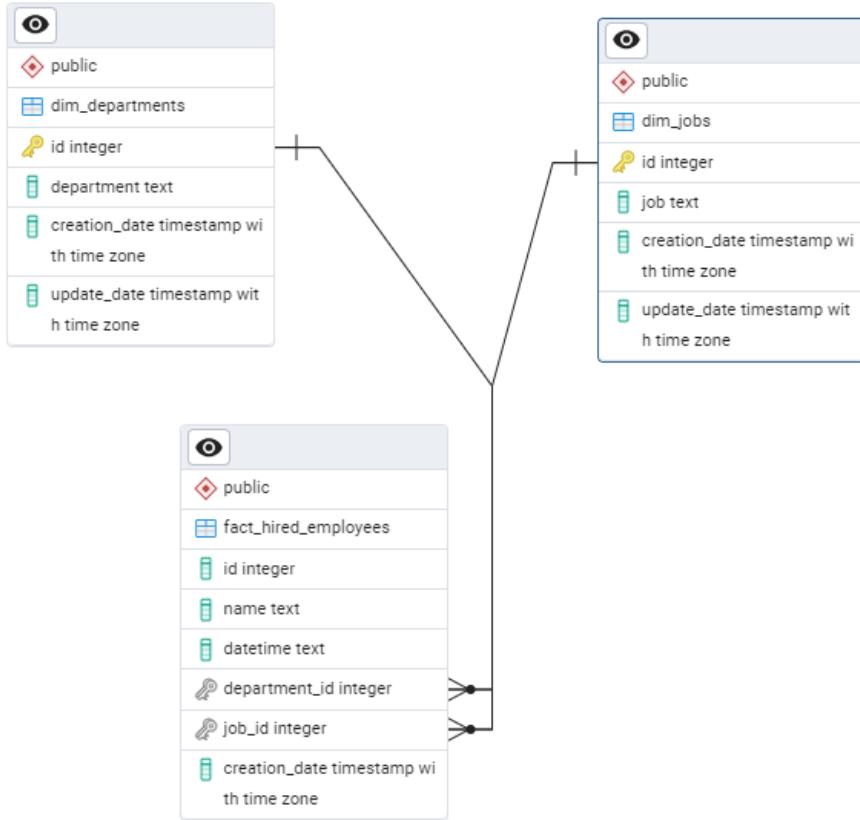


Entity Relation Diagram



Testing

1. API

Test Case 1: Check if the api is running

Description: Run the code for the api ('api.py') and check that there are no errors in the execution. Also check that the api is running on <http://127.0.0.1:5000>.

Expected results: The api is running on the established port, and there are no errors on the terminal due to the execution.

Execution:

```
api.py U x
api.py > ...
1  from flask import Flask, request, jsonify
2  import pandas as pd
3  import psycopg2
4  from psycopg2 import sql
5  import os
6  from dotenv import load_dotenv
7  import hashlib
8
9  # Load environment variables from the .env file
10 load_dotenv()
11
12 app = Flask(__name__)
13
14 # Connection to PostgreSQL
15 def connect_db():
16     try:
17         conn = psycopg2.connect(
18             dbname=os.getenv("DB_NAME"),
19             user=os.getenv("DB_USER"),
20             password=os.getenv("DB_PASSWORD"),
21             host=os.getenv("DB_HOST"),
22             port=os.getenv("DB_PORT"))
23
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Api & C:\Users\User\anaconda3\python.exe c:/users/User/OneDrive/Documentos/AT/Globant/globant_challenge(Api/api.py
* Serving Flask app 'api'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 212-937-318
* Detected change in 'c:/Users/User/OneDrive\Documentos\AT\Globant\globant_challenge\Api\api.py', reloading
* Detected change in 'c:/Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Api\api.py', reloading
* Detected change in 'c:/Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Api\api.py', reloading
* Detected change in 'c:/Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Api\api.py', reloading
* Debugger is active!
* Debugger PIN: 212-937-318
```



The screenshot shows the Postman interface with a successful API call. The URL is `http://127.0.0.1:5000`. The response status is `200 OK`, time is `6 ms`, and size is `196 B`. The response body is:

```
1  Flask server is working
```

Results: The api is running according to the expected.

Test Case 2: Load a csv file for dim_departments using POST

Description: Now that the api is running, use postman to send a POST with the departments.csv file to load the dim_departments table using the route

http://127.0.0.1:5000/upload_csv/dim_departments .

Before to execute the POST ensure to truncate all the tables in the database.

Expected results:

- The message in postman should be 'Data inserted correctly!'.
- Check the *dim_departments* table to confirm that the load is right.
- The number of rows should be 12.
- The table *file_uploads* should have the loaded file and all the columns populated.

Execution:

-Truncate tables:

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Servers (1)

- PostgreSQL 17
 - Databases (3)
 - fs_challenge
 - globant_challenge
 - public
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (4)
 - dim_departments
 - dim_jobs
 - fact_hired_employees
 - file_uploads
 - Trigger Functions
 - Types
 - Views
 - Subscriptions
 - postgres
 - Login/Group Roles (16)
 - Tablespaces

Total rows: 1 of 1 Query complete 00:00:00.071 Ln 1, Col 1

Scratch Pad X

```

1 ✓ truncate table
2   public.c_fact_hired_employees;
3
4 ✓ truncate table
5   public.dim_departments cascade;
6
7 ✓ truncate table
8   public.dim_jobs cascade;
9
10 ✓ truncate table
11   public.file_uploads;
12
13
14
15
16
17

```

Data Output Messages Notifications

NOTICE: truncate cascades to table "fact_hired_employees"
 NOTICE: truncate cascades to table "fact_hired_employees"
 TRUNCATE TABLE

Query returned successfully in 71 msec.

8:56 p.m.
10/10/2024

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Servers (1)

- PostgreSQL 17
 - Databases (3)
 - fs_challenge
 - globant_challenge
 - public
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (4)
 - dim_departments
 - dim_jobs
 - fact_hired_employees
 - file_uploads
 - Trigger Functions
 - Types
 - Views
 - Subscriptions
 - postgres
 - Login/Group Roles (16)
 - Tablespaces

Total rows: 0 of 0 Query complete 00:00:00.092 Ln 2, Col 28

Scratch Pad X

```

1 ✓ SELECT *
2   FROM public.dim_departments;
3
4
5
6
7
8

```

Data Output Messages Notifications

	id	department	creation_date	update_date
[PK]	integer	text	timestamp with time zone	timestamp with time zone

5:55 p.m.
10/10/2024

-Now execute the POST in Postman

POST http://127.0.0.1:5000/upload_csv/dim_departments

Body (8)

Params Authorization Headers (8) Body (1) Pre-request Script Tests Settings Cookies

None form-data x-www-form-urlencoded raw binary

Key	Value
file	departments.csv

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize HTML

Status: 200 OK Time: 6 ms Size: 196 B Save Response

```
1 Flask server is working
```

POST http://127.0.0.1:5000/upload_csv/dim_departments

Body (8)

Params Authorization Headers (8) Body (1) Pre-request Script Tests Settings Cookies

None form-data x-www-form-urlencoded raw binary

Key	Value
file	departments.csv

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

Status: 201 CREATED Time: 116 ms Size: 214 B Save Response

```
1 "message": "Data inserted correctly!"
```

-Check the tables in the database:

Session 1 (Top):

```

1 SELECT *
2   FROM public.dim_departments;
3
4
5
6
7
8
9

```

id [PK] integer	department	creation_date	update_date
1	Product Management	2024-10-10 18:00:21.786041+05	[null]
2	Sales	2024-10-10 18:00:21.786041+05	[null]
3	Research and Development	2024-10-10 18:00:21.786041+05	[null]
4	Business Development	2024-10-10 18:00:21.786041+05	[null]
5	Engineering	2024-10-10 18:00:21.786041+05	[null]
6	Human Resources	2024-10-10 18:00:21.786041+05	[null]
7	Services	2024-10-10 18:00:21.786041+05	[null]
8	Support	2024-10-10 18:00:21.786041+05	[null]
9	Marketing	2024-10-10 18:00:21.786041+05	[null]
10	Training	2024-10-10 18:00:21.786041+05	[null]
11	Legal	2024-10-10 18:00:21.786041+05	[null]
12	Accounting	2024-10-10 18:00:21.786041+05	[null]

Total rows: 12 of 12 Query complete 00:00:00.094 Ln 2, Col 18

Session 2 (Bottom):

```

1 select * from file_uploads
2
3
4
5
6
7

```

id [PK] integer	file_name	file_hash	creation_date
1	departments.csv	71fffb7fa7afa03ca29867a9313104c3a23b70639fe4a4d347a2b1d2223...	2024-10-10 18:00:21.832959+05

Total rows: 1 of 1 Query complete 00:00:00.098 Ln 2, Col 1

Results: The file was loaded into the dim_departments table and also the row with the hash was inserted into the file_uploads table.

Test Case 3: Load a csv file for dim_jobs using POST

Description: Now that the api is running, use postman to send a POST with the jobs.csv file to load the dim_jobs table using the route http://127.0.0.1:5000/upload_csv/dim_jobs.

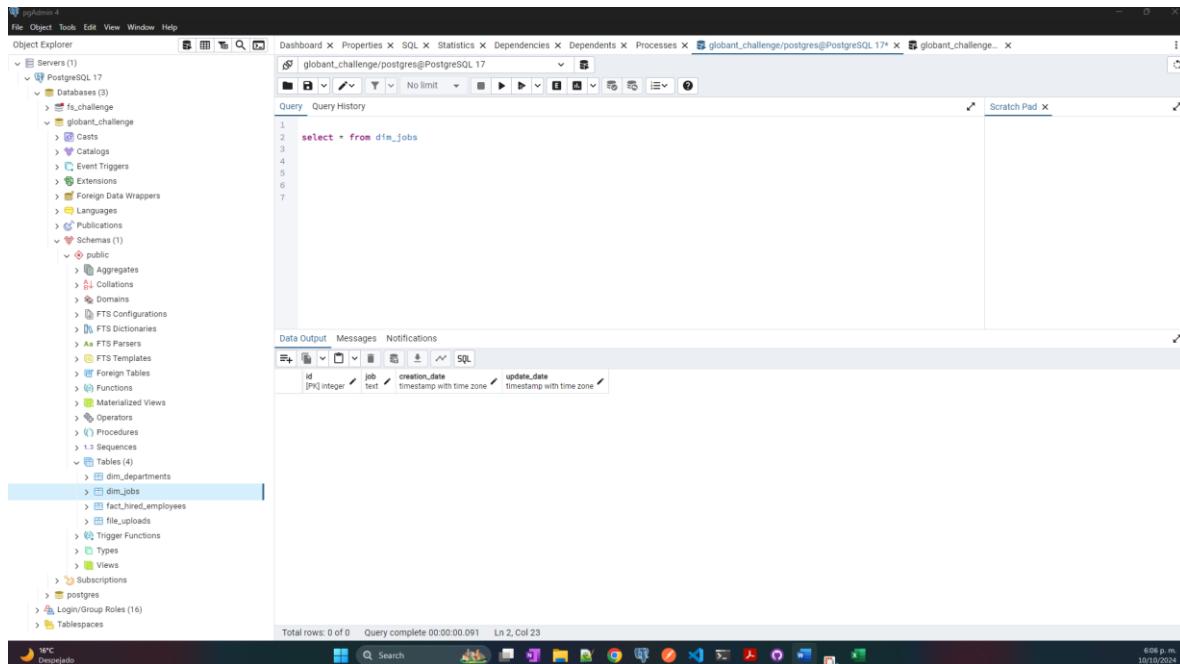
Before to execute the POST check that the dim_jobs table is empty.

Expected results:

- The message in postman should be 'Data inserted correctly!'.
- Check the *dim_jobs* table to confirm that the load is right.
- The number of rows should be 183.
- The table *file_uploads* should have the loaded file and all the columns populated.

Execution:

-Check that the *dim_jobs* table is empty:



The screenshot shows the pgAdmin 4 interface. In the Object Explorer on the left, under the 'Tables' section, the *dim_jobs* table is selected and highlighted with a blue border. The main pane displays a SQL query window with the following code:

```
1
2 select * from dim_jobs
3
4
5
6
7
```

Below the query window is a Data Output grid with the following schema:

	<code>id</code>	<code>[PK] integer</code>	<code>job</code>	<code>text</code>	<code>creation_date</code>	<code>timestamp with time zone</code>	<code>update_date</code>	<code>timestamp with time zone</code>
Total rows:	0 of 0							

At the bottom of the pgAdmin window, the status bar shows "Query complete 00:00:00.091 Ln 2, Col 23". The system tray at the bottom of the screen shows the date and time as "10/10/2024 6:06 p.m." and the weather as "16°C Despejado".

-Now execute the POST in Postman

POST http://127.0.0.1:5000/upload_csv/dim_jobs

Send

Params Authorization Headers (8) Body Tests Settings Cookies

Body

Key Value

file jobs.csv

Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1
2 "message": "Data inserted correctly!"
3

```

Status: 201 CREATED Time: 203 ms Size: 214 B Save Response

Search 6:07 p.m. 10/10/2024

-Check the tables in the database:

File Object Tools Edit View Window Help

Object Explorer

Servers (1)

PostgreSQL 17

Databases (3)

fs_challenge

globant_challenge

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (4)

dim_departments

dim_jobs

fact_hired_employees

file_uploads

Trigger Functions

Types

Views

Subscriptions

Postgres

Login/Group Roles (16)

Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents Processes globant_challenge/postgres@PostgreSQL 17* globant_challenge..

Query History

```

1
2 select * from dim_jobs
3
4
5
6
7

```

Data Output Messages Notifications

	id [PK] integer	job	creation_date timestamp with time zone	update_date timestamp with time zone
1	1	Marketing Assistant	2024-10-10 18:07:22.764992-05	[null]
2	2	VP Sales	2024-10-10 18:07:22.764992-05	[null]
3	3	Biostatistician IV	2024-10-10 18:07:22.764992-05	[null]
4	4	Account Representative II	2024-10-10 18:07:22.764992-05	[null]
5	5	VP Marketing	2024-10-10 18:07:22.764992-05	[null]
6	6	Environmental Specialist	2024-10-10 18:07:22.764992-05	[null]
7	7	Software Consultant	2024-10-10 18:07:22.764992-05	[null]
8	8	Office Assistant III	2024-10-10 18:07:22.764992-05	[null]
9	9	Information Systems Manager	2024-10-10 18:07:22.764992-05	[null]
10	10	Desktop Support Technician	2024-10-10 18:07:22.764992-05	[null]
11	11	Financial Advisor	2024-10-10 18:07:22.764992-05	[null]
12	12	Computer Systems Analyst I	2024-10-10 18:07:22.764992-05	[null]
13	13	Automation Specialist IV	2024-10-10 18:07:22.764992-05	[null]
14	14	Help Desk Technician	2024-10-10 18:07:22.764992-05	[null]
15	15	Office Assistant II	2024-10-10 18:07:22.764992-05	[null]

Total rows: 183 of 183 Query complete 00:00:00.119 Ln 2, Col 22

100% Despejado 6:08 p.m. 10/10/2024

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under 'Servers (1) > PostgreSQL 17 > globant_challenge > Tables (4)', the 'file_uploads' table is selected. The 'Data Output' tab displays the following data:

	id [PK] integer	file_name character varying (255)	file_hash character varying (64)	creation_date timestamp with time zone
1	12	departments.csv	71fefb7aa7afa03ca29867a9313046c3a23b70639e4a4d34d7a2b1d2223a	2024-10-10 18:00:21.832959-05
2	13	jobs.csv	b52b7eeaa311516393f4af7447fa21b10f8b157ac974a54299310e477ea4	2024-10-10 18:07:22.874840-05

Total rows: 2 of 2 Query complete 00:00:00.182 Ln 2, Col 27

Results: The file was loaded into the *dim_jobs* table with 183 rows and also the row with the hash was inserted into the *file_uploads* table.

Test Case 4: Load a csv file for *fact_hired_employees* using POST

Description: Now that the api is running, use postman to send a POST with the *hired_employees.csv* file to load the *fact_hired_employees* table using the route http://127.0.0.1:5000/upload_csv/fact_hired_employees.

Before to execute the POST check that the *fact_hired_employees* table is empty.

Expected results:

- The message in postman should be 'Data inserted correctly!'.
- Check the *fact_hired_employees* table to confirm that the load is right.
- The number of rows should be 1999.
- The table *file_uploads* should have the loaded file and all the columns populated.

Execution:

-Check that the *fact_hired_employees* table is empty:

-Now execute the POST in Postman

Key	Value
<input checked="" type="checkbox"/> file	hired_employees.csv
Key	Value

```

1   "message": "Data inserted correctly!"
2
3

```

-Check the tables in the database:

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, the 'fact_hired_employees' table is selected under the 'Tables (4)' section of the 'globant_challenge' schema. A query window displays the following SQL code:

```
1
2 select * from fact_hired_employees;
```

The resulting Data Output table contains 1999 rows of employee hiring data. The columns are: id, name, datetime, department_id, job_id, and creation_date. The first few rows are:

	id	name	datetime	department_id	job_id	creation_date
1	1	Harold Vogt	2021-11-07T02:48:42Z	2	96	2024-10-10 18:36:36.25504-05
2	2	Ty Hofer	2021-05-30T05:43:46Z	8	[null]	2024-10-10 18:36:36.25504-05
3	3	Lyman Hade	2021-09-01T23:27:38Z	5	52	2024-10-10 18:36:36.25504-05
4	4	Lott Crotwhe	2021-10-16T11:04:21Z	12	71	2024-10-10 18:36:36.25504-05

The status bar at the bottom indicates 'Total rows: 1000 of 1999' and 'Query complete 00:00:00.111 Ln 2, Col 1'.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, the 'file_uploads' table is selected under the 'Tables (4)' section of the 'globant_challenge' schema. A query window displays the following SQL code:

```
1
2 select * from file_uploads;
```

The resulting Data Output table contains 3 rows of file upload data. The columns are: id, file_name, file_hash, and creation_date. The first few rows are:

	id	file_name	file_hash	creation_date
1	12	departments.csv	71fefb7faaf3ca29867a9313048c3a230706399e4a034d74201d22233a	2024-10-10 18:00:21.832959-05
2	13	jobs.csv	b52b7eeae12315165934af447e321b1cd7d8124ce974a542efca1bb477ca4..	2024-10-10 18:07:22.874845-05
3	17	hired_employees.csv	f402a0874a161507239930e77cse692cwe9363ea3bea27cce1a8a26c094b..	2024-10-10 18:36:36.25504-05

The status bar at the bottom indicates 'Total rows: 3 of 3' and 'Query complete 00:00:00.081 Ln 2, Col 27'.

Results: The file was loaded into the *fact_hired_employees* table with 1999 rows and also the row with the hash was inserted into the *file_uploads* table.

Test Case 5: Try to load a csv file previously loaded using POST

Description: Now that the api is running, use postman to send a POST with the *jobs.csv* file to try to load the table *dim_jobs* using the same file loaded previously. Use the route http://127.0.0.1:5000/upload_csv/dim_jobs.

Before to execute the POST check that the *dim_jobs* table is populated and the file *jobs* is in the *file_uploads* table.

Expected results:

- The message in postman should be ‘The file has already been uploaded before’.
- Check the *dim_jobs* table didn’t load the file again.
- The number of rows should be 183 rows.
- The table *file_uploads* has no changes.

Execution:

-Check that the *dim_jobs* table has 183 rows:

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'Tables' section, the 'dim_jobs' table is selected. In the main pane, a SQL query is run:

```

1 select * from dim_jobs;
2
3
4
5
6

```

The results show 183 rows of data, each representing a job title and its creation date:

	id [PK] integer	job text	creation_date timestamp with time zone	update_date timestamp with time zone
1	1	Marketing Assistant	2024-10-10 18:07:22.764992-05	[null]
2	2	VP Sales	2024-10-10 18:07:22.764992-05	[null]
3	3	BioStatistician IV	2024-10-10 18:07:22.764992-05	[null]
4	4	Account Representative II	2024-10-10 18:07:22.764992-05	[null]
5	5	VP Marketing	2024-10-10 18:07:22.764992-05	[null]
6	6	Environmental Specialist	2024-10-10 18:07:22.764992-05	[null]
7	7	Software Consultant	2024-10-10 18:07:22.764992-05	[null]
8	8	Office Assistant III	2024-10-10 18:07:22.764992-05	[null]
9	9	Information Systems Manager	2024-10-10 18:07:22.764992-05	[null]
10	10	Desktop Support Technician	2024-10-10 18:07:22.764992-05	[null]
11	11	Financial Advisor	2024-10-10 18:07:22.764992-05	[null]
12	12	Computer Systems Analyst I	2024-10-10 18:07:22.764992-05	[null]
13	13	Automation Specialist IV	2024-10-10 18:07:22.764992-05	[null]
14	14	Help Desk Technician	2024-10-10 18:07:22.764992-05	[null]
15	15	Office Assistant II	2024-10-10 18:07:22.764992-05	[null]
16	16	VP Quality Control	2024-10-10 18:07:22.764992-05	[null]
17	17	Office Assistant IV	2024-10-10 18:07:22.764992-05	[null]
18	18	Financial Analyst	2024-10-10 18:07:22.764992-05	[null]
19	19	Electrical Engineer	2024-10-10 18:07:22.764992-05	[null]
20	20	Chemical Engineer	2024-10-10 18:07:22.764992-05	[null]
21	21	Social Worker	2024-10-10 18:07:22.764992-05	[null]
22	22	VP Product Management	2024-10-10 18:07:22.764992-05	[null]
23	23	Administrative Officer	2024-10-10 18:07:22.764992-05	[null]
24	24	Paralegal	2024-10-10 18:07:22.764992-05	[null]

Total rows: 183 of 183 Query complete 00:00:00.101 Ln 2, Col 23

-Now execute the POST in Postman

POST http://127.0.0.1:5000/upload_csv/dim_jobs

http://127.0.0.1:5000/upload_csv/dim_jobs

Send

Params Authorization Headers (8) Body **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary

Key	Value	...	Bulk Edit
<input checked="" type="checkbox"/> file	jobs.csv		△
Key	Value		

Body Cookies Headers (5) Test Results Status: 400 BAD REQUEST Time: 59 ms Size: 233 B Save Response

```

1 "error": "The file has already been uploaded before"
2
3

```

File Object Tools Edit View Window Help

Search 6:48 p.m. 10/10/2024

-Check the tables in the database:

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Servers (1) PostgreSQL 17

- Databases (3)
 - fs_challenge
 - globant_challenge
 - public
- Catalogs
- Event Triggers
- Extensions
- Foreign Data Wrappers
- Languages
- Publications
- Schemas (1)
 - public
- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (4)
 - dim_departments
 - dim_jobs
 - fact_hired_employees
 - file_uploads
- Trigger Functions
- Types
- Views

globant_challenge/postgres@PostgreSQL 17*

Query History

```

1 select * from dim_jobs;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

```

Data Output Messages Notifications

id [PK] integer	job text	creation_date timestamp with time zone	update_date timestamp with time zone
1	Marketing Assistant	2024-10-10 18:07:22.764992+05	[null]
2	VP Sales	2024-10-10 18:07:22.764992+05	[null]
3	Risk Statistician IV	2024-10-10 18:07:22.764992+05	[null]
4	Account Representative II	2024-10-10 18:07:22.764992+05	[null]
5	VP Marketing	2024-10-10 18:07:22.764992+05	[null]
6	Environmental Specialist	2024-10-10 18:07:22.764992+05	[null]
7	Software Consultant	2024-10-10 18:07:22.764992+05	[null]
8	Office Assistant III	2024-10-10 18:07:22.764992+05	[null]
9	Information Systems Manager	2024-10-10 18:07:22.764992+05	[null]
10	Desktop Support Technician	2024-10-10 18:07:22.764992+05	[null]
11	Financial Advisor	2024-10-10 18:07:22.764992+05	[null]
12	Computer Systems Analyst I	2024-10-10 18:07:22.764992+05	[null]
13	Automation Specialist IV	2024-10-10 18:07:22.764992+05	[null]
14	Help Desk Technician	2024-10-10 18:07:22.764992+05	[null]
15	Office Assistant II	2024-10-10 18:07:22.764992+05	[null]
16	VP Quality Control	2024-10-10 18:07:22.764992+05	[null]
17	Office Assistant IV	2024-10-10 18:07:22.764992+05	[null]
18	Financial Analyst	2024-10-10 18:07:22.764992+05	[null]
19	Electrical Engineer	2024-10-10 18:07:22.764992+05	[null]
20	Chemical Engineer	2024-10-10 18:07:22.764992+05	[null]
21	Social Worker	2024-10-10 18:07:22.764992+05	[null]
22	VP Product Management	2024-10-10 18:07:22.764992+05	[null]
23	Administrative Officer	2024-10-10 18:07:22.764992+05	[null]
24	Paralegal	2024-10-10 18:07:22.764992+05	[null]

Total rows: 183 of 183 Query complete 00:00:00.130 Ln 2, Col 22 ✓ Success

10/10/2024 6:48 p.m.

id	file_name	file_hash	creation_date
1	departments.csv	71febd7fa57af03ca299679313104c3a23070639fe444d34c7a091e22233a	2024-10-10 10:20:31.823959+05
2	jobs.csv	b525f8ea13131639364647432191c570123467445429c9106477c34	2024-10-10 18:07:22.874845+05
3	hired_employees.csv	f4020a807416150723d99304e77be992cbe9383ea7bea27cc1e8a20c094b...	2024-10-10 18:36:36.25504+05

Results: The file wasn't loaded into the *dim_jobs* table. The table has the same 183 rows and no rows were inserted in the *file_uploads* table.

Test Case 6: Try to load a csv file with the incorrect number of columns using POST

Description: Now that the api is running, use postman to send a POST with the *hired_employees.csv* file to try to load the table *dim_jobs*, since the *hired_employees* file has more columns than expected by the *dim_jobs* table, an error should appear. Use the route http://127.0.0.1:5000/upload_csv/dim_jobs.

Expected results:

- The message in postman should be 'The csv file should have 2 columns'.
- Check the *dim_jobs* table isn't affected.
- The number of rows should be 183 rows.
- The table *file_uploads* has no changes.

Execution:

-Check that the *dim_jobs* table has 183 rows:

The screenshot shows the pgAdmin 4 interface. The left sidebar (Object Explorer) lists the database structure under 'Servers (1)'. Under 'globant_challenge' schema, there are tables like 'dim_departments', 'dim_jobs', and 'fact_hired_employees'. The main area shows a query window with the following SQL:

```
1
2 select * from dim_jobs;
```

The results pane displays a table with 183 rows of data:

	<code>id</code>	<code>job</code>	<code>creation_date</code>	<code>update_date</code>
1	1	Marketing Assistant	2024-10-10 18:07:22.764992-05	[null]
2	2	VP Sales	2024-10-10 18:07:22.764992-05	[null]
3	3	Biostatistician IV	2024-10-10 18:07:22.764992-05	[null]
4	4	Account Representative II	2024-10-10 18:07:22.764992-05	[null]
5	5	VP Marketing	2024-10-10 18:07:22.764992-05	[null]
6	6	Environmental Specialist	2024-10-10 18:07:22.764992-05	[null]
7	7	Software Consultant	2024-10-10 18:07:22.764992-05	[null]
8	8	Office Assistant III	2024-10-10 18:07:22.764992-05	[null]
9	9	Information Systems Manager	2024-10-10 18:07:22.764992-05	[null]
10	10	Desktop Support Technician	2024-10-10 18:07:22.764992-05	[null]
11	11	Financial Advisor	2024-10-10 18:07:22.764992-05	[null]
12	12	Computer Systems Analyst I	2024-10-10 18:07:22.764992-05	[null]
13	13	Automation Specialist IV	2024-10-10 18:07:22.764992-05	[null]
14	14	Help Desk Technician	2024-10-10 18:07:22.764992-05	[null]
15	15	Office Assistant II	2024-10-10 18:07:22.764992-05	[null]
16	16	VP Quality Control	2024-10-10 18:07:22.764992-05	[null]
17	17	Office Assistant IV	2024-10-10 18:07:22.764992-05	[null]
18	18	Financial Analyst	2024-10-10 18:07:22.764992-05	[null]
19	19	Electrical Engineer	2024-10-10 18:07:22.764992-05	[null]
20	20	Chemical Engineer	2024-10-10 18:07:22.764992-05	[null]
21	21	Social Worker	2024-10-10 18:07:22.764992-05	[null]
22	22	VP Product Management	2024-10-10 18:07:22.764992-05	[null]
23	23	Administrative Officer	2024-10-10 18:07:22.764992-05	[null]
24	24	Paralegal	2024-10-10 18:07:22.764992-05	[null]

-Now execute the POST in Postman

The screenshot shows the Postman application. A POST request is being made to `http://127.0.0.1:5000/upload_csv/dim_jobs`. The 'Body' tab is active, showing a file named `hired_employees.csv` attached. The response at the bottom shows a status of `400 BAD REQUEST` with the error message: `"error": "The csv file should have 2 columns"`.

-Check the tables in the database:

The screenshot shows the pgAdmin 4 interface with two sessions. Both sessions show the results of a query on the `dim_jobs` table. The first session (top) shows 183 rows, and the second session (bottom) also shows 183 rows. The columns are `id`, `job`, `creation_date`, and `update_date`. The data is identical in both cases.

	<code>id</code>	<code>job</code>	<code>creation_date</code>	<code>update_date</code>
1	1	Marketing Assistant	2024-10-10 18:07:22.764992-05	[null]
2	2	VP Sales	2024-10-10 18:07:22.764992-05	[null]
3	3	BioStatistician IV	2024-10-10 18:07:22.764992-05	[null]
4	4	Account Representative II	2024-10-10 18:07:22.764992-05	[null]
5	5	VP Marketing	2024-10-10 18:07:22.764992-05	[null]
6	6	Environmental Specialist	2024-10-10 18:07:22.764992-05	[null]
7	7	Software Consultant	2024-10-10 18:07:22.764992-05	[null]
8	8	Office Assistant III	2024-10-10 18:07:22.764992-05	[null]
9	9	Information Systems Manager	2024-10-10 18:07:22.764992-05	[null]
10	10	Desktop Support Technician	2024-10-10 18:07:22.764992-05	[null]
11	11	Financial Advisor	2024-10-10 18:07:22.764992-05	[null]
12	12	Computer Systems Analyst I	2024-10-10 18:07:22.764992-05	[null]
13	13	Automation Specialist IV	2024-10-10 18:07:22.764992-05	[null]
14	14	Help Desk Technician	2024-10-10 18:07:22.764992-05	[null]
15	15	Office Assistant II	2024-10-10 18:07:22.764992-05	[null]
16	16	VP Quality Control	2024-10-10 18:07:22.764992-05	[null]
17	17	Office Assistant IV	2024-10-10 18:07:22.764992-05	[null]
18	18	Financial Analyst	2024-10-10 18:07:22.764992-05	[null]
19	19	Electrical Engineer	2024-10-10 18:07:22.764992-05	[null]
20	20	Chemical Engineer	2024-10-10 18:07:22.764992-05	[null]
21	21	Social Worker	2024-10-10 18:07:22.764992-05	[null]
22	22	VP Product Management	2024-10-10 18:07:22.764992-05	[null]
23	23	Administrative Officer	2024-10-10 18:07:22.764992-05	[null]
24	24	Paralegal	2024-10-10 18:07:22.764992-05	[null]

Total rows: 183 of 183 Query complete 00:00:00.164 Ln 2, Col 19

Successfully run. Total query runtime: 164 msec. 183 rows affected

Results: The file wasn't loaded into the `dim_jobs` table. The table has the same 183 rows and no rows were inserted in the `file_uploads` table.

Test Case 7: Try to load an invalid table

Description: Now that the api is running, use postman to send a POST with the `hired_employees.csv` file to try to load the `file_uploads` table, since the `file_uploads` isn't a valid table to be loaded using csv files, an error should appear. Use the route http://127.0.0.1:5000/upload_csv/file_uploads.

Expected results:

- The message in postman should be 'Invalid table'.
- The table *file_uploads* has no changes.

Execution:

-Check the *file_uploads* table:

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer tree view is expanded to show the database structure, including servers, databases, globant_challenge schema, and tables like departments, jobs, fact_hired_employees, and file_uploads. The file_uploads table is selected. In the center, a SQL tab displays the query: "select * from file_uploads;". Below the query, the results table shows three rows of data:

id	file_name	file_hash	creation_date
1	departments.csv	71ff077ea7afa0ca29887a931046c3a23b70639fe4a4d34d7a291d22233a	2024-10-10 18:00:21.832959-05
2	jobs.csv	b52d7eef131516593d4af447e321bfc7d8124e974a542e9cb1b5477ca4	2024-10-10 18:07:22.87845-05
3	hired_employees.csv	f4020ab07416150723d993b04e77cd69f2ce9363ea3bea27cc1a8a2dc0940	2024-10-10 18:36:25504-05

At the bottom right of the results pane, a message says "Successfully run. Total query runtime: 95 msec. 3 rows affected."

-Now execute the POST in Postman

The screenshot shows a POST request to `http://127.0.0.1:5000/upload_csv/file_uploads`. The 'Body' tab is selected, showing a form-data key 'file' with value 'hired_employees.csv'. The response status is 400 BAD REQUEST with the message: "error": "Invalid table".

-Check the table in the database:

The screenshot shows the pgAdmin 4 interface with the 'file_uploads' table selected in the 'globant_challenge' database. The table has three rows with the following data:

id	file_name	file_hash	creation_date
1	departments.csv	71f9fb7aa7afaf03ca29867a93131046c3a23b70639fe44d34d7a2b1c22233a	2024-10-10 18:00:21.832999-05
2	jobs.csv	b5267e4ea13151659364a4471e321b1c27d8124ce874a5429ca1b5477ca4...	2024-10-10 18:07:22.874645-05
3	hired_employees.csv	f4020a807416150723d99304e7cde692cbe9353ea3bea27ce1a8a2dc094b...	2024-10-10 18:36:39.255004-05

Results: The file wasn't loaded into the `file_uploads` table. The table has the same 3 rows.

Test Case 8: Try to load a file with more than 1000 rows

Description: Now that the api is running, use postman to send a POST with the `hired_employees.csv` file to try to load the `fact_hired_employees` table, since the csv file contains more than 1000 rows, an error should appear. Use the route http://127.0.0.1:5000/upload_csv/fact_hired_employees.

Expected results:

- The message in postman should be 'Batch limit exceeded. You can only insert up to 1000 rows at a time'.
 - The table *fact_hired_employees* has no changes.
 - The table *file_uploads* shouldn't have the new row with the file.

Execution:

-Check the *file_uploads* table:

The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL database named `globant_challenge`. The Object Explorer on the left lists various database objects such as Schemas, Tables, and Views. The main pane displays a query editor with the following SQL code:

```
globant_challenge/postgres@PostgreSQL_17
globant_challenge

Query Query History
1 ~ select * from
2   public.dim_jobs;
3
4 --truncate table dim_jobs;
5
6 ~ select * from
7   file_uploads;
8

Data Output Messages Notifications
SQL
```

The results grid shows two rows of data from the `dim_jobs` table:

	<code>id</code>	<code>file_name</code>	<code>file_hash</code>	<code>creation_date</code>
1	33	jobs.csv	b52b7eeef1311659364d4471e321b1cd7d8124ce974a542e9ca1b0477ca...	2024-10-11 14:57:06.469446+05
2	34	departments.csv	71fefb7fa7faf03ca298679f3131046c3a23b70639feaa4d3467a2b1d223...	2024-10-11 14:57:43.182079-05

At the bottom, it indicates "Total rows: 2 of 2" and "Query complete 00:00:00.125 Ln 12, Col 1". The status bar at the bottom right shows "2:58 p.m. 11/18/2024".

-Check the fact_hired_employees table:

```

--truncate table file_uploads;
select * from fact_hired_employees;

```

-Check the file to be loaded, the file has more than 1000 rows:

A1968	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1968	Gunn	Wheatcroft	2021-09-17T12:10:28Z	2,67																								
1969	Dulcie	Whinray	2021-10-27T15:12:49Z	2,61																								
1970	Eadie	Whittington	2021-08-08T06:06:40Z	8,101																								
1971	Katherine	Whitton	2021-12-17T17:49:37Z	8,26																								
1972	Bridgette	Wickney	2021-09-12T18:10:42Z	1,51																								
1973	Leanne	Widmer	2021-07-05T10:05:05Z	22,131																								
1974	Hever	Willerton	2021-06-17T06:05:34Z	128																								
1975	Gina	Wilton	2021-11-15T18:17:51Z	9,40																								
1976	Gerard	Windmill	2021-03-10T08:19:76Z	7,74																								
1977	Tremain	Withnall	2021-03-16T01:52:08Z	8,72																								
1978	Hanwell	Wigden	2021-08-23T11:30:08Z	2,117																								
1979	Marley	Wolffart	2021-12-13T02:36:36Z	8,4																								
1980	Elaine	Wood	2021-07-10T11:10:29Z	2,59																								
1981	Hattie	Woodard	2021-04-29T00:00:37Z	7,105																								
1982	Kerny	Woofinden	2021-06-23T20:56:05Z	9,51																								
1983	Felicia	Wooldridge	2021-08-09T04:54:47Z	9,80																								
1984	Violette	Woollin	2021-08-03T10:15:32Z	4,55																								
1985	Leanne	Worrell	2021-07-10T11:10:29Z	7																								
1986	Starla	Wreford	2021-04-24T14:10:15Z	1,148																								
1987	Shana	Wyldish	2021-12-23T10:05:05Z	92,102																								
1988	Danya	Wynne	2021-10-12T03:46:10Z	8,8,102																								
1989	Sherie	Yakowlin	2021-10-12T01:18:14Z	302,8,74																								
1990	Julian	Yandell	2022-01-09T14:30:23Z	2,63																								
1991	Alissa	Yanuzzi	2021-09-11T11:11:29Z	7,99																								
1992	Tatjana	Yates	2021-11-15T08:08:41Z	3,6																								
1993	Magey	Yau	2021-04-11T12:51:33Z	7,80																								
1994	Hedi	Yeowell	2022-01-07T06:21:57Z	2,5,18																								
1995	Goren	Yong	2021-06-22T21:57:53Z	2,8,28																								
1996	Cristoboro	Youngs	2021-04-01T17:48:42Z	2,3,23																								
1997	Leanne	Yule	2021-08-10T11:10:29Z	2,132																								
1998	Jerry	Yuen	2021-10-03T14:12:50Z	2,108																								
1999	Jerrin	Zebedee	2022-01-18T04:47:17Z	2,80																								
2000																												
2001																												
2002																												
2003																												
2004																												
2005																												

-Now execute the POST in Postman

The screenshot shows the Postman interface with a failed API request. The URL is `http://127.0.0.1:5000/upload_csv/fact_hired_employees`. The request method is POST. The body contains a file named `hired_employees.csv`. The response status is 400 BAD REQUEST with a message: "error": "Batch limit exceeded. You can only insert up to 1000 rows at a time".

-Check the tables in the database:

The screenshot shows the pgAdmin 4 interface. The Object Explorer on the left lists various database objects like schemas, tables, and functions. The central area shows a SQL query window with the following code:

```

11
12 v select * from
13   file_uploads;
14
15
16 v select * from
17   fact_hired_employees;
18
19

```

The Data Output tab shows the schema for the `fact_hired_employees` table:

<code>id</code>	<code>name</code>	<code>datetime</code>	<code>department_id</code>	<code>job_id</code>	<code>creation_date</code>
-----------------	-------------------	-----------------------	----------------------------	---------------------	----------------------------

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree view of database objects including tables like dim_departments, dim_jobs, fact_hired_employees, and file_uploads. The right pane has tabs for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The SQL tab is active, showing a query history with the following SQL code:

```

11
12 v select * from
13   file_uploads;
14
15
16 v select * from
17   fact_hired_employees;
18
19

```

The Data Output tab shows the results of the last query:

Total rows: 2 of 2 Query complete 00:00:00.100 Ln 12, Col 1				
SQL				
	id [PK] integer	file_name character varying (255)	file_hash character varying (64)	creation_date timestamp with time zone
1	33	jobs.csv	b52b7eae1311659364af4471e321b1cd7d8124ce974a542e9ca1bb477ca...	2024-10-11 14:57:06.469646-05
2	34	departments.csv	71fefb7aa7afa03ca29867a93131046c3e23b70639fe4a4d3407a2b1e2223...	2024-10-11 14:57:43.182079-05

Results: The file wasn't loaded into the `fact_hired_employees` table. And the api returns an error.

2. SQL

- Number of employees hired for each job and department in 2021 divided by quarter. The table must be ordered alphabetically by department and job.

Output example:

department	job	Q1	Q2	Q3	Q4
Staff	Recruiter	3	0	7	11
Staff	Manager	2	1	0	2
Supply Chain	Manager	0	1	3	0

The query used creates a view with the resulting data:

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Servers (1)

- PostgreSQL 17
 - Databases (3)
 - fs_challenge
 - globant_challenge
 - public
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Publications
 - Schemas (1)
 - public
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - As FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (4)
 - dim_departments
 - dim_jobs
 - fact_hired_employees
 - fileuploads
 - Trigger Functions
 - Types
- Views (1)
 - requirement_1
- Subscriptions
- postres
- Login/Group Roles (16)
- Tablespaces

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Section 2-1.sql X globant_challenge... X globant_challenge... X public.fact_hired_e... X

Query History

```

11 --table must be ordered alphabetically by department and job.
12 -----
13 
14 ✓ create or replace view requirement_1 as (
15     select
16         split_part(dept.job, '-', 1) department,
17         split_part(dept.job, '-', 2) job,
18         coalesce(q1, 0) q1,
19         coalesce(q2, 0) q2,
20         coalesce(q3, 0) q3,
21         coalesce(q4, 0) q4
22     from crossstab(
23         $$
24             select
25                 --Concatenate the department and job columns to avoid weird behavior when pivot data
26                 coalesce(dim_departments.department, 'N/A') || '-' || coalesce(dim_jobs.job, 'N/A') dept_job,
27                 extract(QUARTER from fact_hired_employees.datetime::TIMESTAMPTZ) AT TIME ZONE 'UTC' Q,
28                 count(fact_hired_employees.id) total_hired
29             from
30                 public.fact_hired_employees
31                 left join public.dim_jobs
32                     on fact_hired_employees.job_id = dim_jobs.id
33                 left join public.dim_departments
34                     on fact_hired_employees.department_id = dim_departments.id
35             where
36                 extract(YEAR FROM (fact_hired_employees.datetime::TIMESTAMPTZ) AT TIME ZONE 'UTC') = 2021
37             group by
38                 dim_departments.department,
39                 dim_jobs.job,
40                 -
41         Data Output Messages Notifications
42     NOTICE: extension "tablefunc" already exists, skipping
43     CREATE VIEW
44 
```

Query returned successfully in 74 msec.

Total rows: 3 of 3 Query complete 00:00:00.074 Ln 22, Col 16

7:11 p.m.
10/10/2024

Dashboard X Properties X SQL X Statistics X Dependencies X Dependents X Section 2-1.sql X globant_challenge... X globant_challenge/postgres@PostgreSQL 17*

Query History

```

1 ✓ select * from
2 public.requirement_1;

```

Data Output Messages Notifications

	department	job	q1	q2	q3	q4
	text	text	integer	integer	integer	integer
1	Accounting	Account Representative IV	1	0	0	0
2	Accounting	Actuary	0	1	0	0
3	Accounting	Analyst Programmer	0	0	1	0
4	Accounting	Budget/Accounting Analyst III	0	1	0	0
5	Accounting	Cost Accountant	0	1	0	0
6	Accounting	Database Administrator III	0	0	0	1
7	Accounting	Desktop Support Technician	0	0	1	0
8	Accounting	Food Chemist	1	0	0	0
9	Accounting	Graphic Designer	0	1	0	0
10	Accounting	Health Coach III	0	0	0	1
11	Accounting	Health Coach IV	0	0	1	0
12	Accounting	Help Desk Technician	0	0	1	0
13	Accounting	Junior Executive	0	0	1	0
14	Accounting	Legal Assistant	0	0	1	1
15	Accounting	Media Manager III	0	1	0	0
16	Accounting	Programmer Analyst IV	0	1	1	1
17	Accounting	Programmer III	0	0	1	0
18	Accounting	Research Assistant IV	0	0	1	0
19	Accounting	Sales Representative	1	0	1	0
20	Accounting	Senior Cost Accountant	0	0	0	1
21	Accounting	Senior Developer	0	0	0	1
22	Accounting	Software Test Engineer IV	0	0	1	0
23	Accounting	Statistician I	0	0	0	1

Total rows: 958 Query complete 00:00:00.119 Ln 2, Col 12

7:11 p.m.
10/10/2024

You can see the full results on the file /SQL/requirement_1_results.csv .

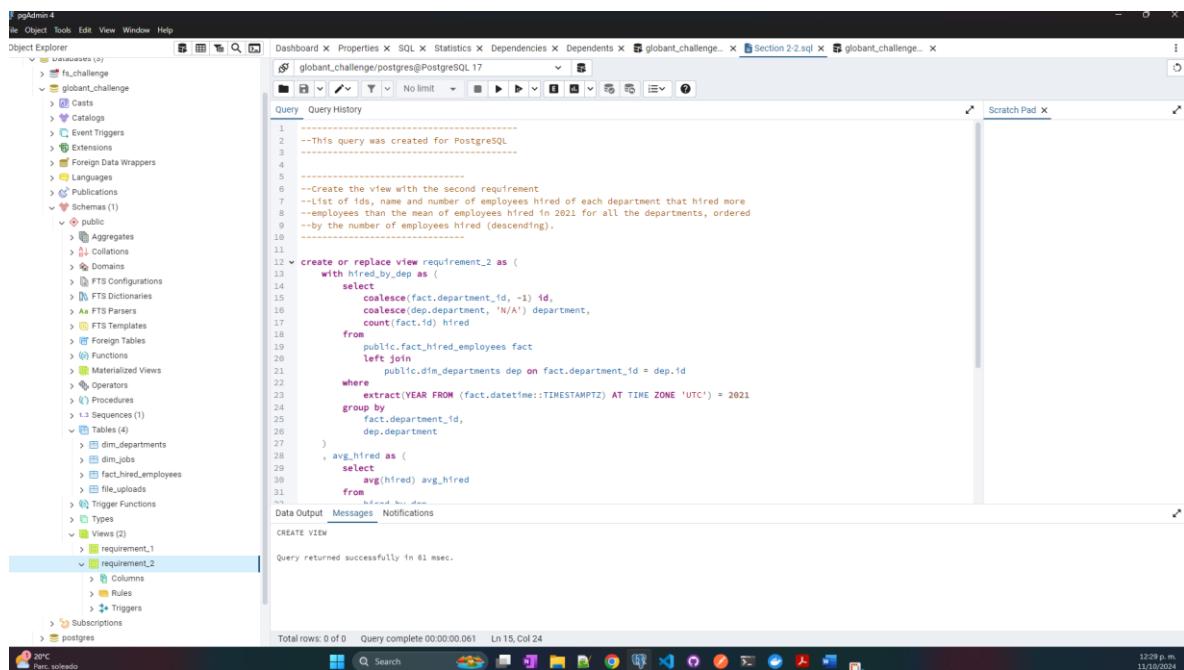
- List of ids, name and number of employees hired of each department that hired more

employees than the mean of employees hired in 2021 for all the departments, ordered by the number of employees hired (descending).

Output example:

id	department	hired
7	Staff	45
9	Supply Chain	12

The query used, creates a view with the resulting data:



The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure including Schemas, Tables, Views, and other objects.
- Query Editor:** Displays the SQL code for creating the view requirement_2.
- Status Bar:** Shows "Query returned successfully in 81 msec." and "Total rows: 0 of 0".
- System Bar:** Shows the date and time as "17:09 p.m. 11/10/2024".

```

1 --This query was created for PostgreSQL
2 -----
3 -----
4 -----
5 -----
6 --Create the view with the second requirement
7 --List of ids, name and number of employees hired of each department that hired more
8 --employees than the mean of employees hired in 2021 for all the departments, ordered
9 --by the number of employees hired (descending).
10 -----
11 -----
12 -- create or replace view requirement_2 as (
13     with hired_by_dep as (
14         select
15             coalesce(fact.department_id, -1) id,
16             coalesce(dep.department, 'N/A') department,
17             count(fact.id) hired
18         from
19             public.fact_hired_employees fact
20             left join
21                 public.dim_departments dep on fact.department_id = dep.id
22         where
23             extract(YEAR FROM (fact.datetime::TIMESTAMPTZ) AT TIME ZONE 'UTC') = 2021
24         group by
25             fact.department_id,
26             dep.department
27     )
28     , avg_hired as (
29         select
30             avg(hired) avg_hired
31         from
32             requirement_1
33     )
34     select
35         id, department, hired, avg_hired
36     from
37         requirement_2
38     order by
39         department, hired desc
40     limit 10
41 )
42
43 CREATE VIEW
44 
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, displaying a tree view of database objects. The main area is the Query Editor, showing a SQL query and its results. The results table has columns: id, department, hired. The data is as follows:

	id	department	hired
1	8	Support	221
2	5	Engineering	208
3	6	Human Resources	204
4	7	Services	204
5	4	Business Development	187
6	3	Research and Development	151
7	9	Marketing	143

You can see the full results on the file /SQL/requirement_2_results.csv .

3. Bonus Track! Cloud, Testing & Containers

Add the following to your solution to make it more robust:

Host your architecture in any public cloud (using the services you consider more adequate)

The database was created on Azure:

The screenshot shows the Microsoft Azure portal home page. It features a top navigation bar with links for Microsoft Learn, Azure Monitor, Microsoft Defender for Cloud, Cost Management, and Azure mobile app. Below the navigation is a section for 'Azure services' with icons for Create a resource, SQL databases, All resources, Subscriptions, Virtual machines, Quickstart Center, Azure AI services, Kubernetes services, App Services, and More services. The main content area displays 'Resources' under the 'Recent' tab, showing two items: 'cloud_glochallenge' (Database for PostgreSQL - Flexible Server) and 'glo_challenge' (Resource group). There are also sections for 'Navigate' (Subscriptions, Resource groups, All resources, Dashboard), 'Tools' (Microsoft Learn, Azure Monitor, Microsoft Defender for Cloud, Cost Management), and 'Useful links' (Technical Documentation, Azure Services, Recent Azure Updates, Azure mobile app).

The screenshot shows the Azure portal interface for managing a PostgreSQL database named 'cloud-glo-challenge'. The 'Connection details' section contains the following environment variables:

```

export PGHOST=cloud-glo-challenge.postgres.database.azure.com
export PGUSER=admin_glo_challenge
export PORT=5432
export PGPASSWORD="your-password"

```

Below this, a note states: "After setting these variables, you can connect to your database server using various PostgreSQL utilities (psql, pg_dump, pg_restore, pgbench, createdb) without specifying connection options. For example, you can now simply type psql to connect."

Here you can see the connection from pgAdmin.

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure under the 'cloud_db' database, including tables like 'dim_departments', 'dim_jobs', 'fact_hired_employees', and 'file_uploads'.
- Properties Panel:** Shows the properties for the 'file_uploads' table, including its definition, security, parameters, default privileges, and advanced settings. The table is defined as follows:


```

      1 -- Table: public.file_uploads
      2
      3 -- SEQUENCE: public.file_uploads_id_seq
      4 DROP TABLE IF EXISTS public.file_uploads;
      
```
- SQL Editor:** Contains the SQL code for creating the table and sequence.
- Connection Details:** Shows the connection information for the 'cloud_db' database, including host name/address ('cloud_glo_challenge.postgres.database.azure.com'), port ('5432'), maintenance database ('postgres'), username ('admin_glo_challenge'), and role ('').
- Scratch Pad:** An empty scratch pad window.

Also, the docker container was uploaded into azure:

Subscription (group) : Azure subscription 1

Subscription ID : 7469d168-3930-46e8-8637-b34e7f54fb7a

Location : East US 2

Deployments : 2 Succeeded

Tags (edit) : Add tags

Resources Recommendations (1)

Name	Type	Location
appGloChallenge	App Service	West US 2
cloudGloChallengePlan	App Service plan	West US 2
cloud-glo-challenge	Azure Database for PostgreSQL - Flexible Server	East US 2
cloudglochallenge	Container registry	East US 2

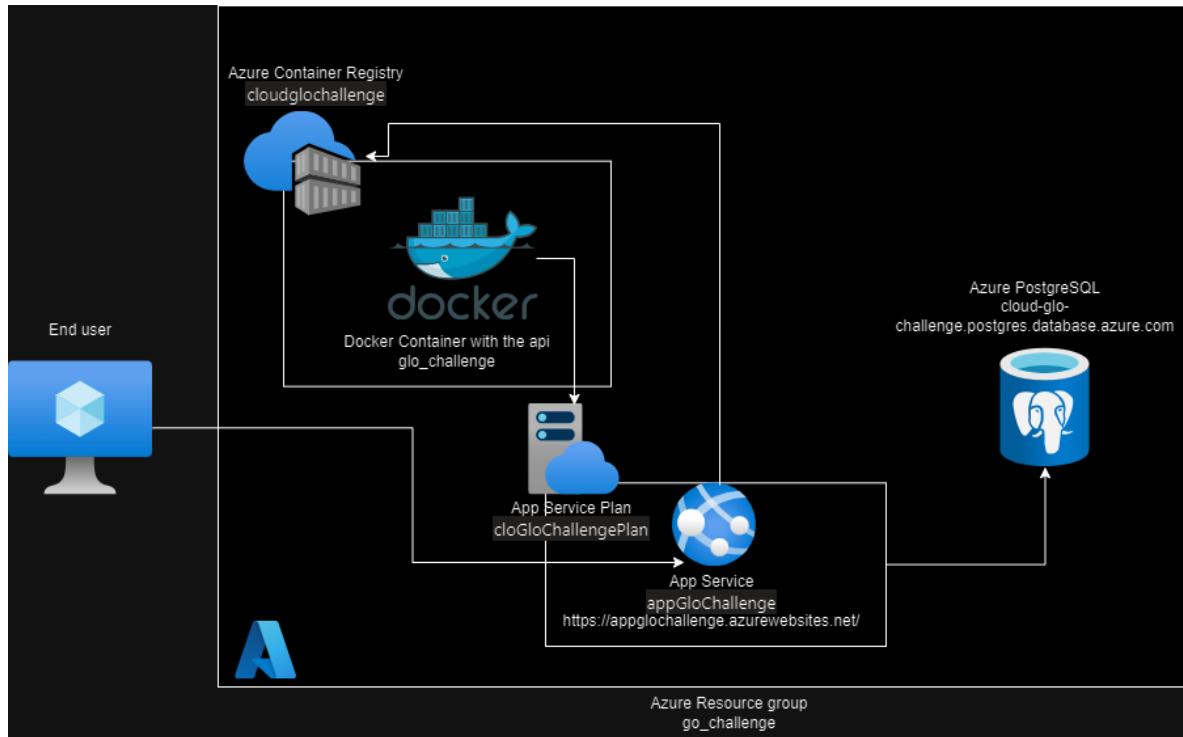
Here you can access to the api.

<https://appglochallenge.azurewebsites.net/>

Flask server is working



Architecture diagram:



Add automated tests to the API

- You can use whichever library that you want
 - Different tests types, if necessary, are welcome

The file test_api.py was created to test automatically the api, the library used was pytest.

```
❶ api.py U test_api.py U x
❷ test_api.py > ...
18 # Test for the csv upload
19 def test_upload_csv(client):
20     # csv file created to test
21     data = BytesIO(b"50,col2\n51,value2\n53,value4")
22     data.name = 'test_file.csv'
23
24 rv = client.post('/upload_csv/dim_departments',
25                   data={'file': (data, 'test_file.csv')},
26                   content_type='multipart/form-data')
27
28 assert rv.status_code == 201
29 assert b'Data inserted correctly!' in rv.data
30
31 # Test for the csv upload, table incorrect
32 def test_upload_csv_2(client):
33     # csv file created to test
34     data = BytesIO(b"1000,col2\n1001,value2\n1002,value4")
35     data.name = 'test_file.csv'
36
37 rv = client.post('/upload_csv/dim_depar',
38                   data={'file': (data, 'test_file.csv')},
39                   content_type='multipart/form-data')
40
41 assert rv.status_code == 400
42 assert b'Invalid table' in rv.data
43
44 # Test for the csv upload, a file with the wrong structure for the table
45 # ... don't upload csv files! ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\user\OneDrive\Documentos\AT\globant\globant_challenge\Api> pytest
PS C:\Users\user\OneDrive\Documentos\AT\globant\globant_challenge\Api> pytest
platform win32 -- Python 3.11.7, pytest-7.4.0, pluggy-1.0.0
rootdir: C:\Users\user\OneDrive\Documentos\AT\globant\globant_challenge\Api
plugins: anyio-4.2.0, flask-1.3.0
collected 5 items

test_api.py .....
=====
5 passed in 0.99s =====
PS C:\Users\user\OneDrive\Documentos\AT\globant\globant_challenge\Api> []
```

You can find the test file under the **Api** folder.

Containerize your application

o Create a Dockerfile to deploy the package

The docker container was created, you can find the code used to configure the docker under the *docker_api* folder.

Here is the execution of the build for docker:

```
C:\Windows\system32\cmd.e: + - x

:\Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Docker Api>docker build -t glo_challenge .
+] Building 1.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 552B
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/python:3.11-slim@sha256:5501a4fe605abe24de87c2f3d6cf9fd760354416a0cad0296cf2
=> => resolve docker.io/library/python:3.11-slim@sha256:5501a4fe605abe24de87c2f3d6cf9fd760354416a0cad0296cf2
=> [internal] load build context
=> => transferring context: 5.67kB
=> CACHED [2/6] WORKDIR /app
=> CACHED [3/6] RUN apt-get update && apt-get install -y libpq-dev gcc
=> CACHED [4/6] COPY requirements.txt .
=> CACHED [5/6] RUN pip install --no-cache-dir -r requirements.txt
=> [6/6] COPY . .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:9def3a9e780b6abb4d0dc2d762a20b205869883598d3578e8c5d83b168ea33c
=> => exporting config sha256:ffd637a90416cae0c915367a8fb10ad9622dc63e6ef8bf031db6675b6040137
=> => exporting attestation manifest sha256:63d988d9d41ca72a5f2e656e819c02fb827d9b2490459ea989932cf3b52f9f8d
=> => exporting manifest list sha256:8af08cffdd2bc99ea9d5520e64e14b7af1152a5947bd9fd6a1ccb6e55a8bb5e7
=> => naming to docker.io/library/glo_challenge:latest
=> => unpacking to docker.io/library/glo_challenge:latest

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview

:\Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Docker Api>docker run -p 5000:5000 glo_challenge
* Serving Flask app 'api'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
```

And the docker running with the api:

```
C:\Windows\system32\cmd.e => => unpacking to docker.io/library/glo_challenge:latest 0.0s
What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Docker Api>docker run -p 5000:5000 glo_challenge
* Serving Flask app 'api'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 461-683-321

C:\Users\User\OneDrive\Documentos\AT\Globant\globant_challenge\Docker Api>docker run -p 5000:5000 glo_challenge
* Serving Flask app 'api'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 439-577-306
172.17.0.1 - [11/Oct/2024 16:01:18] "GET / HTTP/1.1" 200 -
172.17.0.1 - [11/Oct/2024 16:05:31] "POST /upload_csv/dim_jobs HTTP/1.1" 500 -
```

