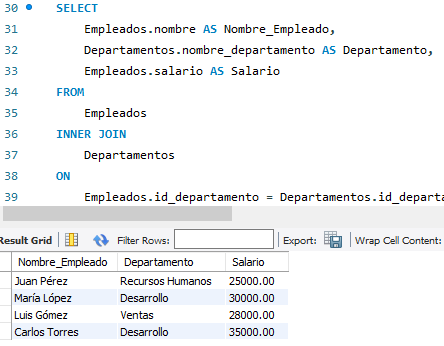


**LEFT JOIN**

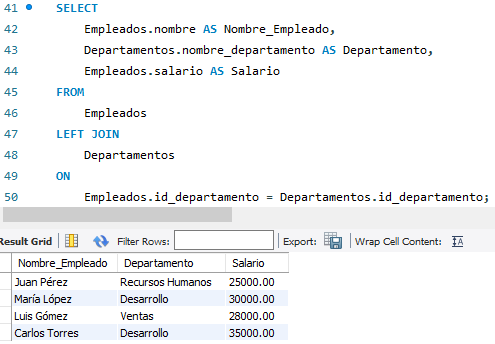


**RIGTH JOIN**

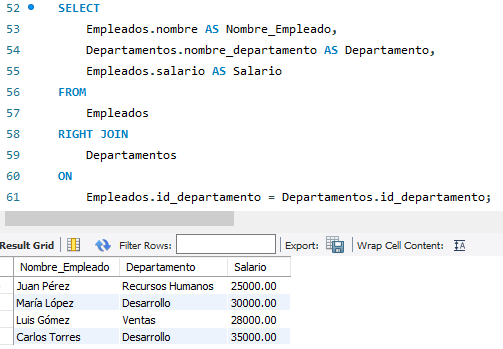
**Apuntes 15/01/2025**



**INNER JOIN**



**LEFT JOIN**



**RIGHT JOIN**

**Link de script de la practica:**

**Ejemplo Resuelto 1: SELECT**

**Consulta:** Listar los nombres y edades de todos los empleados.

**Pasos:**

1. Identificar las columnas necesarias: nombre, edad.
2. Usar SELECT y especificar la tabla empleados.

**Código:**

SELECT nombre, edad FROM empleados;

**Resultado:**

| **nombre** | **edad** |
| --- | --- |
| Ana | 25 |
| Carlos | 30 |
| Beatriz | 35 |

**Ejemplo Resuelto 2: WHERE**

**Consulta:** Obtener los empleados cuyo salario es mayor a 50,000.

**Pasos:**

1. Identificar la condición: salario > 50000.
2. Especificar la tabla empleados.
3. Aplicar el filtro con WHERE.

**Código:**

SELECT nombre, salario

FROM empleados

WHERE salario > 50000;

**Resultado:**

| **nombre** | **salario** |
| --- | --- |
| Carlos | 60000 |
| Beatriz | 70000 |

**Ejemplo Resuelto 3: JOIN**

**Consulta:** Mostrar los nombres de los empleados junto con el nombre de su departamento.

**Pasos:**

1. Identificar las tablas: empleados y departamentos.
2. Determinar la columna de relación: empleados.departamento\_id = departamentos.id.
3. Usar INNER JOIN para combinar ambas tablas.

**Código:**

SELECT empleados.nombre AS empleado, departamentos.nombre\_departamento AS departamento

FROM empleados

INNER JOIN departamentos

ON empleados.id\_departamento = departamentos.id\_departamento;

**Resultado:**

| **empleado** | **departamento** |
| --- | --- |
| Ana | Ventas |
| Carlos | Finanzas |
| Beatriz | Recursos Humanos |

[**https://es.educaplay.com/recursos-educativos/21913389-4\_sql\_basico\_consultas\_select\_where\_y\_join.html**](https://es.educaplay.com/recursos-educativos/21913389-4_sql_basico_consultas_select_where_y_join.html)

[actividad1SQLBasicoConsultasSelectWhereJoin.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias%2BActividades/15012025/actividad1SQLBasicoConsultasSelectWhereJoin.png)

**Ejercicios Propuestos con Solución**

**Ejercicio 1: SELECT**

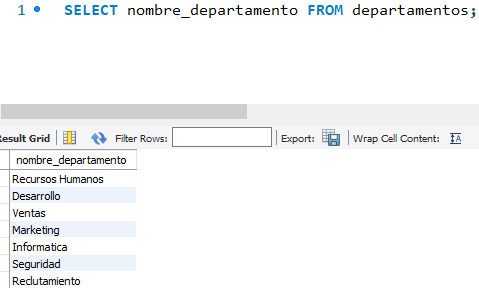
**Consulta:** Listar los nombres de todos los departamentos.

**Código:**

SELECT nombre\_departamento FROM departamentos;

**Resultado:**

|  |
| --- |
| nombre |
| Ventas |
| Finanzas |
| Recursos Humanos |



**Ejercicio 2: WHERE**

**Consulta:** Obtener los empleados que trabajan en el departamento de "Ventas".

**Código:**

SELECT nombre

FROM empleados

WHERE departamento\_id = (SELECT id FROM departamentos WHERE nombre = 'Ventas');

**Resultado:**

|  |
| --- |
| nombre |
| Ana |

**Ejercicio 3: JOIN**

**Consulta:** Listar los salarios de los empleados junto con el nombre de su departamento.

**Código:**

SELECT empleados.nombre, empleados.salario, departamentos.nombre AS departamento

FROM empleados

INNER JOIN departamentos

ON empleados.departamento\_id = departamentos.id;

**Resultado:**

|  |  |  |
| --- | --- | --- |
| nombre | salario | departamento |
| Ana | 45000 | Ventas |
| Carlos | 60000 | Finanzas |
| Beatriz | 70000 | Recursos Humanos |

<https://es.educaplay.com/recursos-educativos/21913675-conceptos_y_consultas_sql.html>

[actividad2ConceptosConsultasSQL.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias%2BActividades/15012025/actividad2ConceptosConsultasSQL.png)

***5. SQL Avanzado: Subconsultas, funciones de agregación, optimización.***

**1. Teoría:**

1. **Subconsultas**  
   Una subconsulta es una consulta anidada dentro de otra consulta SQL. Se utilizan para realizar operaciones complejas como filtrar datos con condiciones basadas en otras tablas.
   * Pueden aparecer en la cláusula WHERE, FROM, o SELECT.
   * Ejemplo: Recuperar datos donde el resultado depende de otra consulta.
2. **Funciones de agregación**  
   Estas funciones realizan cálculos en un conjunto de valores y devuelven un único valor. Ejemplos comunes incluyen:
   * COUNT (): Cuenta filas.
   * SUM (): Calcula la suma de un conjunto de valores.
   * AVG (): Calcula el promedio.
   * MAX () y MIN (): Encuentran el valor máximo y mínimo.
3. **Optimización de consultas**  
   Técnicas para mejorar el rendimiento:
   * Utilizar índices en columnas clave.
   * Reducir el uso de subconsultas y optar por uniones (JOIN) si es más eficiente.
   * Limitar los resultados con LIMIT.
   * Evitar el uso de SELECT \*, especificando las columnas necesarias.

**2. Ejemplos resueltos con solución**

**Ejemplo 1: Subconsulta en una cláusula WHERE**

**Problema:** Encuentra los empleados cuyo salario está por encima del promedio.  
**Consulta:**

SELECT nombre, salario

FROM empleados

WHERE salario > (SELECT AVG(salario) FROM empleados);

**Explicación:** La subconsulta calcula el salario promedio, y la consulta principal filtra los empleados con salarios mayores a este.

**Ejemplo 2: Funciones de agregación con GROUP BY**

**Problema:** Muestra el total de ventas por región.  
**Consulta:**

SELECT region, SUM (ventas) AS total ventas

FROM ventas

GROUP BY region;

**Explicación:** La función SUM calcula el total de ventas para cada región agrupada por la columna region.

**Ejemplo 3: Optimización con índices y LIMIT**

**Problema:** Recupera los 5 productos más caros de la tabla productos.  
**Consulta:**

SELECT nombre, precio

FROM productos

ORDER BY precio DESC

LIMIT 5;

**Explicación:** Ordena los productos por precio de forma descendente y devuelve los primeros 5 resultados. El uso de LIMIT reduce el procesamiento de datos.

**3. Ejemplos propuestos con solución**

**Ejemplo 1: Subconsulta con IN**

**Problema:** Encuentra los clientes que han realizado pedidos en enero de 2025.  
**Consulta propuesta:**

SELECT nombre

FROM clientes

WHERE id\_cliente IN (SELECT id\_cliente

FROM pedidos

WHERE fecha\_pedido BETWEEN '2025-01-01' AND '2025-01-14');

**Solución:**  
La subconsulta devuelve los IDs de los clientes con pedidos en enero de 2025, que luego son usados para filtrar la tabla clientes.

**Ejemplo 2: Funciones de agregación combinadas**

**Problema:** Muestra el promedio y la suma de las ventas por categoría.  
**Consulta propuesta:**

SELECT categoria, AVG (ventas) AS promedio ventas, SUM (ventas) AS suma ventas

FROM productos

GROUP BY categoria;

**Solución:**  
Las funciones AVG y SUM se aplican a las ventas agrupadas por categoría.

**Ejemplo 3: Optimización eliminando subconsultas**

**Problema:** Recupera el nombre y la región de los clientes con pedidos superiores a $500.  
**Consulta propuesta:**

SELECT c.nombre, c.region

FROM clientes c

JOIN pedidos p ON c.cliente\_id = p.cliente\_id

WHERE p.monto > 500;

Show databases; Se usa para visualizar todas las bases de datos que se tienen. Show tables; Se usa para visuallizar tolas las tablas dentro de la una BD.

**Solución:**  
En lugar de una subconsulta, se usa una unión para relacionar clientes con pedidos, mejorando la eficiencia.

***Script visto en clase (extra)***

CREATE DATABASE IF NOT EXISTS TiendaDB;  
 USE TiendaDB;

CREATE TABLE IF NOT EXISTS Productos (  
 id\_producto INT AUTO\_INCREMENT PRIMARY KEY,  
 nombre\_producto VARCHAR(100),  
 precio DECIMAL(10, 2),  
 fecha\_ultimo\_precio DATE  
 );  
 CREATE TABLE IF NOT EXISTS Clientes (  
 id\_cliente INT AUTO\_INCREMENT PRIMARY KEY,  
 nombre\_cliente VARCHAR(100)  
 );  
 CREATE TABLE IF NOT EXISTS Ventas (  
 id\_venta INT AUTO\_INCREMENT PRIMARY KEY,  
 id\_cliente INT,  
 id\_producto INT,  
 cantidad INT,  
 fecha DATE,  
 FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente),  
 FOREIGN KEY (id\_producto) REFERENCES Productos(id\_producto)  
 );

INSERT INTO Productos (nombre\_producto, precio, fecha\_ultimo\_precio) VALUES  
 ('Camiseta', 20.00, '2025-01-01'),  
 ('Pantalón', 30.00, '2025-01-01'),  
 ('Zapatos', 50.00, '2025-01-01'),  
 ('Chaqueta', 70.00, '2025-01-01');

INSERT INTO Clientes (nombre\_cliente) VALUES  
 ('Juan Pérez'),  
 ('Ana Gómez'),  
 ('Carlos López'),  
 ('María Rodríguez');  
 INSERT INTO Ventas (id\_cliente, id\_producto, cantidad, fecha) VALUES  
 (1, 1, 2, '2025-01-10'),  
 (1, 3, 1, '2025-01-12'),  
 (2, 2, 1, '2025-01-11'),  
 (3, 1, 3, '2025-01-14');  
 Obtener ventas con detalles de productos

Obtener ventas con detalles de productos

SELECT V.id\_venta, V.cantidad, P.nombre\_producto, P.precio  
 FROM Ventas V  
 INNER JOIN Productos P ON V.id\_producto = P.id\_producto;

Ver todos los productos y las ventas correspondientes

SELECT P.nombre\_producto, P.precio, V.id\_venta, V.cantidad  
 FROM Productos P  
 LEFT JOIN Ventas V ON P.id\_producto = V.id\_producto;

Obtener ventas sin detalles de productos

SELECT V.id\_venta, V.cantidad, P.nombre\_producto  
 FROM Ventas V  
 RIGHT JOIN Productos P ON V.id\_producto = P.id\_producto;

INNER JOIN con múltiples tablas: Obtener ventas, productos y clientes

SELECT V.id\_venta, C.nombre\_cliente, P.nombre\_producto, V.cantidad  
 FROM Ventas V  
 INNER JOIN Productos P ON V.id\_producto = P.id\_producto  
 INNER JOIN Clientes C ON V.id\_cliente = C.id\_cliente;

LEFT JOIN con productos sin ventas: Obtener productos que no han sido vendidos

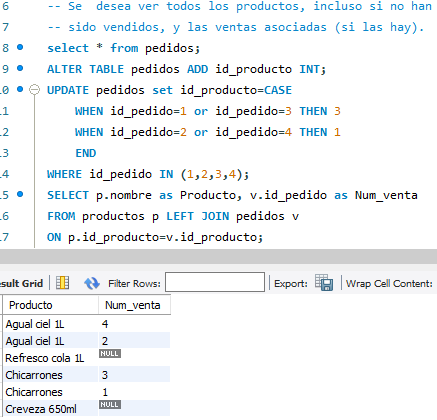
SELECT P.nombre\_producto  
 FROM Productos P  
 LEFT JOIN Ventas V ON P.id\_producto = V.id\_producto  
 WHERE V.id\_venta IS NULL;

RIGHT JOIN con ventas sin productos

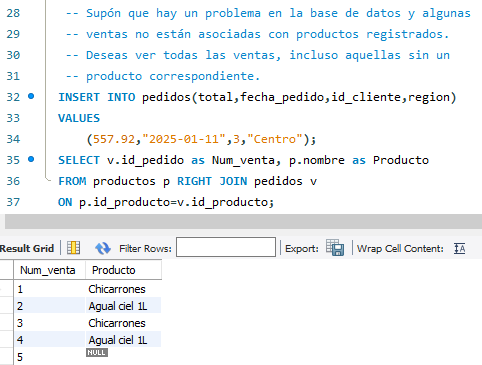
SELECT V.id\_venta, V.cantidad, P.nombre\_producto  
 FROM Ventas V  
 RIGHT JOIN Productos P ON V.id\_producto = P.id\_producto;

INNER JOIN con precios actuales de productos y ventas

SELECT V.id\_venta, V.cantidad, P.nombre\_producto, P.precio  
 FROM Ventas V  
 INNER JOIN Productos P ON V.id\_producto = P.id\_producto  
 WHERE P.fecha\_ultimo\_precio <= V.fecha;



Ejercicio propuesto 2



Ejercicio propuesto 6

Eres parte del equipo de desarrollo de una tienda en línea que vende diversos productos como consolas de videojuegos, tenis, ropa y artículos para jóvenes. El negocio está creciendo rápidamente y necesitas crear un sistema para gestionar la información de los productos, clientes y ventas. Tu tarea es resolver varios problemas utilizando SQL, que ayudarán a la empresa a obtener información clave sobre su funcionamiento.

Objetivos 1:

Crear la base de datos y las tablas necesarias.

Categorías de productos ( Xbox, Tenis, Ropa, Artículos para jóvenes)

Productos (detalles como nombre, precio, stock y categoría)

Clientes (nombre, correo electrónico)

Ventas (relación de qué producto compró cada cliente, cantidad y fecha de la venta)

Objetivos: 2:

Realizar las siguientes consultas SQL para obtener información relevante para la tienda:

-Obtener el total de compras realizadas por cada cliente usando INNER JOIN

<https://es.educaplay.com/recursos-educativos/21913908-instrucciones_sql.html>

[actividad3InstruccionesSQL.png](https://us-east-1.console.aws.amazon.com/s3/object/mybucketacademiadataengineer?region=us-east-1&bucketType=general&prefix=Evidencias%2BActividades/15012025/actividad3InstruccionesSQL.png)

Python es la herramienta perfecta para manejar, transformar y analizar datos en AWS. Combina la simplicidad del lenguaje con el poder de los servicios en la nube, como S3, Glue, Redshift y Lambda, permitiéndote procesar grandes volúmenes de información de manera eficiente y escalable. Con Python puedes escribir scripts para interactuar directamente con los servicios de AWS, automatizando tareas y optimizando tus pipelines de datos.

***Módulo 3: AWS Glue y Procesos ETL***

***Duración Total:*** *15 horas*

**Objetivo del Módulo:**

Proporcionar a los participantes los conocimientos y habilidades necesarios para diseñar, implementar y gestionar procesos de Extracción, Transformación y Carga (ETL) utilizando AWS Glue, optimizando el manejo de datos en entornos escalables y en la nube.

**Distribución de Horas:**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Teoría (%) | Práctica (%) | Descripción |
| 1. Procesamiento ETL con AWS Glue | 30% | 70% | Introducción al concepto de ETL y su aplicación en AWS Glue. Práctica: crear un pipeline básico de ETL. |
| 1. Creación y automatización de jobs de ETL | 20% | 80% | Explicación teórica sobre los jobs en Glue. Práctica: configuración y automatización de jobs con triggers y workflows. |
| 1. Catalogación de datos con Glue | 25% | 75% | Conceptos básicos sobre el Data Catalog de AWS Glue. Práctica: catalogar y consultar datasets en S3. |

* + - 1. ***Procesamiento ETL con AWS Glue***

**1.1 ¿Qué es ETL?**

ETL (Extract, Transform, Load)

Es un proceso para:

Extraer datos desde diferentes fuentes (bases de datos, archivos, APIs, etc.).

Transformar esos datos para adaptarlos a un formato específico o enriquecerlos.

Cargar los datos procesados en un destino, como un almacén de datos o un lago de datos.

**1.2 ¿Qué es AWS Glue?**

AWS Glue es un servicio totalmente administrado de Amazon Web Services que simplifica el desarrollo y ejecución de procesos ETL a escala.

*1.2.1Componentes principales de AWS Glue:*

1. **Data Catalog**: Almacena metadatos de las fuentes de datos.
2. **Crawlers:** Descubren y catalogan automáticamente los datos.
3. **Jobs:** Ejecutan los procesos ETL, escritos en Python o Scala usando Apache Spark.
4. **Triggers:** Automatizan la ejecución de los jobs según un horario o evento.

<https://es.educaplay.com/recursos-educativos/21914199-procesamiento_etl_con_aws_glue.html>

1.3 Ejemplo resuelto: Crear un pipeline básico de ETL

**Caso:**

Necesitamos procesar un archivo CSV en Amazon S3, transformar sus datos (por ejemplo, calcular el promedio de ventas), y almacenarlos en otro bucket de S3 como un archivo Parquet.

***Pasos y solución:***

1. Crear un Crawler:
2. Configurar un crawler en AWS Glue para catalogar el archivo CSV.
3. Ejecutar el crawler para registrar los metadatos en el Data Catalog.
4. Crear un Job de ETL:
5. En el Job, seleccionar el archivo catalogado como origen.
6. Escribir un script para transformar los datos.
7. Configurar el destino como otro bucket de S3 en formato Parquet.
8. Script básico para el Job:

import sys

from awsglue.transforms import \*

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB\_NAME'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args['JOB\_NAME'], args)

# Cargar datos desde el Data Catalog

datasource = glueContext.create\_dynamic\_frame.from\_catalog(database="mi\_base\_datos", table\_name="mi\_tabla")

# Transformar los datos (calcular el promedio de ventas)

transformed = datasource.toDF()

transformed = transformed.groupBy("categoria").avg("ventas").withColumnRenamed("avg(ventas)", "promedio\_ventas")

# Escribir los datos transformados en S3 como Parquet

output = glueContext.write\_dynamic\_frame.from\_options(

frame=DynamicFrame.fromDF(transformed, glueContext, "output"),

connection\_type="s3",

connection\_options={"path": "s3://mi-bucket-destino/transformados/"},

format="parquet"

)

job.commit()

1. Probar el pipeline:
2. Ejecutar el job y verificar que el archivo transformado se almacene en el bucket destino.

***Ejemplos propuestos con solución***

**Ejemplo 1**: Filtrar datos en un pipeline ETL

Problema: Filtra las ventas mayores a $1000 de un archivo CSV y guárdalas en un bucket de S3 en formato JSON.

Solución:

Modificar el script del job para agregar un filtro:

# Filtrar las ventas mayores a $1000

filtered = datasource.toDF().filter("ventas > 1000")

# Escribir los datos filtrados en formato JSON

output = glueContext.write\_dynamic\_frame.from\_options(

frame=DynamicFrame.fromDF(filtered, glueContext, "output"),

connection\_type="s3",

connection\_options={"path": "s3://mi-bucket-destino/filtrados/"},

format="json"

)

**Ejemplo 2**: Enriquecer datos en un pipeline ETL

Problema: Enriquecer un dataset con datos de otra tabla, agregando la descripción del producto.

Solución:

Usar una unión (join) entre dos tablas:

# Cargar datos adicionales desde otra tabla

productos = glueContext.create\_dynamic\_frame.from\_catalog(database="mi\_base\_datos", table\_name="productos")

# Unir datasets por el campo 'producto\_id'

joined = datasource.toDF().join(productos.toDF(), "producto\_id")

# Escribir los datos enriquecidos en S3

output = glueContext.write\_dynamic\_frame.from\_options(

frame=DynamicFrame.fromDF(joined, glueContext, "output"),

connection\_type="s3",

connection\_options={"path": "s3://mi-bucket-destino/enriquecidos/"},

format="parquet"

)

* + - 1. ***Creación y automatización de jobs de ETL***

**2.1 ¿Qué son los jobs de ETL en Glue?**

AWS Glue es un servicio de ETL (Extract, Transform, Load) totalmente administrado que facilita la preparación y carga de datos para análisis. Un *job* en Glue representa un script (usualmente en PySpark o Python) que procesa datos. Los pasos típicos son:

1. **Extracción (Extract):** Conectar a fuentes de datos como S3, bases de datos relacionales, o NoSQL.
2. **Transformación (Transform):** Limpiar, filtrar, agregar o enriquecer los datos.
3. **Carga (Load):** Almacenar los datos transformados en destinos como Amazon S3, Redshift, o RDS.

Glue incluye herramientas como el **Data Catalog**, que permite definir esquemas y metadatos, y herramientas para configurar triggers y workflows para la automatización.

**2.2 Configuración de jobs con triggers y workflows**

1. **Trigger:** Mecanismo que inicia un job en función de un evento (horario, dependencia, etc.).
2. **Workflow:** Orquestación de múltiples jobs o crawlers, organizándolos en un flujo de trabajo.

**Ejemplo Resuelto 1: Creación y ejecución de un job en Glue**

[**https://es.educaplay.com/recursos-educativos/21916692-repaso\_de\_aws\_glue.html**](https://es.educaplay.com/recursos-educativos/21916692-repaso_de_aws_glue.html)

**1. Requisitos previos:**

* Subir datos a un bucket de S3.
* Crear un catálogo en Glue que apunte al bucket de datos.

**2. Creación del job:**

* Navega al panel de Glue en la consola de AWS.
* Selecciona "Jobs" > "Add Job".
* Configura el job:
  + Nombre: ETL\_Job\_Example
  + Tipo: Spark.
  + Script: Proporcionar uno propio o usar el editor visual.

**3. Script básico de PySpark para transformación:**

import sys

from awsglue.transforms import \*

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB\_NAME'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args['JOB\_NAME'], args)

# Lee datos desde Glue Catalog

datasource0 = glueContext.create\_dynamic\_frame.from\_catalog(

database="default",

table\_name="example\_table"

)

# Transformación: Filtrar datos

filtered\_data = Filter.apply(frame=datasource0, f=lambda x: x["age"] > 30)

# Escribir resultados en S3

glueContext.write\_dynamic\_frame.from\_options(

frame=filtered\_data,

connection\_type="s3",

connection\_options={"path": "s3://my-bucket/transformed-data/"},

format="json"

)

job.commit()

**4. Ejecución del job:**

* Configura un trigger basado en tiempo (ejemplo: cada hora).
* Ejecuta el job desde la consola.

**Ejemplo Propuesto 2: Configuración de un workflow con múltiples jobs**

**Descripción:**

Automatiza el flujo:

1. Ejecuta un crawler para actualizar el catálogo.
2. Ejecuta un job que procesa datos.
3. Ejecuta un segundo job para cargar los resultados en Redshift.

**Solución:**

1. Configura el crawler en Glue.
2. Crea los dos jobs con scripts básicos.
3. Configura un workflow:
   * Paso 1: Inicia el crawler.
   * Paso 2: Trigger para iniciar el primer job cuando el crawler termine.
   * Paso 3: Trigger para iniciar el segundo job cuando termine el primero.
4. Activa el workflow.

1. ***Catalogación de datos con Glue***

**Conceptos básicos sobre el Data Catalog de AWS Glue**

El **Data Catalog de AWS Glue** es un repositorio centralizado donde se almacenan metadatos de tus datasets. Estos metadatos permiten que las herramientas de análisis como Athena, Redshift y EMR puedan interactuar con los datos sin necesidad de conocer sus detalles físicos.

**Componentes principales:**

1. **Bases de datos:** Espacios lógicos en el catálogo que agrupan tablas relacionadas.
2. **Tablas:** Representaciones de los datos con información como ubicación en S3, formato (JSON, Parquet, CSV), y esquema (nombres de columnas, tipos de datos).
3. **Crawlers:** Servicios que inspeccionan los datos en S3 para inferir automáticamente esquemas y crear o actualizar tablas en el Data Catalog.
4. **Particiones:** Subconjuntos de datos organizados por claves (por ejemplo, year=2024/month=12).

**Beneficios:**

* **Automatización:** Detecta automáticamente cambios en los esquemas de datos.
* **Interoperabilidad:** Compatible con múltiples servicios de AWS.
* **Escalabilidad:** Diseñado para manejar grandes volúmenes de datos.

<https://es.educaplay.com/recursos-educativos/21916881-el_data_catalog_de_aws.html>

**Ejemplo Resuelto 1: Catalogar un dataset en S3**

**1. Requisitos previos:**

* Dataset almacenado en S3 (por ejemplo, s3://my-bucket/sales-data/).
* Permisos adecuados en IAM para Glue y S3.

**2. Configurar un Crawler:**

1. Ve a **AWS Glue > Crawlers** y selecciona "Add Crawler".
2. Configura:
   * Nombre: sales-data-crawler.
   * Fuente de datos: Selecciona la ubicación en S3.
   * Rol: Crea o utiliza un rol con permisos para Glue y S3.
   * Configuración de salida: Asocia el crawler con una base de datos (puedes crear una nueva, como sales\_data\_db).
3. Ejecuta el crawler.

**3. Resultados del Crawler:**

* Una tabla llamada sales\_data aparecerá en el catálogo dentro de la base de datos seleccionada.
* Detalles del esquema, formato y particiones estarán disponibles.

**4. Consulta del dataset:**

Utiliza Amazon Athena para consultar la tabla:

SELECT \*

FROM sales\_data

WHERE year = '2024';

**Ejemplo Propuesto 2: Catalogar y consultar datos particionados**

**Descripción:**

Un bucket S3 contiene datos de ventas organizados por particiones: s3://my-bucket/sales-data/year=2024/month=01/.

1. Configura un crawler para catalogar estos datos con sus particiones.
2. Consulta los datos en Athena para obtener las ventas de enero de 2024.

**Solución:**

1. Crea el crawler como en el ejemplo anterior.
2. En Athena, la consulta sería:

SELECT \*

FROM sales\_data

WHERE year = '2024' AND month = '01';

**Ejemplo Propuesto 3: Actualización automática del catálogo**

**Descripción:**

Automatiza la catalogación para que, al subir nuevos datos a S3, el catálogo se actualice automáticamente.

**Solución:**

1. Configura un evento de S3 para invocar un Lambda al detectar nuevos archivos.
2. En la función Lambda, utiliza el SDK de Glue para iniciar el crawler:

import boto3

def lambda\_handler(event, context):

glue\_client = boto3.client('glue')

response = glue\_client.start\_crawler(Name='sales-data-crawler')

return response

1. Prueba subiendo un archivo a S3 y verifica la actualización del catálogo.

AWS Glue y Procesos ETL  
ETL (Extracción, Transformación y Carga) es un proceso que maneja la recolección, limpieza y almacenamiento de datos. AWS Glue es un servicio serverless que automatiza ETL, escalando automáticamente sin necesidad de infraestructura. Ofrece un catálogo de datos para organizar metadatos y usa PySpark para la transformación de datos. AWS Glue es ideal para manejar grandes volúmenes de datos y está integrado con servicios como S3, Redshift y Athena.

***Módulo 4: Gestión de Datos con Amazon Redshift y Aurora.***

***Duración Total:*** *20 horas*

**Objetivo del Módulo:**

Proveer a los participantes los conocimientos y habilidades prácticas necesarias para diseñar, implementar y optimizar soluciones de gestión de datos utilizando Amazon Redshift y Amazon Aurora, enfocándose en la construcción de almacenes de datos escalables, la administración eficiente de bases de datos relacionales, y la aplicación de estrategias de respaldo y recuperación para garantizar la disponibilidad y resiliencia de los datos.

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Porcentaje Teoría | Porcentaje Práctica | Descripción |
| 1. Implementación y optimización de data warehouses con Amazon Redshift | 30% | 70% | Configuración, gestión y mejora del rendimiento de almacenes de datos en Amazon Redshift. |
| 1. Gestión de bases de datos relacionales con Amazon Aurora | 40% | 60% | Uso de Amazon Aurora para administrar bases de datos relacionales con alta disponibilidad y escalabilidad. |
| 1. Estrategias de respaldo y recuperación | 50% | 50% | Creación de planes para respaldos automáticos, recuperación ante desastres y restauración de datos. |

1. **Implementación y optimización de data warehouses con Amazon Redshift**
2. **Amazon Redshift**  
   Amazon Redshift es un servicio de almacenamiento de datos (data warehouse) en la nube que permite consultas rápidas y escalables para grandes volúmenes de datos.
   1. **Características principales**:
      1. Uso de nodos de clúster para procesar datos en paralelo.
      2. Compatible con SQL para consultas.
      3. Integración con herramientas como Amazon S3 para ETL.
   2. **Casos de uso**: Generación de reportes analíticos y dashboards.
3. **Amazon Aurora**  
   Aurora es un motor de base de datos relacional totalmente administrado, compatible con MySQL y PostgreSQL.
   1. **Características principales**:
      1. Escalabilidad automática hasta 128 TB por base de datos.
      2. Resiliencia con múltiples zonas de disponibilidad.
   2. **Casos de uso**: Aplicaciones empresariales que requieren alta disponibilidad y velocidad.
4. **Respaldo y Recuperación**
   1. Configuración de respaldos automáticos y manuales para garantizar la recuperación ante fallos.
   2. Uso de snapshots para restaurar datos en puntos específicos en el tiempo.

**Ejemplo Resuelto 1: Consultas en Amazon Redshift**

**Problema:** Crear una consulta en Amazon Redshift para obtener el total de ventas por región en una tabla llamada sales.  
**Solución:**

SELECT region, SUM (sales\_amount) AS total\_sales

FROM sales

GROUP BY region

ORDER BY total\_sales DESC;

**Explicación:**

* Agrupa las ventas (SUM) por región.
* Ordena los resultados en orden descendente de total de ventas.

**Ejemplo Resuelto 2: Configuración de Respaldo en Aurora**

**Problema:** Configurar un respaldo automático para una base de datos en Aurora. **Solución:**

1. Inicia sesión en la consola de AWS.
2. Ve a "RDS" > "Bases de Datos".
3. Selecciona la base de datos y habilita "Backups automáticos".
4. Configura el periodo de retención (7 días, por ejemplo).
5. Guarda los cambios.

**Resultado esperado:** Aurora realizará respaldos automáticos diariamente y los conservará por 7 días.

**Ejemplo Propuesto 1: Creación de una Vista en Redshift**

**Problema:** Crear una vista que muestre los clientes con más de $10,000 en compras.  
**Solución:**

CREATE VIEW high\_value\_customers AS

SELECT customer\_id, SUM (purchase\_amount) AS total\_purchases

FROM sales

GROUP BY customer\_id

HAVING SUM(purchase\_amount) > 10000;

**Explicación:**  
Esta vista permite consultar clientes con un alto volumen de compras de forma recurrente.

**Ejemplo Propuesto 2: Restauración de un Respaldo en Aurora**

**Problema:** Restaurar un respaldo manual en Aurora para un caso de prueba.  
**Solución:**

1. En la consola de AWS, ve a "Snapshots".
2. Selecciona el snapshot deseado.
3. Haz clic en "Restaurar Snapshot".
4. Configura los detalles de la nueva instancia y finaliza el proceso.

**Resultado esperado:** Se crea una nueva instancia basada en el respaldo especificado.

1. **Gestión de Bases de Datos Relacionales con Amazon Aurora**

Amazon Aurora es un servicio de base de datos relacional en la nube que ofrece alta disponibilidad, escalabilidad automática, y compatibilidad con MySQL y PostgreSQL.

* **Ventajas principales**:
  + **Rendimiento**: Hasta 5 veces más rápido que MySQL y 3 veces más rápido que PostgreSQL.
  + **Alta disponibilidad**: Replica automáticamente datos en múltiples zonas de disponibilidad.
  + **Resiliencia**: Detección y recuperación automática de fallos en milisegundos.

**Casos de uso**:

* Aplicaciones transaccionales empresariales.
* Sistemas de comercio electrónico con alta concurrencia.
* Bases de datos para aplicaciones SaaS.

<https://es.educaplay.com/recursos-educativos/21928757-conceptos_aws.html>

**Ejemplo Resuelto 1: Creación de una Base de Datos Aurora**

**Problema:** Crear una instancia de Amazon Aurora compatible con MySQL.  
**Solución**:

1. **Accede a AWS RDS**:
   * Ve a la consola de AWS y selecciona "RDS".
2. **Elige Crear Base de Datos**:
   * Haz clic en "Create database".
3. **Selecciona Aurora**:
   * En "Motor de base de datos", selecciona **Amazon Aurora**.
   * Escoge la compatibilidad deseada (MySQL en este caso).
4. **Configura el Clúster**:
   * **Clase de instancia**: Selecciona db.t3.medium para propósitos de prueba.
   * **Configuración de almacenamiento**: Habilita la escalabilidad automática.
5. **Configura Acceso**:
   * Proporciona un nombre de usuario y contraseña.
   * Configura el grupo de seguridad para permitir conexiones desde tu IP.
6. **Crea la Base de Datos**:
   * Revisa la configuración y haz clic en "Create database".

**Resultado esperado:** Una base de datos Aurora estará disponible para conexiones después de unos minutos.

**Ejemplo Resuelto 2: Consulta Básica en Aurora**

**Problema:** Insertar y consultar datos en una tabla en Aurora.  
**Solución**:

1. **Conéctate a Aurora**:
   * Usa un cliente SQL (como MySQL Workbench) para conectarte a tu instancia de Aurora.
2. **Crea una Tabla**:

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

position VARCHAR(50),

salary DECIMAL(10, 2)

);

1. **Inserta Datos**:

INSERT INTO employees (name, position, salary)

VALUES

('Alice', 'Manager', 75000.00),

('Bob', 'Developer', 60000.00),

('Charlie', 'Analyst', 50000.00);

1. **Consulta los Datos**:

SELECT \* FROM employees WHERE salary > 55000;

**Resultado esperado:** La consulta devolverá empleados con salario superior a $55,000.

**Ejemplo Propuesto 1: Creación de un Índice**

**Problema:** Crear un índice en la tabla employees para mejorar la velocidad de las consultas por position.  
**Solución**:

1. **Crea el Índice**:

CREATE INDEX idx\_position ON employees (position);

1. **Verifica el Índice**:

SHOW INDEX FROM employees;

**Resultado esperado:** El índice idx\_position aparecerá en la lista de índices de la tabla.

**Ejemplo Propuesto 2: Configuración de Respaldo Automático**

**Problema:** Configurar respaldos automáticos en Amazon Aurora.  
**Solución Paso a Paso**:

1. **Accede a AWS RDS**:
   * Ve a la consola de AWS y selecciona "RDS".
2. **Selecciona tu Base de Datos**:
   * Haz clic en la base de datos Aurora que quieres configurar.
3. **Configura el Respaldo**:
   * Ve a "Modify".
   * En la sección "Backups", habilita los respaldos automáticos.
   * Configura el periodo de retención (por ejemplo, 7 días).
4. **Guarda los Cambios**:
   * Haz clic en "Continue" y luego en "Modify DB Instance".

**Resultado esperado:** Aurora realizará respaldos automáticos diarios y los conservará según el periodo configurado.

1. **Estrategias de Respaldo y Recuperación**

Las estrategias de respaldo y recuperación son esenciales para garantizar la disponibilidad y protección de datos ante fallos o desastres.

* **Respaldo (Backup):** Copia de los datos almacenada en un lugar seguro para su recuperación en caso de pérdida.
* **Recuperación:** Proceso de restaurar los datos desde un respaldo para minimizar el impacto de interrupciones.

**Tipos de respaldos**:

1. **Completo**: Copia todos los datos.
2. **Incremental**: Copia solo los datos que cambiaron desde el último respaldo.
3. **Diferencial**: Copia los datos que cambiaron desde el último respaldo completo.

**Mejores prácticas**:

* Automatizar respaldos.
* Probar la recuperación regularmente.
* Almacenar copias en ubicaciones separadas (on-premise y en la nube).

<https://es.educaplay.com/recursos-educativos/21920157-db_amazon.html>

**Ejemplo Resuelto 1: Configuración de Respaldo Automático en Amazon RDS**

**Problema:** Configurar respaldos automáticos para una base de datos en RDS.  
**Solución**:

1. **Accede a AWS RDS**:
   * Ve a la consola de AWS y selecciona "RDS".
2. **Selecciona la Base de Datos**:
   * Elige la instancia para configurar respaldos.
3. **Habilita Respaldo Automático**:
   * Haz clic en "Modify".
   * En la sección "Backup", habilita "Backups automáticos".
   * Configura el periodo de retención (por ejemplo, 7 días).
4. **Guarda los Cambios**:
   * Haz clic en "Continue" y luego en "Modify DB Instance".

**Resultado esperado:** Los respaldos automáticos se realizarán diariamente y se conservarán durante el periodo configurado.

**Ejemplo Resuelto 2: Recuperación de Datos desde un Snapshot en Aurora**

**Problema:** Restaurar una base de datos Aurora desde un snapshot manual.  
**Solución**:

1. **Accede a AWS RDS**:
   * Ve a "Snapshots".
2. **Selecciona el Snapshot**:
   * Elige el snapshot manual que deseas usar.
3. **Restaura desde Snapshot**:
   * Haz clic en "Restore Snapshot".
   * Configura los detalles de la nueva instancia (nombre, clase, etc.).
4. **Completa la Restauración**:
   * Haz clic en "Restore DB Instance".

**Resultado esperado:** Se creará una nueva instancia de Aurora con los datos del snapshot.

**Ejemplo Propuesto 1: Configuración de Respaldo Manual en Redshift**

**Problema:** Crear un respaldo manual de un clúster de Amazon Redshift.  
**Solución**:

1. **Accede a AWS Redshift**:
   * Ve a la consola de AWS y selecciona "Redshift".
2. **Selecciona el Clúster**:
   * Elige el clúster para el respaldo.
3. **Crea el Respaldo**:
   * Haz clic en "Actions" > "Create Snapshot".
   * Asigna un nombre al snapshot.
4. **Confirma el Respaldo**:
   * Haz clic en "Create Snapshot".

**Resultado esperado:** Se generará un snapshot manual que se almacenará en AWS.

**Ejemplo Propuesto 2: Simulación de Recuperación ante Desastres**

**Problema:** Recuperar datos de un snapshot después de una falla crítica en un clúster de Redshift.  
**Solución**:

1. **Accede a AWS Redshift**:
   * Ve a la consola de AWS y selecciona "Snapshots".
2. **Elige el Snapshot**:
   * Selecciona el snapshot más reciente para la restauración.
3. **Restaura el Clúster**:
   * Haz clic en "Restore from Snapshot".
   * Configura el nombre y los detalles del nuevo clúster.
4. **Completa la Restauración**:
   * Haz clic en "Restore".

**Resultado esperado:** Se creará un nuevo clúster basado en el snapshot seleccionado, minimizando el tiempo de inactividad.

Amazon Redshift y Aurora son soluciones de gestión de datos altamente escalables para análisis y procesamiento de datos. Redshift optimiza consultas rápidas y consultas analíticas masivas, mientras que Aurora garantiza alta disponibilidad y desempeño para bases de datos relacionales. Ambos servicios son fundamentales para arquitecturas de datos robustas en entornos de gran escala.

***Módulo 5: Almacenamiento y Gestión de Datos con Amazon S3.***

***Duración Total:*** *10 horas*

**Objetivo del Módulo:**

Proporcionar a los participantes los conocimientos y habilidades necesarios para configurar, optimizar y gestionar almacenamiento de datos en Amazon S3, asegurando la seguridad y acceso controlado a los datos, e integrando S3 con otros servicios de AWS para una solución eficiente y escalable en la nube.

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Porcentaje Teoría | Porcentaje Práctica | Descripción |
| 1. Configuración y optimización de S3 | 40% | 60% | Implementación de almacenamiento eficiente en Amazon S3, incluyendo gestión de buckets y objetos. |
| 1. Seguridad y políticas de acceso | 35% | 65% | Configuración de permisos y políticas para el acceso seguro a los datos almacenados en S3. |
| 1. Integración de S3 con otros servicios AWS | 25% | 75% | Uso de S3 junto con servicios como Lambda, Redshift o Glue para crear soluciones integradas. |

* + 1. ***Configuración y optimización de S3:***

***1. Configuración de Amazon S3***

***Amazon S3 (Simple Storage Service) es un servicio de almacenamiento de objetos escalable, duradero y seguro.***

***Pasos básicos para la configuración:***

1. ***Crear un Bucket:***
   * ***Un bucket es un contenedor para almacenar objetos (archivos y metadatos).***
2. ***Configurar el Acceso:***
   * ***Gestiona el acceso mediante políticas de bucket o configuraciones de permisos.***
3. ***Subir Objetos:***
   * ***Sube archivos al bucket y organiza mediante prefijos y carpetas virtuales.***
4. ***Habilitar Versionado:***
   * ***Permite guardar múltiples versiones de un mismo objeto.***

***Configuraciones avanzadas:***

* ***Configuración de ciclo de vida: Define reglas para mover datos entre clases de almacenamiento***
* ***O eliminarlas automáticamente.***
* ***Habilitación de logs de acceso: Rastrea solicitudes de acceso al bucket.***
* ***Encriptación: Protege los datos en reposo usando KMS o SSE.***

***2. Optimización de Amazon S3***

***Clases de almacenamiento eficientes:***

* ***S3 Standard: Para acceso frecuente.***
* ***S3 Intelligent-Tiering: Mueve automáticamente objetosa niveles más económicos***
* ***Según patrones de acceso.***
* ***S3 Glacier: Para datos archivados y acceso esporádico.***

***Prácticas recomendadas:***

* ***Minimizar costos:***
  + ***Usar compresión antes de cargar objetos.***
  + ***Implementar reglas de ciclo de vida para mover datos obsoletos a Glacier.***
* ***Mejorar el rendimiento:***
  + ***Dividir datos grandes en partes al subirlos (multipart upload).***
  + ***Habilitar Transfer Acceleration para subir/descargar datos más rápido.***

***Gestión de permisos:***

* ***Aplicar el principio de menor privilegio.***
* ***Utilizar roles de IAM para acceso temporal en lugar de credenciales duraderas.***

[***https://es.educaplay.com/recursos-educativos/21929054-configuracion\_y\_optimizacion\_de\_s3.html***](https://es.educaplay.com/recursos-educativos/21929054-configuracion_y_optimizacion_de_s3.html)

***Ejemplo Resuelto***

***Objetivo: Crear un bucket, subir un archivo y configurar reglas de ciclo de vida.***

1. ***Crear un Bucket:***

***aws s3api create-bucket --bucket mi-bucket-ejemplo --region us-east-1***

1. ***Subir un archivo:***

***aws s3 cp archivo.txt s3://mi-bucket-ejemplo/***

1. ***Configurar reglas de ciclo de vida para mover objetos a Glacier después de 30 días:***

***{***

***"Rules": [***

***{***

***"ID": "MoverAGlacier",***

***"Filter": {},***

***"Status": "Enabled",***

***"Transitions": [***

***{***

***"Days": 30,***

***"StorageClass": "GLACIER"***

***}***

***]***

***}***

***]***

***}***

***Comando para aplicar:aws s3api put-bucket-lifecycle-configuration --bucket mi-bucket-ejemplo - lifecycle configuration*** [***file://ciclo-vida.json***](file://ciclo-vida.json)

1. ***Validar la configuración:***

***aws s3api get-bucket-lifecycle-configuration --bucket mi-bucket-ejemplo***

***Ejemplo Propuesto***

***Objetivo:***

* ***Crear un bucket con versión habilitada.***
* ***Subir un archivo modificado varias veces y listar todas sus versiones.***
* ***Mover versiones anteriores a Glacier después de 15 días.***

***Solución:***

1. ***Crear el bucket con versionado habilitado:***

***aws s3api create-bucket --bucket mi-bucket-versionado --region us-east-1***

***aws s3api put-bucket-versioning --bucket mi-bucket-versionado --versioning-configuration Status=Enabled***

1. ***Subir el archivo con modificaciones:***

***echo "Primera versión" > archivo.txt***

***aws s3 cp archivo.txt s3://mi-bucket-versionado/***

***echo "Segunda versión" > archivo.txt***

***aws s3 cp archivo.txt s3://mi-bucket-versionado/***

1. ***Listar todas las versiones del archivo:***

***aws s3api list-object-versions --bucket mi-bucket-versionado***

1. ***Configurar ciclo de vida para mover versiones antiguas a Glacier:***

***{***

***"Rules": [***

***{***

***"ID": "MoverVersionesAntiguas",***

***"Filter": {},***

***"Status": "Enabled",***

***"NoncurrentVersionTransitions": [***

***{***

***"NoncurrentDays": 15,***

***"StorageClass": "GLACIER"***

***}***

***]***

***}***

***]***

***}***

***Comando para aplicar:***

***aws s3api put-bucket-lifecycle-configuration --bucket mi-bucket-versionado --lifecycle-configuration - file://ciclo-vida.json***

***aws s3api get-bucket-lifecycle-configuration --bucket mi-bucket-versionado***

***2. Seguridad y Políticas de Acceso en Amazon S3***

***1. Conceptos clave***

* ***Políticas de bucket: Permiten controlar el acceso a nivel de bucket. Se configuran mediante JSON***
* ***y se aplican a todos los objetos del bucket.***
* ***Listas de control de acceso (ACLs): Métodos heredados para definir permisos de lectura***
* ***y escritura a nivel de objeto o bucket.***
* ***Roles y políticas de IAM: Los usuarios o servicios de AWS pueden acceder a los recursos de S3***
* ***según las políticas asociadas a ellos.***
* ***Acceso público: S3 permite deshabilitar por completo el acceso público a los buckets***
* ***para evitar exposición accidental.***
* ***Encriptación: Protege los datos en tránsito (HTTPS) y en reposo (SSE-S3, SSE-KMS, o SSE-C).***

***2. Configuración de permisos y políticas***

***Prácticas recomendadas:***

1. ***Principio de menor privilegio: Otorgar solo los permisos necesarios.***
2. ***Deshabilitar el acceso público: Configuración global para prevenir exposición no deseada.***
3. ***Auditoría de acceso: Usar CloudTrail o habilitar el registro de eventos de acceso al bucket.***

***Tipos de políticas de acceso:***

1. ***Política de bucket:***
   * ***Controla quién puede acceder al bucket y qué operaciones están permitidas.***
2. ***Política de IAM:***
   * ***Define permisos a nivel de usuario, grupo o rol.***
3. ***Política basada en objetos: Define permisos específicos para un objeto.***

***Ejemplo Resuelto***

***Objetivo: Configurar un bucket con acceso restringido para un grupo de usuarios específicos.***

1. ***Crear un bucket:***

***aws s3api create-bucket --bucket bucket-seguro --region us-east-1***

1. ***Deshabilitar el acceso público al bucket:***

***aws s3api put-public-access-block --bucket bucket-seguro --public-access-block-configuration BlockPublicAcls=true IgnorePublicAcls=true BlockPublicPolicy=true RestrictPublicBuckets=true***

1. ***Crear una política de bucket para permitir acceso a un grupo de usuarios de IAM: Política JSON (politica-bucket.json):***

***{***

***"Version": "2012-10-17",***

***"Statement": [***

***{***

***"Effect": "Allow",***

***"Principal": { "AWS": "arn:aws:iam::123456789012:role/GrupoPermitido" },***

***"Action": "s3:\*",***

***"Resource": [***

***"arn:aws:s3:::bucket-seguro",***

***"arn:aws:s3:::bucket-seguro/\*"***

***]***

***}***

***]***

***}***

***aws s3api put-bucket-policy --bucket bucket-seguro --policy file://politica-bucket.json***

1. ***Probar acceso con un usuario o rol autorizado:***

***aws s3 ls s3://bucket-seguro --profile usuario-autorizado***

***Ejemplo Propuesto***

***Objetivo:***

***Configurar un bucket con:***

* ***Acceso público deshabilitado.***
* ***Permisos para que solo un usuario específico de IAM pueda subir objetos.***
* ***Restringir la eliminación de objetos a otro rol específico.***

***Solución:***

1. ***Crear el bucket:***

***aws s3api create-bucket --bucket bucket-restringido --region us-east-1***

1. ***Deshabilitar acceso público:***

***aws s3api put-public-access-block --bucket bucket-restringido --public-access-block-configuration BlockPublicAcls=true IgnorePublicAcls=true BlockPublicPolicy=true RestrictPublicBuckets=true***

1. ***Configurar política de bucket: Política JSON (politica-restringida.json):***

***{***

***"Version": "2012-10-17",***

***"Statement": [***

***{***

***"Effect": "Allow",***

***"Principal": { "AWS": "arn:aws:iam::123456789012:user/UsuarioSubida" },***

***"Action": "s3:PutObject",***

***"Resource": "arn:aws:s3:::bucket-restringido/\*"***

***},***

***{***

***"Effect": "Deny",***

***"Principal": "\*",***

***"Action": "s3:DeleteObject",***

***"Resource": "arn:aws:s3:::bucket-restringido/\*",***

***"Condition": {***

***"StringNotEquals": {***

***"aws:PrincipalArn": "arn:aws:iam::123456789012:role/RolAdministrador"***

***}***

***}***

***}***

***]***

***}***

***aws s3api put-bucket-policy --bucket bucket-restringido --policy file://politica-restringida.json***

1. ***Probar con el usuario autorizado para subir archivos:***

***aws s3 cp archivo.txt s3://bucket-restringido/ --profile UsuarioSubida***

1. ***Intentar eliminar un objeto como otro usuario:***

***aws s3 rm s3://bucket-restringido/archivo.txt --profile OtroUsuario***

***(Debe fallar a menos que sea el rol "RolAdministrador").***

***3.. Integración de S3 con otros servicios AWS:***

***Amazon S3 (Simple Storage Service) es un servicio de almacenamiento en la nube que permite guardar grandes cantidades de datos de manera segura y escalable. Al integrarlo con otros servicios AWS, como Lambda, Redshift o Glue, se crean soluciones más robustas para procesamiento de datos, análisis y automatización.***

***Servicios Integrados con S3:***

1. ***AWS Lambda:***
   * ***Utiliza S3 como fuente o destino para el procesamiento de datos sin servidores.***
   * ***Ejemplo: S3 trigger para ejecutar funciones Lambda que procesan imágenes subidas a S3.***
2. ***AWS Redshift:***
   * ***Utiliza S3 como almacenamiento externo para cargar datos y realizar análisis avanzados.***
   * ***Ejemplo: Carga de datos desde S3 a Redshift para consultas analíticas rápidas.***
3. ***AWS Glue:***
   * ***Usa S3 como fuente para extractar, transformar y cargar (ETL) datos.***

***Ejemplo: S3 como punto de inicio para cargar datos a un catálogo de Glue y ejecutar trasformaciones.***

***Ejemplo Resuelto:***

***Integración S3 y AWS Lambda:***

* ***Objetivo: Procesar imágenes subidas a S3 automáticamente.***

1. ***Configurar un bucket S3 para almacenar imágenes.***
2. ***Crear una función Lambda que se activa cada vez que se sube una imagen a S3.***
3. ***Implementar lógica en Lambda para realizar procesamiento, como redimensionamiento o análisis.***

***Ejemplo Propuesto:***

***Integración S3 y AWS Redshift:***

* ***Cargar datos desde un bucket S3 a Redshift para análisis avanzado.***
* ***Paso 1: Configurar un bucket S3 con archivos CSV.***
* ***Paso 2: Usar AWS Glue para preparar y transformar los datos.***
* ***Paso 3: Cargar los datos procesados en Redshift para consultas.***

Amazon S3 es una solución de almacenamiento en la nube confiable y escalable para datos no estructurados. Proporciona almacenamiento seguro con acceso a través de diversas APIs y permite almacenamiento a largo plazo o procesamiento de datos en tiempo real. Ideal para almacenar grandes volúmenes de datos de manera eficiente y económica.