

# Data Employees

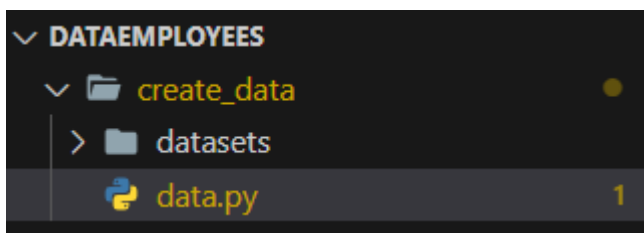
Juan Carlos Gomez Berrio

Laboratorio de ingeniería de datos y Cloud Computing

Link Repositorio: <https://github.com/BerrioGB/DataEmployees>

## Creación Automática de los datos

Para este proceso se guardó dentro de la carpeta create\_data el archivo data.py



el cual consiste en los siguientes pasos:

1. Importar las librerías que fueron necesarios las cuales son pandas y numpy
2. definir los diccionarios de datos que se usarán: departamentos\_data, jobs\_data, empleados\_data estos contendrán tanto los identificadores como los nombres de los departamentos, cargos y empleados.
3. Se genera la tabla principal la cual es Empleados automáticamente asignando identificadores del 1 al 101, asignando 100 nombres automáticos los cuales se denominará "Nombre 1, Nombre 2", etc hasta 100. mientras que para el identificador del cargo se asignará de manera aleatoria entre las posibilidades que se agregaron previamente

```
# Empleados
np.random.seed(0)
empleados_data = {
    "empleado_id": np.arange(1, 101),
    "Nombre_Empleado": [f"Nombre {i}" for i in range(1, 101)],
    "job_id": np.random.choice(["101", "102", "201", "202", "301", "302"], 100)
}
```

4. Se generan los data frame para una visualización previa de los datos, con la novedad de que al data frame de empleados en base a su identificador de cargo se extrae el primer número de su cargo para definir el departamento al que pertenece su cargo, gracias a esto ningún cargo va poder estar dentro de un departamento incorrecto.

```
# Conversion a dataframe
departamentos_df = pd.DataFrame(departamentos_data)
jobs_df = pd.DataFrame(jobs_data)
empleados_df = pd.DataFrame(empleados_data)

empleados_df["Departamento_id"] = empleados_df["job_id"].str[0]

#impresiones
print("Departamentos DataFrame:")
print(departamentos_df)

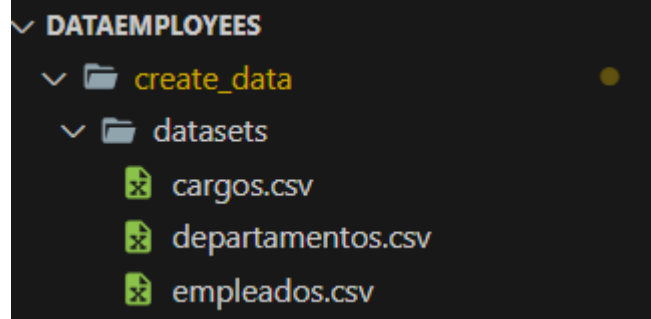
print("\nCargos DataFrame:")
print(jobs_df)

print("\nEmpleados DataFrame:")
print(empleados_df)
```

# Exportar datos a csv

Luego de verificar la calidad de los datos generados automáticamente con unas pocas líneas de código logramos exportar los data frame que teníamos a formato csv dentro de una carpeta llamada data sets donde estarán almacenados, se escogió el formato csv ya que es el mas optimo por la cantidad de los datos además de ser ampliamente compatible con todo tipo de aplicaciones, sistemas y bases de datos.

```
43 #Exportando los datos a csv
44 departamentos_df.to_csv("./datasets/departamentos.csv",index=False)
45 jobs_df.to_csv("./datasets/cargos.csv",index=False)
46 empleados_df.to_csv("./datasets/empleados.csv",index=False)
```

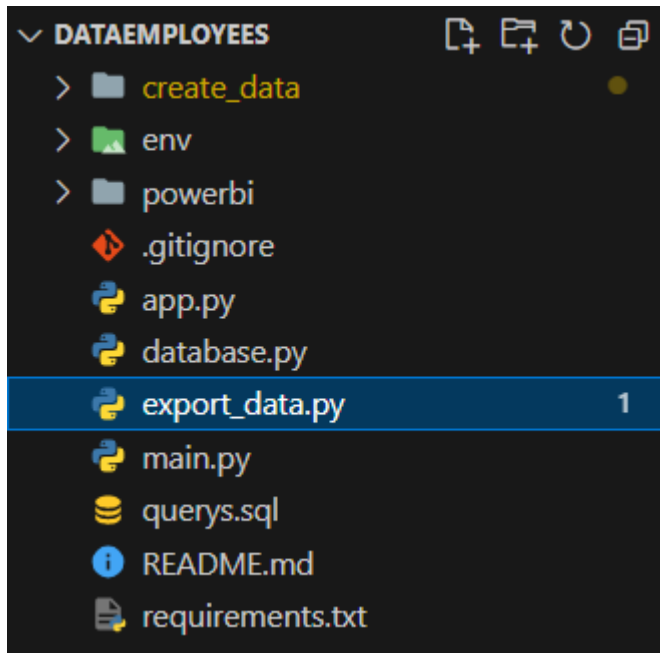


```
DATAEMPLOYEES
├── create_data
└── datasets
    ├── cargos.csv
    ├── departamentos.csv
    └── empleados.csv
```

# Exportar archivos csv a Azure

Migramos estos archivos csv hacia la nube de azure almacenando nuestros csv dentro de un contenedor de una cuenta de almacenamiento para esto necesitamos el siguiente código que se encuentra en el archivo `export_data.py` ubicado en la raíz del proyecto

Para este proceso necesitaremos la librería `azure.storage.blob`



1. Se realiza la conexión con el contenedor de azure

```
# Credenciales y nombre del contenedor
connection_string = "DefaultEndpointsProtocol=https;AccountName=
container_name = "empleadoscontainer"
```

2. Se crea el objeto Blob Service Client que permite interactuar con el servicio de Azure Blob Storage donde se subirán los archivos csv

```
# Crear cliente BlobServiceClient
blob_service_client = BlobServiceClient.from_connection_string(connection_string)
```

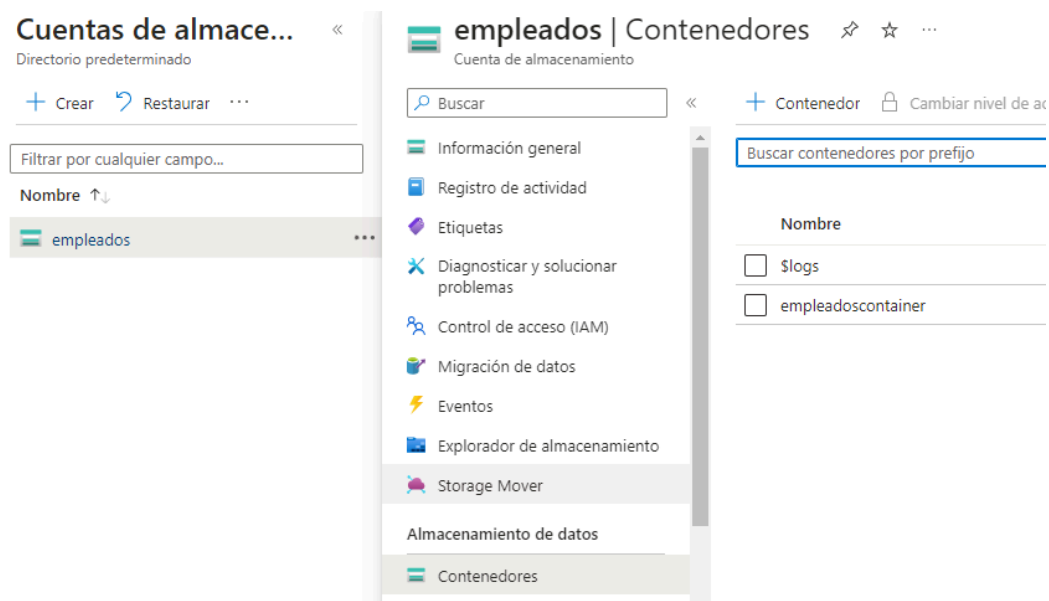
3. Cargamos los archivos csv desde su respectiva carpeta data sets y utilizamos un ciclo que nos permite recorrer la lista de csv y subirlos de manera automática dentro del contenedor de azure.

```
# Cargar archivo CSV
archivos_csv = ["create_data/datasets/cargos.csv", "create_data/datasets/departamentos.csv"]

#Cargar archivos
for archivo in archivos_csv:
    with open(archivo, "rb") as data:
        blob_client = blob_service_client.get_blob_client(container_name, archivo)
        blob_client.upload_blob(data)

print("Archivo CSV cargado en Azure Blob Storage.")
```

4. El proceso que se realiza a la nube previo al código es crear la cuenta de almacenamiento llamada en este caso empleados, posteriormente creamos en la pestaña contenedores un contenedor que almacena los csv denominado empleados container.





# empleadoscontainer ...

Contenedor

Buscar <<

Cargar Cambiar nivel de acceso Actualizar

Información general

Diagnosticar y solucionar problemas

Control de acceso (IAM)

## Configuración

Tokens de acceso compartido

Directiva de acceso

Propiedades

Metadatos

**Método de autenticación:** Clave de acceso ([Cambiar a la configuración de autenticación](#))  
**Ubicación:** [empleadoscontainer](#) / [create\\_data](#) / [datasets](#)

Buscar blobs por prefijo (distingue mayúsculas de minúsculas)

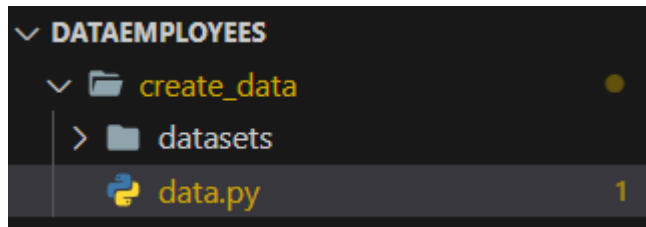
Agregar filtro

### Nombre

- |                          |                   |
|--------------------------|-------------------|
| <input type="checkbox"/> | [..]              |
| <input type="checkbox"/> | cargos.csv        |
| <input type="checkbox"/> | departamentos.csv |
| <input type="checkbox"/> | empleados.csv     |

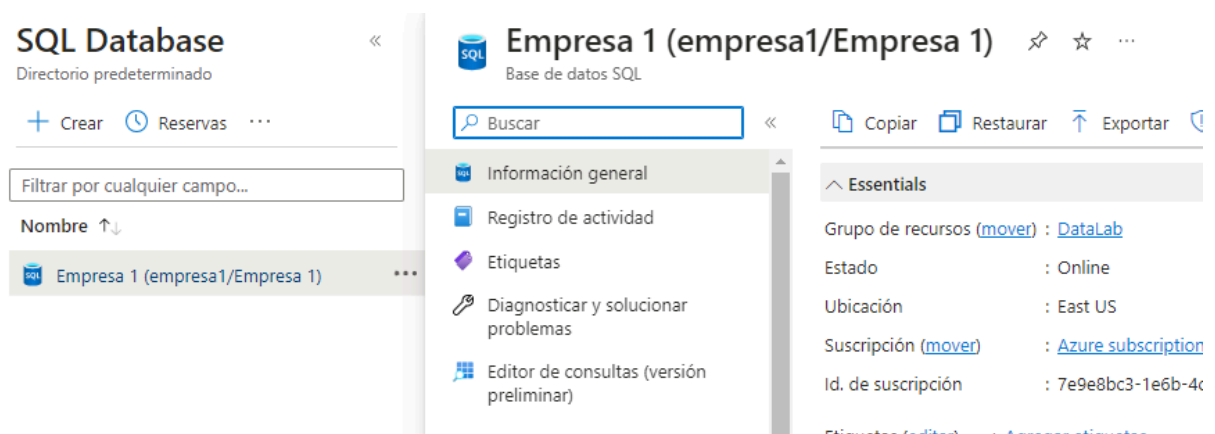
# Migrar Datos a SQL Server Azure

este proceso se encuentra en el archivo data.py donde en primera instancia importamos la libreria pyodbc



1. Se crea la base de datos sql dentro del portal de azure y realizamos la conexión a la misma con las credenciales respectivas

Error: En esta parte se cometió el error de no usar variables de entorno y usar las credenciales directamente en el código



```
48  ###inyeccion de los datos
49  import pyodbc
50
51  connection_string = 'Driver={SQL Server};Server=tcp:empresa1.database.windows.net,14
52
53  connection = pyodbc.connect(connection_string)
54
55  cursor = connection.cursor()
56
```

2. Creamos una variable para almacenar la query de inserción de los datos dejando espacio para los valores que se asignan posteriormente mediante un ciclo que recorre cada data frame usando el método iterrows y ejecutando el query con los valores extraídos de los dataframe.

```
###Insertar Jobs o cargos
insert_query = """INSERT INTO cargos (job_id, cargo) VALUES (?, ?)"""

for index, row in jobs_df.iterrows():
    job_id = row['job_id']
    cargo = row['cargo']
    cursor.execute(insert_query, (job_id, cargo))
```

```
66  ###Insertar Departamentos
67  insert_query = """INSERT INTO departamentos (Departamento_id, Departamento) VALUES (?, ?)"""
68
69  for index, row in departamentos_df.iterrows():
70      Departamento_id = row['Departamento_id']
71      Departamento = row['Departamento']
72      cursor.execute(insert_query, (Departamento_id, Departamento))
```

```
###Insertar Empleados
insert_query = """INSERT INTO empleados (empleado_id, Nombre_Empleado, job_id, Departamento_id) VALUES (?, ?, ?, ?)"""

for index, row in empleados_df.iterrows():
    empleado_id = row['empleado_id']
    Nombre_Empleado = row['Nombre_Empleado']
    job_id = row['job_id']
    Departamento_id = row['Departamento_id']

    cursor.execute(insert_query, (empleado_id, Nombre_Empleado, job_id, Departamento_id))
```

3. Por último notificamos los cambios a la base de datos y cerramos la conexión a la misma para evitar el gasto de recursos

```
### Cerrar la conexion para evitar el consumo de recursos
connection.commit()
cursor.close()
connection.close()
```



Verificamos que los datos se hayan migrado correctamente a la base de datos en Azure

The screenshot shows the Azure Data Studio interface. On the left, the 'Enterprise 1 (adminsql)' server is connected, and the 'Tables' folder is expanded, showing 'dbo.cargos', 'dbo.departamentos', 'dbo.empleados', and 'dbo.sysdiagrams'. The main editor shows a SQL query: `select * from [dbo].[empleados]`. Below the editor, the 'Results' tab is active, displaying a table with 15 rows of employee data.

empleado_id	Nombre_Empleado	job_id	Departamento_id
1	Nombre 1	301	3
10	Nombre 10	201	2
100	Nombre 100	101	1
11	Nombre 11	301	3
12	Nombre 12	101	1
13	Nombre 13	101	1
14	Nombre 14	301	3
15	Nombre 15	201	2

## Desarrollo de la query

Se desarrollaron tres queries las cuales se encuentran en un archivo llamado queries.sql estos consisten en:

1. Query que muestra el nombre del empleado, su cargo y el departamento al que pertenece

```
/*Consulta que muestra el nombre del empleado
SELECT
    E.Nombre_Empleado,
    C.cargo,
    D.Departamento
FROM empleados AS E
JOIN departamentos AS D
ON E.departamento_id = D.departamento_id
JOIN cargos as C
ON E.job_id = C.job_id;
GO
```

Nombre_Empleado	cargo	Departamento
Nombre 1	Director de recursos humanos	RRHH
Nombre 10	Community manager	Marketing
Nombre 100	Analista desarrollo	TI
Nombre 11	Director de recursos humanos	RRHH
Nombre 12	Analista desarrollo	TI
Nombre 13	Analista desarrollo	TI
Nombre 14	Director de recursos humanos	RRHH
Nombre 15	Community manager	Marketing

2. Query que trae los datos de todas las tres tablas

```
SELECT
    E.Empleado_id,
    E.Nombre_Empleado,
    C.job_id,
    C.cargo,
    D.Departamento_id,
    D.Departamento
FROM empleados AS E
JOIN departamentos AS D
ON E.departamento_id = D.departamento_id
JOIN cargos as C
ON E.job_id = C.job_id;
GO
```

Empleado_id	Nombre_Empleado	job_id	cargo	Departamento_id	Departamento
1	Nombre 1	301	Director de recursos hu...	3	RRHH
10	Nombre 10	201	Community manager	2	Marketing
100	Nombre 100	101	Analista desarrollo	1	TI
11	Nombre 11	301	Director de recursos hu...	3	RRHH
12	Nombre 12	101	Analista desarrollo	1	TI
13	Nombre 13	101	Analista desarrollo	1	TI
14	Nombre 14	301	Director de recursos hu...	3	RRHH
15	Nombre 15	201	Community manager	2	Marketing

3. Query que trae solamente los ids de todas las tablas

```
SELECT
    E.Empleado_id,
    C.job_id,
    D.Departamento_id
FROM empleados AS E
JOIN departamentos AS D
ON E.departamento_id = D.departamento_id
JOIN cargos as C
ON E.job_id = C.job_id;
GO
```

Empleado_id	job_id	Departamento_id
1	301	3
10	201	2
100	101	1
11	301	3
12	101	1
13	101	1
14	301	3

## Desarrollo de API en Flask

Las librerías necesarias para llevar a cabo la api son collections, flask y pyodbc

1. El primer paso es crear la conexión con la base de datos igual que en procedimientos anteriores

```
#Conexion a la base de datos

connection_string = 'Driver={SQL Server};Server=tcp:empresa1.databa

connection = pyodbc.connect(connection_string)

cursor = connection.cursor()
```

2. utilizando el `app.route` creamos la ruta raiz que cargara por defecto todos los campos de todas las tablas, para esto agregaremos dentro de la funcion principal `root` habilitar la conexion con la base de datos

```
#Ruta Principal: Consulta core que trae todos los  
@app.route('/')  
def root():  
    try:  
        conn = pyodbc.connect(connection_string)  
        cursor = conn.cursor()
```

3. Traeremos el query correspondiente para la consulta que necesitamos y lo guardaremos en una variable para a su vez ejecutarlo y guardar todas las filas mediante el `fetchall`

```
query = """  
    SELECT  
        E.Empleado_id,  
        E.Nombre_Empleado,  
        C.job_id,  
        C.cargo,  
        D.Departamento_id,  
        D.Departamento  
    FROM empleados AS E  
    JOIN departamentos AS D  
    ON E.departamento_id = D.departamento_id  
    JOIN cargos as C  
    ON E.job_id = C.job_id  
    """  
  
cursor.execute(query)  
rows = cursor.fetchall()
```

4. Los datos que recibiremos los organizaremos mediante un ciclo for que nos permitirá organizar la entrada de los datos mediante un diccionario que sera el que se retorna como respuesta

```
#La data que se recibe se convierte en diccionario
data = []
for row in rows:
    new_dict = {
        'empleado_id': row[0],
        'Nombre_Empleado': row[1],
        'job_id': row[2],
        'cargo': row[3],
        'Departamento_id': row[4],
        'Departamento': row[5]
    }

    data.append(new_dict)

return jsonify(data)
```

5. Adicional a esto usamos en todo este proceso un manejo de error que ante un potencial error de la base de datos nos lo notificara y finalmente cerrando la conexión.

```
except pyodbc.Error as ex:
    print("Database connection error:", ex)
    return jsonify({'error': 'Error connecting to database'}), 500

finally:
    if conn:
        conn.close()
```

Utilizando Thunder Client podemos verificar el uso de la api que creamos y sus rutas

The screenshot displays the Thunder Client interface. At the top, a request is configured with the method 'GET' and the URL 'http://127.0.0.1:5000/'. Below this, the 'Query' tab is selected, showing the response details. The status is '200 OK', the size is '18.02 KB', and the time is '275 ms'. The 'Response' tab is active, showing a JSON array with two objects. A 'Copy' button is visible on the right side of the response area.

```
1  [  
2    {  
3      "Departamento": "RRHH",  
4      "Departamento_id": "3",  
5      "Nombre_Empleado": "Nombre 1",  
6      "cargo": "Director de recursos humanos",  
7      "empleado_id": "1",  
8      "job_id": "301"  
9    },  
10   {  
11     "Departamento": "Marketing",  
12     "Departamento_id": "2",  
13     "Nombre_Empleado": "Nombre 10",  
14     "cargo": "Community manager",  
15     "empleado_id": "10",  
16     "job_id": "201"  
17   },  
18 ]
```

GET ⌵ http://127.0.0.1:5000/empleados

Send

Status: 200 OK   Size: 10.8 KB   Time: 4.85 s

Response


Headers <sup>5</sup>

Cookies

Results

Docs

```
1  [
2    {
3      "Cargo": "Director de recursos humanos",
4      "Departamento": "RRHH",
5      "Nombre_Empleado": "Nombre 1"
6    },
7    {
8      "Cargo": "Community manager",
9      "Departamento": "Marketing",
10     "Nombre_Empleado": "Nombre 10"
11   },
12   {
13     "Cargo": "Analista desarrollo",
14     "Departamento": "TI",
15     "Nombre_Empleado": "Nombre 100"
16   },
17   {
18     "Cargo": "Director de recursos humanos",
19     "Departamento": "RRHH",
20     "Nombre_Empleado": "Nombre 11"
21   },
```

GET  http://127.0.0.1:5000/ids

Send

Status: 200 OK   Size: 8 KB   Time: 671 ms

Response

Headers <sup>5</sup>

Cookies

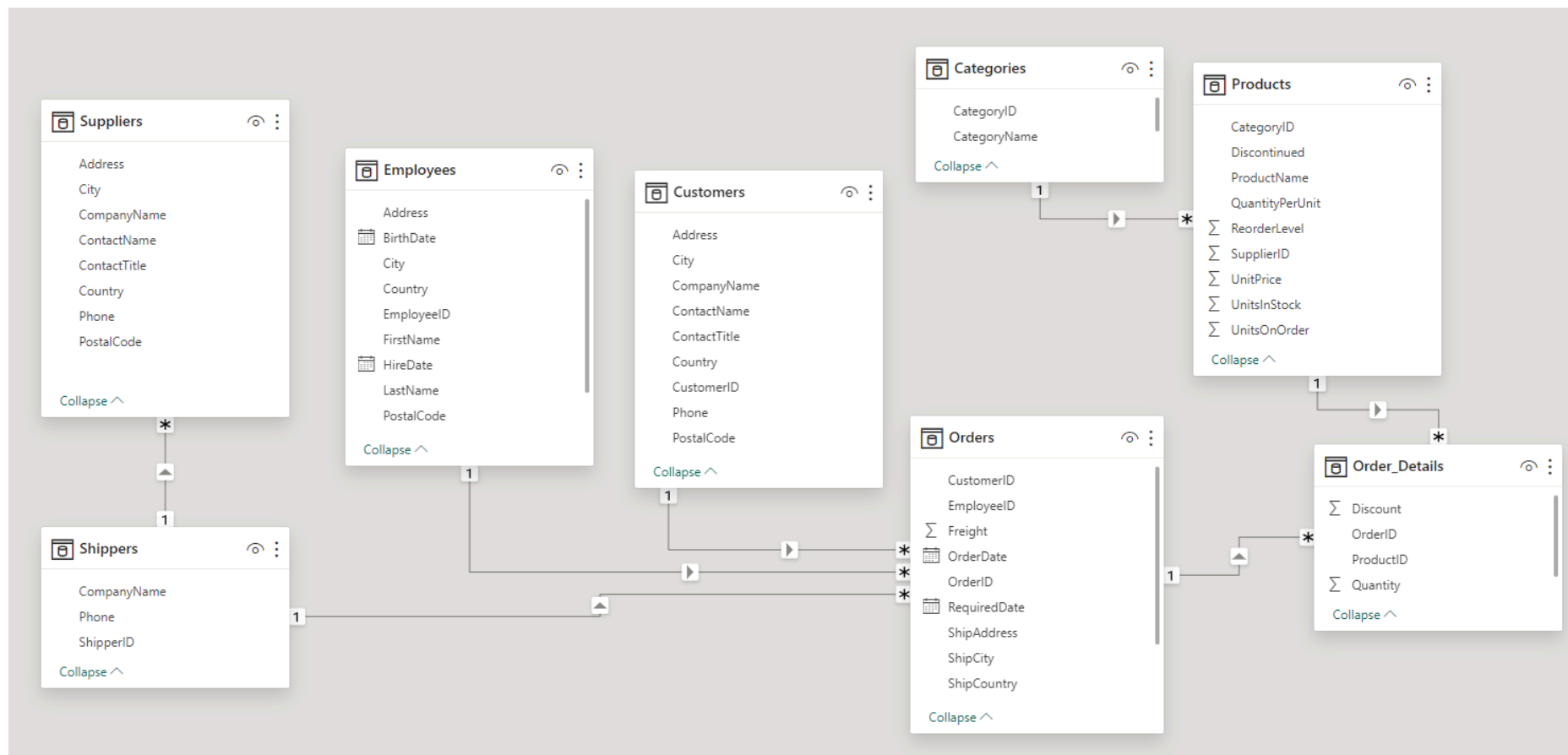
Results

Docs

```
1  [
2    {
3      "Departamento_id": "3",
4      "empleado_id": "1",
5      "job_id": "301"
6    },
7    {
8      "Departamento_id": "2",
9      "empleado_id": "10",
10     "job_id": "201"
11   },
12   {
13     "Departamento_id": "1",
14     "empleado_id": "100",
15     "job_id": "101"
16   }
```



# Modelo Dimensional



# Modelamiento de los datos

1. Se extraen los datos de la url mediante el OData
2. Se escogen las tablas que necesitamos las cuales se sacaron a partir del diagrama que se compartió inicialmente
3. Procedemos con la limpieza de los datos llevando a cabo la eliminación de columnas las cuales tengan muchos datos vacíos o repetidos y eliminando datos irrelevantes
4. Dentro de la limpieza y preparación de los datos también se llevó a cabo la creación de unas columnas restantes de identificadores los cuales son necesarios para llevar a cabo el relacionamiento
5. Como resultado de la limpieza de las tablas tenemos como resultado tablas normalizadas que los id cumplan con la condición de que sean diferentes y únicos en igualdad de cantidad estas dos condiciones, columnas sin filas repetidas y sin espacios en blanco
6. En base a la preparación de los datos llevamos a cabo las relaciones y la estructura del modelo teniendo en cuenta nuestros hechos principales que son las órdenes

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	PostalCode	Country
1	Davolio	Nancy	Sales Representative	Ms.	08/12/1948 12:00:00 a. m.	01/05/1992 12:00:00 a. m.	507 - 20th Ave. E. Apt. 2A	Seattle	98122	USA
2	Fuller	Andrew	Vice President, Sales	Dr.	19/02/1952 12:00:00 a. m.	14/08/1992 12:00:00 a. m.	908 W. Capital Way	Tacoma	98401	USA
3	Leverling	Janet	Sales Representative	Ms.	30/08/1963 12:00:00 a. m.	01/04/1992 12:00:00 a. m.	722 Moss Bay Blvd.	Kirkland	98033	USA
4	Peacock	Margaret	Sales Representative	Mrs.	19/09/1937 12:00:00 a. m.	03/05/1993 12:00:00 a. m.	4110 Old Redmond Rd.	Redmond	98052	USA
5	Buchanan	Steven	Sales Manager	Mr.	04/03/1955 12:00:00 a. m.	17/10/1993 12:00:00 a. m.	14 Garrett Hill	London	SW1 8JR	UK
6	Suyama	Michael	Sales Representative	Mr.	02/07/1963 12:00:00 a. m.	17/10/1993 12:00:00 a. m.	Coventry House Miner Rd.	London	EC2 7JR	UK
7	King	Robert	Sales Representative	Mr.	29/05/1960 12:00:00 a. m.	02/01/1994 12:00:00 a. m.	Edgeham Hollow Winchester Way	London	RG1 9SP	UK
8	Callahan	Laura	Inside Sales Coordinator	Ms.	09/01/1958 12:00:00 a. m.	05/03/1994 12:00:00 a. m.	4726 - 11th Ave. N.E.	Seattle	98105	USA
9	Dodsworth	Anne	Sales Representative	Ms.	27/01/1966 12:00:00 a. m.	15/11/1994 12:00:00 a. m.	7 Houndstooth Rd.	London	WG2 7LT	UK

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	PostalCode	Country	Phone
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	12209	Germany	030-0074321
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	05021	Mexico	(5) 555-4729
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	05023	Mexico	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	WA1 1DP	UK	(171) 555-7788
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå	S-958 22	Sweden	0921-12 34 65
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	68306	Germany	0621-08460
BLONP	Blondesdls père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg	67000	France	88.60.15.31
BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67	Madrid	28023	Spain	(91) 555 22 82
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	13008	France	91.24.45.40
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada	(604) 555-4729
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London	EC2 5NT	UK	(171) 555-1212
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires	1010	Argentina	(1) 135-5555
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993	México D.F.	05022	Mexico	(5) 555-3392
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	3012	Switzerland	0452-076545
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas, 23	Sao Paulo	05432-043	Brazil	(11) 555-7647
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK	(171) 555-2282
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator	Walsertweg 21	Aachen	52066	Germany	0241-039123
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinquante Otages	Nantes	44000	France	40.67.88.88

CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress	ShipCity	ShipCountry	OrderID
HANAR	4	08/07/1996 12:00:00 a. m.	05/08/1996 12:00:00 a. m.	12/07/1996 12:00:00 a. m.	2	65.83	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	Brazil	3
SUPRD	4	09/07/1996 12:00:00 a. m.	06/08/1996 12:00:00 a. m.	11/07/1996 12:00:00 a. m.	2	51.3	Suprêmes délices	Boulevard Tirou, 255	Charleroi	Belgium	5
QUEDE	4	19/07/1996 12:00:00 a. m.	16/08/1996 12:00:00 a. m.	30/07/1996 12:00:00 a. m.	2	3.05	Que Delícia	Rua da Panificadora, 12	Rio de Janeiro	Brazil	14
RATTC	4	30/08/1996 12:00:00 a. m.	27/09/1996 12:00:00 a. m.	05/09/1996 12:00:00 a. m.	2	147.26	Rattlesnake Canyon Grocery	2817 Milton Dr.	Albuquerque	USA	47
RICAR	4	06/09/1996 12:00:00 a. m.	04/10/1996 12:00:00 a. m.	13/09/1996 12:00:00 a. m.	2	29.76	Ricardo Adocicados	Av. Copacabana, 267	Rio de Janeiro	Brazil	52
SUPRD	4	10/09/1996 12:00:00 a. m.	08/10/1996 12:00:00 a. m.	09/10/1996 12:00:00 a. m.	2	6.27	Suprêmes délices	Boulevard Tirou, 255	Charleroi	Belgium	55
ISLAT	4	26/09/1996 12:00:00 a. m.	24/10/1996 12:00:00 a. m.	03/10/1996 12:00:00 a. m.	2	41.76	Island Trading	Garden House Crowther Way	Cowes	UK	68
BOLID	4	10/10/1996 12:00:00 a. m.	07/11/1996 12:00:00 a. m.	14/10/1996 12:00:00 a. m.	2	77.92	Bólido Comidas preparadas	C/ Araquil, 67	Madrid	Spain	79
SPLIR	4	15/10/1996 12:00:00 a. m.	26/11/1996 12:00:00 a. m.	23/10/1996 12:00:00 a. m.	2	191.67	Split Rail Beer & Ale	P.O. Box 555	Lander	USA	82
FRANK	4	30/10/1996 12:00:00 a. m.	13/11/1996 12:00:00 a. m.	04/11/1996 12:00:00 a. m.	2	54.83	Frankenversand	Berliner Platz 43	München	Germany	95
WHITC	4	01/11/1996 12:00:00 a. m.	29/11/1996 12:00:00 a. m.	05/11/1996 12:00:00 a. m.	2	23.29	White Clover Markets	1029 - 12th Ave. S.	Seattle	USA	97
WANDK	4	07/11/1996 12:00:00 a. m.	05/12/1996 12:00:00 a. m.	15/11/1996 12:00:00 a. m.	2	0.78	Die Wandernde Kuh	Adenauerallee 900	Stuttgart	Germany	101
BOTTM	4	20/12/1996 12:00:00 a. m.	17/01/1997 12:00:00 a. m.	24/12/1996 12:00:00 a. m.	2	47.42	Bottom-Dollar Markets	23 Tsawassen Blvd.	Tsawassen	Canada	142
RICSU	4	20/01/1997 12:00:00 a. m.	17/02/1997 12:00:00 a. m.	30/01/1997 12:00:00 a. m.	2	137.35	Richter Supermarkt	Starenweg 5	Genève	Switzerland	172
PICCO	4	27/01/1997 12:00:00 a. m.	24/02/1997 12:00:00 a. m.	03/03/1997 12:00:00 a. m.	2	31.29	Piccolo und mehr	Geisweg 14	Salzburg	Austria	180
BOTTM	4	30/01/1997 12:00:00 a. m.	13/02/1997 12:00:00 a. m.	07/02/1997 12:00:00 a. m.	2	44.17	Bottom-Dollar Markets	23 Tsawassen Blvd.	Tsawassen	Canada	184
SAVEA	4	10/02/1997 12:00:00 a. m.	10/03/1997 12:00:00 a. m.	28/02/1997 12:00:00 a. m.	2	86.53	Save-a-lot Markets	187 Suffolk Ln.	Boise	USA	193
RICAR	4	14/02/1997 12:00:00 a. m.	14/03/1997 12:00:00 a. m.	07/03/1997 12:00:00 a. m.	2	68.66	Ricardo Adocicados	Av. Copacabana, 267	Rio de Janeiro	Brazil	200
RANCH	4	17/02/1997 12:00:00 a. m.	17/03/1997 12:00:00 a. m.	24/02/1997 12:00:00 a. m.	2	38.82	Rancho grande	Av. del Libertador 900	Buenos Aires	Argentina	201
VICTE	4	27/02/1997 12:00:00 a. m.	27/03/1997 12:00:00 a. m.	28/02/1997 12:00:00 a. m.	2	25.09	Victualles en stock	2, rue du Commerce	Lyon	France	212
FURIB	4	04/03/1997 12:00:00 a. m.	01/04/1997 12:00:00 a. m.	14/03/1997 12:00:00 a. m.	2	89	Furia Bacalhau e Frutos do Mar	Jardim das rosas n. 32	Lisboa	Portugal	217
BONAP	4	11/03/1997 12:00:00 a. m.	08/04/1997 12:00:00 a. m.	14/03/1997 12:00:00 a. m.	2	64.56	Bon app'	12, rue des Bouchers	Marseille	France	223
LINOD	4	25/03/1997 12:00:00 a. m.	08/04/1997 12:00:00 a. m.	31/03/1997 12:00:00 a. m.	2	64.45	LINO-Delicateses	Ave. 5 de Mayo Porlamar	I. de Margarita	Venezuela	238
COMMI	4	02/04/1997 12:00:00 a. m.	30/04/1997 12:00:00 a. m.	09/04/1997 12:00:00 a. m.	2	65.99	Comércio Mineiro	Av. dos Lusíadas, 23	Sao Paulo	Brazil	247
LILAS	4	08/04/1997 12:00:00 a. m.	06/05/1997 12:00:00 a. m.	16/04/1997 12:00:00 a. m.	2	102.02	LILA-Supermercado	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto	Venezuela	252
TORTU	4	25/04/1997 12:00:00 a. m.	09/05/1997 12:00:00 a. m.	05/05/1997 12:00:00 a. m.	2	218.15	Tortuga Restaurante	Avda. Azteca 123	México D.F.	Mexico	271
WARTH	4	05/05/1997 12:00:00 a. m.	02/06/1997 12:00:00 a. m.	15/05/1997 12:00:00 a. m.	2	59.50	Wartian Herkku	Torikatu 38	Oulu	Finland	276