



► Result Arkon Test

Data Engineer

Resume

- ▶ My github repositories:

https://github.com/anitarr/Arkon_Test

- ▶ Python Code:

https://github.com/anitarr/Arkon_Test/blob/master/src/my_code.ipynb

- ▶ SQL Code

https://github.com/anitarr/Arkon_Test/blob/master/sql/test5-7.sql

- ▶ PostgreSQL DB in Docker

https://github.com/anitarr/Arkon_Test/blob/master/Readme.md

- ▶ Connecting postgres with Python

https://github.com/anitarr/Arkon_Test/blob/master/postgresDB.ipynb

Initialize a new Git repository

▶ `git init`

Add files to the staging area

▶ `git add -A`

Commit the changes

▶ `git commit -m "First commit"`

Add the GitHub remote

▶ `git remote add origin
https://github.com/anitarr/Arkon_test.git`

Push the changes to GitHub

▶ `git push -u origin main`

Git and Github

Python code

Question 1

#Importing the pandas and numpy libraries

- ▶ import pandas as pd
- ▶ import numpy as np

#Read data from “.parquet” files

- ▶ df_parquet =
pd.read_parquet(r"D:\DATA_Analysis\Arkon\data\data2
(1).parquet")

#Read data from “.csv” files

- ▶ df_data1 =
pd.read_csv(r"D:\DATA_Analysis\Arkon\data\Data1.csv")

#Question 1. Join the two datasets into one

- ▶ df_union = pd.concat([df_parquet, df_data1])

#Delete NAN and duplicate data

- ▶ df_union_dna = df_union.dropna().drop_duplicates()

Question 2. Unique 'starships' values

- ▶ `valores_unicos = df_union["starships"].unique()`

Question 3. Generate a record count on the group [Skin_color, eye_color]

- ▶ `count_s_e = df_union.groupby(['skin_color', 'eye_color']).size().reset_index(name='Count')`

Question 4. Generate a table with the duplicate 'Names' and how many times they are repeated.

- ▶ `name_counts = df_union['name'].value_counts().reset_index()`
- ▶ `name_counts.columns = ['Name', 'Count']`
- ▶ `duplicates = name_counts[name_counts['Count'] > 1]`

Python code

Question 3,4,5

Python code

Questions 5,6,7

#Question 5. Filter in python

```
▶ filtered_df = df_union[
    (df_union['height'] >= 180) &
    (df_union['height'] <= 190) &
    (df_union['sex'] == 'male') &
    (df_union['hair_color'] != 'none')]
```

#Question 6. Record count on the group ['skin_color', 'eye_color']

```
▶ avg = df_union['mass'].mean()
▶ # create new column 'flat' and show results
▶ df_union['flat'] = df_union['mass'].apply(lambda x: 1 if x > avg
    else 0)
▶ mass_flat = df_union[['name', 'mass', 'flat']]
```

#Question 7. Metrics column 'species'

```
▶ result = df_union.groupby('species').agg(
    avg_height=('height', 'mean'),
    max_height=('height', 'max'),
    min_height=('height', 'min')).reset_index()
```

/*Pregunta 2 Unique 'starships' values */

▶ SELECT DISTINCT starships FROM data_union;

/*Pregunta 3 Generate a record count on the group [Skin_color, eye_color] */

▶ SELECT COUNT(*), skin_color, eye_color FROM data_union
group by skin_color, eye_color;

/*Pregunta 4 Generate a table with the duplicate 'Names' and how many times they are repeated.*/

▶ SELECT name, COUNT(*) AS cantidad_duplicados
FROM data_union
GROUP BY name
HAVING COUNT(*) > 1;

SQL code

Questions 2,3,4

SQL code

Questions 5,6,7

/* Question 5 . Filter in SQL*/

▶ SELECT name FROM testdata.data_union
WHERE height BETWEEN 180 AND 190 AND sex = 'male' AND
hair_color!='none';

/* Question 6 Record count on the group ['skin_color', 'eye_color']*/

▶ SELECT name, mass,
CASE
WHEN mass > (SELECT AVG(mass) FROM data_union) THEN 1
ELSE 0
END AS bandera
FROM data_union;

/* Question 7 Metrics column 'species' */

▶ SELECT species,
AVG(height) AS altura_promedio,
MAX(height) AS altura_maxima,
MIN(height) AS altura_minima
FROM data_union
GROUP BY species;

PostgreSQL DB in Docker

Pull/Download Official Postgres Image From Docker Hub

- ▶ `docker pull postgres`

Create and Run Postgres Container

- ▶ `docker run -d --name arkon_data -p 5432:5432 -e POSTGRES_PASSWORD=pass1234 postgres`

Initialize the database connection

```
▶ db = PostgresDB(  
    host="localhost",  
    database="postgres",  
    user="postgres",  
    password="pass1234"  
)  
db.connect()
```

Postgres DB with python

Generate SQL for creating the table

- ▶ `table_name = "data_union"`
- ▶ `schema_name = "arkon_data"`
- ▶ `columns = []`
- ▶ `for column_name, dtype in df_union.dtypes.items():`
- ▶ `sql_type = map_dtype_to_sql(dtype)`
- ▶ `columns.append(f"{column_name} {sql_type}")`
- ▶ `create_table_query = f"""`
- ▶ `CREATE SCHEMA IF NOT EXISTS {schema_name};`
- ▶ `CREATE TABLE IF NOT EXISTS {schema_name}.{table_name}`
- ▶ `(`
- ▶ `id SERIAL PRIMARY KEY,`
- ▶ `{', '.join(columns)}`
- ▶ `);`
- ▶ `"""`
- ▶ `db.execute_query(create_table_query)`

Postgres DB with python

Insert data into the table

```
▶ insert_query_template = f"""  
    INSERT INTO {schema_name}.{table_name} ({',  
    '.join(df_union.columns)}) VALUES ({',  
    '.join(['%s'] * len(df_union.columns))});  
    """
```

```
▶ cursor = db.connection.cursor()  
▶ for index, row in df_union.iterrows():  
▶ cursor.execute(insert_query_template,  
    tuple(row))  
▶ db.connection.commit()  
▶ cursor.close()
```

Close the database connection

```
▶ db.close_connection()
```

Postgres DB with python

Questions





Thanks