Create and Manage Cloud Spanner Instances: Challenge Lab

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

2. Click through the following windows:

   - Continue through the Cloud Shell information window.
   - Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `PROJECT_ID`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

4. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core]
project = "PROJECT_ID"
```

Note: For full documentation of `gcloud`, in Google Cloud, refer to [the gcloud CLI overview guide](#).

Challenge Scenario

In your role as the corporate Database Administrator, you have been tasked with standing up a new Cloud Spanner database for your company's Banking Operations group. You have been provided a list specifcations for this database related to tables datasets to load.

Task 1. Create a Cloud Spanner instance

1. Your first task is to create an instance.
2. You may complete this step using the Cloud Console or the gcloud CLI.
3. Your instance must have following attributes:

```
export REGION=us-east1
export BUCKET_NAME=qwiklabs-gcp-01-026491e087be
```

1. The first step in using Cloud Spanner is to create an instance. An instance is an allocation of Google Cloud compute and storage resources. From the Console, open

the navigation menu (☰) > **View All Products**. Under **Databases** section, click **Spanner**.

2. Accept any acknowledgement or information window that may appear.

3. Then click **Create a Provisioned Instance**.

4. Fill in the following fields, leave the remainder with the default values:

Item	**Value**
Name	**banking-ops-instance**
Region	`us-east1`
Allocate Compute Capacity	**Unit - Nodes // Quantity - 1**
Item	**Value**

5. Click **Create**. Now you can see your instance on the Instance Details page. Here you have an overview of how the instance is performing, utilization, etc.. The next step is to create a database.

Task 2. Create a database

1. From the instance details page, click **Create database**.

2. For the database name, enter **banking-ops-instance**.

3. Skip the **Define your schema (optional)** step for now. You'll define your schema in the next section.

4. Click **Create**.

5. You're now on the **Overview** page for the new database you created. You can see that the page is similar to the Instance one, but the statistics refer to the specific database. Also note the new options on the left menu.

Task 2. Create a Cloud Spanner database

1. Your next task is to create a database.

2. You may complete this step using the Cloud Console or the gcloud CLI.

3. Your database must have following attribute:

Item	**Value**
Name	**banking-ops-db**

An example gcloud CLI command to create a database is as follows:

```
gcloud spanner databases create my-sample-db \
--instance=my-sample-instance
```

Task 3. Create tables in your database

1. Your database must have a total of four (4) tables - **Portfolio**, **Category**, **Product**, and **Customer**.

2. The tables must be defined as listed below.

An example DDL command to create a table is as follows:

```
CREATE TABLE Sample (
  SampleId INT64 NOT NULL,
  SampleName STRING(MAX)
) PRIMARY KEY (SampleId);
```

Table: **Portfolio**

Primary Key: **PortfolioId**

Column	**Datatype**
PortfolioId	INT64 NOT NULL
Name	STRING(MAX)
ShortName	STRING(MAX)
PortfolioInfo	STRING(MAX)

Table: **Category**

Primary Key: **CategoryId**

Column	**Datatype**

CategoryId	INT64 NOT NULL
PortfolioId	INT64 NOT NULL
CategoryName	STRING(MAX)
PortfolioInfo	STRING(MAX)

Table: **Product**

Primary Key: **ProductId**

Column	**Datatype**
ProductId	INT64 NOT NULL
CategoryId	INT64 NOT NULL
PortfolioId	INT64 NOT NULL
ProductName	STRING(MAX)
ProductAssetCode	STRING(25)
ProductClass	STRING(25)

Table: **Customer**

Primary Key: **CustomerId**

Column	**Datatype**
CustomerId	STRING(36) NOT NULL
Name	STRING(MAX) NOT NULL
Location	STRING(MAX) NOT NULL

```
CREATE TABLE Portfolio (
  PortfolioId INT64 NOT NULL,
  Name STRING(MAX),
  ShortName STRING(MAX),
  PortfolioInfo STRING(MAX)
) PRIMARY KEY (PortfolioId);
```

```
CREATE TABLE Category (
  CategoryId INT64 NOT NULL,
  PortfolioId INT64 NOT NULL,
  CategoryName STRING(MAX),
  PortfolioInfo STRING(MAX)
) PRIMARY KEY (CategoryId);
```

```
CREATE TABLE Product (
  ProductId INT64 NOT NULL,
  CategoryId INT64 NOT NULL,
  PortfolioId INT64 NOT NULL,
  ProductName STRING(MAX),
  ProductAssetCode STRING(25),
  ProductClass STRING(25)
) PRIMARY KEY (ProductId);

CREATE TABLE Customer (
  CustomerId STRING(36) NOT NULL,
  Name STRING(MAX) NOT NULL,
  Location STRING(MAX) NOT NULL
) PRIMARY KEY (CustomerId);
```

Task 4. Load simple datasets into tables

1. Three of your tables, **Portfolio**, **Category**, and **Product**, will be loaded with simple, low-volume datasets.

2. You may employ any method to load these tables.

Note: The data elements provided are ordered to match the order of the columns of the corresponding table.

An example DML command to load a single row into a table is as follows:

```
INSERT INTO
  Sample (SampleId, SampleName)
VALUES
  (1, "Banking");
```

Table: **Portfolio**

```
1, "Banking", "Bnkg", "All Banking Business"
2, "Asset Growth", "AsstGrwth", "All Asset Focused Products"
3, "Insurance", "Insurance", "All Insurance Focused Products"
```

INSERT INTO Portfolio (PortfolioId, Name, ShortName, PortfolioInfo) VALUES

(1, "Banking", "Bnkg", "All Banking Business"),

(2, "Asset Growth", "AsstGrwth", "All Asset Focused Products"),

(3, "Insurance", "Insurance", "All Insurance Focused Products");

Table: **Category**

```
1,1,"Cash"
2,2,"Investments - Short Return"
3,2,"Annuities"
4,3,"Life Insurance"
```

INSERT INTO Category (CategoryId, PortfolioId, CategoryName, PortfolioInfo) VALUES

(1, 1, "Cash", NULL),

(2, 2, "Investments - Short Return", NULL),

(3, 2, "Annuities", NULL),

(4, 3, "Life Insurance", NULL);

Table: **Product**

```
 1,1,1,"Checking Account","ChkAcct","Banking LOB"
 2,2,2,"Mutual Fund Consumer Goods","MFundCG","Investment LOB"
 3,3,2,"Annuity Early Retirement","AnnuFixed","Investment LOB"
 4,4,3,"Term Life Insurance","TermLife","Insurance LOB"
 5,1,1,"Savings Account","SavAcct","Banking LOB"
 6,1,1,"Personal Loan","PersLn","Banking LOB"
 7,1,1,"Auto Loan","AutLn","Banking LOB"
 8,4,3,"Permanent Life Insurance","PermLife","Insurance LOB"
 9,2,2,"US Savings Bonds","USSavBond","Investment LOB"
```

```
INSERT INTO Product (ProductId, CategoryId, PortfolioId, ProductName,
ProductAssetCode, ProductClass) VALUES
  (1, 1, 1, "Checking Account", "ChkAcct", "Banking LOB"),
  (2, 2, 2, "Mutual Fund Consumer Goods", "MFundCG", "Investment LOB"),
  (3, 3, 2, "Annuity Early Retirement", "AnnuFixed", "Investment LOB"),
  (4, 4, 3, "Term Life Insurance", "TermLife", "Insurance LOB"),
  (5, 1, 1, "Savings Account", "SavAcct", "Banking LOB"),
  (6, 1, 1, "Personal Loan", "PersLn", "Banking LOB"),
  (7, 1, 1, "Auto Loan", "AutLn", "Banking LOB"),
  (8, 4, 3, "Permanent Life Insurance", "PermLife", "Insurance LOB"),
  (9, 2, 2, "US Savings Bonds", "USSavBond", "Investment LOB");
```

Task 5. Load a complex dataset

1. You will load the **Customer** table with a much larger set of data.

2. A file named **Customer_List_500.csv** contains 500 rows of data and is located in the following public Cloud Storage bucket. You may reference or download it as necessary.

gsutil URI

```
gs://cloud-training/OCBL375/Customer_List_500.csv
```

HTTP URL

```
https://storage.googleapis.com/cloud-
training/OCBL375/Customer_List_500.csv
```

3. You may recall from the lab **Cloud Spanner - Loading Data and Performing Backups** that a few options exist to load larger datasets. These include using Dataflow or a client library in Batch mode. You may choose to create simple insert statements. The decision is yours but you **must** load all 500 rows.

4. Utilize any method that you prefer to load the 500 row datafile. Some methods will require edits to the datafile which will require downloading it to your local machine. Please be sure to make a backup file if you choose that option.

5. **Note**: if you use Dataflow should ensure that you specify the `us-east1` Regional Endpoint and reset the Dataflow API via the following gcloud command:

```
gcloud services disable dataflow.googleapis.com --force
gcloud services enable dataflow.googleapis.com
```

Copied!

content_copy

6. Also if you use a Dataflow template you will be required to provide a Mainfest file named **manifest.json**. Below is a sample **manifest.json** that you can use to guide creation of a file appropriate for loading the **Customer** table.

Note: This sample **cannot be used as is**, you must update it accordingly.

Note:

```
{
    "tables": [
        {
            "table_name": "TABLE_NAME",
            "file_patterns": [
                "gs://BUCKET/FOLDER/FILENAME.SUFFIX"
            ],
            "columns": [
                {"column_name" : "UPDATE_COLUMN1", "type_name" :
"UPDATE_DATATYPE" },
                {"column_name" : "UPDATE_COLUMN2", "type_name" :
"UPDATE_DATATYPE" },
                {"column_name" : "UPDATE_COLUMN3", "type_name" :
"UPDATE_DATATYPE" }
            ]
        }
    ]
}
```

SOLUTION:

Download the csv file

```
gsutil cp gs://cloud-training/OCBL375/Customer_List_500.csv .
```

Enable APIs

```
gcloud services enable dataflow.googleapis.com
gcloud services enable spanner.googleapis.com
```

Create manifest.json file

```
cat > manifest.json << EOF_CP
{
  "tables": [
    {
      "table_name": "Customer",
      "file_patterns": [
        "gs://cloud-training/OCBL375/Customer_List_500.csv"
      ],
      "columns": [
        {"column_name" : "CustomerId", "type_name" : "STRING" },
        {"column_name" : "Name", "type_name" : "STRING" },
        {"column_name" : "Location", "type_name" : "STRING" }
      ]
    }
  ]
}
EOF_CP
```

Create S3 bucket

```
gcloud storage buckets create gs://$BUCKET_NAME --location=$REGION
gsutil cp manifest.json gs://$BUCKET_NAME
```

Run datafow job

```
gcloud dataflow jobs run spanner-import-job \
    --gcs-location=gs://dataflow-templates/latest/GCS_Text_to_Cloud_Spanner \
    --region=$REGION\
    --parameters=instanceId="banking-ops-instance",databaseId="banking-ops-
db",importManifest="gs://$BUCKET_NAME/manifest.json"
```

Task 6. Add a new column to an existing table

1. As part of your DBA responsibilities you are required to add a new column called **MarketingBudget** to the **Category** table.

2. The column **MarketingBudget** must have a datatype of **INT64**.

3. Adding a new column is accomplished by a DDL command. You may issue the DDL via a gcloud command, the Cloud Console, or client library call.

An example gcloud CLI command to add a column to a table is as follows:

```
gcloud spanner databases ddl update my-sample-db \
--instance=my-sample-instance \
--ddl='ALTER TABLE Sample ADD COLUMN SampleValue INT64;'
```

SOLUTION:

```
gcloud spanner databases ddl update banking-ops-db \
--instance=banking-ops-instance \
--ddl="ALTER TABLE Category ADD COLUMN MarketingBudget INT64;"
```

Verify the data

```
gcloud spanner databases ddl describe banking-ops-db --instance=banking-ops-
instance
```