



Innovación con datos en la nube

*Oracle Autonomous JSON
Database - Guía para
laboratorio Hands-On*

Johanna Gutiérrez
Diego Bueno
Junio 2021



Este trabajo está sujeto a una licencia Creative Commons Atribuição-Compartilhamento 4.0 Internacional. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/4.0/>.

Guía para Laboratorio *Hands-On*

Introducción.....	4
Lab 1. Accediendo a su cuenta de Oracle Cloud.....	5
Lab 2. Creando una instancia de Oracle Autonomous JSON Database.....	9
Lab 3. Explorando Oracle Autonomous JSON Database	14
Resumen de Oracle Autonomous JSON Database	14
Consola del servicio Oracle Autonomous JSON Database	19
Lab 4. Inicia tu propia aventura con Oracle Autonomous JSON.....	25
Conéctese a su base de datos usando SQL Developer Web.....	26
Cree un usuario para trabajar con JSON en la base de datos Oracle	28
Cargue un archivo JSON y trabaje con tablas relacionales	30
Cargue un archivo JSON y trabaje con documentos y colecciones JSON	49
Exposición de los datos para aplicaciones	59

Introducción

En este laboratorio práctico, trabajaremos en la creación de una instancia de *Oracle Autonomous JSON Database* siguiendo las mejores prácticas sobre *Oracle Cloud Infrastructure*.

Una vez creada la instancia de *Oracle Autonomous JSON Database*, este taller proporcionará un tutorial sobre cómo la base de datos Autónoma de Oracle no solo puede almacenar, indexar y tener consistencia transaccional (*ACID*) con documentos *JSON*, sino cómo puede aprovechar todo el poder de la base de datos para seguridad avanzada, desarrollo de aplicaciones y un conjunto completo de API de acceso simple a documentos de Oracle (*SODA*).

Estos son los objetivos que queremos cumplir en este *Hands-On*:

- Cargar un archivo *JSON* en la base de datos *Oracle*
- Trabajar con *JSON* en la base de datos *Oracle* con tablas relacionales
- Trabajar con *JSON* en la base de datos *Oracle* como documentos *JSON*
- Proporcionar puntos clave para el desarrollo de aplicaciones sin esquemas o relaciones

Estas son las herramientas de Desarrollo que proporciona la *Oracle Autonomous JSON Database*:

- *Database Actions*: Cargue, explore, transforme, modele y catalogue sus datos. Use una hoja de trabajo *SQL*, cree interfaces *REST* y aplicaciones de bajo código, administre usuarios y conexiones, cree y aplique modelos de aprendizaje automático. Esto a través de la versión web de *SQL Developer* de *Oracle*.
- *Oracle Application Express (APEX)*: es una plataforma de desarrollo de código bajo que le permite crear aplicaciones empresariales escalables y seguras con características de clase mundial que se pueden implementar en cualquier lugar.
- *Oracle Machine Learning User Administration*: es una interfaz de cuaderno *SQL* para que los científicos de datos realicen el aprendizaje automático en la base de datos autónoma, basado en *Apache Zeppelin*.
- *SODA (Simple Oracle Document Access) Drivers*: es un conjunto de APIs que le permiten trabajar con documentos *JSON* administrados por Oracle Database sin necesidad de utilizar *SQL*. Los controladores SODA están disponibles para *REST*, *Java*, *Node.js*, *Python*, *PL / SQL* y *C*.

Es importante que los conceptos fundamentales de estas funcionalidades sean claros para una buena experiencia en nuestra nube.



Lab 1.

Accediendo a su cuenta de *Oracle Cloud*

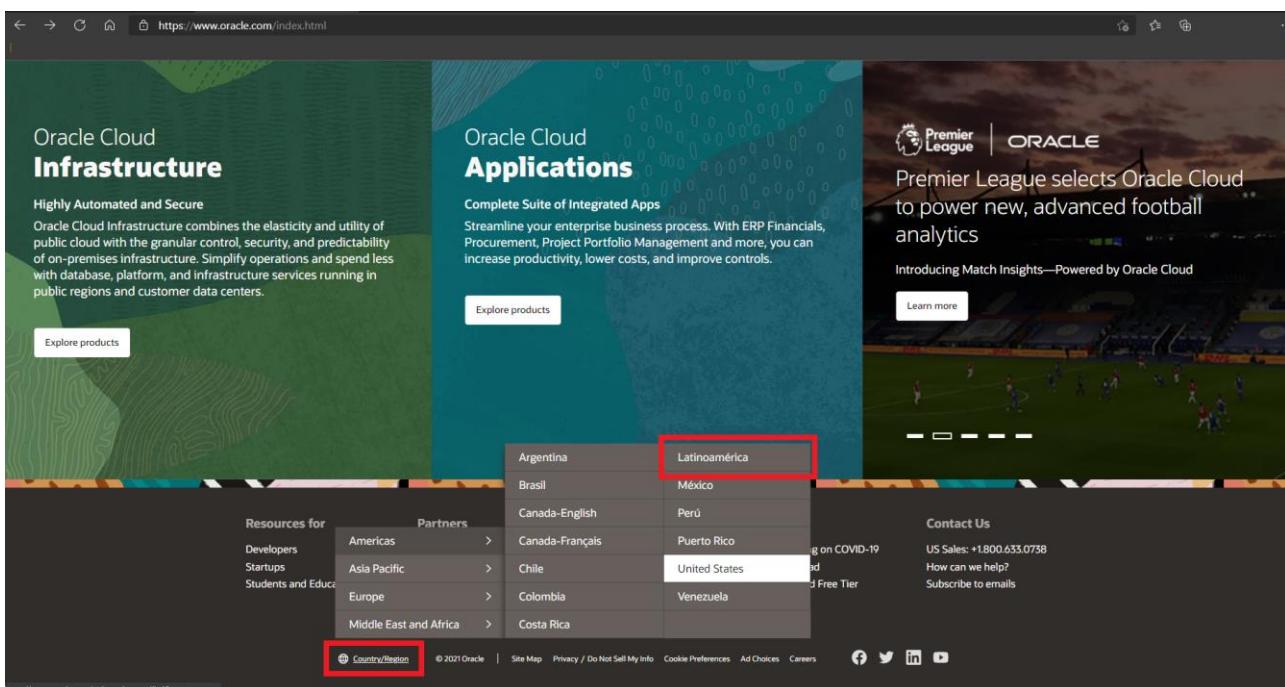
Lab 1. Accediendo a su cuenta de Oracle Cloud

Objetivos

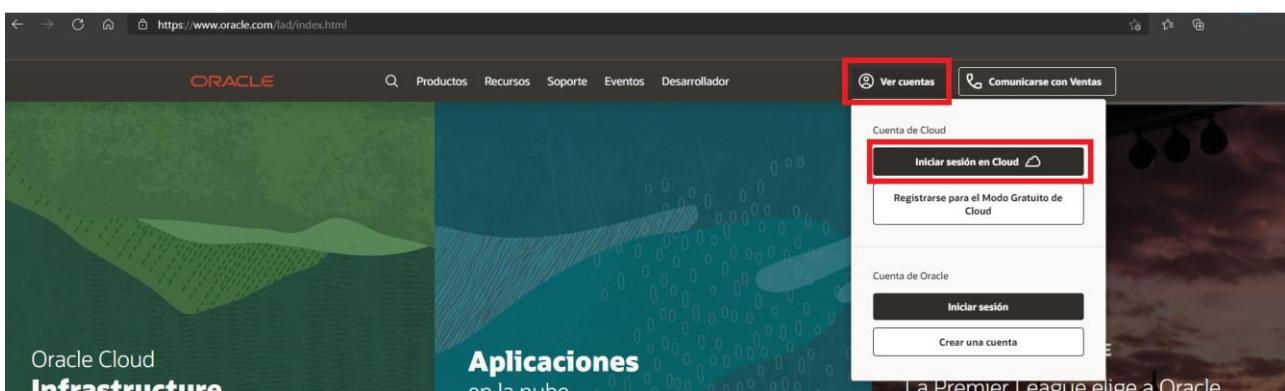
- Acceder a la consola de *Oracle Cloud*
- Conocer los servicios de infraestructura y plataforma
- Familiarizarse con el ambiente de *Oracle Cloud*

En esta sección aprenderá más sobre el acceso inicial al ambiente de *Oracle Cloud*.

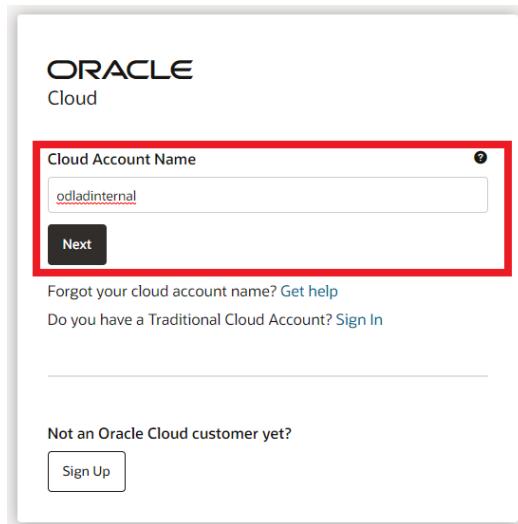
Vaya a la dirección www.oracle.com/cloud/. Usted puede cambiar la región y el idioma de esta página antes de acceder al entorno:



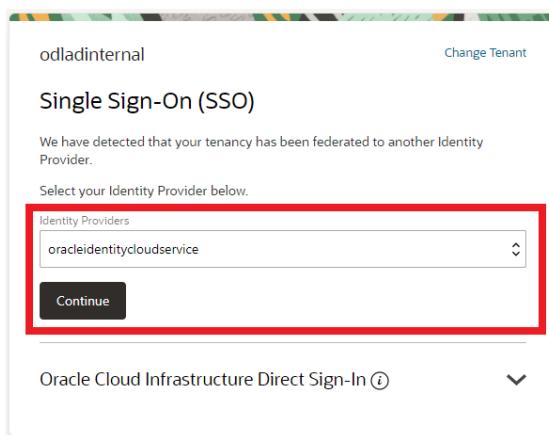
Una vez seleccionando la región, para este caso Latinoamérica, hacer clic en **Ver cuentas** y luego en **Iniciar sesión en Cloud**:



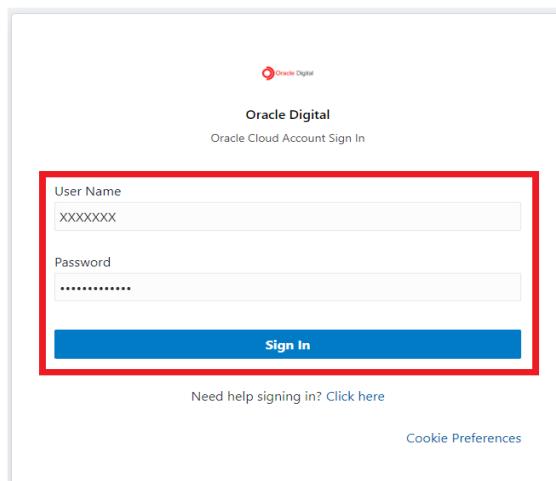
Para efectuar el **login**, se debe hacer con la opción de **Cloud Account Name** y en el cuadro colocar el nombre de la **cuenta Cloud** y luego hacer clic en **Next**:



Posterior a esto, asegurarse de seleccionar en el campo de **Identity Providers**, la opción de **oracleidentitycloudservice** y hacer clic en **Continue**:



Luego, colocaremos nuestro usuario y contraseña en cada campo para acceder a la cuenta y hacer clic en **Sign In**:



Se muestra la pantalla principal de su entorno. En él se pueden ver algunas acciones rápidas para crear algunos recursos, algunos artículos sobre soluciones dentro de la nube de Oracle que pueden ayudar, la parte de aprender que conduce a la documentación, la cual es muy detallada.

En la barra superior está el menú que enumera todas las pestañas de la consola en la nube, la lupa para búsquedas en el entorno de la nube, información sobre qué región se está viendo, en el caso de la imagen de abajo, se muestra la región de Ashburn. El siguiente cuadro al lado de la región es para acceder al *Cloud Shell* de la consola. En la campana es donde se hacen los anuncios relacionados con la nube, el signo de interrogación es donde hay algunos temas de ayuda y también donde puede contactar al soporte o abrir un ticket para aumentar los límites del servicio en la nube. En el mundo es en donde el usuario puede cambiar el idioma de la consola en la nube y finalmente en el símbolo de usuario puede encontrar su información.

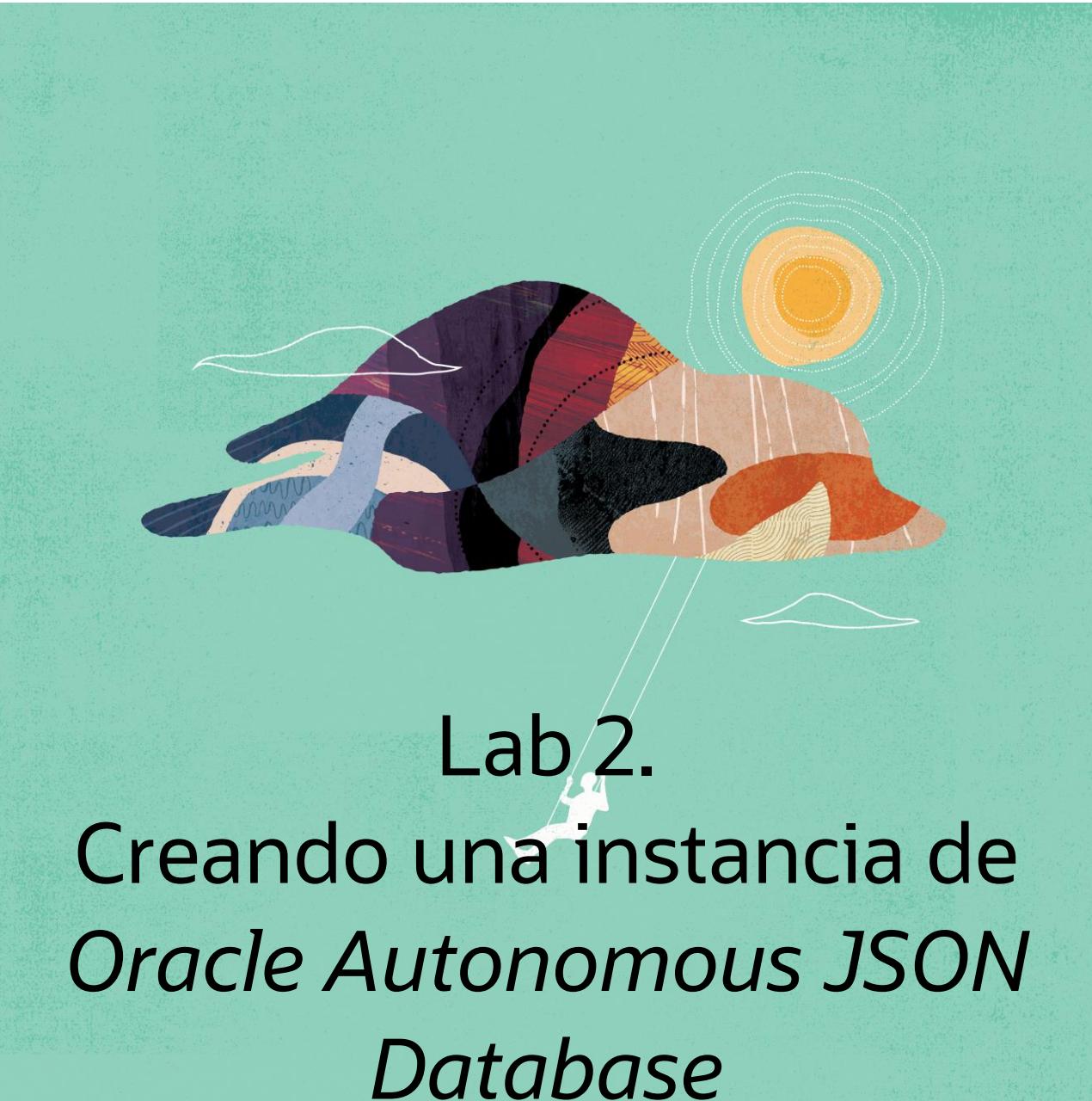
The screenshot shows the Oracle Cloud Home page with several key sections highlighted:

- Quick Actions:** A grid of cards for various services:
 - COMPUTE: Create a VM instance (2-4 mins)
 - AUTONOMOUS TRANSACTION PROCESSING: Create an ATP database (3.5 mins)
 - AUTONOMOUS DATA WAREHOUSE: Create an ADW database (3-5 mins)
 - NETWORKING: Set up a network with a wizard (2-3 mins)
 - RESOURCE MANAGER: Create a stack (2-6 mins)
 - OBJECT STORAGE: Store data (2-6 mins)
 - NETWORKING: Set up a load balancer (5 mins)
 - ORACLE CLOUD DEVELOPMENT KIT: Set up an instance with developer tools (10-15 mins)
 - SEARCH: View all my resources
- Start Exploring:** A section with links to:
 - Get Started: Deploy Websites & Apps, Explore Developer Tools, Manage Bills
 - Key Concepts and Terminology DOCUMENTATION: To get started with Oracle Cloud Infrastructure, familiarize yourself with some key concepts and terminology.
 - Introduction to APEX BLOG: Oracle Application Express (APEX) is a low-code development framework that enables you to rapidly build modern, data-driven applications for your browser - no external tools required. See how you can use APEX to develop and deploy compelling low-code apps in minutes.
 - Get Started with FREE training from Oracle University DOCUMENTATION: Introduction to Resource Manager DOCUMENTATION: Oracle Database 19c Performance Tuning Guide
- Account Center:** A sidebar with:
 - User Management: Add a user to your tenancy
 - Billing: Analyze costs, Manage payment method
 - What's New: Database Migration is now available in all commercial regions (May 18, 2021), Support for GPU shapes (May 26, 2021), Store Terraform configurations in Object Storage buckets (May 18, 2021), Data Flow Supports Spark-Submit (May 18, 2021), MySQL Database Server direct data import (May 18, 2021), View release notes...
 - Get Help: Contact Support, Developer Tools, Documentation, Oracle Cloud Community Forum, Oracle Cloud Conference

Por último y no menos importante, en la parte superior izquierda en la tres líneas horizontales o en el menú “hamburguesa”, al hacer clic se despliega el menú principal con todos los diferentes productos y opciones a los que puede acceder:

The screenshot shows the Oracle Cloud Home page with the main navigation menu (the "hamburger" icon) highlighted. The menu includes the following categories:

- Home
- Compute
- Storage
- Networking
- Oracle Database
- Databases
- Analytics & AI
- Developer Services
- Identity & Security
- Observability & Management
- Hybrid
- Migration
- Governance & Administration
- Marketplace
- OCI Classic Services



Lab 2.

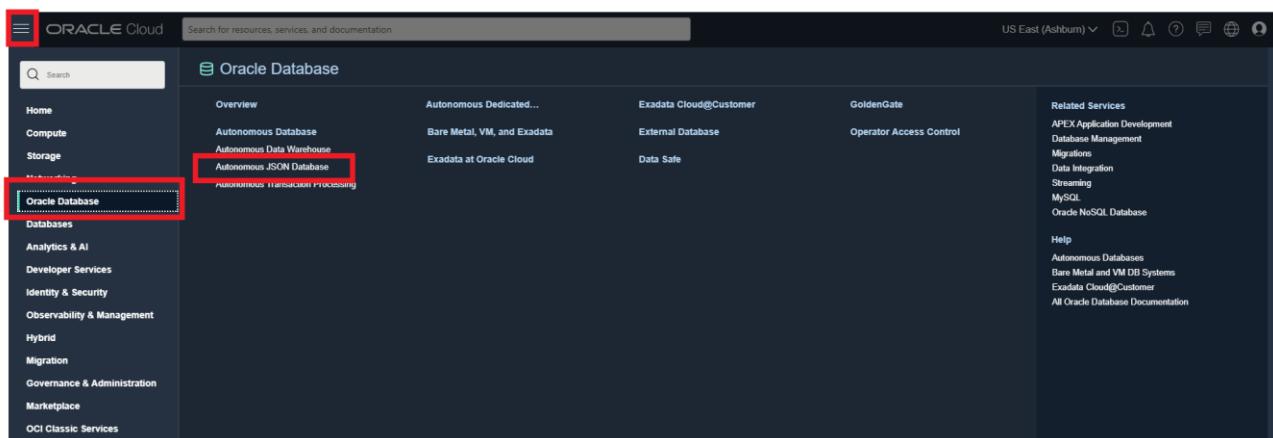
Creando una instancia de *Oracle Autonomous JSON Database*

Lab 2. Creando una instancia de Oracle Autonomous JSON Database

Objetivos

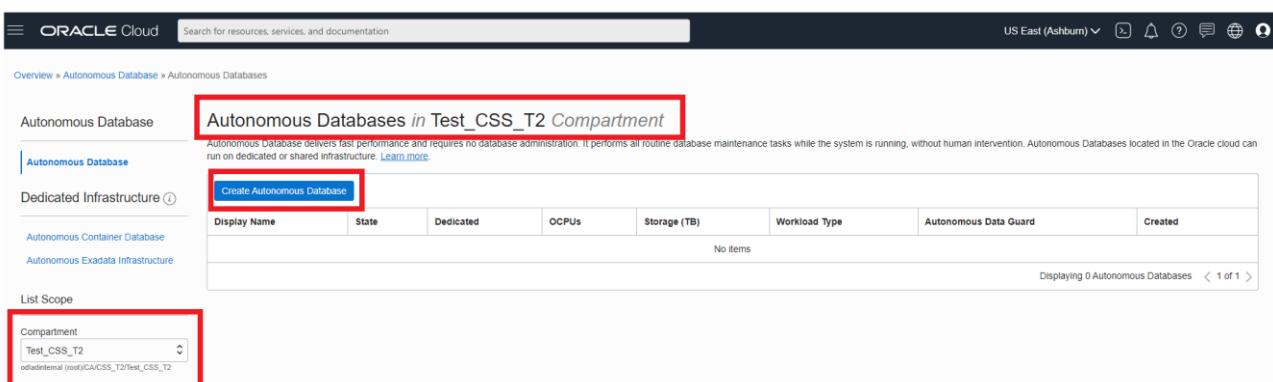
- Aprovisionar una instancia de Oracle Autonomous JSON Database
- Funcionalidades y demás generalidades de la instancia creada

Después de iniciar sesión en el entorno *Cloud*, se accederá al menú “hamburguesa”, hacer clic en la opción **Oracle Database** y luego en **Autonomous JSON Database**:



The screenshot shows the Oracle Cloud interface. On the left, there's a navigation sidebar with various service links. The 'Oracle Database' link is highlighted with a red box. Under it, the 'Autonomous JSON Database' link is also highlighted with a red box. The main content area is titled 'Oracle Database' and contains several tabs: Overview, Autonomous Dedicated..., Bare Metal, VM, and Exadata, Exadata Cloud@Customer, Operator Access Control, External Database, Data Safe, and Autonomous Transaction Processing. To the right, there are sections for 'Related Services' (APEX Application Development, Database Management, Migrations, Data Integration, Streaming, MySQL, Oracle NoSQL Database) and 'Help' (Autonomous Databases, Bare Metal and VM DB Systems, Exadata Cloud@Customer, All Oracle Database Documentation).

Una vez seleccionada la opción anterior, aparecerá la siguiente ventana, en ella en parte inferior izquierda vamos a seleccionar el *Compartiment* sobre el cual vamos a crear la instancia. Para este caso, selecciono el *Compartiment Test_CSS_T2* para luego hacer clic en **Create Autonomous Database**:



The screenshot shows the 'Autonomous Database' creation page. At the top, it says 'Autonomous Databases in Test_CSS_T2 Compartiment'. Below that, there's a brief description of Autonomous Database. In the center, there's a table with columns: Display Name, State, Dedicated, OCPUs, Storage (TB), Workload Type, Autonomous Data Guard, and Created. The table is currently empty. At the bottom of the page, there's a 'List Scope' section where 'Compartiment' is set to 'Test_CSS_T2'. The 'Create Autonomous Database' button is highlighted with a red box.

No se recomienda crear ningún recurso en el compartimento raíz. Si requiere crear un compartimento, solo vaya al menú, *Identity & Security* y *Compartments*. Puede ver más detalles en el siguiente enlace: <https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingcompartments.htm>

Una vez hicimos clic en **Create Autonomous Database**, procedemos a ejecutar los siguientes pasos:

- ❖ Nos aseguramos de validar que el **Compartment** corresponde al mismo que seleccionamos previamente.
- ❖ Llenar el campo **Display name**.
- ❖ Llenar el campo **Database name**.
- ❖ Seleccionamos **JSON** en el **workload type**.
- ❖ Seleccionamos **Shared Infrastructure** en el **deployment type**.

Create Autonomous Database

Provide basic information for the Autonomous Database

Compartment
Test_CSS_T2

Display name
CE-DABH-AJSON

Database name
DABHAJSON

Choose a workload type

Data Warehouse Built for decision support and data warehouse workloads. Fast queries over large volumes of data.	Transaction Processing Built for transactional workloads. High concurrency for short-running queries and transactions.	JSON Built for JSON-centric application development. Developer-friendly document APIs and native JSON storage.	APEX Built for Oracle APEX application development. Creation and deployment of low-code applications, with database included.
---	---	--	--

Choose a deployment type

Shared Infrastructure Run Autonomous Database on shared Exadata infrastructure.	Dedicated Infrastructure Run Autonomous Database on dedicated Exadata infrastructure.
---	--

Dedicated Exadata infrastructure is not available for Oracle Autonomous JSON Database.

- ❖ Vemos la versión de la base de datos, la cual por ahora solo puede ser 19c.
- ❖ Colocamos el número de **OCPUs** (capacidad de cómputo), que deber ser mínimo 1.
- ❖ Colocamos la cantidad de **Storage** (almacenamiento) requerido en *Terabytes* y deber ser mínimo 1.
- ❖ Podemos seleccionar la opción de **Auto scaling** si así lo queremos, para este ejercicio, no la vamos a seleccionar.

Configure the database

Always Free ⓘ
 Show only Always Free configuration options

Choose database version
19c

OCPUs count
1

The number of OCPUs cores to enable. Available cores are subject to your tenancy's service limits.

Storage (TB)
1

The amount of storage to allocate.

Auto scaling
Allows system to use up to three times the provisioned number of cores as the workload increases. [Learn more](#)

- ❖ Posteriormente en la opción de **Create administrator credentials**, el usuario por defecto será **ADMIN**.
- ❖ Procedemos a colocar una clave y confirmarla.

Create administrator credentials ⓘ

Username *Read-Only*

ADMIN

ADMIN username cannot be edited.

Password

.....

Confirm password

.....

- ❖ Seguido a esto, seleccionamos el tipo de acceso de red en el **Access Type**, para este caso vamos a seleccionar **Secure Access from everywhere**.
- ❖ Luego seleccionamos el **license type**, que para el caso de la **Oracle Autonomous JSON Database**, siempre será **License included**.
- ❖ También es posible colocar contactos que sean notificados por correo para cualquier tipo de mantenimiento que se le haga a la instancia o para cuando la instancia termine de crearse, llegue una notificación al correo. En este caso voy a colocar mi correo.

Choose network access

Access Type

Secure access from everywhere

Restrict access to specified IP addresses and VCNs.

Configure access control rules ⓘ

Private endpoint access only

Restrict access to a private endpoint within an OCI VCN.

Choose a license type

Bring Your Own License (BYOL)

Bring my organization's Oracle Database software licenses to the Database service.

[Learn more](#)

License Included

Subscribe to new Oracle Database software licenses and the Database service.

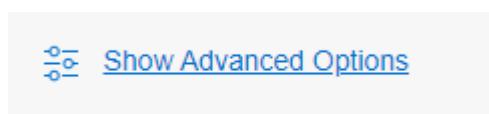
Provide up to 10 maintenance contacts

Contact Email

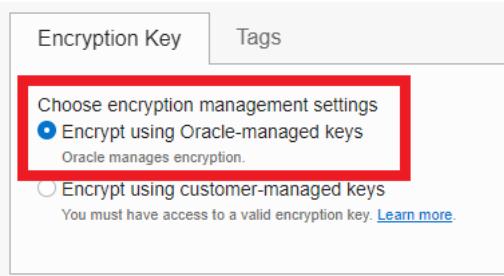
diego.bueno@oracle.com

Add Contact

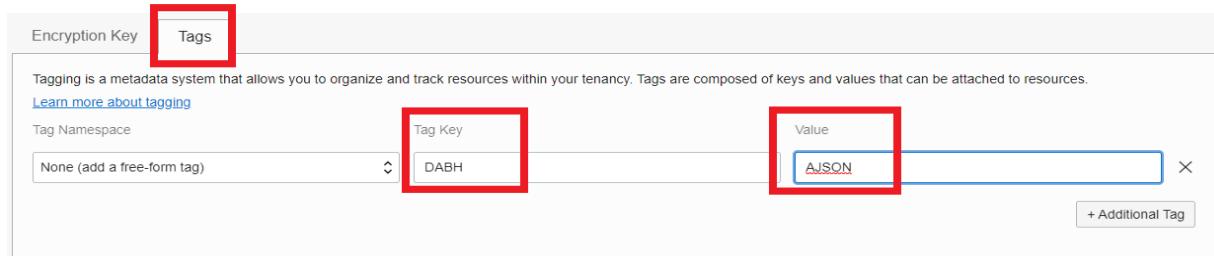
- ❖ Como último paso hacemos clic en **Show Advanced Options** para ver las últimas opciones disponibles antes de crear la instancia.



- ❖ Una vez allí, vamos a seleccionar la opción de **Encrypt using Oracle-managed keys**, en la sección de **Encryption Key**.



- ❖ Luego vamos a la opción de **Tags** (Etiquetado), la cual nos permite tener un control de los recursos que se crean. Para este caso yo voy a colocar dos etiquetas.

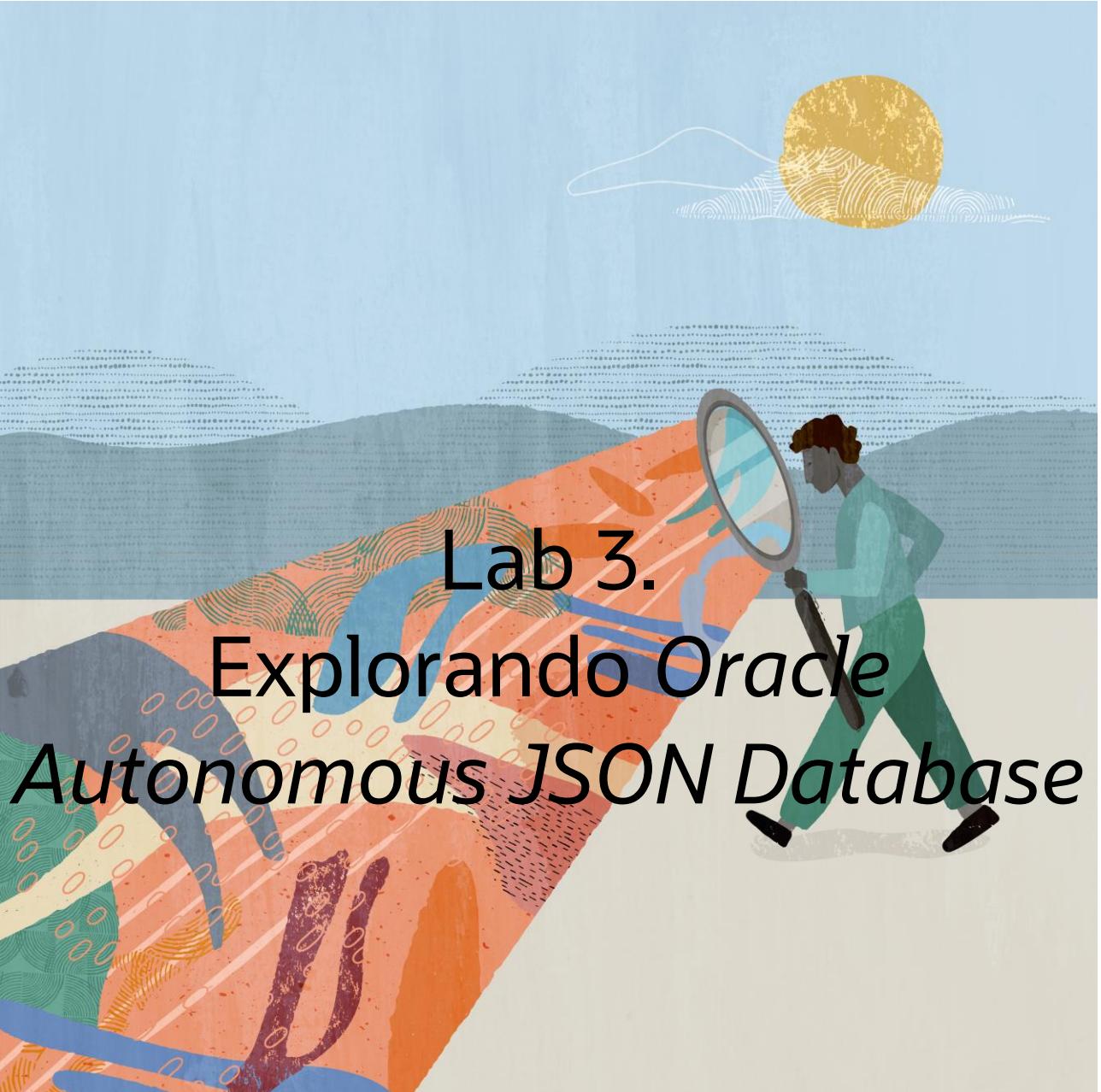


- ❖ Ya que tenemos todas las opciones diligenciadas y seleccionadas, hacemos clic en **Create Autonomous Database**.



El proceso de creación de la base de datos autónoma de JSON tomará alrededor de 3 minutos en crearse. Cuando la instancia ya está creada, saldrá en verde como *Available* y podremos empezar a utilizarla.

General Information	Infrastructure
Database Name: DABH-AJSON Workload Type: JSON Database Compartment: odiadinternal (root)/CA/CSS_T2/Test_CSS_T2 OCID: ocid1.database.oc1..myj2pa Show Copy Created: Mon, Jun 7, 2021, 02:37:40 UTC OCPU Count: 1 Auto Scaling: Disabled Storage: 1 TB License Type: License included Database Version: 19c Lifecycle State: Available Instance Type: Paid Mode: Read/Write	Dedicated Infrastructure: No Autonomous Data Guard: Status: Disabled
APEX Instance Instance Name: CE-DABH-AJSON	Backup Last Automatic Backup: No active backups exist for this database. Manual Backup Store: Not Configured
	Network Access Type: Allow secure access from everywhere Access Control List: Disabled
	Maintenance Next Maintenance: Sun, Jun 13, 2021, 08:00:00 UTC - 12:00:00 UTC View History Customer Contacts: Configured Manage
	Data Safe



Lab 3. Explorando Oracle Autonomous JSON Database

Lab 3. Explorando *Oracle Autonomous JSON Database*

Objetivos

- Resumen de *Oracle Autonomous JSON Database*
- Consola del servicio *Oracle Autonomous JSON Database*

Resumen de *Oracle Autonomous JSON Database*

Autonomous JSON Database es una base de datos de procesamiento autónomo de transacciones de *Oracle*, pero está especializada en el desarrollo de aplicaciones de estilo *NoSQL* que utilizan documentos *JavaScript Object Notation (JSON)*. Está creada para el desarrollo de aplicaciones centradas en *JSON*. Ofrece *API* de documentos fáciles de usar y almacenamiento nativo *JSON*.

Un caso de uso común para una columna *JSON* es proporcionar algunos datos sin esquema en un contexto relacional. Una columna *JSON* puede contener datos *JSON* arbitrarios, que se pueden consultar o proyectar como datos relacionales.

Las siguientes características, en particular, respaldan el desarrollo de aplicaciones de alto rendimiento y alta seguridad:

- **Administración automática de bases de datos:** Las tareas rutinarias de administración de la base de datos, como la aplicación de parches y la realización de copias de seguridad, se realizan automáticamente, por lo que puede concentrarse en desarrollar su aplicación.
- **Afinamiento automático del rendimiento:** Dedique menos tiempo a definir y ajustar su base de datos.
- **Alto rendimiento preconfigurado:** Cuando se conecta a la base de datos con un cliente de *Oracle* utilizando grupos de conexiones, aprovecha las funciones de alto rendimiento configuradas en el lado de la base de datos de su conexión.
- **Servicios de base de datos predefinidos y específicos para cargas de trabajo:** Las aplicaciones cliente pueden conectarse a la base de datos mediante un servicio de conexión que mejor se adapte al tipo de operaciones de base de datos que necesitan. (Para la mayoría de las aplicaciones que usan documentos *JSON*, usa el servicio de conexión típico para el procesamiento de transacciones, *tp*.)

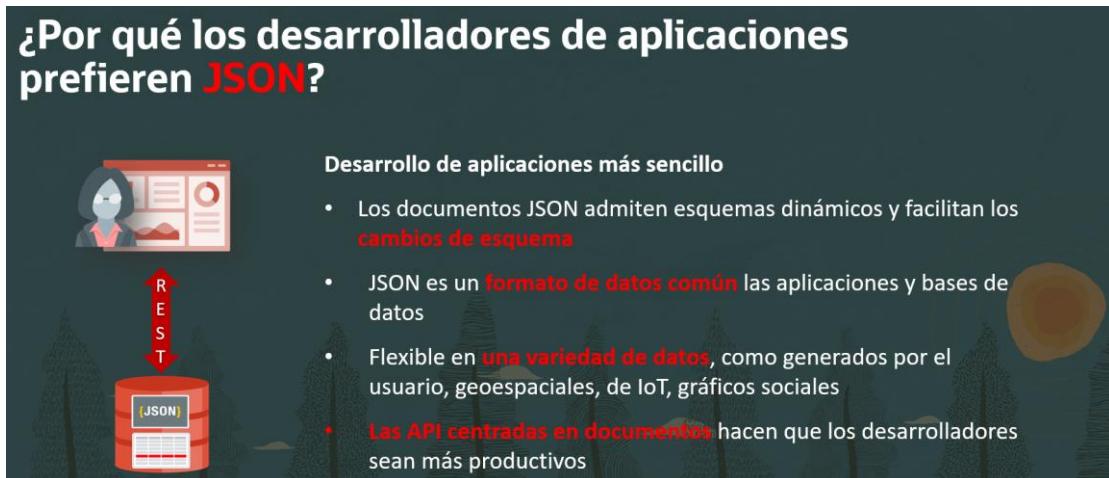
Atributos Oracle Autonomous Database



Autonomous JSON Database está especializada en el desarrollo de aplicaciones de estilo *NoSQL* que utilizan documentos *JavaScript Object Notation (JSON)* mediante *API* de acceso simple a documentos de *Oracle (SODA)*. También puede utilizar *Oracle Application Express (APEX)* para el desarrollo de cuadros de mando de bajo código sobre sus datos *JSON*.

Pero al igual que para el procesamiento autónomo de transacciones, los datos *JSON* almacenados en una base de datos *JSON* también son totalmente accesibles mediante el lenguaje de consulta estructurado (*SQL*), para análisis e interfaz con herramientas relacionales. Puede promover una base de datos *JSON* autónoma en cualquier momento a una base de datos de procesamiento de transacciones autónomo, que le permite almacenar más datos que no sean *JSON*.

¿Por qué los desarrolladores de aplicaciones prefieren *JSON*?



Autonomous JSON Database proporciona todas las mismas características que una *Autonomous Transaction Processing*, con esta importante limitación: puede almacenar solo hasta 20GB de datos que no sean colecciones de documentos *JSON*. No hay límite de almacenamiento para las colecciones *JSON*.

Oracle Autonomous JSON Database - AJD

Más que un simple almacén de documentos, la base de datos convergente de Oracle admite acceso NoSQL y SQL

- Autónomo
- Soporte completo de SQL
- Transacciones ACID
- Seguridad avanzada
- Desarrollo de código bajo APEX
- 20GB de datos no JSON. Expansión instantánea con un clic a ATP.



El desarrollo de aplicaciones centradas en documentos de estilo *NoSQL* es particularmente flexible porque las aplicaciones utilizan datos sin esquema. Esto le permite reaccionar rápidamente a los requisitos cambiantes de la aplicación. No es necesario normalizar los datos en tablas relacionales y no hay impedimento para cambiar la estructura u organización de los datos en cualquier momento y de ninguna manera. Un documento JSON tiene una estructura interna, pero no se impone ninguna relación en documentos JSON separados.

Con *Oracle Autonomous JSON Database*, sus aplicaciones JSON centradas en documentos suelen utilizar *Simple Oracle Document Access (SODA)*, que es un conjunto de *API* de estilo *NoSQL* para varios lenguajes de desarrollo de aplicaciones y para el estilo arquitectónico de transferencia de estado representacional (*REST*). Puede utilizar cualquier *API* de *SODA* para acceder a cualquier colección de *SODA*.

Las colecciones de documentos *SODA* están respaldadas por vistas y tablas de bases de datos ordinarias. Para utilizar otros tipos de datos, sujeto al límite de 20 GB, normalmente necesita algún conocimiento del lenguaje de consulta estructurado (*SQL*) y cómo se almacenan esos datos en la base de datos.

Ejemplos de Oracle SODA

Node.js

```
conn = await oracledb.getConnection(...);
db = conn.get SodaDatabase();
col = await
db.createCollection("purchase_orders");
await col.drop();
```

Python

```
conn = cx_Oracle.connect(...);
db = conn.get SodaDatabase();
col = db.createCollection("purchase_orders");
col.drop();
```

Java

```
OracleClient client = new OracleRDBMSClient();
db = client.getDatabase(jdbcConn);
OracleCollection col =
db.admin.createCollection("purchase_orders");
col.admin().drop();
```

PL/SQL (and Oracle Application Express)

```
col := dbms_soda.create_collection('purchase_orders');

select dbms_soda.drop_collection('purchase_orders')
from dual;
```

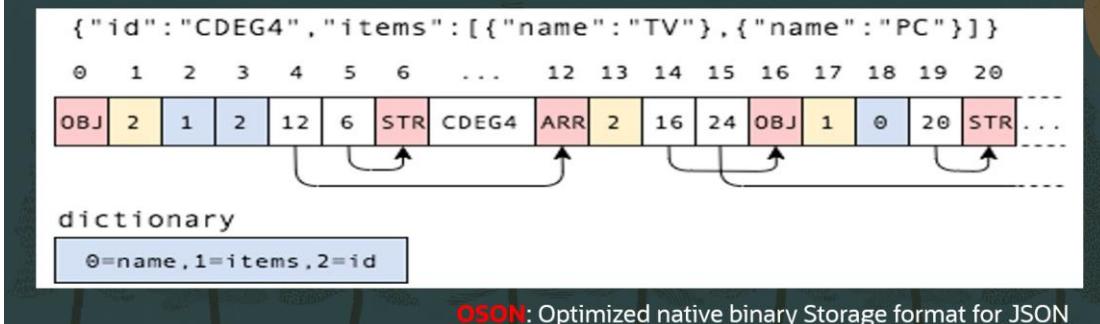
Con *Oracle Autonomous JSON Database*, una colección *SODA* solo puede contener datos *JSON*. Por ejemplo, no puede tener una colección de documentos de imagen o una colección que contenga tanto documentos *JSON* como documentos de imagen. Esta es una limitación relativa al procesamiento autónomo de transacciones, donde puede definir colecciones tan heterogéneas.

Independientemente del tipo de datos que utilicen sus aplicaciones, ya sea *JSON* o cualquier otra cosa, puede aprovechar todas las funciones de *Oracle Database*. Esto es cierto independientemente del tipo de base de datos autónoma de *Oracle* que utilice.

Los datos *JSON* se almacenan de forma nativa en la base de datos. En una colección *SODA* en una base de datos autónoma, los datos *JSON* se almacenan en el formato binario nativo de *Oracle*, *OSON*.

OSON - Formato de almacenamiento binario nativo optimizado para JSON

- Rendimiento de consulta rápido
- Actualizaciones eficientes
- Almacenamiento reducido



Los datos *JSON* binarios nativos (formato *OSON*) amplían el lenguaje *JSON* al agregar tipos escalares, como la fecha, que corresponden a los tipos de *SQL* y no forman parte del estándar *JSON*. *Oracle Database* también admite el uso de objetos *JSON* textuales que representan valores escalares *JSON*, incluidos dichos valores no estándar.

Consola del servicio Oracle Autonomous JSON Database

En la sección de creación de la instancia, vimos que esta se creó satisfactoriamente y se muestra como *Available* o Disponible.

The screenshot shows the Oracle Cloud interface for an Autonomous Database named 'CE-DABH-AJSON'. The database is marked as 'AVAILABLE'. The main page displays 'General Information' such as Database Name (DABHAJSON), Workload Type (JSON Database), Compartiment (odadinternal (root)/CA/CSS_T2/test_CSS_T2), OCID (...mvj2pa), and Creation Date (Mon, Jun 7, 2021, 02:37:40 UTC). It also lists OCPU Count (1), Auto Scaling (Disabled), Storage (1 TB), License Type (License included), Database Version (19c), Lifecycle State (Available), Instance Type (Paid), Mode (ReadWrite), and APEX Instance (Instance Name: CE-DABH-AJSON). On the right side, there are sections for Infrastructure (Dedicated Infrastructure: No), Autonomous Data Guard (Status: Disabled), Backup (Last Automatic Backup: No active backups exist for this database), Network (Access Type: Allow secure access from everywhere, Access Control List: Disabled), Maintenance (Next Maintenance: Sun, Jun 13, 2021, 06:00:00 UTC - 12:00:00 UTC), and Data Safe. The 'More Actions' dropdown menu is open, showing options like Stop, Restart, Restore, Create Clone, Update Network Access, Change Workload Type, Administrator Password, Manage Encryption Key, Rename Database, Move Resource, Add Tags, and Terminate.

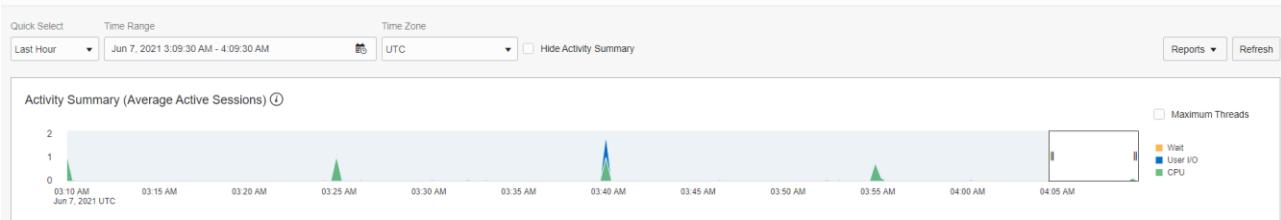
Aquí podemos ver además de la información general de la instancia, tales como el nombre de la base de datos, el tipo de carga de trabajo, el compartimento, la cantidad de OCPUs, el almacenamiento, el tipo de licencia, entre otros, están las siguientes opciones, las cuales permiten hacer lo que se describe a continuación:

- **DB Connection:** Muestra una ventana con las cadenas de conexión (alta, media, baja) y permite al usuario descargar las credenciales del cliente para conectarse posteriormente.

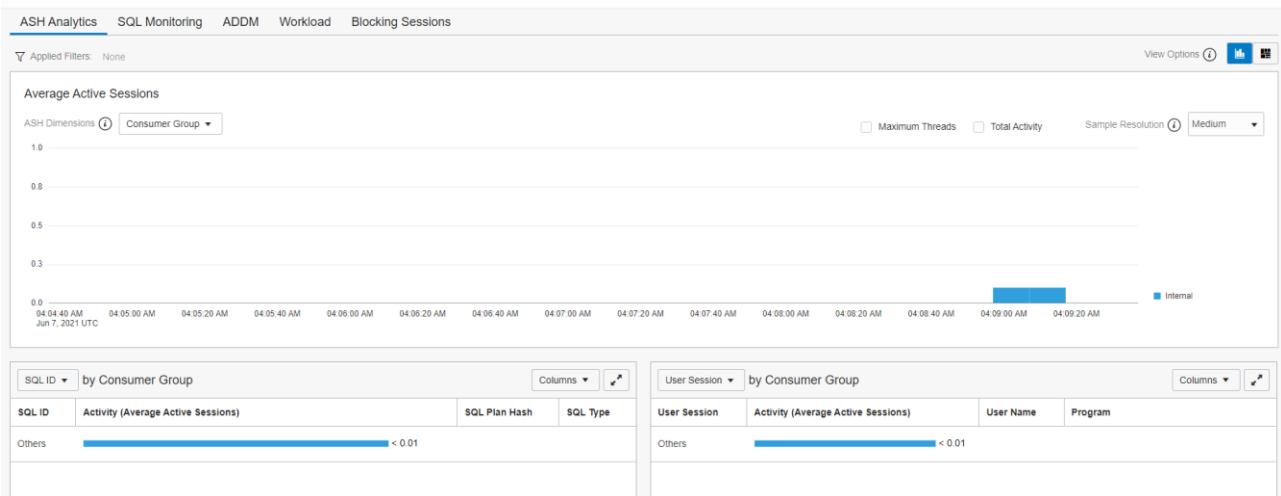
This screenshot shows the 'Database Connection' section of the Oracle Cloud interface. It starts with a message: 'You will need the client credentials and connection information to connect to your database. The client credentials include the wallet.' Below this is a 'Download Client Credentials (Wallet)' section. It contains a note: 'To download your client credentials, select the type of wallet, then click **Download Wallet**. You will be asked to create a password for the wallet.' A dropdown menu for 'Wallet Type' is open, showing 'Instance Wallet'. At the bottom of this section are two buttons: 'Download Wallet' and 'Rotate Wallet'. A note at the bottom states: 'Wallet last rotated: -'.

- **Performance Hub:** Abre una pestaña que monitorea el desempeño de la base de datos a diferentes niveles y proporcionando información detallada del estado de esta.

Performance Hub - CE-DABH-AJSON



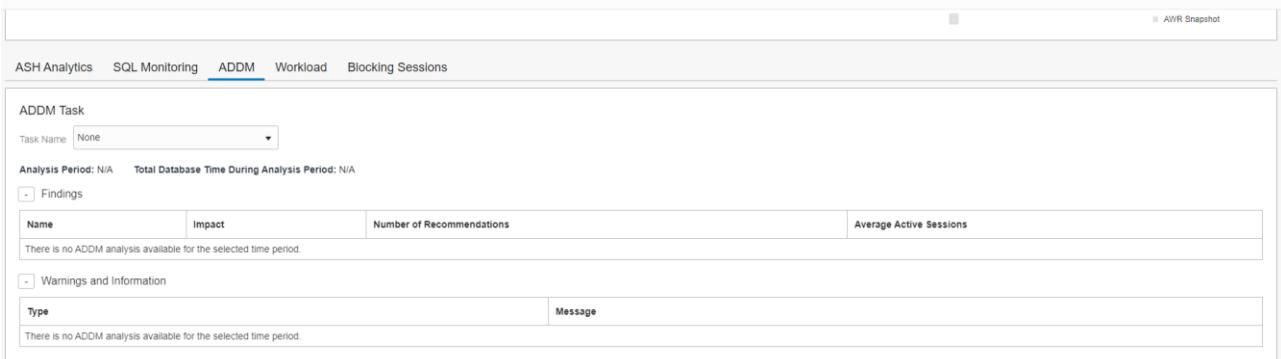
Performance Hub - CE-DABH-AJSON



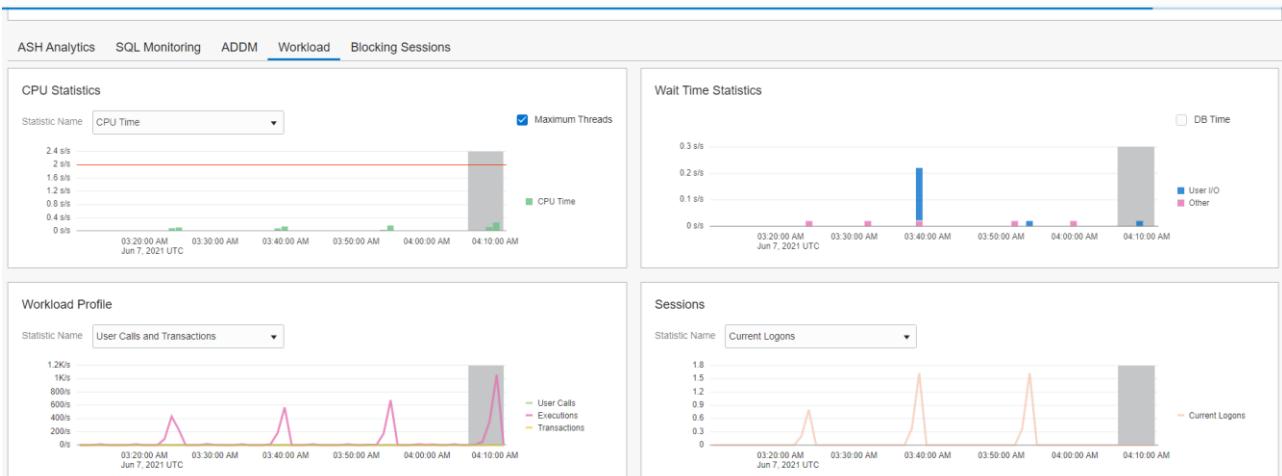
Performance Hub - CE-DABH-AJSON



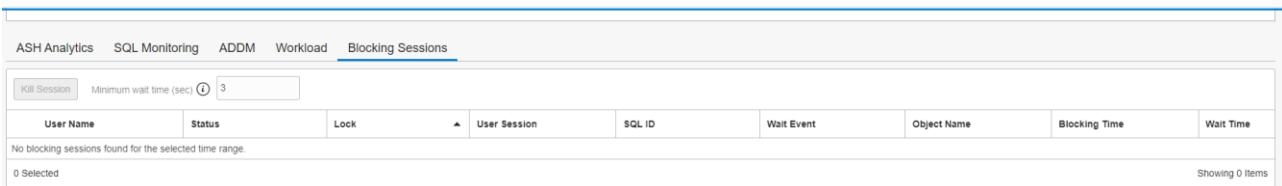
Performance Hub - CE-DABH-AJSON



Performance Hub - CE-DABH-AJSON

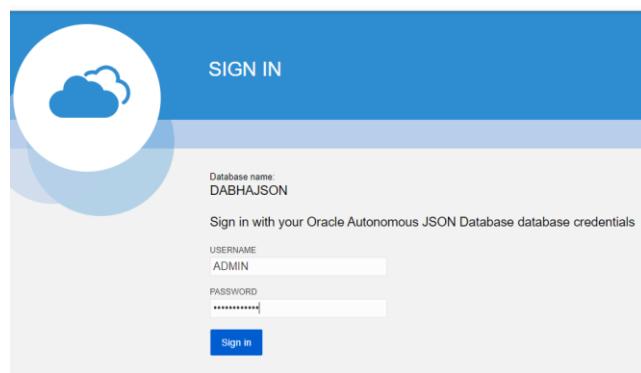


Performance Hub - CE-DABH-AJSON



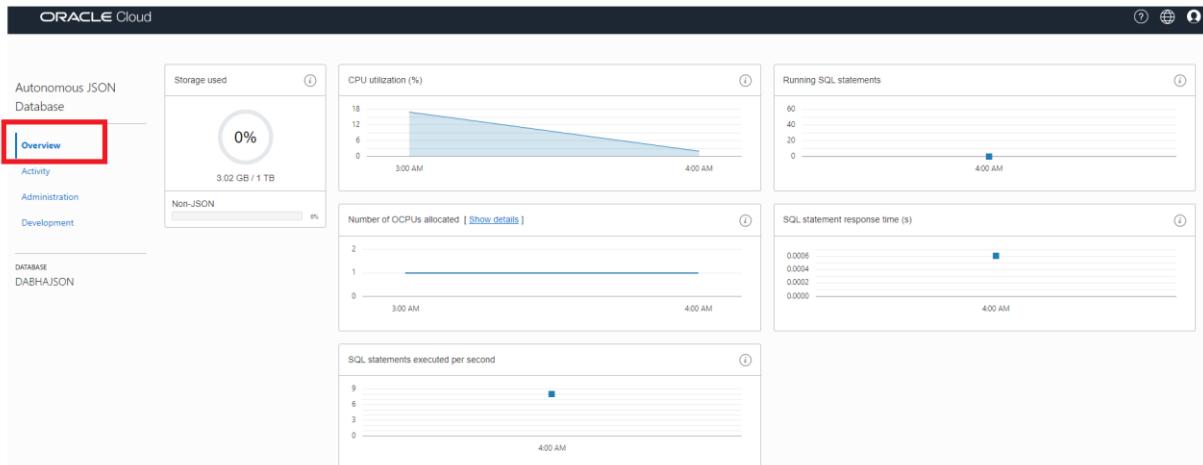
- **Service Console:** Lleva al usuario a la consola de servicios de la base de datos.

Una vez se hace clic sobre **Service Console**, automáticamente nos llevará a una ventana de logueo, en la cual usaremos el usuario **ADMIN** (usuario por defecto) y la clave que asignamos cuando se hizo la creación de la instancia.

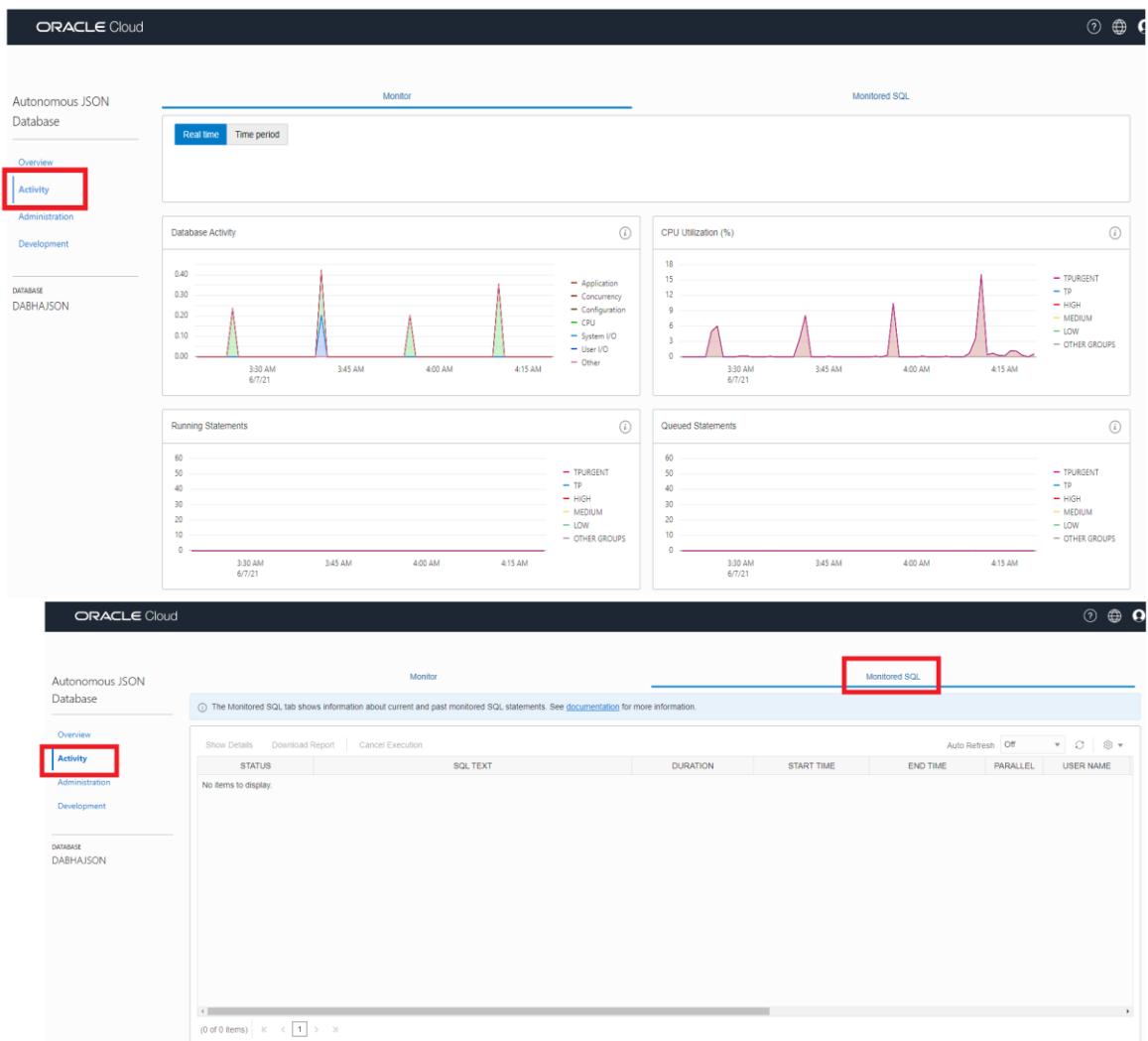


Tan pronto nos logueamos, aparecerá la siguiente ventana con cuatro opciones al lado izquierdo:

Overview: En él, el usuario verifica cómo se está utilizando Autonomous en potencia y almacenamiento computacional, cuánta OCPU se asignan y cómo su rendimiento está ejecutando las consultas.



Activity: en esta opción el usuario monitorea las actividades de Autonomous en tiempo real y puede alcanzar el nivel de detalle de la consulta al acceder a la pestaña "SQL monitoreado" que se muestra en la imagen.



Administration: En esta parte de la consola el usuario puede descargar las credenciales de acceso, de la misma manera que se mencionó anteriormente en este tutorial, el administrador de la base de datos puede restablecer las reglas de los recursos dedicados para cada tipo de conexión o devolverlas a las predeterminadas. *Oracle Autonomous JSON Database* tiene tres formas de conectarse, estas son:

- “*High*”: Conexión que permite el paralelismo de consultas, tiene altos recursos dedicados a esta conexión y baja concurrencia.
- “*Medium*”: Conexión que permite el paralelismo de consultas, menos recursos dedicados y más concurrencia entre procesamientos.
- “*Low*”: Conexión que ejecuta consultas en serie, luego utiliza el paralelismo de consultas existente en la base de datos, pocos recursos asignados y alta concurrencia entre procesos.

The screenshot shows the Oracle Cloud interface for Autonomous JSON Database. On the left, there's a sidebar with navigation links: Overview, Activity, Administration (which is selected), Development, DATABASE, and DABHAJSON. The main area contains several cards with different management options:

- Download Client Credentials (Wallet)**: Instructions for connecting to the database using a wallet file.
- Set Resource Management Rules**: A card for managing rules to allocate CPU/I/O shares to consumer groups.
- Set Administrator Password**: A card for changing the password for the database administrator user.
- Manage Oracle ML Users**: A card for creating and managing Oracle Machine Learning users.
- Send Feedback to Oracle**: A card for providing feedback through the Oracle Customer Connect forum.

También en esta consola, el usuario puede cambiar la contraseña de inicio de sesión **ADMIN** de *Autonomous*, crear usuarios de *Oracle Machine Learning*, que es una herramienta que permite a los usuarios ejecutar *scripts* de *Machine Learning* en SQL o PL/SQL dentro de *Autonomous* utilizando su *Machine Learning*, está en desarrollo la ejecución de scripts en R y Python, al que se accede a través de la interfaz web y está basado en *Apache Zeppelin*.

Y finalmente, el usuario puede enviar comentarios a *Oracle*, a través de ese enlace, será redirigido a un foro donde no solo podrá enviar comentarios sino también compartir sus dudas e historias de éxito.

Development: En esta parte de la consola el usuario puede acceder a las herramientas de desarrollo en *Autonomous*, todas tipo web y son:

- Descargar *Oracle Instant Client*: *Oracle Instant Client* permite que las aplicaciones se conecten a una base de datos *Oracle* local o remota para el desarrollo y la implementación de la producción. Las bibliotecas *Instant Client* brindan la conectividad de red necesaria, así como capacidades de datos base y de alto nivel, para aprovechar al máximo la base de datos *Oracle*. Es la base de las API de *Oracle* para lenguajes y entornos populares, incluidos *Node.js*, *Python* y *PHP*, y proporciona acceso a aplicaciones *OCI*, *OCCI*, *JDBC*, *ODBC* y *Pro * C*. Herramientas incluidas con *Instant Client*, como *SQL * Plus* y *Oracle Data Pump* proporcionan un acceso rápido y conveniente a los datos.

- Descargar controladores *SODA*: es un conjunto de *API* para usar colecciones de documentos *JSON* almacenados en *Oracle Database*. Los controladores *SODA* están disponibles para *Java*, *Node.js*, *Python*, *C*, *PL/SQL* y *REST*.
- *APEX*: *Oracle Application Express (APEX)* es una plataforma de desarrollo de código bajo que le permite crear aplicaciones asombrosas y escalables en una sola plataforma que se puede implementar en cualquier lugar, que es totalmente compatible con *Autonomous Database*.
- *Database Actions*: Cargue, explore, transforme, modele y catalogue sus datos. Use una hoja de trabajo *SQL*, cree interfaces *REST* y aplicaciones de bajo código, administre usuarios y conexiones, cree y aplique modelos de aprendizaje automático. Esto a través de la versión web de *SQL Developer* de *Oracle*.
- *Oracle Machine Learning*: los cuadernos de *Oracle Machine Learning SQL* brindan fácil acceso a las implementaciones de bases de datos escalables y paralelizadas de *Oracle* de una biblioteca de algoritmos de aprendizaje automático de *Oracle Advanced Analytics* (clasificación, regresión, detección de anomalías, agrupación, asociaciones, importancia de atributos, extracción de recursos, series de tiempo etc.), *SQL*, *PL / SQL* y funciones de análisis y estadísticas de *Oracle SQL*.
- *Servicios RESTful* y *SODA*: proporciona interfaces *HTTPS* para trabajar con el contenido de su base de datos Oracle en uno o más esquemas habilitados para *REST*.

The screenshot shows the Oracle Cloud interface for the Autonomous JSON Database. The left sidebar has a navigation menu with 'Autonomous JSON Database' at the top, followed by 'Overview', 'Activity', 'Administration' (with 'Development' highlighted), and 'DATABASE DABHAJSON' at the bottom. The main content area displays several cards:

- Download Oracle Instant Client**: A card describing the Oracle Instant Client as a free, light-weight set of tools, libraries and SDKs for building and connecting applications.
- Download SODA Drivers**: A card describing SODA Drivers as a set of APIs for using collections of JSON documents stored in Oracle Database.
- Oracle APEX**: A card describing Oracle APEX as a low code application development framework.
- Database Actions**: A card describing Database Actions as a way to load, explore, transform, model, and catalog your data using a SQL worksheet.
- Oracle Machine Learning Notebooks**: A card describing Oracle Machine Learning Notebooks as a collaborative, Apache Zeppelin-based user interface for data scientists and broader SQL users of Autonomous Database.
- RESTful Services and SODA**: A card describing RESTful Services (ORDS) as providing HTTPS interfaces for working with the contents of your Oracle Database in one or more REST enabled schemas.

- *Scale Up/Down*: Abre una ventana que permite al usuario escalar la base de datos, tanto hacia arriba como hacia abajo, tanto la capacidad de almacenamiento como de procesamiento, y todos estos cambios son independientes y sin tiempo de inactividad. También permite al usuario activar el *Auto Scaling*, esta característica permite a la base de datos que aumente automáticamente su potencia computacional hasta 3 veces, cuando sea necesario, esto con base en la cantidad de *OCPUs* que tenía sin ningún aumento automático.



Lab 4.

Inicia tu propia aventura con
Oracle Autonomous JSON

Lab 4. Inicia tu propia aventura con Oracle Autonomous JSON Database

Objetivos

Conéctese a su base de datos usando *SQL Developer Web*

Cree un usuario para trabajar con *JSON* en la base de datos *Oracle*

Cargue un archivo *JSON* en la base de datos y trabaje con tablas relacionales

Cargue un archivo *JSON* en la base de datos y trabaje con documentos y colecciones *JSON*

Exposición de los datos para aplicaciones

Conéctese a su base de datos usando *SQL Developer Web*

Objetivo

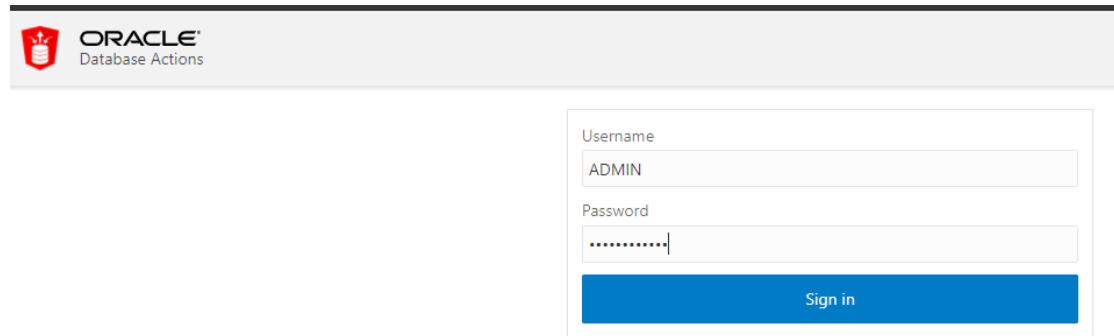
- Conectarse a la base de datos autónoma a través de *SQL Developer Web*, disponible en la consola de servicio.

Tomando como base el hecho que la instancia de *Oracle Autonomous JSON Database* fue aprovisionada satisfactoriamente y además de esto, ya hubo una primera interacción y exploración de las diferentes opciones que existen en la consola de servicio, vamos a conectarnos a la base de datos través del *SQL Developer Web* que está disponible en la opción de *Database Actions*.

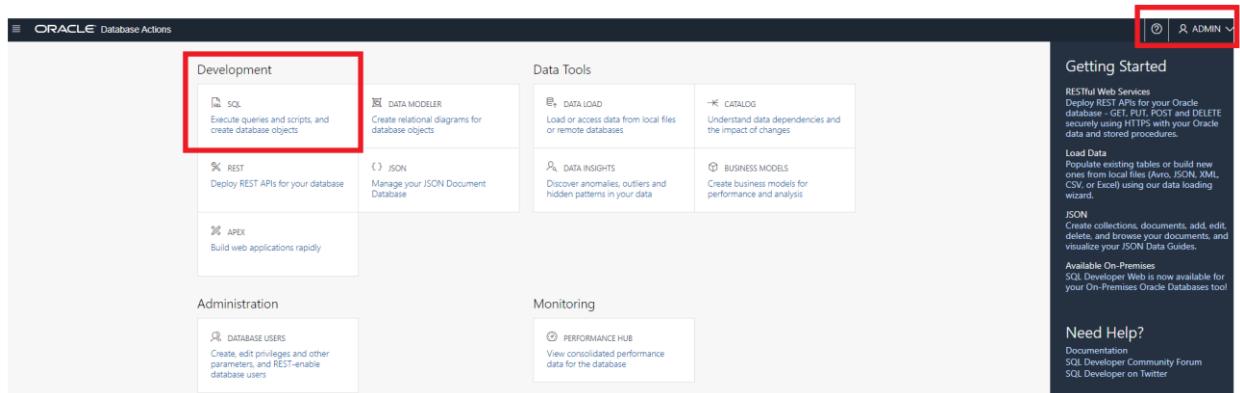
- Paso 1: Hacer clic en **Tools** y luego en la sección de **Database Actions**, hacemos clic en **Open Database Actions**.

The screenshot shows the Oracle Autonomous Database Details page. At the top, there's a large green banner with 'AJD' in white. Below it, the database name 'CE-DABH-AJSD' is displayed. A red box highlights the 'Tools' tab in the navigation bar. Underneath, a red box highlights the 'Database Actions' section. Within this section, a yellow box highlights the 'Open Database Actions' button. The 'Database Actions' section contains text about loading, exploring, transforming, modeling, and cataloging data, along with a link to learn more. Below this, another section for 'Oracle ML User Administration' is partially visible.

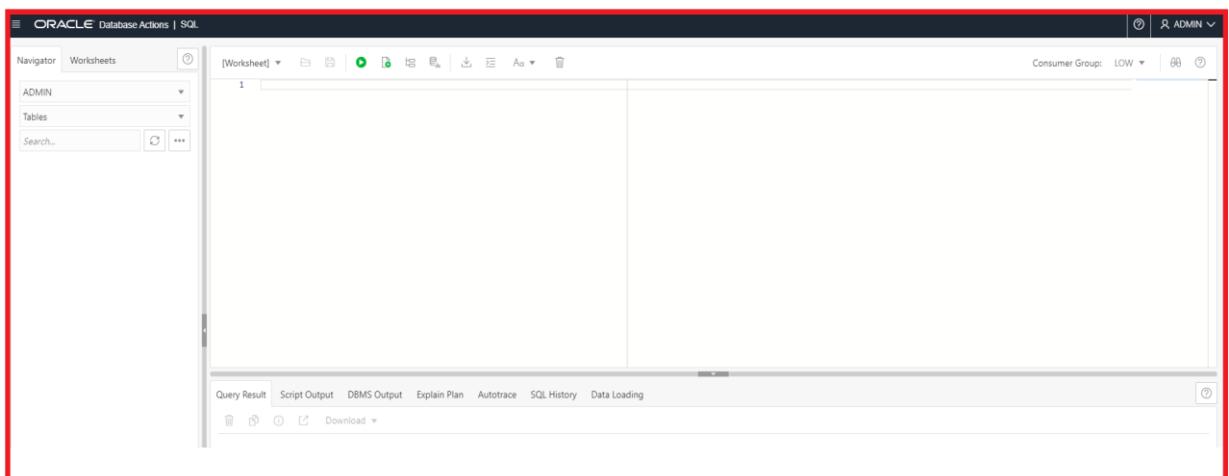
- Paso 2: Usamos el usuario **ADMIN** y la clave que le asignamos, para loguearnos.



- Paso 3: Una vez logueados, seleccionamos la opción de **SQL** para abrir el **SQL Developer Web**. Asimismo, en la parte superior derecho podemos ver el usuario **ADMIN** con el cual, hicimos el logueo previamente.



- Paso 4: Seleccionando la opción anterior, ya estaremos sobre el **SQL Developer Web** para empezar a trabajar allí.



Conclusión: Ahora está conectado a su base de datos autónoma mediante **SQL Developer Web**.

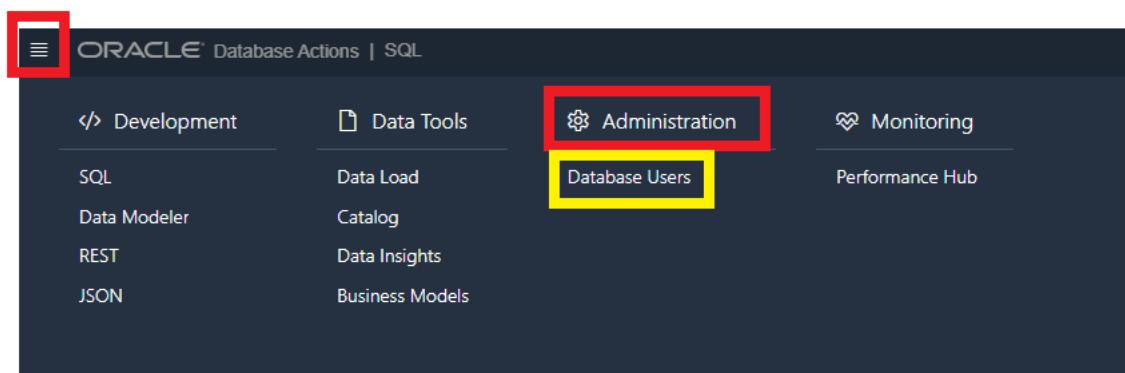
Cree un usuario para trabajar con JSON en la base de datos Oracle

Objetivos

- Crear un usuario de base de datos.
- Otorgar roles, cuotas y REST necesarios para el usuario.
- Acceder a la interfaz de usuario de acciones de la base de datos con el nuevo usuario creado.

Tomamos como punto de partida, la conclusión del punto anterior en donde en este momento ya estamos conectados a la base de datos autónoma mediante **SQL Developer Web**.

- Paso 1: Requerimos crear un esquema de base de datos para nuestras tablas, datos y lógica empresarial. Para esto, vamos a crear un usuario de base de datos; hacemos clic en el menú de acciones de la base de datos en la parte superior izquierda de la página y luego en la sección **Administration**, hacemos haciendo clic en **Database Users**.



Paso 2: Hacemos clic la lado derecho en la opción **+ Create User**.

A screenshot of the 'Database Users' page in the Oracle Database Actions interface. The page shows a list of current users. At the top right, there's a button labeled '+ Create User' with a red box around it. Below the button, there's a table listing users: ADBSNMP, ADMIN (REST Enabled), GGADMIN, and RMAN\$VPC. Each user row has a small profile icon, a name, and a status indicator. The ADMIN row also includes details like ORDS Alias: admin, Last Login: 6/7/2021, 4:21:50 PM, and Password Expires in 359 days. The URL https://slidgxiipd6hbyw9-dabjson.adb.us-ashburn-1.oraclecloudapps.com/ords/admin/_sdw/ is shown below the user details. The bottom right corner of the table area has a 'Sort by' dropdown set to 'User Name: ASC'.

Paso 3: Diligenciamos el nombre de usuario, para este caso vamos a usar **GARY**, luego asignamos una clave y la confirmamos. Tener muy presente que la contraseña debe tener entre 12 y 30 caracteres y contener al menos una letra mayúscula, una letra minúscula y un número. La contraseña no puede contener el carácter de comillas dobles ("") ni el nombre de usuario "admin".

Posteriormente habilitamos la opción de **Web Access**, lo cual nos permitirá utilizar servicios *REST* con este esquema de base de datos desde el principio, así como utilizar las herramientas de Acciones de base de datos.

Por último, en el campo **Quota on tablespace DATA**, seleccionamos la opción **UNLIMITED** y en **Web access advanced features** se debe dejar como **Base path**.

Create User

User

Granted Roles (2)

User Name *
GARY

Password *
.....

Confirm Password *
.....

Quota on tablespace DATA
UNLIMITED

Web Access

Graph (2)

OML (2)

Web access advanced features

Paso 4: Ahora vamos a la opción de **Granted Roles** en la misma ventana que tenemos abierta, y vamos a seleccionar los roles **PDB_DBA** y **SODA_APP** en la columna de **Granted** y hacemos clic en **Create User**.

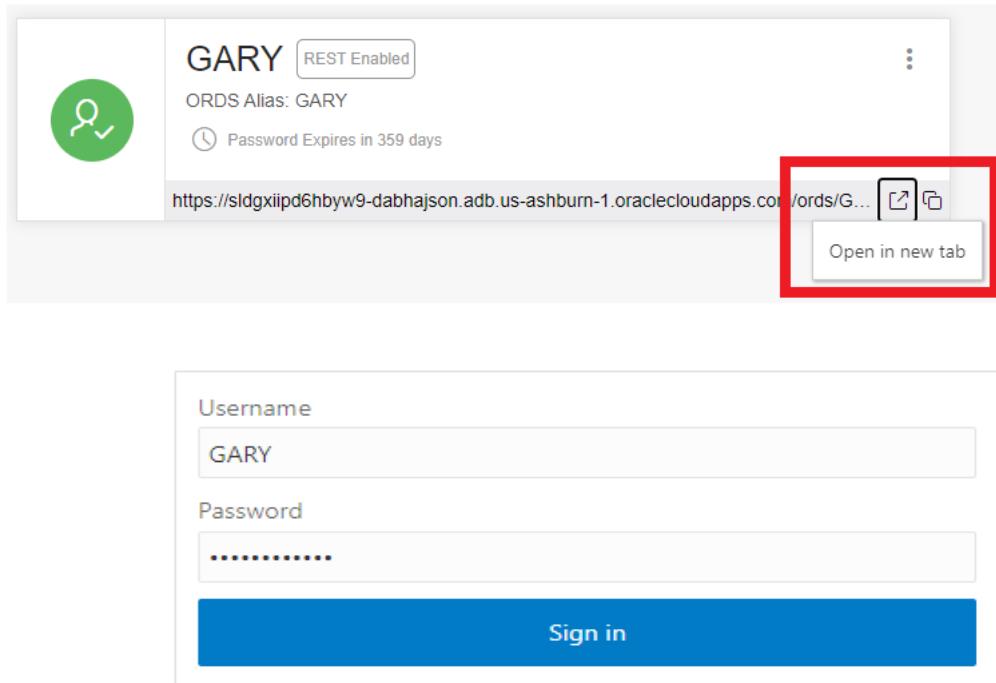
Create User

Granted Roles (4)

Role	Granted	Admin	Default
PROVISIONER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PYQADMIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RDFCTX_ADMIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RECOVERY_CATALOG_OWNER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RECOVERY_CATALOG_OWNER_VPD	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RECOVERY_CATALOG_USER	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SCHEDULER_ADMIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SELECT_CATALOG_ROLE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SYSUMF_ROLE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
XS_CACHE_ADMIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
XS_CONNECT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
XS_NAMESPACE_ADMIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
XS_SESSION_ADMIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CONNECT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PDB_DBA	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RESOURCE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SODA_APP	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Create User Cancel

Paso 5: Ya tenemos el usuario creado, para este caso **GARY**, por tanto, vamos a hacer clic sobre el área señalada abajo y nos logueamos con el usuario **GARY** usando la contraseña que establecimos previamente.



Conclusión: En esta sección, creamos un nuevo usuario de base de datos y le otorgamos los roles y la cuota necesarios para continuar con el *Hands-On*.

Cargue un archivo JSON en la base de datos y trabaje con tablas relacionales

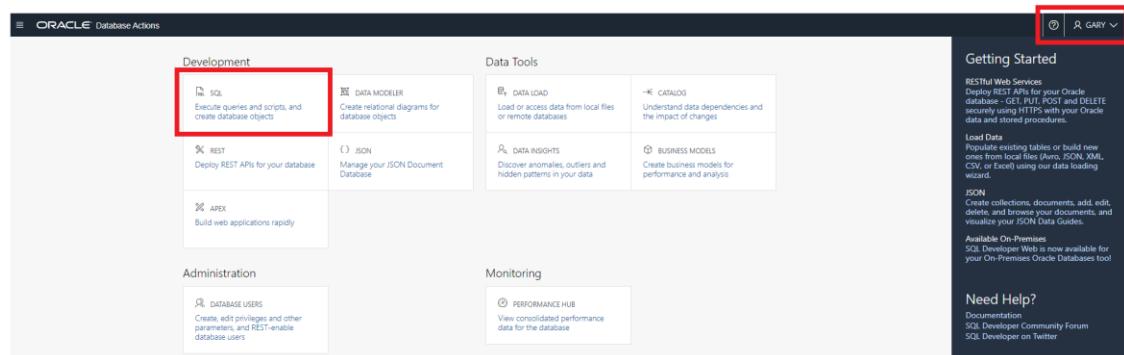
Objetivos

- Cargar datos JSON en tablas relacionales.
- Comprender cómo *Oracle* almacena datos JSON en tablas de relaciones.
- Trabajar con los datos JSON con *SQL*.

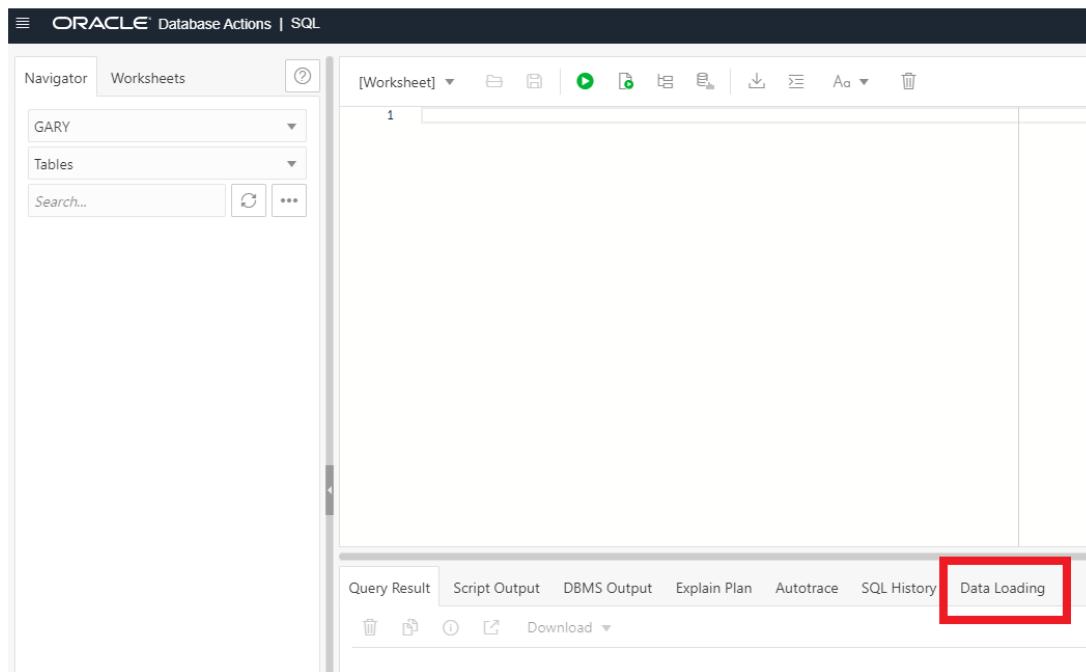
Previamente creamos el usuario **GARY** y ya estamos logueados con dicho usuario, por ende vamos a continuar.

Parte 1 – Cargando datos en la tabla relacional.

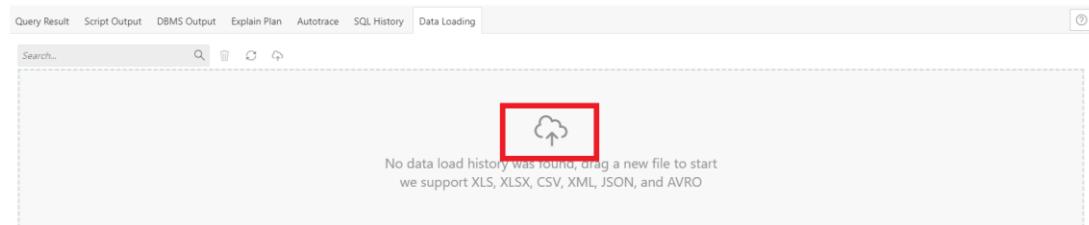
Paso 1: Después de iniciar sesión con el usuario GARY en la sección anterior, estamos sobre la página de inicio o resumen de acciones de la base de datos. Hacemos clic en SQL.

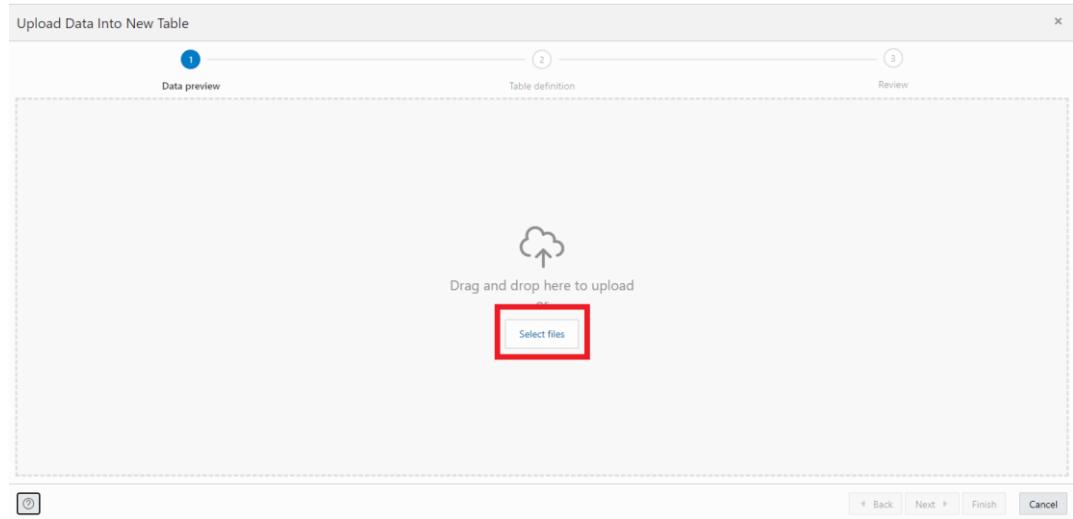


Paso 2: Ahora estamos listos para cargar datos en la base de datos, para ello usaremos la opción **Data Loading**, hacemos clic allí.



Paso 3: Vamos a cargar un archivo tipo **JSON**, para este caso el archivo se llama ***airportDelays.json*** y tiene información sobre vuelos, terminales y aerolíneas, entre otros. Vamos a buscar ese archivo para cargarlo, siguiendo los pasos que se muestran abajo.





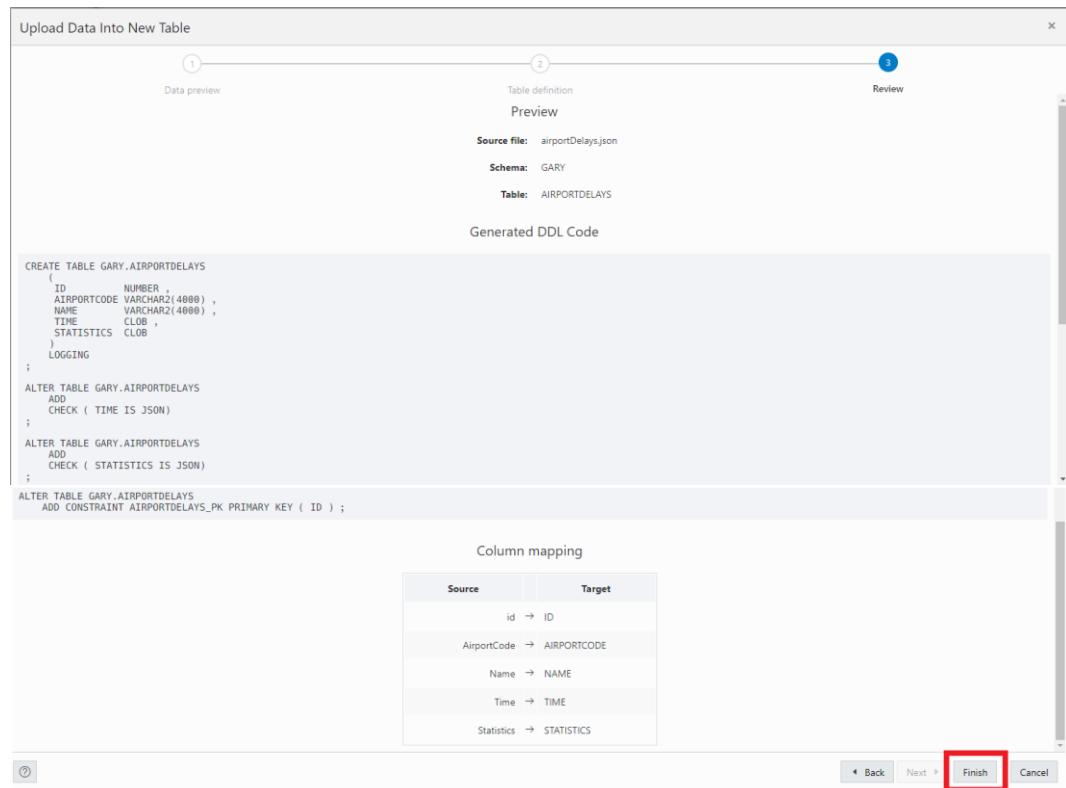
Paso 4: Tan pronto cargamos el archivo, hacemos clic en **Next**:

	ID	AirportCode	Name	Time	Statistics
1	1	ATL	Atlanta, GA: Hartsfield-J...	["Label": "2003/06", "Mo...	(*# of Delays*:{"Carrier":...
2	2	BOS	Boston, MA: Logan Inter...	["Label": "2003/06", "Mo...	(*# of Delays*:{"Carrier":...
3	3	BWI	Baltimore, MD: Baltimore...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
4	4	CLT	Charlotte, NC: Charlotte...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
5	5	DCA	Washington, DC: Ronald...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
6	6	DEN	Denver, CO: Denver Intern...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
7	7	DFW	Dallas/Fort Worth, TX: D...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
8	8	DTW	Detroit, MI: Detroit Metro...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
9	9	EWR	Newark, NJ: Newark Lib...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
10	10	FLL	Fort Lauderdale, FL: Fort...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
11	11	IAD	Washington, DC: Washin...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
12	12	IAH	Houston, TX: George Bu...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
13	13	JFK	New York, NY: John F. K...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...
14	14	LAS	Las Vegas, NV: McCarran...	["Label": "2003/06", "Mo...	(*# of Delays*: {"Carrier":...

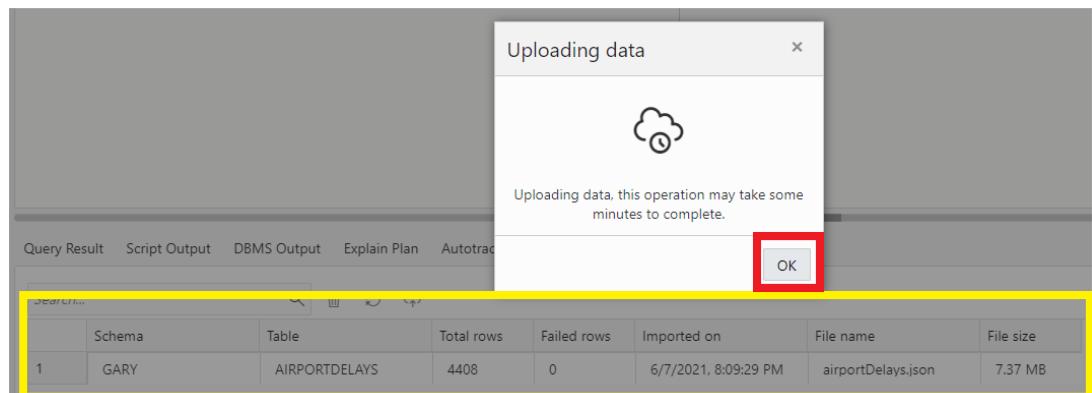
Paso 5: Nos aseguramos de seleccionar como **Primary Key - PK** la línea **ID/NUMBER** y hacemos clic en **Next**.

Column Name	Column Type	Length/Precision	Scale	Default	PK	NULL	Format mask	Row 1	Row 2	Row
<input checked="" type="checkbox"/> ID	NUMBER				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		1	2	3
<input checked="" type="checkbox"/> AIRPORTCODE	VARCHAR2	4000			<input type="checkbox"/>	<input checked="" type="checkbox"/>		ATL	BOS	BWI
<input checked="" type="checkbox"/> NAME	VARCHAR2	4000			<input type="checkbox"/>	<input checked="" type="checkbox"/>		Atlanta, GA: Hartsfield-J...	Boston, MA: Logan Inter...	Baltimore, MD:
<input checked="" type="checkbox"/> TIME	CLOB (JSON)				<input type="checkbox"/>	<input checked="" type="checkbox"/>		["Label": "2003/06", "Mo...	["Label": "2003/06", "Mo...	["Label": "2003/06", "Mo...
<input checked="" type="checkbox"/> STATISTICS	CLOB (JSON)				<input type="checkbox"/>	<input checked="" type="checkbox"/>		(*# of Delays*: {"Carrier":...	(*# of Delays*: {"Carrier":...	(*# of Delays*: {"Carrier":...

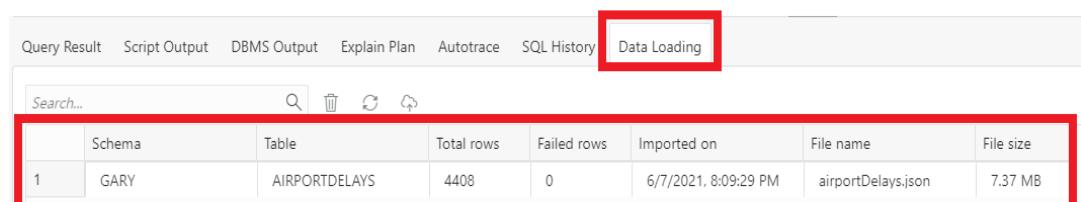
Paso 6: A continuación podemos ver un resumen de la carga del archivo y procedemos a dar clic en **Finish**.



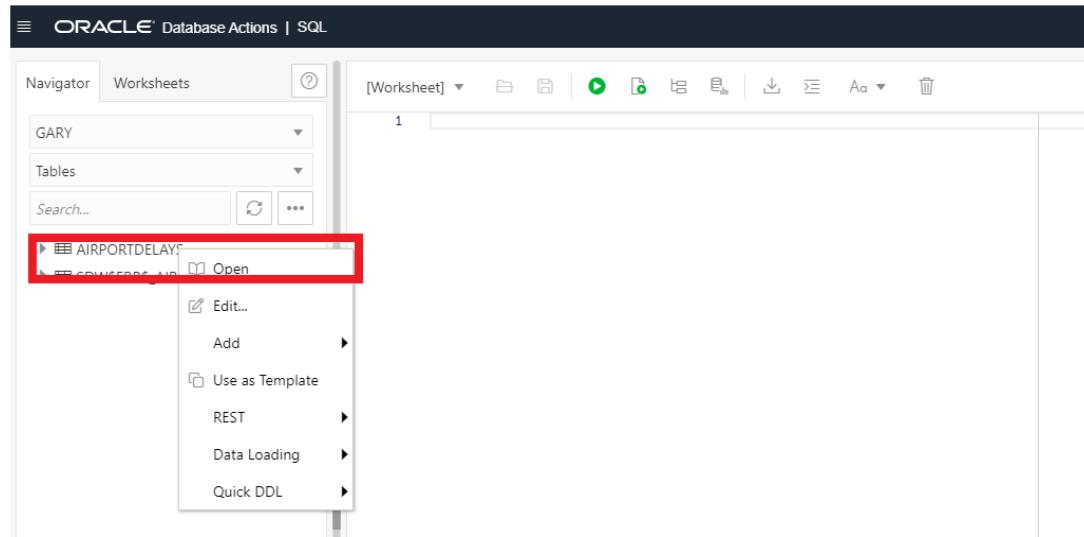
Paso 7: La opción de **Data Loading** ahora procesará el archivo creando una tabla y cargará los datos del archivo JSON en dicha tabla. Una vez veamos al fondo que el archivo ya se cargó, hacemos clic en **OK**.



Paso 8: Una vez hecho esto, se verá una fila en la pestaña Carga de datos que indica cuántas filas se cargaron, si alguna falló y el nombre de la tabla.



Paso 9: Podemos echar un vistazo a nuestra tabla recién creada y los datos en ella usando el navegador a la izquierda de la hoja de trabajo SQL. Hacemos clic con el botón derecho en el nombre de la tabla (**AIRPORTDELAYS**) y seleccionamos la opción **Open**.



Paso 10: En la ventana que se despliega después de la acción anterior, podemos ver la definición de los datos, *Constraints* y demás información que se creó después de subir el archivo.

This screenshot shows the detailed view of the 'GARY.AIRPORTDELAYS' table. On the left, a sidebar lists table metadata: Columns, Data, Constraints, Grants, Statistics, Triggers, Dependencies, Details, Partitions, and Indexes. The 'Data' section is highlighted with a red box. The main area displays a table with 21 rows of data, each containing an ID, airport code, name, time, and statistics. The entire table area is highlighted with a yellow box. At the bottom right of the window, there's a 'Close' button.

	id	airportcode	name	time	statistics
1	10	FLL	Fort Lauderdale, FL: Ft. Lauderdale-Hollywood Int'l	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
2	11	IAD	Washington, DC: Washington Dulles International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
3	12	IAH	Houston, TX: George Bush Intercontinental	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
4	13	JFK	New York, NY: John F. Kennedy	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
5	14	LAS	Las Vegas, NV: McCarran International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
6	15	LAX	Los Angeles, CA: Los Angeles International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
7	1	ATL	Atlanta, GA: Hartsfield-Jackson Atlanta	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
8	2	BOS	Boston, MA: Logan International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
9	3	BWI	Baltimore, MD: Baltimore Washington	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
10	4	CLT	Charlotte, NC: Charlotte Douglas International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
11	5	DCA	Washington, DC: Ronald Reagan Washington National	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
12	6	DEN	Denver, CO: Denver International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
13	7	DFW	Dallas/Fort Worth, TX: Dallas/Fort Worth International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
14	8	DTW	Detroit, MI: Detroit Metropolitan Wayne County	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
15	9	EWR	Newark, NJ: Newark Liberty International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
16	22	PDX	Portland, OR: Portland International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
17	23	PHL	Philadelphia, PA: Philadelphia International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
18	24	PHX	Phoenix, AZ: Phoenix Sky Harbor International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
19	25	SAN	San Diego, CA: San Diego International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
20	26	SEA	Seattle, WA: Seattle-Tacoma International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}
21	27	SFO	San Francisco, CA: San Francisco International	{"Label": "2003/06", "Value": 1}	{"# of Delays": 1, "Cancellations": 0, "Arrivals": 1, "Departures": 0}

Parte 2 – Trabajando con datos JSON en una tabla relacional.

Paso 11: Comencemos de manera simple, consultemos la tabla y devolvamos un recuento (*). Vamos a recuperar la columna JSON de estadísticas basada en un código de aeropuerto.

Ejecutamos la instrucción en la hoja de SQL y este es el resultado:

```
select a.statistics  
from airportdelays a  
where a.airportcode = 'SFO'  
fetch first 1 row only;
```

Query Result		Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
						Download ▾	Execution time: 0.182 seconds
statistics							
1	{"# of Delays": {"Carrier": 416, "Late Aircraft": 312, "National Aviation System": 1080, "Security": 14, "Weather Codes": ["SNW", "RAIN", "SUN", "CLDY"], "Weather": 59}, "Carriers": {"Aircraft Types": [{"make": "Boeing", "models": ["717", "737", "757"]}], "Count": 1}, "Count": 1}						

Paso 12: Ahora, ¿Qué pasaría si quisiéramos filtrar esto según el mes y el año que se encuentra en la columna de tiempo que contiene JSON? Queremos encontrar dónde está el código del aeropuerto SFO, dónde el mes es junio y el año es 2010.

Ejecutamos la instrucción en la hoja de SQL y este es el resultado:

```
select a.statistics  
      from airportdelays a  
     where a.airportcode = 'SFO'  
        and a.time."Month Name" = 'June'  
        and a.time.Year = '2010';
```

Query Result		Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
						Download ▾	Execution time: 0.025 seconds
statistics							
1	{"# of Delays": {"Carrier": 593, "Late Aircraft": 1048, "National Aviation System": 1648, "Security": 1, "Weather Codes": ["SNW", "RAIN", "SUN", "CLDY"], "Weather": 85}, "Carriers": {"Aircraft Types": [{"make": "Boeing", "models": ["717", "737", "757"]}], "Count": 1}, "Count": 1}						

O simplemente podríamos usar el elemento "Etiqueta":

```
select a.statistics  
      from airportdelays a  
     where a.airportcode = 'SFO'  
        and a.time.Label = '2010/06';
```

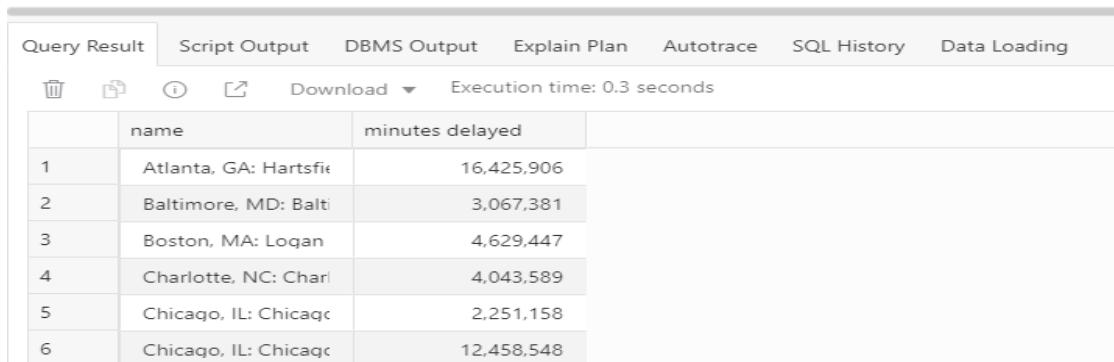
Query Result		Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
						Download ▾	Execution time: 0.078 seconds
statistics							
1	{"# of Delays": {"Carrier": 593, "Late Aircraft": 1048, "National Aviation System": 1648, "Security": 1, "Weather Codes": ["SNW", "RAIN", "SUN", "CLDY"], "Weather": 85}, "Carriers": {"Aircraft Types": [{"make": "Boeing", "models": ["717", "737", "757"]}], "Count": 1}, "Count": 1}						

En cualquier caso, vemos que podemos recorrer el *JSON* con *Dot-Notation* o la notación de puntos.

Paso 13: También podemos usar funciones analíticas como la suma usando *Dot-Notation* o la notación de puntos. Muy útil para crear cuadros y gráficos sobre estos datos con fines de generación de informes. Para este ejemplo, vamos a sumar la cantidad de minutos de retraso para cada aeropuerto en todo este conjunto de datos y agrupar por aeropuerto.

Ejecutamos la instrucción en la hoja de *SQL* y este es el resultado:

```
select a.name,  
       sum(a.statistics."Minutes Delayed".Carrier) "Minutes Delayed"  
from  
airportdelays a  
group by a.name  
order by a.name;
```



	name	minutes delayed
1	Atlanta, GA: Hartsfield-Jackson	16,425,906
2	Baltimore, MD: Baltimore	3,067,381
3	Boston, MA: Logan	4,629,447
4	Charlotte, NC: Charlotte	4,043,589
5	Chicago, IL: Chicago	2,251,158
6	Chicago, IL: O'Hare	12,458,548

Paso 14: *Oracle* también tiene muchas funciones *JSON* integradas para trabajar con datos de documentos, lo que eleva la funcionalidad que se encuentra con *Dot-Notation*.

json_value: la función *SQL/JSON JSON_VALUE* encuentra un valor *JSON* escalar especificado en datos *JSON* y lo devuelve como un valor *SQL* definido (fecha, número, marca de tiempo, sdo_geometry, etc.).

Podemos comenzar fácilmente con solo obtener el elemento Etiqueta en el bloque *JSON* de tiempo.

Ejecutamos la instrucción en la hoja de *SQL* y este es el resultado:

```
select json_value (  
    time,  
    '$.Label'  
) date_label  
from airportdelays a  
where a.id = 5;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
				Download ▾	Execution time: 0.015 seconds	
date_label						
1	2003/06					

Paso 15: Vamos a profundizar un poco más y obtengamos la cantidad de vuelos cancelados que se encuentran en el bloque Estadísticas JSON. Asimismo, le diremos a la consulta que devuelva este valor como un número usando el número de retorno después de la ruta JSON:

Ejecutamos la instrucción en la hoja de SQL y este es el resultado:

```
select json_value (
    Statistics,
    '$.Flights.Cancelled' returning number
) canceled_flights
from airportdelays a
where a.id = 5;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
				Download ▾	Execution time: 0.008 seconds	
canceled_flights						
1	74					

Paso 16: ¿Qué pasa si ahora seleccionamos un valor que sabemos que será una matriz? (También agregamos el error en caso de que no solo obtengamos un valor nulo).

Ejecutamos la instrucción en la hoja de SQL y este es el resultado:

```
select json_value(statistics, ".$.Carriers"."Aircraft Types"[*].models'
error on error)
from airportdelays
where id = 1032;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
				Download ▾		
ORA-40470: JSON_VALUE evaluated to multiple values						

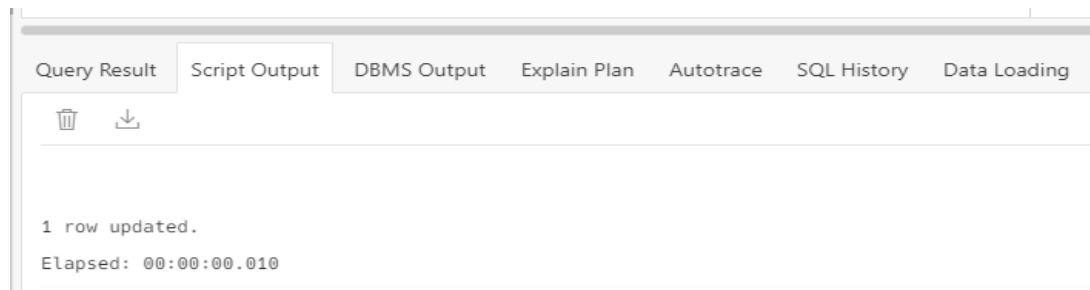
Vemos el error *ORA-40470: JSON_VALUE evaluado en varios valores*. *json_value* solo puede devolver valores JSON escalares (número, cadena, booleano o nulo).

Paso 17: Es posible usar la función *json_mergepatch* para actualizar partes específicas de un documento JSON. Puede pensar que *JSON Merge Patch* fusiona el contenido de la fuente y el parche.

Supongamos que desea que su aerolínea se vea realmente muy bien y elimínela de la cadena de aerolíneas en la sección Nombres del bloque JSON de estadísticas. Podemos hacer esto reemplazando o actualizando el nombre con otro en esa parte particular del documento JSON. Eliminaremos *United* y lo reemplazaremos con *Oracle*.

Ejecutamos la instrucción en la hoja de *SQL* y este es el resultado:

```
update airportdelays
set statistics = json_mergepatch (
    statistics,
    '{"Carriers": {"Names": ""||(
        select replace(json_value (
            Statistics, '$.Carriers.Names')
        , 'United Air Lines Inc.', 'Oracle Air Lines
        Inc.')
        from airportdelays a
        where id = 10)||"""}}
)
where id = 10;
```



The screenshot shows the Oracle SQL Developer interface with the 'Query Result' tab active. The output window displays the message '1 row updated.' and the elapsed time 'Elapsed: 00:00:00.010'.

Ahora procedemos a ejecutar una consulta, después del cambio hecho:

```
select json_value (
    Statistics,
    '$.Carriers.Names'
) "Airline Names"
from airportdelays a
where a.id = 10;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
airline names						

Paso 18: Es posible usar la función `json_transform` para cambiar los datos JSON de entrada (o partes de datos JSON), especificando una o más operaciones de modificación que realizan cambios en los datos JSON. A diferencia de `json_mergepatch`, `json_transform` puede apuntar a los atributos específicos que desea cambiar.

Continuando con nuestras alteraciones de datos clandestinos, trabajemos con la sección JSON de *Minutes Delayed* y veamos si podemos hacer que nuestros números se vean un poco más favorables. A continuación una muestra del JSON con el que trabajaremos:

```

"Minutes Delayed": {
    "Carrier": 61606,
    "Late Aircraft": 68335,
    "National Aviation System": 118831,
    "Security": 518,
    "Total": 268764,
    "Weather": 19474
}

```

Podemos usar `json_transform` para reemplazar el valor de “Total” para que sea 0 con la siguiente instrucción SQL:

```

update airportdelays
set statistics = json_transform (
    statistics,
    replace '$."Minutes Delayed".Total' = '0'
)
where id = 10;

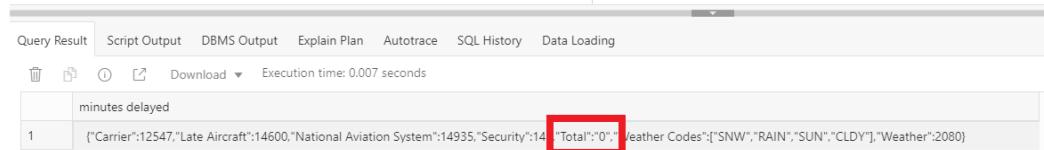
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
1 row updated.						
Elapsed: 00:00:00.010						
1 row updated.						
Elapsed: 00:00:00.009						

Y comprobando la ejecución previa, primero podemos ver la sección completa de JSON.

Ejecutamos la instrucción en la hoja de SQL y este es el resultado:

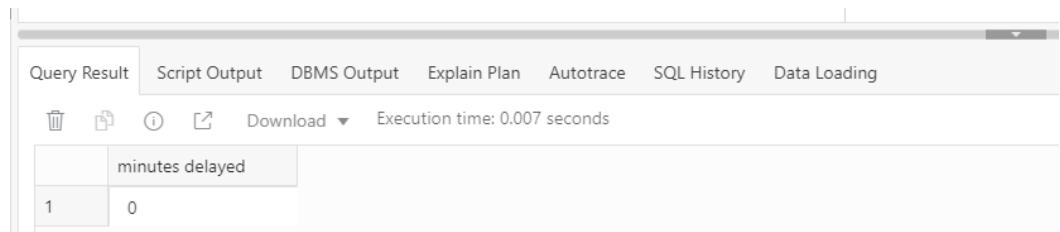
```
select a.statistics."Minutes Delayed"  
from airportdelays a  
where id = 10;
```



minutes delayed	
1	{"Carrier":12547,"Late Aircraft":14600,"National Aviation System":14935,"Security":14,"Total":0,"Weather Codes":["SNW","RAIN","SUN","CLDY"],"Weather":2080}

O también es posible revisar ese atributo en particular:

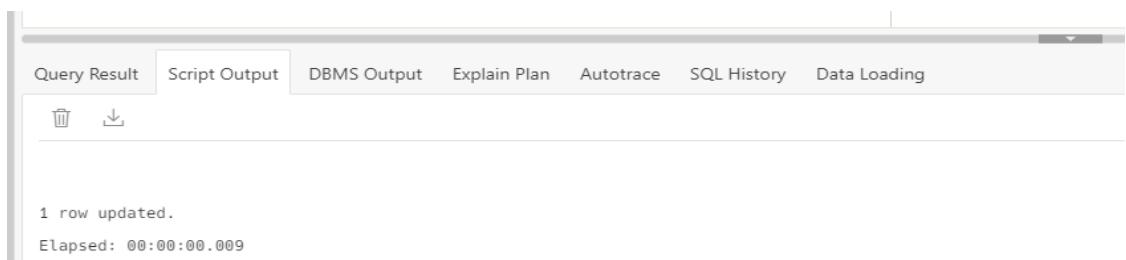
```
select a.statistics."Minutes Delayed".Total  
from airportdelays a  
where id = 10;
```



minutes delayed	
1	0

Obtuvimos un resultado de 0 en la consulta anterior. También podemos usar `json_transform` para eliminar atributos completos. Si quisiéramos eliminar varios atributos en esta sección *Minutes Delayed*, podemos ejecutar la siguiente sentencia SQL para cada atributo:

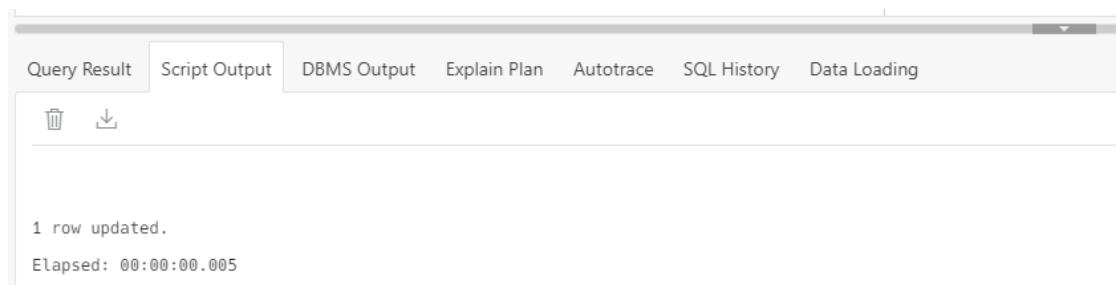
```
update airportdelays  
set statistics = json_transform (  
    statistics,  
    remove '$."Minutes Delayed".Carrier',  
)  
where id = 10;
```



Query Result	
1 row updated.	
Elapsed: 00:00:00.009	

O podemos encadenar múltiples declaraciones de eliminación y reemplazo con la instrucción `json_transform`:

```
update airportdelays
set statistics = json_transform (
    statistics,
    replace '$."Minutes Delayed".Total' = '0',
    remove '$."Minutes Delayed".Carrier',
    remove '$."Minutes Delayed"."Late Aircraft"'
)
where id = 10;
```



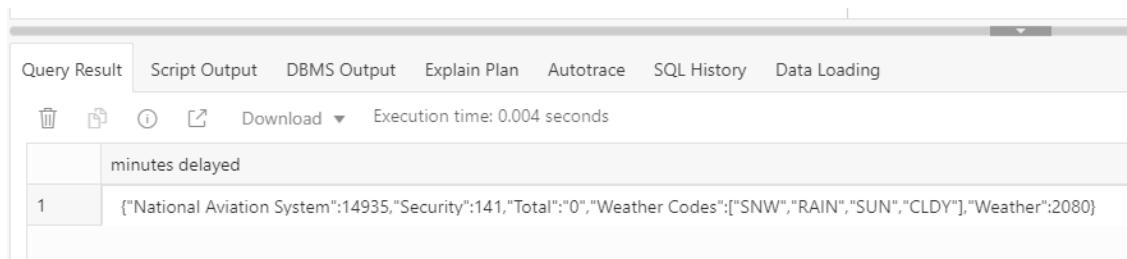
Query Result Script Output DBMS Output Explain Plan Autotrace SQL History Data Loading

1 row updated.

Elapsed: 00:00:00.005

Y como siempre, revisemos nuestro trabajo para asegurarnos que los cambios fueron ejecutados correctamente:

```
select a.statistics."Minutes Delayed"
from airportdelays a
where id = 10;
```



Query Result Script Output DBMS Output Explain Plan Autotrace SQL History Data Loading

Execution time: 0.004 seconds

	minutes delayed
1	{"National Aviation System":14935,"Security":141,"Total":"0","Weather Codes":["SNW","RAIN","SUN","CLDY"],"Weather":2080}

Paso 19: La función `json_query` de SQL/JSON, selecciona y devuelve uno o más valores de datos JSON. Por lo tanto, es posible usar `json_query` para recuperar fragmentos de un documento JSON en JSON.

Podemos ejecutar la siguiente consulta para ejemplificar:

```
select JSON_QUERY(statistics, '$.# of Delays')
from airportdelays
where id = 1032;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
1	json_query(statistics,'\$.#ofdelays')					

Execution time: 0.059 seconds

```
{"Carrier":485,"Late Aircraft":686,"National Aviation System":545,"Security":2,"Weather Codes":["SNW","RAIN","SUN","CLDY"],"Weather":50}
```

Repasemos un problema que tuvimos anteriormente con *json_value* y las matrices de retorno.

Recuerdan que tuvimos el error *ORA-40470: JSON_VALUE* evaluado en varios valores, ya que *json_value* solo puede devolver valores JSON escalares (número, cadena, booleano o nulo).

¿Puede *json_query* venir al rescate?

```
select json_query(statistics, '$."Carriers"."Aircraft Types"[*].models'
error on error)
from airportdelays
where id = 1032;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
1	ORA-40480: result cannot be returned without array wrapper					

Y nuevamente obtenemos un error, pero esta vez el *ORA-40480*: el resultado no se puede devolver sin un contenedor de matriz.

Ok, está bien, entonces vamos a agregar un contenedor de matriz a la consulta y veamos qué pasa:

```
select JSON_QUERY(statistics, '$."Carriers"."Aircraft Types"[*].models'
with array wrapper error on error)
from airportdelays
where id = 1032;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
1	json_query(statistics,'\$.carriers.aircrafttypes[*].models'witharraywrappererroronerror)					

Execution time: 0.007 seconds

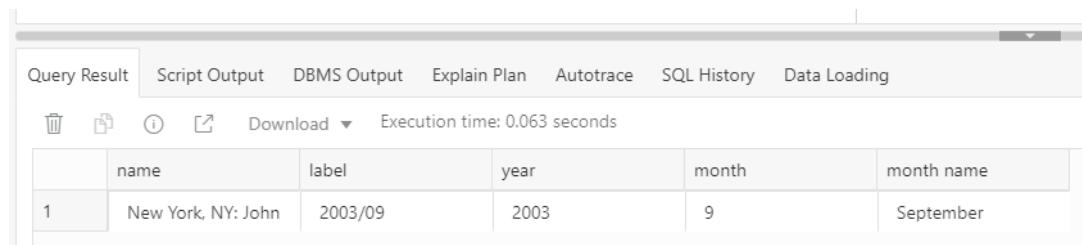
```
[["717","737","757","767","777","787"],["A320","A321","A330","A340","A350","A380"]]
```

Haciendo esos ajustes a la consulta, el resultado es satisfactorio.

Paso 20: La función `JSON_TABLE` de `SQL/JSON` crea una vista relacional de datos `JSON`. Asigna el resultado de una evaluación de datos `JSON` en filas y columnas relacionales.

Podemos comenzar tomando el tiempo `JSON` y convirtiéndolo en una tabla:

```
select a.name, v.*  
from airportdelays a, json_table (  
    a.time, '$'  
    columns (  
        Label,  
        Year,  
        Month, "Month Name"  
    ) )v  
where a.id = 100;
```



	name	label	year	month	month name
1	New York, NY: John	2003/09	2003	9	September

De esta forma hemos tomado el tiempo en formato `JSON` y creó un resultado de tabla relacional a partir de él.

```
"Time": {  
    "Label": "2003/09",  
    "Month": 9,  
    "Month Name": "September",  
    "Year": 2003  
}
```

¿Qué tal si probamos con más de una fila? Podemos alterar el `SQL` para obtener, digamos, 10 filas, lo que devuelve varias filas en un formato de tabla única:

```
select a.name, v.*  
from airportdelays a, json_table (  
    a.time, '$'  
    columns (  
        Label, Year,  
        Month, "Month Name"  
    ) )v  
fetch first 10 rows only;
```

Query Result Script Output DBMS Output Explain Plan Autotrace SQL History Data Loading

Download ▾ Execution time: 0.067 seconds

	name	label	year	month	month name
1	Fort Lauderdale, FL:	2003/06	2003	6	June
2	Washington, DC: W:	2003/06	2003	6	June
3	Houston, TX: Georg	2003/06	2003	6	June
4	New York, NY: John	2003/06	2003	6	June
5	Las Vegas, NV: McC	2003/06	2003	6	June
6	Los Angeles, CA: Lo	2003/06	2003	6	June
7	Atlanta, GA: Hartsfir	2003/06	2003	6	June
8	Boston, MA: Logan	2003/06	2003	6	June
9	Baltimore, MD: Balt	2003/06	2003	6	June
10	Charlotte, NC: Charl	2003/06	2003	6	June

¿Qué tal si ahora queremos obtener datos en matrices anidadas?

```
select t.*
from airportdelays,
    json_table(Statistics, '$.Carriers."Aircraft Types"[*]')
        columns(
            make varchar2(400) path '$.make',
            nested path '$.models[*]'
            columns(
                models varchar2(400) path '$'
            )
        )
    ) as t
where id = 100;
```

Query Result Script Output DBMS Output Explain Plan Autotrace SQL History Data Loading

Download ▾ Execution time: 0.013 seconds

	make	models
1	Boeing	717
2	Boeing	737
3	Boeing	757
4	Boeing	767
5	Boeing	777
6	Boeing	787
7	Airbus	A320
8	Airbus	A321
9	Airbus	A330
10	Airbus	A340
11	Airbus	A350
12	Airbus	A380

Como pueden ver, es posible obtener datos de matrices con la función `json_table` en combinación con la ruta anidada.

También podemos combinar tablas relacionales y JSON. En lugar de una nueva fila por modelo, volvamos a condensarlos en solo 2 filas:

```
select t.*  
from airportdelays,  
      json_table(Statistics, '$.Carriers."Aircraft Types"[*]'  
                  columns(  
                      make varchar2(400) path '$.make',  
                      models varchar2(400) format json path '$.models'  
                  )  
      ) as t  
where id = 100;
```

	make	models
1	Boeing	[717,"737","757","767","777","787"]
2	Airbus	["A320","A321","A330","A340","A350","A380"]

Lo que hicimos fue combinar la segunda ruta anidada y luego en la consulta hicimos que generara o formateara el resultado en JSON con formato JSON.

Incluso podemos ir un paso más allá y pivotear matrices anidadas como columnas como se ve en el siguiente ejemplo:

```
select a.airportcode, t.* , md.*  
from airportdelays a,  
      json_table (  
          a.time, '$'  
          columns (  
              Label,  
              "Month Name",  
              Year  
          )  
      ) t,  
      json_table (  
          a.Statistics, '$."Minutes Delayed"'  
          columns (  
              Carrier,
```

```

    "National Aviation System",
    "Late Aircraft",
    Security,
    Weather,
    "Weather Codes" varchar2(200) format json path
'$.Weather Codes",
nested path '$.Weather Codes"
columns (
    code1 varchar2(100) path '$[0]',
    code2 varchar2(100) path '$[1]',
    code3 varchar2(100) path '$[2]',
    code4 varchar2(100) path '$[3]'
),
Total
)
) md
where a.id = 100;

```

Query Result										
Script Output DBMS Output Explain Plan Autotrace SQL History Data Loading										
Download Execution time: 0.018 seconds										
	airportcode	label	month name	year	carrier	national aviation system	late aircraft	security	weather	weather codes
1	JFK		September	2003	13176	14391	7732	93	996	[{"SNW": "RAIN", "SUI":

Paso 21: La condición *json_exists* de *SQL/JSON* le permite usar una expresión de ruta de *SQL/JSON* como un filtro de fila, para seleccionar filas según el contenido de los documentos *JSON*.

Nuevamente, comencemos con facilidad y recuperaremos todos los identificadores donde el año es 2004 y el mes es 6 (junio):

```

select a.id
from airportdelays a
where json_exists(a.time, '$?(@.Year == "2004" && @.Month == "6")');

```

Query Result									
Script Output DBMS Output Explain Plan Autotrace SQL History Data Loading									
Download Execution time: 0.18 seconds									
	id								
1		349							
2		350							
3		351							
4		352							
5		353							
6		354							
7		358							
8		359							
9		360							
10		361							

Pero vamos a ir un paso más allá y no solo recuperaremos todos los identificadores donde el año es 2004 y el mes es 6 (junio), sino que el total de minutos retrasados es mayor que 500000:

```
select a.id, a.airportcode, a.Statistics."Minutes Delayed".Total
from airportdelays a
where json_exists(a.time, '$?(@.Year == "2004" && @.Month == "6")')
and json_exists(a.Statistics, '$?(@."Minutes Delayed".Total > 500000)');
```

	id	airportcode	minutes delayed
1	349	ATL	714316
2	369	ORD	554653
3	355	DFW	520185

Como podemos ver, `json_exists` nos permitirá filtrar de manera rápida y eficiente nuestros documentos JSON para obtener los resultados que queremos.

Hagamos una prueba más, pero ahora ejecutando una explicación del plan en el siguiente SQL que acabamos de usar:

```
select a.id, a.airportcode, a.Statistics."Minutes Delayed".Total
from airportdelays a
where json_exists(a.time, '$?(@.Year == "2004" && @.Month == "6")')
and json_exists(a.Statistics, '$?(@."Minutes Delayed".Total > 500000)');
```

OPERATION	OBJECT NAME	OBJECT TYPE	CARDINALITY	COST	PARTITION START	PARTITION STOP
SELECT STATEMENT	null	(null)	1	37	(null)	(null)
TABLE ACCESS	AIRPORTDELAYS	TABLE	1	37	(null)	(null)

Ahora vamos a ejecutar la misma sentencia con *SQL*, pero usando el formato *Dot Notation* o notación de puntos y ejecutaremos una explicación del plan para ver cuál fue el resultado:

```
select a.id, a.airportcode, a.Statistics."Minutes Delayed".Total
from airportdelays a
where a.time.Year = 2004
and a.time.Month = 6
and a.Statistics."Minutes Delayed".Total > 500000
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
		Save	Print	Copy		
OPERATION	OBJECT NAME	OBJECT TYPE	CARDINALITY	COST	PARTITION START	PARTITION STOP
SELECT STATEMENT	null	(null)	180	334	(null)	(null)
NESTED LOOPS	null	(null)	180	334	(null)	(null)
TABLE ACCESS	AIRPORTDELAYS	TABLE	220	37	(null)	(null)

Podemos ver que las expresiones de ruta de SQL/JSON son mucho más eficientes.

Paso 22: Bueno, pero ahora requiero crear un *JSON* a partir de datos relacionales. ¿Necesita hacer un documento *JSON* a partir de la tabla relacional? ¡También es posible con la siguiente instrucción!

```
select json_object( * ) jdoc
from airportdelays;
```

Query Result	Script Output	DBMS Output	Explain Plan	Autotrace	SQL History	Data Loading
Execution time: 0.135 seconds						
jdoc						
1	{"ID":10,"AIRPORTCODE":"FLL","NAME":"Fort Lauderdale, FL: Fort Lauderdale-Hollywood International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":247,"Late Aircraft":102,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
2	{"ID":11,"AIRPORTCODE":"IAD","NAME":"Washington, DC: Washington Dulles International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":320,"Late Aircraft":102,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
3	{"ID":12,"AIRPORTCODE":"IAH","NAME":"Houston, TX: George Bush Intercontinental/Houston","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":329,"Late Aircraft":102,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
4	{"ID":13,"AIRPORTCODE":"JFK","NAME":"New York, NY: John F. Kennedy International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":376,"Late Aircraft":226,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
5	{"ID":14,"AIRPORTCODE":"LAS","NAME":"Las Vegas, NV: McCarran International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":511,"Late Aircraft":678,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
6	{"ID":15,"AIRPORTCODE":"LAX","NAME":"Los Angeles, CA: Los Angeles International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":830,"Late Aircraft":765,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
7	{"ID":1,"AIRPORTCODE":"ATL","NAME":"Atlanta, GA: Hartsfield-Jackson Atlanta International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":1009,"Late Aircraft":495,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
8	{"ID":2,"AIRPORTCODE":"BOS","NAME":"Boston, MA: Logan International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":374,"Late Aircraft":495,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
9	{"ID":3,"AIRPORTCODE":"BWI","NAME":"Baltimore, MD: Baltimore/Washington International Thurgood Marshall","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":300,"Late Aircraft":472,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
10	{"ID":4,"AIRPORTCODE":"CLT","NAME":"Charlotte, NC: Charlotte Douglas International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":300,"Late Aircraft":472,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
11	{"ID":5,"AIRPORTCODE":"DCA","NAME":"Washington, DC: Ronald Reagan Washington National","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":283,"Late Aircraft":323,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
12	{"ID":6,"AIRPORTCODE":"DEN","NAME":"Denver, CO: Denver International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":516,"Late Aircraft":323,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
13	{"ID":7,"AIRPORTCODE":"DFW","NAME":"Dallas/Fort Worth, TX: Dallas/Fort Worth International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":986,"Late Aircraft":371,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
14	{"ID":8,"AIRPORTCODE":"DTW","NAME":"Detroit, MI: Detroit Metro Wayne County","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":376,"Late Aircraft":371,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
15	{"ID":9,"AIRPORTCODE":"EWR","NAME":"Newark, NJ: Newark Liberty International","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":322,"Late Aircraft":519,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					
16	{"ID":10,"AIRPORTCODE":"DCA","NAME":"Washington, DC: Ronald Reagan Washington National","TIME":{"Label":"2003/06","Month":6,"Month Name":"June","Year":2003}, "STATISTICS":{"# of Delays":{"Carrier":300,"Late Aircraft":472,"National Aviation":102,"Weather Delays":102}, "# of Weather Delays":102}, "Weather":102}					

Paso 23: Por último, vamos a poner todo junto. Analicemos un poco estos datos ... tenemos retrasos ... así que tal vez podamos echar un vistazo a qué aeropuertos tuvieron más retrasos meteorológicos en cada mes durante el lapso de estos datos.

```
with weather_delays as (
    select json_value(time, '$.Label') time_label,
           max(json_value(statistics,'$.#"# of Delays".Weather' returning number error on error)) max_weather
        from airportdelays a
       group by json_value(time, '$.Label')
)
select a.airportcode,
       json_value(a.time, '$.Label') "Year/Month",
       json_value(a.statistics, '$.#"# of Delays".Weather' returning number) "Weather Delays"
  from airportdelays a,
       weather_delays wd
 where wd.time_label = json_value(a.time, '$.Label')
```

```

and json_value(a.statistics, '$.#' of "Delays".Weather'
returning number) = wd.max_weather
order by json_value(a.time, '$.Label');

```

	airportcode	year/month	weather delays
1	ATL	2003/06	328
2	ATL	2003/07	367
3	ATL	2003/08	361
4	ATL	2003/09	161
5	ATL	2003/10	170
6	ORD	2003/11	202
7	SLC	2003/12	282
8	ATL	2004/01	599
9	ATL	2004/02	787
10	ATL	2004/03	227
11	DFW	2004/04	217
12	ATL	2004/05	556
13	DFW	2004/06	729

Conclusión: En esta sección, cargamos un archivo JSON en una tabla relacional y trabajamos con esos datos usando SQL y funciones JSON/SQL usando la hoja de trabajo SQL de la sección **Database Actions**.

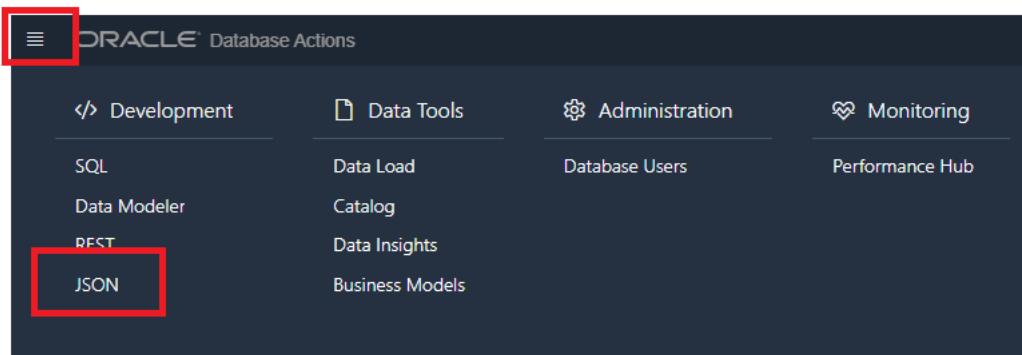
Cargue un archivo JSON en la base de datos y trabaje con documentos y colecciones JSON

Objetivos

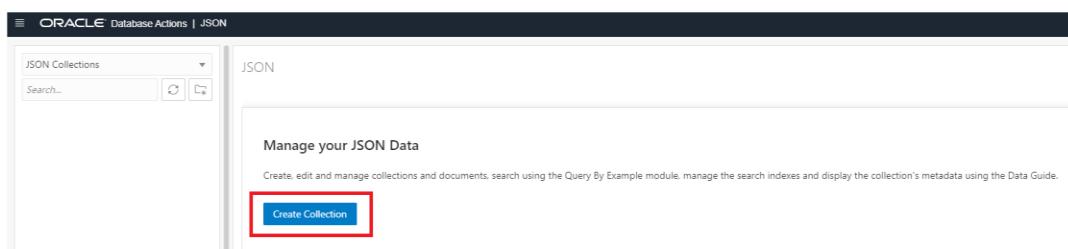
- Cargar datos JSON en tablas relacionales.
- Comprender cómo Oracle almacena datos JSON en tablas de relaciones.
- Trabajar con los datos JSON con SQL.

Parte 1 – Creando una Colección usando la opción de Database Actions.

Paso 1: El primer paso es crear una **colección** para nuestros documentos JSON. Podemos hacer esto de dos formas. El primer método es utilizar la interfaz de usuario en acciones de base de datos. Podemos comenzar seleccionando JSON en el menú de **Database Actions** en la parte superior izquierda. Es necesario asegurarnos que se cree con el usuario **GARY** que es con el que venimos trabajando.



Paso 2: En la hoja de trabajo JSON, haga clic en el botón **Create Collection** en medio de la página.

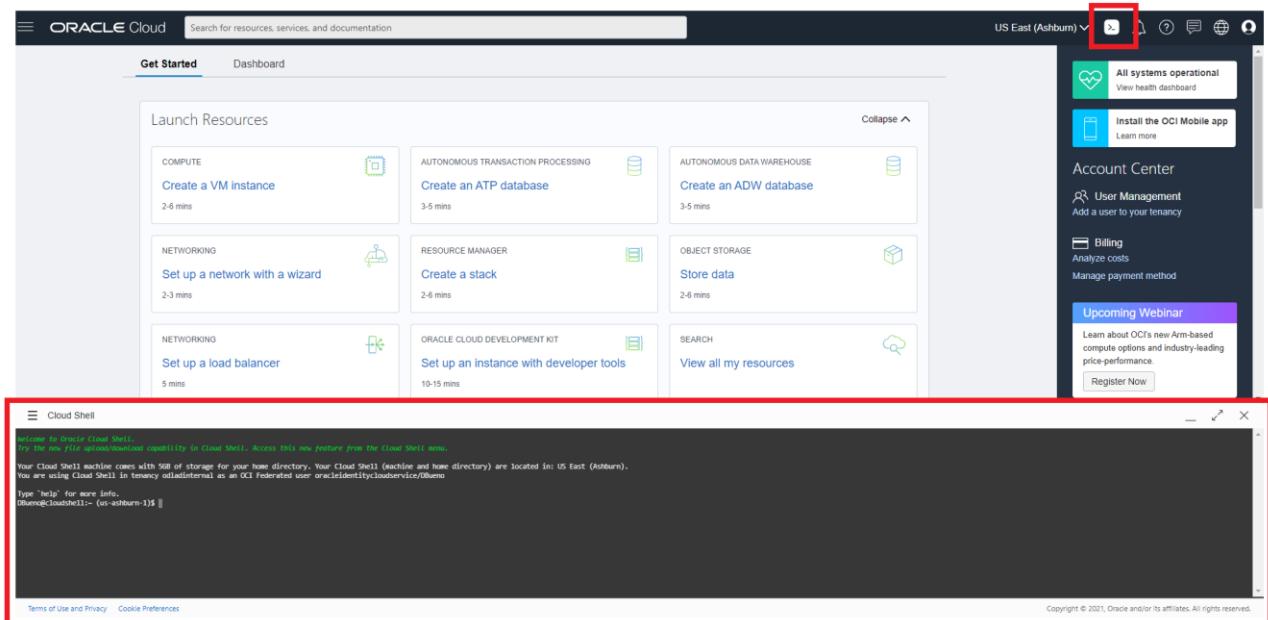


Paso 3: en la nueva pestaña que se despliega, en el campo de nombre vamos a colocar **airportdelayscollection** y damos clic en **Create**.

A screenshot of the "New Collection" dialog box. The "Collection Name" field contains "airportdelayscollection" and is highlighted with a red box. At the bottom right of the dialog, there is a "Create" button highlighted with a red box. The dialog also includes sections for General, Key Column, Content Column, and Version Column settings.

Existe otra forma para crear la Colección y es usando *SODA* para *REST APIs*, vamos a proceder con esa parte.

Paso 1: Para hacer esto, abrimos un *OCI Cloud Shell*. Podemos hacer esto haciendo clic en el ícono de *Cloud Shell* en la esquina superior derecha de la consola web de *OCI*. Una vez se despliegue en la parte inferior, ya podemos usarlo.



Paso 2: Para usar *SODA* para *API REST*, necesitamos construir la *URL*. Para comenzar, usamos **cURL** y pasamos la combinación de nombre de usuario / contraseña. Asegúrese de utilizar la contraseña que estableció para nuestro usuario en el laboratorio de configuraciones de usuario.

```
curl -u "GARY:PASSWORD"
```

Además, podemos agregar el **-i** que le dice al comando **cURL** que incluya los encabezados de respuesta *HTTP* en la salida. Esto es útil para depurar.

```
curl -u "GARY:PASSWORD" -i
```

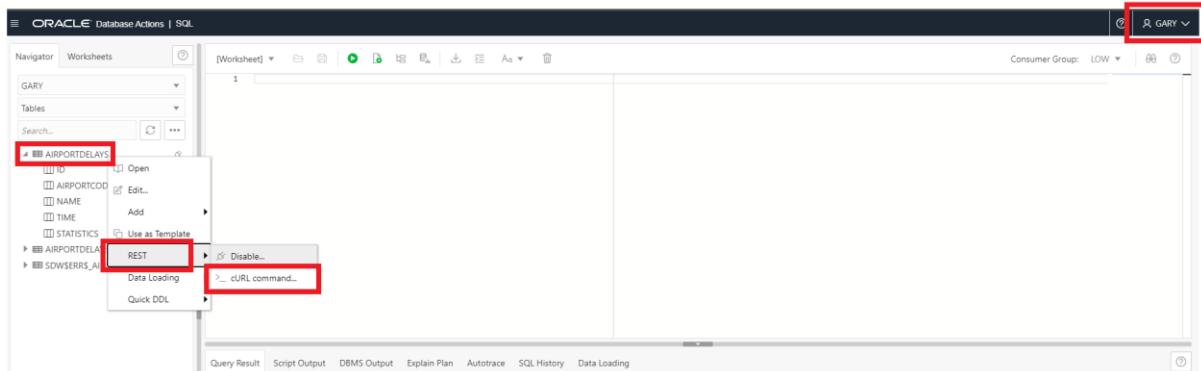
A continuación, esto creará una colección, por lo que usaremos el método **PUT HTTP**.

```
curl -u " GARY:PASSWORD" -i -X PUT
```

Por último, agregaremos la *URL*. La *URL* se crea con el nombre de *host* seguido de *ords*, seguido de nuestro nombre de esquema **GARY**.

Para saber cuál es la *URL*, vamos a la hoja de *SQL* en **Database Actions**, buscamos la tabla que creamos de **AIRPORTDELAYS**, damos clic derecho sobre el nombre y

allí en la opción de **REST**, lo habilitamos, si ya está habilitado allí nos va a mostrar la opción de > **cURL command**, de allí tomamos la línea requerida:



cURL for the table AIRPORTDELAYS

This screenshot shows the "cURL for the table AIRPORTDELAYS" interface. It lists various HTTP methods: GET ALL, GET Single, POST, BATCH LOAD, PUT, and DELETE. The "GET Single" method is selected, and its corresponding cURL command is displayed: "curl --location \ 'https://sldgxiipd6hbyw9-dabha.json.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/airportdelays/'". This line is also highlighted with a red box.

Luego, agregamos la palabra “**soda**” para indicar que queremos usar las API de SODA seguidas de ”**latest**“ y el nombre de la colección **airportdelayscollection**.

De esta manera, la línea nos debería quedar algo así:

```
curl -u "GARY:xxxxxxxxx" -i -X PUT https://sldgxiipd6hbyw9-dabha.json.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/soda/latest/airportdelayscollection
```

Con esta información ya tenemos la línea completa y procedemos a ejecutar esta línea en el **cloud shell** de OCI:

```
DBueno@cloudshell:~ (us-ashburn-1)$ curl -u "GARY:xxxxxxxxx" -i -X PUT https://sldgxiipd6hbyw9-dabha.json.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/soda/latest/airportdelayscollection
HTTP/1.1 201 Created
Date: Tue, 08 Jun 2021 12:45:49 GMT
Content-Length: 0
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Location: https://sldgxiipd6hbyw9-dabha.json.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/soda/latest/airportdelayscollection/

DBueno@cloudshell:~ (us-ashburn-1)$
```

Parte 2 – Carga de datos JSON en una colección.

En esta sección, comenzaremos por crear una *URL* que le permitirá acceder a las *API* de *SODA* para *REST*. Luego, preparará un archivo y utilizará la *URL* recién creada a través de **cURL** para cargar el archivo en la colección.

Paso 1: Las *API* de *SODA* se utilizarán para cargar los registros en la colección. Construiremos el comando *cURL* de manera similar a como lo hicimos en el paso anterior. Empezamos de nuevo con la combinación de usuario/contraseña. Por favor recuerda usar su contraseña en lugar de *PASSWORD*.

```
curl -u "GARY:PASSWORD"
```

Y como antes, necesitamos agregar la *-i* para la información de retorno del encabezado.

```
curl -u " GARY:PASSWORD" -i
```

Ahora el método *HTTP* para la carga de datos que usaremos es ***POST***.

```
curl -u " GARY:PASSWORD" -i -X POST
```

A continuación, debemos indicar el archivo que queremos cargar. Utilizaremos ***-d*** y luego **@*airportDelays.json*** (para este caso).

```
curl -u " GARY:PASSWORD" -i -X POST -d @airportDelays.json
```

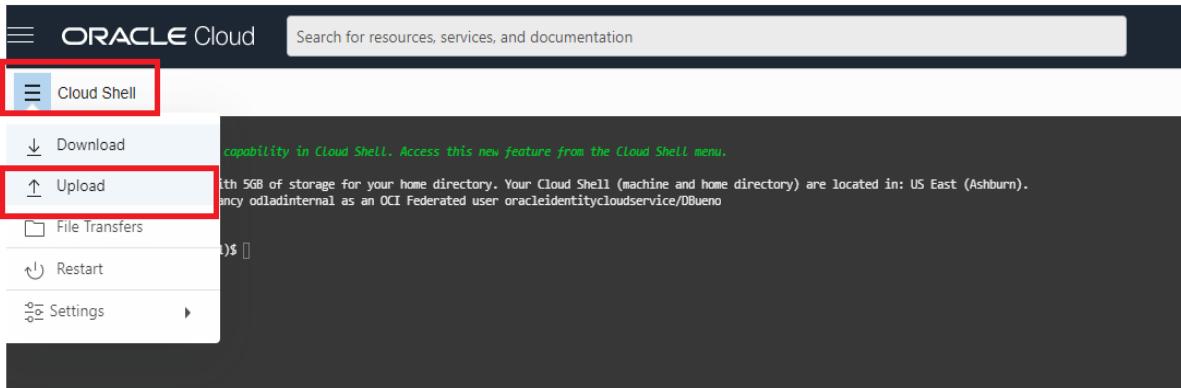
Los encabezados de tipo de contenido deben configurarse para indicar que estamos pasando sobre *JSON*.

```
curl -u " GARY:PASSWORD" -i -X POST -d @airportDelays.json -H "Content-Type: application/json"
```

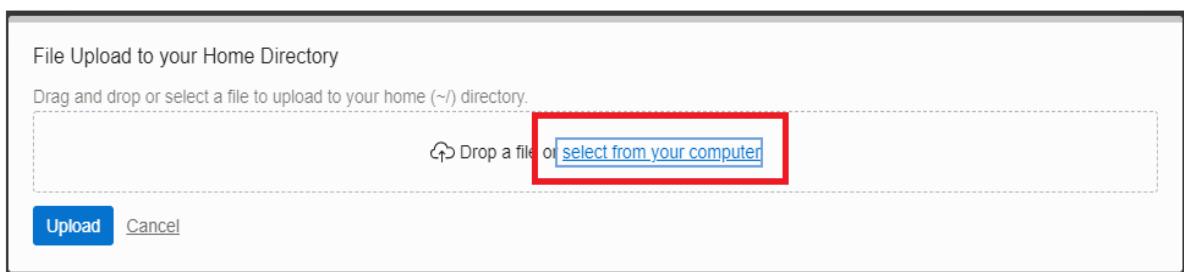
Por último, agregamos la *URL*, pero agregamos una acción de inserción al final con **?action=insert**. Entonces tomamos la *URL* que construimos previamente y agregamos la acción. El comando *cURL* completo será similar al siguiente. Por favor recuerde usar su contraseña en lugar de *PASSWORD*.

```
curl -u "GARY:xxxxxxxx" -i -X POST -d @airportDelays.json -H "Content-Type: application/json" "https://sldgxiipd6hbyw9-dabhajson.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/soda/latest/airportdelayscollection?action=insert"
```

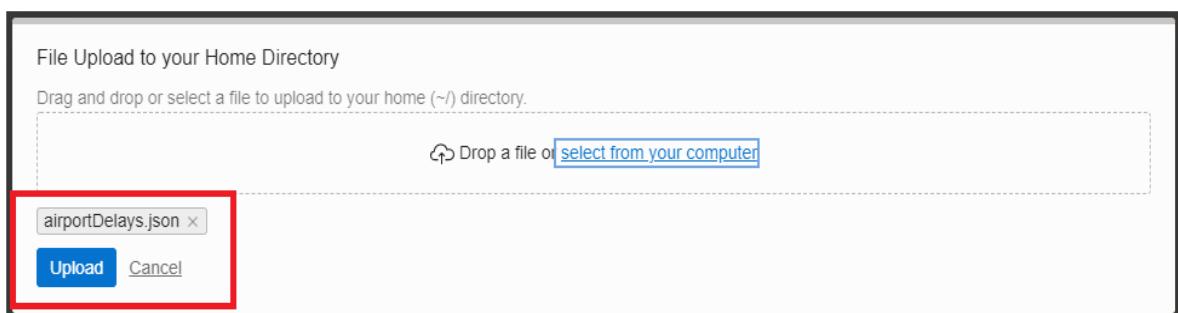
Previo a esto, el archivo *airportDelays.json* lo vamos a cargar en el ***Cloud Shell***, haciendo clic en la parte superior izquierda, después de desplegar el ***Cloud Shell***, luego en la opción ***Upload***:



Posterior a esto se desplegará una ventana, en la cual daremos en clic en **select from your computer**:



Buscamos el archivo y finalmente hacemos clic en **Upload**:



Con los pasos realizados anteriormente, ahora sí puedo ejecutar la siguiente instrucción en el **Cloud Shell**, esto con el fin de cargar el archivo en la colección que creamos previamente:

```
curl -u "GARY:xxxxxxxx" -i -X POST -d @airportDelays.json -H "Content-Type: application/json" "https://sldgxiipd6hbyw9-dabhajson.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/soda/latest/airportdelayscollection?action=insert"
```

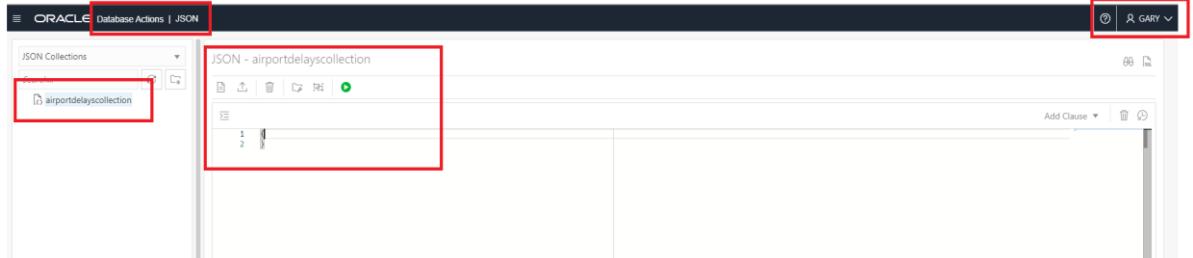
Este es el resultado después de hacer la carga del archivo a través del **Cloud Shell**:

Parte 3 – Trabajando con datos JSON en un almacén de documentos: QBE (Query By Example).

Una especificación de filtro es un patrón expresado en `JSON`. Se usa para seleccionar en una colección, los documentos `JSON` cuyo contenido coincide con él, lo que significa que la condición expresada por el patrón se evalúa como verdadera para el contenido de (solo) esos documentos.

Una especificación de filtro también se denomina consulta por ejemplo (QBE) o simplemente filtro.

Paso 1: Empecemos a trabajar con nuestros documentos en la colección de recogida de retrasos del aeropuerto. Usaremos la hoja de trabajo **JSON** en **Database Actions**. En la hoja de trabajo, tenemos el área principal, similar a lo que veríamos con la hoja de trabajo **SQL** o **SQL Developer**.



Una vez que estamos en la página principal, hacemos clic en el botón Ejecutar consulta (**Run Query**) en la barra de herramientas, esto ejecutará una consulta que

devuelve todos los documentos de nuestra colección en la sección inferior de la página, dado que emitimos un *QBE* vacío indicado por {}. Aquí es donde veremos los resultados de los siguientes *QBE* que ejecutaremos.

```
{
  "id": 1832,
  "AirportCode": "MCO",
  "Name": "Orlando, FL: Orlando International",
  "Time": {
    "Label": "2006/05",
    "Month": 5,
    "Month Name": "May",
    "Year": 2006
  },
  "Statistics": {
    "# of Delays": {
      "Carrier": 485,
      "Late Aircraft": 606,
      "National Aviation System": 545,
      "Security": 2,
      "Weather Codes": [
        "SNW",
        "RAIN",
        "SSN",
        "CLDY"
      ],
      "Weather": 56
    },
    "Carriers": [
      "Aircraft Types": [
        {
          "make": "Boeing",
          "model": [
            "737",
            "737",
            "737",
            "737"
          ]
        }
      ]
    ]
  }
}
```

Paso 2: Para empezar, podemos hacer una consulta similar a la que hicimos cuando trabajamos con datos relacionales. En esa primera instrucción *SQL* recuperó registros donde ***airportcode*** = 'SFO'. Podemos emitir un *QBE* que haga exactamente la misma búsqueda. Vamos a copiar y pegar el siguiente código en la hoja de trabajo *JSON* y a ejecutar la consulta:

```
{
  "AirportCode": "SFO"
}
```

Podemos ver que todos los resultados en la parte inferior de la página tienen el código de aeropuerto de ***SFO***.

```
{
  "id": 868,
  "AirportCode": "SFO",
  "Name": "San Francisco, CA: San Francisco International",
  "Time": {
    "Label": "2005/11",
    "Month": 11,
    "Month Name": "November",
    "Year": 2005
  },
  "Statistics": {
    "# of Delays": {
      "Carrier": 624,
      "Late Aircraft": 473,
      "National Aviation System": 1284,
      "Security": 6,
      "Weather Codes": [
        "SNW",
        "RAIN",
        "SSN",
        "CLDY"
      ],
      "Weather": 56
    },
    "Carriers": [
      "Aircraft Types": [
        {
          "make": "Boeing",
          "model": [
            "737",
            "737",
            "737",
            "737"
          ]
        }
      ]
    ]
  }
}
```

En cualquier momento, podemos hacer doble clic en un resultado en la parte inferior de la página para que aparezca el control deslizante con el contenido del documento JSON. Allí se puede ver el documento JSON completo.

```

1  {
2      "id": 868,
3      "AirportCode": "SFO",
4      "Name": "San Francisco, CA: San Francisco International",
5      "Time": {
6          "Label": "2005/11",
7          "Month": 11,
8          "Month Name": "November",
9          "Year": 2005
10     },
11     "Statistics": {
12         "# of Delays": {
13             "Carrier": 674,
14             "Late Aircraft": 473,
15             "National Aviation System": 1284,
16             "Security": 6,
17             "Weather Codes": [
18                 "SNW",
19                 "RAIN",
20                 "SUN",
21                 "CLDY"
22             ],
23             "Weather": 56
24         },
25         "Carriers": {
26             "Aircraft Types": [
27                 {
28                     "make": "Boeing",
29                     "models": [
30                         "717",
31                         "737",
32                         "757",
33                         "767",
34                         "777",
35                         "787"
36                     ]
37                 },
38                 {
39                     "make": "Airbus",
40                     "models": [
41                         "A320",
42                         "A321",
43                         "A330",
44                         "A340",
45                         "A350",
46                     ]
47                 }
48             ]
49         }
50     }
51 }

```

Paso 3: Podemos agregar al QBE como hicimos con el SQL y filtrar aún más los resultados a todos los documentos que también tienen el nombre del mes de junio y el año de 2010. Vamos a copiar y pegar este QBE en la hoja de trabajo y a ejecutar la consulta:

```
{
    "AirportCode": "SFO",
    "Time": {
        "Month Name": "June",
        "Year": 2010
    }
}
```

Este es el resultado de la consulta:

JSON - airportdelayscollection

```

1 {
2   "AirportCode": "SFO",
3   "Time": {
4     "Month Name": "June",
5     "Year": 2010
6   }
7 }

```

```

{
  "Id": 2463,
  "AirportCode": "SFO",
  "Name": "San Francisco, CA: San Francisco International",
  "Time": {
    "Label": "2010/06",
    "Month": 6,
    "Month Name": "June",
    "Year": 2010
  },
  "Statistics": {
    "# Delays": {
      "Carrier": 593,
      "Late Aircraft": 1048,
      "National Aviation System": 1648,
      "Security": 1
    },
    "Weather Codes": [
      "SNW",
      "RAIN",
      "SUN",
      "CLDY"
    ],
    "Weather": 85
  },
  "Carriers": {
    "Aircraft Types": [
      {
        "make": "Boeing",
        "models": [
          "737",
          "737",
          "737",
          "757",
          "767",
          "777"
        ]
      }
    ]
  }
}

```

Así como lo hicimos con *SQL*, podemos usar la etiqueta “**Label**” para encontrar exactamente el mismo conjunto de resultados:

```
{
  "AirportCode": "SFO",
  "Time": {
    "Label": "2010/06"
  }
}
```

Este es el resultado de la consulta:

JSON - airportdelayscollection

```

1 {
2   "AirportCode": "SFO",
3   "Time": {
4     "Label": "2010/06"
5   }
6 }

```

```

{
  "Id": 2463,
  "AirportCode": "SFO",
  "Name": "San Francisco, CA: San Francisco International",
  "Time": {
    "Label": "2010/06",
    "Month": 6,
    "Month Name": "June",
    "Year": 2010
  },
  "Statistics": {
    "# Delays": {
      "Carrier": 593,
      "Late Aircraft": 1048,
      "National Aviation System": 1648,
      "Security": 1,
      "Weather Codes": [
        "SNW",
        "RAIN",
        "SUN",
        "CLDY"
      ],
      "Weather": 85
    },
    "Carriers": {
      "Aircraft Types": [
        {
          "make": "Boeing",
          "models": [
            "737",
            "737",
            "737",
            "757",
            "767",
            "777"
          ]
        }
      ]
    }
  }
}

```

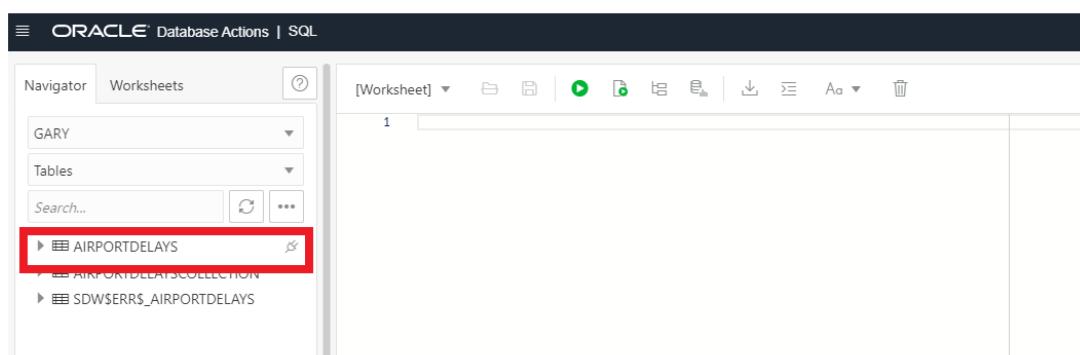
Exposición de los datos para aplicaciones

Objetivos

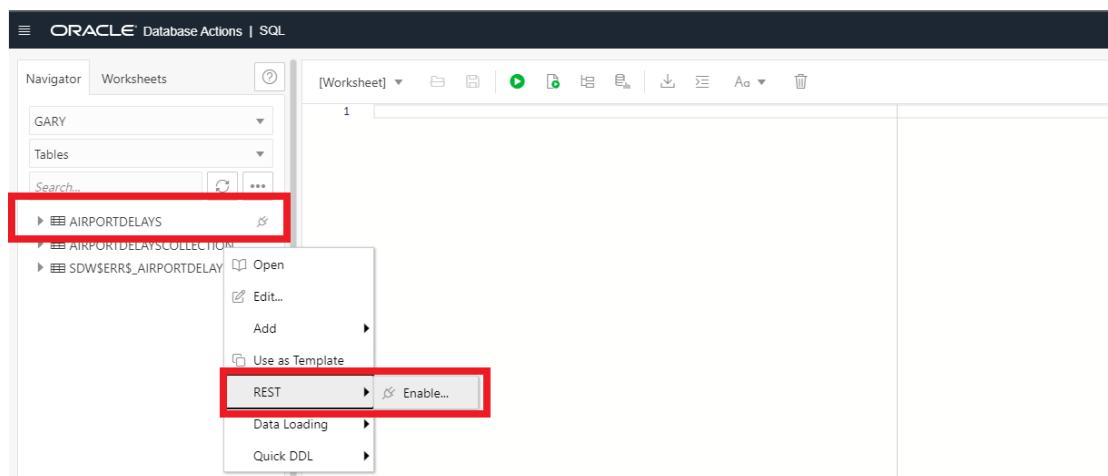
- Revisar cómo exponer datos a aplicaciones

Servicios RESTful en tablas relacionales

Paso 1: Habilitar una tabla relacional no podría ser más fácil. Para hacer esto, busque la tabla que creamos llamada **AIRPORTDELAYS** en el navegador a la izquierda de la hoja de trabajo SQL.



Paso 2: Hacemos clic derecho en el nombre de la tabla (**AIRPORTDELAYS**) y seleccionamos **REST** en el menú emergente y luego **Enable**.



Paso 3: Aparecerá una nueva ventana al lado derecho de la página. Mantenemos los valores predeterminados y cuando esté listo, haga clic en el botón **Enable** en la parte inferior derecha.

REST Enable Object

Object Name	Object Type
AIRPORTDELAYS	TABLE
Object Alias	airportdelays
Preview URL	https://sldgxiipd6byw9-dabha.json.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/airportdelays
Authorization Role	oracle.dbtools.role.autorest.GARY.AIRPORTDELAYS
Authorization Privilege	oracle.dbtools.autorest.privilege.GARY.AIRPORTDELAYS
Require Authentication	(radio button)

Enable **Cancel**

Paso 4: ¡Eso es todo! La tabla está habilitada para *REST*. Podemos trabajar con los puntos finales *REST* usando los comandos *cURL* que la hoja de trabajo *SQL* nos puede proporcionar. Para llegar a estos puntos finales, nuevamente haga clic derecho en el nombre de la tabla (**AIRPORTDELAYS**) como hicimos en el paso anterior, seleccione **REST**, luego **cURL Command**.

ORACLE Database Actions | SQL

The screenshot shows the Oracle Database Actions interface. On the left, there's a sidebar with 'Navigator' and 'Tables' sections. In the main area, a table named 'AIRPORTDELAYS' is selected. A context menu is open over the table name, with 'REST' highlighted. A submenu for 'REST' is displayed, containing 'cURL command...'. The 'cURL for the table AIRPORTDELAYS' section below shows a 'curl' command for the 'GET ALL' endpoint.

```
curl --location \
'https://sldgxiipd6byw9-dabha.json.adb.us-ashburn-1.oraclecloudapps.com/ords/GARY/airportdelays/'
```



Felicidades!
Usted completó el laboratorio
Hands-On de Oracle
Autonomous JSON Database