WORKSHOP 001: Candidates - Job test

By: David Venté Polo

- Table of contents

Description

In this workshop we have an ETL's process based in a CSV file with randomly generated data of candidates from February 22, 2020 until May 13, 2022 with 50.000 rows and 10 columns that after processing are now 16 columns to create a dashboard of candidates hired.

The process start with the CSV file that is read by an Jupyter Notebook to transfer the data to PostgreSQL. Then this data is read by another Jupyter Notebook that applies ETL methodologies to know, organize, calculate and modify it; and same data with additions are loaded again to PostgreSQL. Finally, a dashboard retrieves it to communicate its raison d'être.

- Tools used are:
  - Python 🐍.
  - Git 📦.
  - Ngrok 🛜.
  - Postgres 💽.
  - PgAdmin 4 🖨️.
  - AWS RDS ☁️.
  - GitHub 💼.
  - Looker Studio 📊.
  - Visual Studio Code 📝.

Purpose - Raison d'être

The project is oriented to present in a general way the candidates that are hired under the requirements provided by a job test and to perform it in a visual way by means of sampling specific data to comply with it.

The visualizations that I am expecting are:

- Hires by technology (pie chart).
- Hires by year (horizontal bar chart).
- Hires by seniority (bar chart).
- Hires by country over years (USA, Brazil, Colombia, and Ecuador only) (multiline chart).

This project is the first workshop of the ETL subject of Artificial Intelligence & Data Engineering at the Universidad Autónoma de Occidente under the teaching of Javier Alejandro Vergara Zorilla.

Process

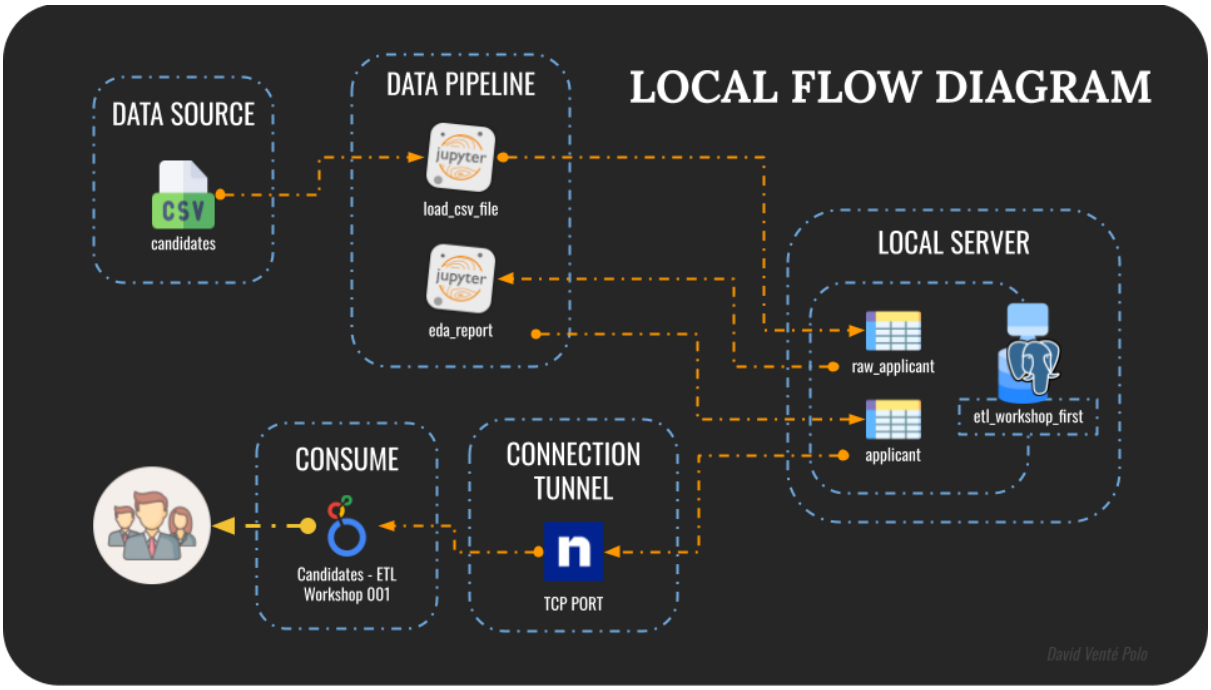
In this part, we will see the process to go from a CSV file named [candidates] to a dashboard in Looker Studio named [Candidates - ETL Workshop 001] shown in the diagrams below, to offer the latter to people.

1. Once I have the requirements and the objective to be achieved, flow diagrams are made to have a field visualization of the process to be developed..
    - **Metadata of diagrams:** In this part, there are a few explaining of every block and his properties that you will look in the diagrams.
        - **Data source block**: There is a CSV file with 50.000 rows of candidates data for this job test. This starting point file is called *candidates.csv*.
        - **Data pipeline block**: In this section, there are two jupyter notebooks, which are:
            - **load_csv_file**. The process of reading the CSV file and loading of data to **raw_applicant** table in a PostgreSQL Database called **etl_workshop_first**.
            - **eda_report**. The process of data collection of **raw_applicant** table in **etl_workshop_first** database to explorate, transform and clean it. Then load data to **applicant** tabe in the database.
        - **Server block**: There are two options, load to the local server or to the AWS RDS server. On either server, the **etl_workshop_first** database will be used which will have two tables, **raw_applicant** and **applicant**.

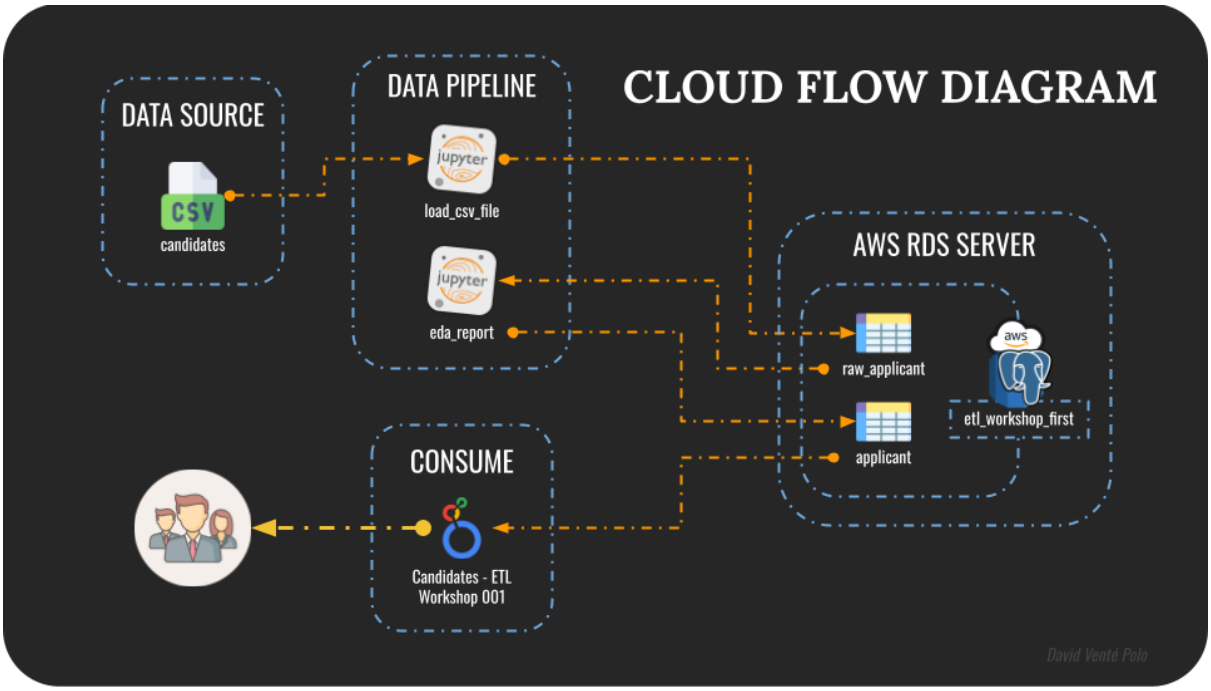
          If the database doesn't exist, **load_csv_file** notebook will create it and its structure.
        - **Connection tunnel block**: In this, I use ngrok to create a public tunnel with *TCP protocol* to connect *Looker Studio* with *Local PostgreSQL* and get all data of **applicant** table.

          AWS RDS give us a *Public Host* to connect if the setting of *AWS VPC* allows it.
        - **Consume block**: Here, we look the dashboard which users will see and can analyze. This dashboard is in *Looker Studio*, which gives us public access.
        - **Users representation**: Represents each of us.

Local flow diagram

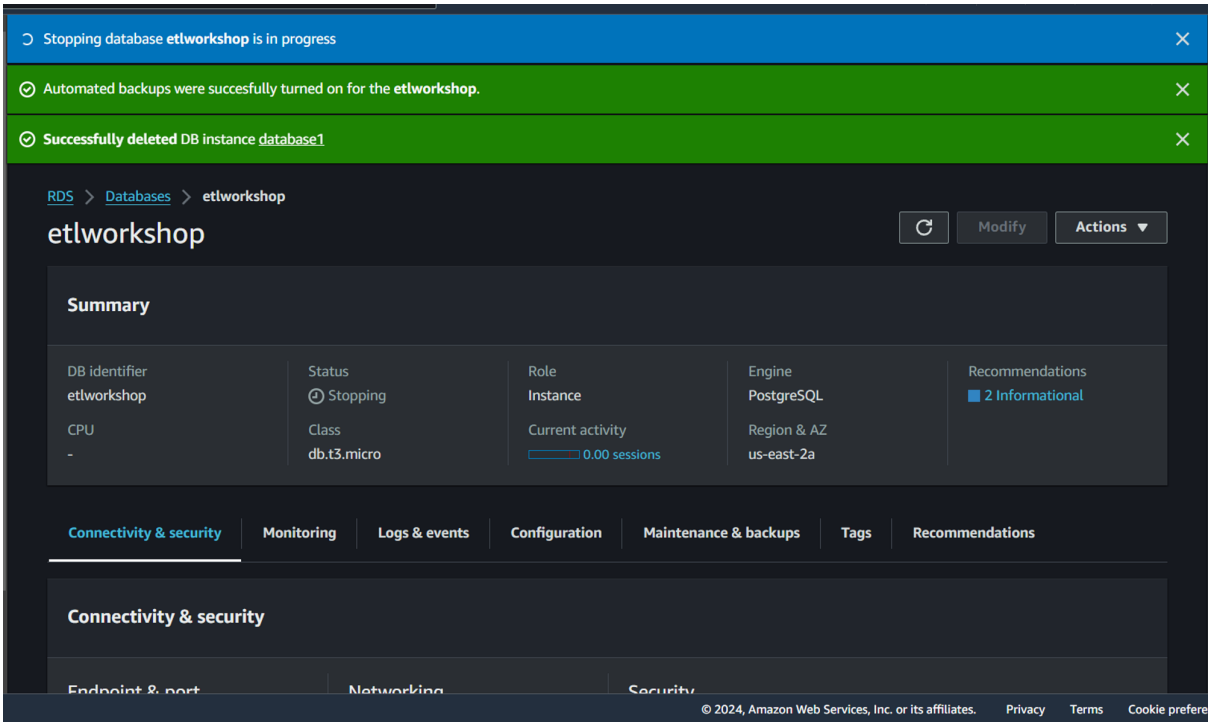


Cloud flow diagram

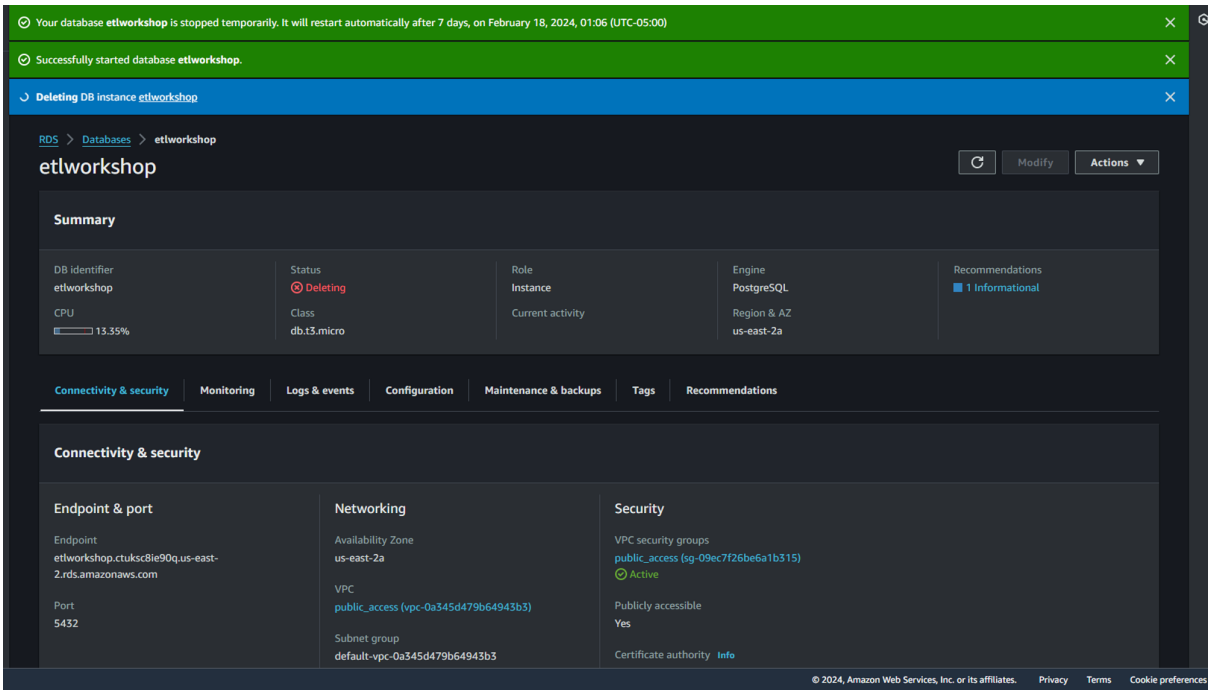


2. Create an account in AWS to use AWS RDS.

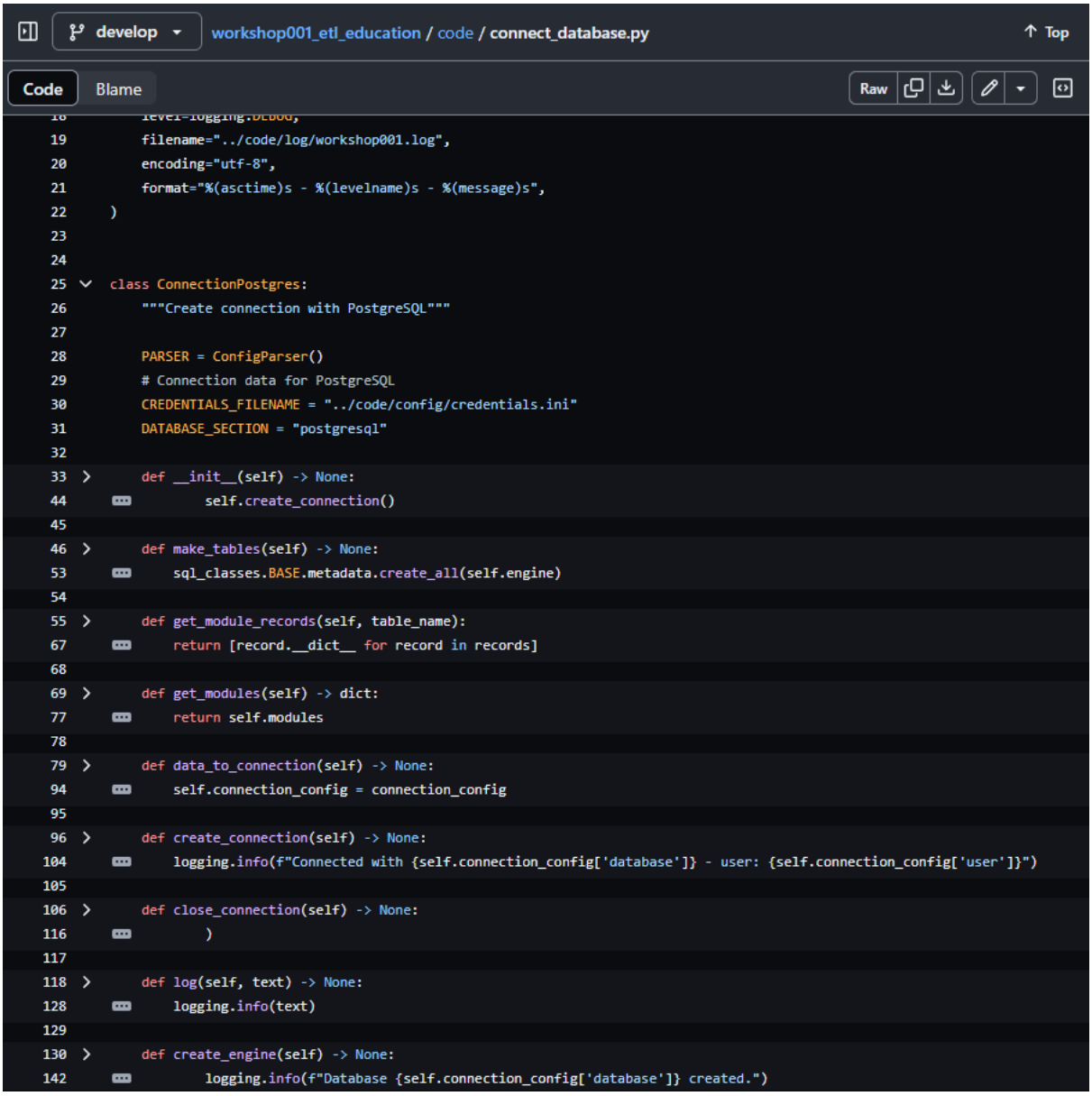
3. Creation on database in AWS RDS and stopping for now to follow with configuration.



After checking Free Tier functions, I see the use of 41 hours of database, that wants to stop it doesn't work and I must delete it.

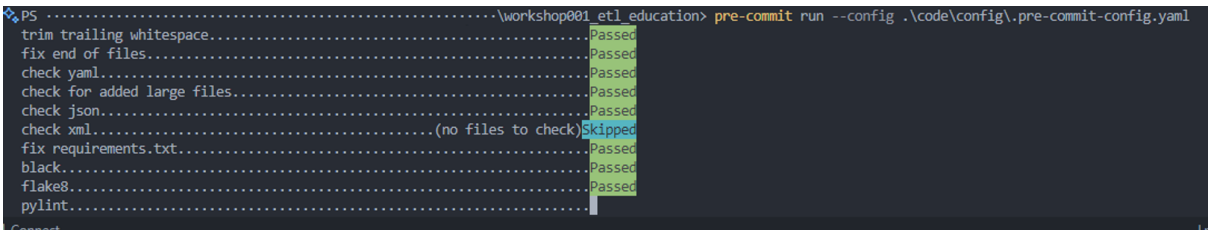


4. **connect_database.** To connect the **load_csv_file** notebook with **candidates** (CSV file), a class called ConnectionPostgres is created that through the SQLAlchemy library that is highly used by its ORM, that depending on the HOST and PORT configuration credentials, links to PostgreSQL on the AWS RDS server or local one.



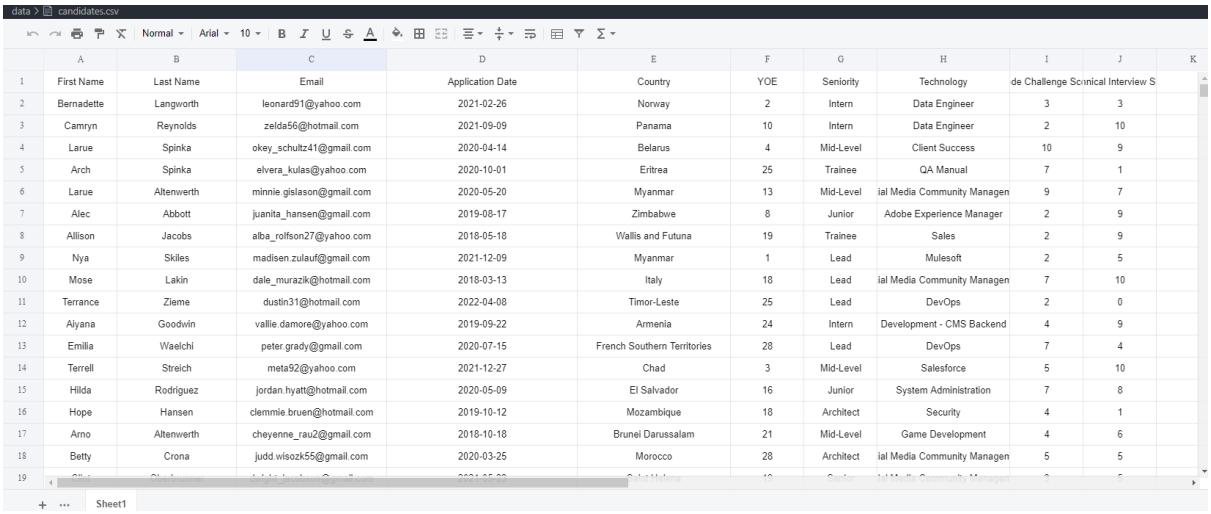
ConnectionPostgres class in connect_database.py

Additional, use of pre-commits to apply good practices.



5. **local_csv_file:** Once the connection has been made, the database and its structure is created by the class if it doesn't exist, including **raw_applicant** and **applicant** tables. When the proccess is executed, the tables will be reset if they already exists.

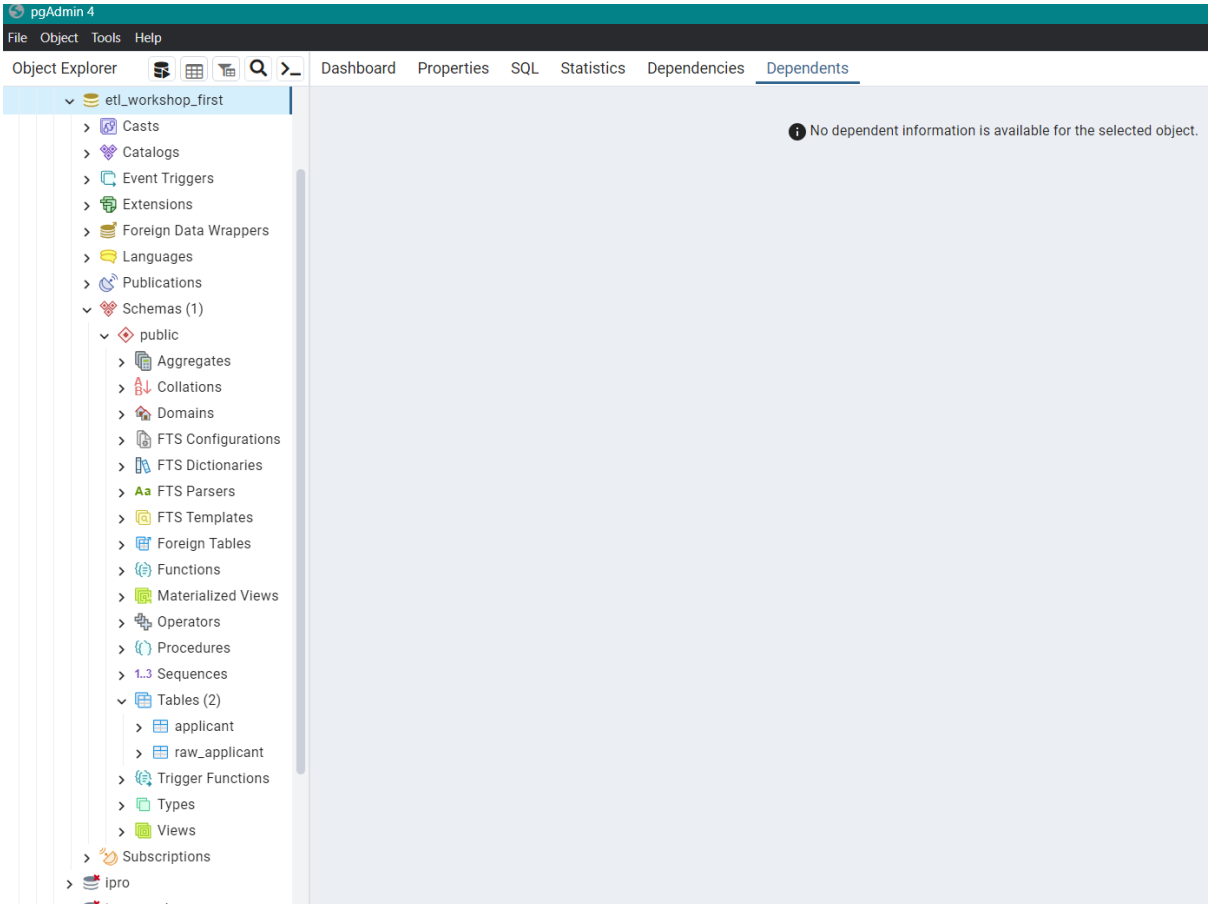
First, I explore the data in tabular from to know the number of columns, names and some data.



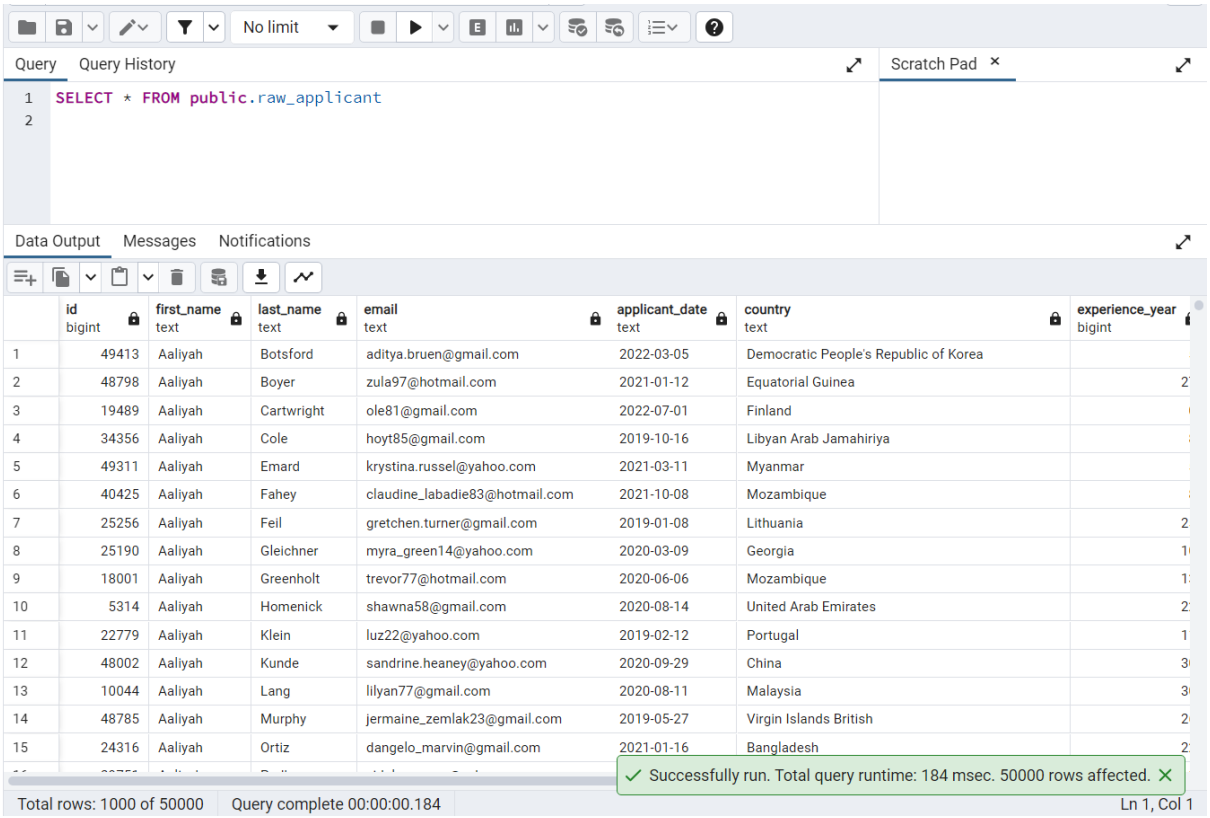
	A	B	C	D	E	F	G	H	I	J	K
1	First Name	Last Name	Email	Application Date	Country	YOE	Seniority	Technology	de Challenge Sc	nical Interview S	
2	Bernadette	Langworth	leonard91@yahoo.com	2021-02-26	Norway	2	Intern	Data Engineer	3	3	
3	Camryn	Reynolds	zelda56@hotmail.com	2021-09-09	Panama	10	Intern	Data Engineer	2	10	
4	Larue	Spinka	okey_schultz41@gmail.com	2020-04-14	Belarus	4	Mid-Level	Client Success	10	9	
5	Arch	Spinka	elvera_kulas@yahoo.com	2020-10-01	Eritrea	25	Trainee	QA Manual	7	1	
6	Larue	Altenwerth	minnie.gislason@gmail.com	2020-05-20	Myanmar	13	Mid-Level	ial Media Community Managen	9	7	
7	Alec	Abbott	juanita_hansen@gmail.com	2019-08-17	Zimbabwe	8	Junior	Adobe Experience Manager	2	9	
8	Allison	Jacobs	alba_rolfson27@yahoo.com	2018-05-18	Wallis and Futuna	19	Trainee	Sales	2	9	
9	Nya	Skiles	madisen.zulauf@gmail.com	2021-12-09	Myanmar	1	Lead	Mulesoft	2	5	
10	Mose	Lakin	dale_murazik@hotmail.com	2018-03-13	Italy	18	Lead	ial Media Community Managen	7	10	
11	Terrance	Zieme	dustin31@hotmail.com	2022-04-08	Timor-Leste	25	Lead	DevOps	2	0	
12	Aiyana	Goodwin	vallie.damore@yahoo.com	2019-09-22	Armenia	24	Intern	Development - CMS Backend	4	9	
13	Emilia	Waelchi	peter.grady@gmail.com	2020-07-15	French Southern Territories	28	Lead	DevOps	7	4	
14	Terrell	Streich	meta92@yahoo.com	2021-12-27	Chad	3	Mid-Level	Salesforce	5	10	
15	Hilda	Rodriguez	jordan.hyatt@hotmail.com	2020-05-09	El Salvador	16	Junior	System Administration	7	8	
16	Hope	Hansen	clemmie.bruen@hotmail.com	2019-10-12	Mozambique	18	Architect	Security	4	1	
17	Arno	Altenwerth	cheyenne_rau2@gmail.com	2018-10-18	Brunei Darussalam	21	Mid-Level	Game Development	4	6	
18	Betty	Crona	judd.wisozk55@gmail.com	2020-03-25	Morocco	28	Architect	ial Media Community Managen	5	5	

Second, viewing the columns and their data, a column named `id` is added to the data from the *candidates* file and the `YOE` column is renamed to `experience_year` .

Third, I load the data to **raw_applicant** table and verify it.



Creation of `etl_workshop_first` database is created in Local PostgreSQL.



	id bigint	first_name text	last_name text	email text	applicant_date text	country text	experience_year bigint
1	49413	Aaliyah	Botsford	aditya.bruen@gmail.com	2022-03-05	Democratic People's Republic of Korea	
2	48798	Aaliyah	Boyer	zula97@hotmail.com	2021-01-12	Equatorial Guinea	2
3	19489	Aaliyah	Cartwright	ole81@gmail.com	2022-07-01	Finland	
4	34356	Aaliyah	Cole	hoyt85@gmail.com	2019-10-16	Libyan Arab Jamahiriya	
5	49311	Aaliyah	Emard	krystina.russel@yahoo.com	2021-03-11	Myanmar	
6	40425	Aaliyah	Fahey	claudine_labadie83@hotmail.com	2021-10-08	Mozambique	
7	25256	Aaliyah	Feil	gretchen.turner@gmail.com	2019-01-08	Lithuania	2
8	25190	Aaliyah	Gleichner	myra_green14@yahoo.com	2020-03-09	Georgia	1
9	18001	Aaliyah	Greenholt	trevor77@hotmail.com	2020-06-06	Mozambique	1
10	5314	Aaliyah	Homenick	shawna58@gmail.com	2020-08-14	United Arab Emirates	2
11	22779	Aaliyah	Klein	luz22@yahoo.com	2019-02-12	Portugal	1
12	48002	Aaliyah	Kunde	sandrine.heaney@yahoo.com	2020-09-29	China	3
13	10044	Aaliyah	Lang	lilyan77@gmail.com	2020-08-11	Malaysia	3
14	48785	Aaliyah	Murphy	jermaine_zemlak23@gmail.com	2019-05-27	Virgin Islands British	2
15	24316	Aaliyah	Ortiz	dangelo_marvin@gmail.com	2021-01-16	Bangladesh	2

Loading of `raw_applicant` database in `etl_workshop_first` .

6. **eda_report:** Now, I perform the EDA process.

   With 50.000 rows and 11 columns we look the data obtained of **raw_applicant** table.

```
Index(['_sa_instance_state', 'email', 'id', 'country', 'seniority',
       'code_challenge_score', 'first_name', 'last_name', 'applicant_date',
       'experience_year', 'technology', 'technical_interview_score'],
      dtype='object')
```

   There is a column '*_sa_instance_state*' unknown. Inquiring is a record that adds SQLAlchemy to store the status of the object instance. After a validation process, this table is deleted.

```
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   email                      50000 non-null  object
 1   id                         50000 non-null  int64
 2   country                    50000 non-null  object
 3   seniority                  50000 non-null  object
 4   code_challenge_score       50000 non-null  int64
 5   first_name                 50000 non-null  object
 6   last_name                  50000 non-null  object
 7   applicant_date             50000 non-null  object
 8   experience_year            50000 non-null  int64
 9   technology                 50000 non-null  object
 10  technical_interview_score  50000 non-null  int64
dtypes: int64(4), object(7)
memory usage: 4.2+ MB
```

By means of the value_counts method, taking into account that there are 50.000 rows and 10 columns, it is found that:

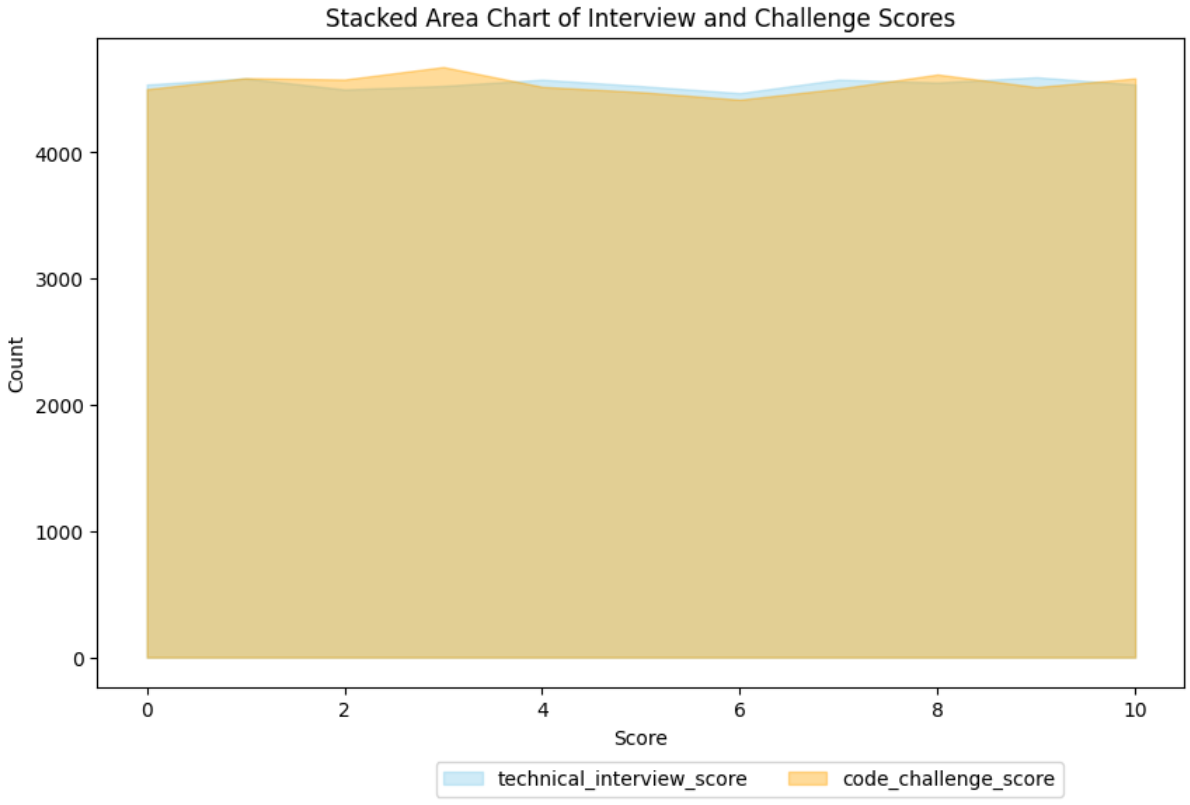
1. Two score columns (`code_challenge_score` *and* `technical_interview_score`) have values among 0 and 10, where 10 is the maximum qualitification.
2. There are several tecnologies that could be agrouped together and provide us with better insights.
3. There are emails that repeat among all records, which could be due to their application as candidates in different areas. So, I will focus in this part.
4. I consider it important to extract the year and month in order to treat them in a better way in the dashboard.

Developing each point:

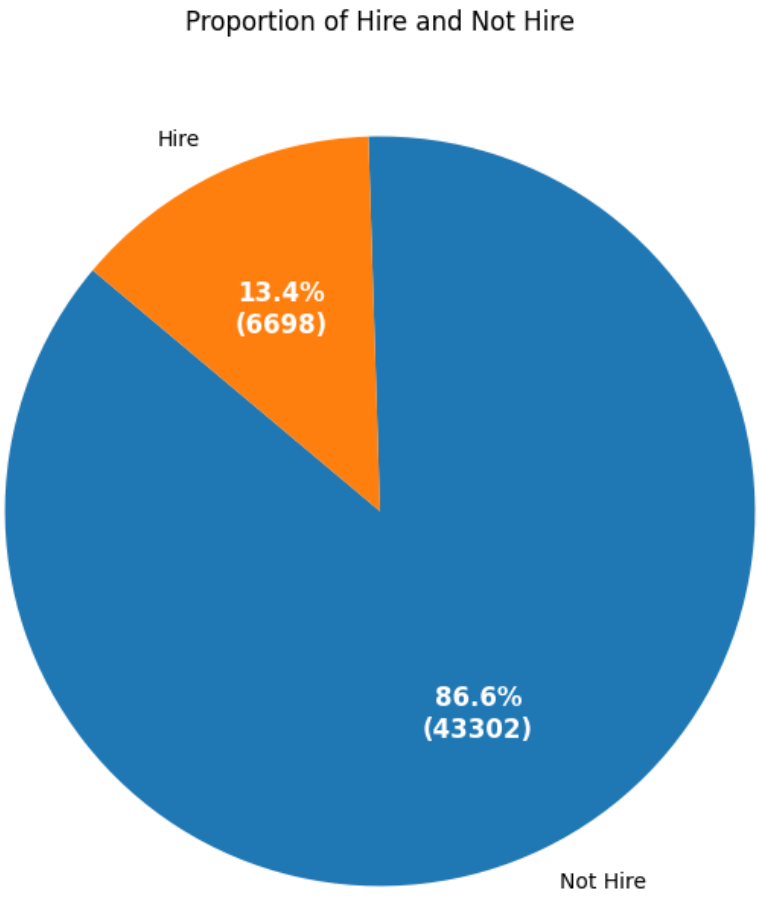
1. Code_challenge_score and technical_interview_score & is hire?

In this point, create a new column on the candidates are hire or not. As is the requirement, a candidate is HIRED when she/he has both scores greater than or equal to 7.

Looking at similarities between the columns, `technical_interview_score` and `code_challenge_score`, I focus on their behaviors according to the score from 0 to 10.



Now, I create the `is_hire` column and the proportion between candidates and non-candidates is observed.



6.698 of 50.000 applicants are hired under the requirements proposed above.

2. Grouping tecnologies to better insights.

In this point, find out common tecnologies in a generalize way and store them in the dataframe.

We have 24 different technologies, so I'm going to group them into the following 8 categories:

- **Software Development**:
  - *Development - Backend*
  - *Development - CMS Backend*
  - *Development - CMS Frontend*
  - *Development - Frontend*
  - *Development - FullStack*
  - *Game Development*
- **Data & Analytics**
  - *Business Intelligence*

    - *Business Analytics / Project Management*
    - *Data Engineer*
- **IT Infrastructure & Operations**:
    - *Database Administration*
    - *System Administration*
    - *DevOps*
- **Quality Assurance**
    - *QA Manual*
    - *QA Automation*
- **Security**
    - *Security*
    - *Security Complilance*
- **Customer Relationship and Success**
    - *Client Success*
    - *Sales*
    - *Social Media Community Management*
    - *Salesforce*
- **Content Management and Design**
    - *Technical Writing*
    - *Design*
    - *Adobe Experience Manager*
- **Integration and Connectivity**
    - *Mulesoft*

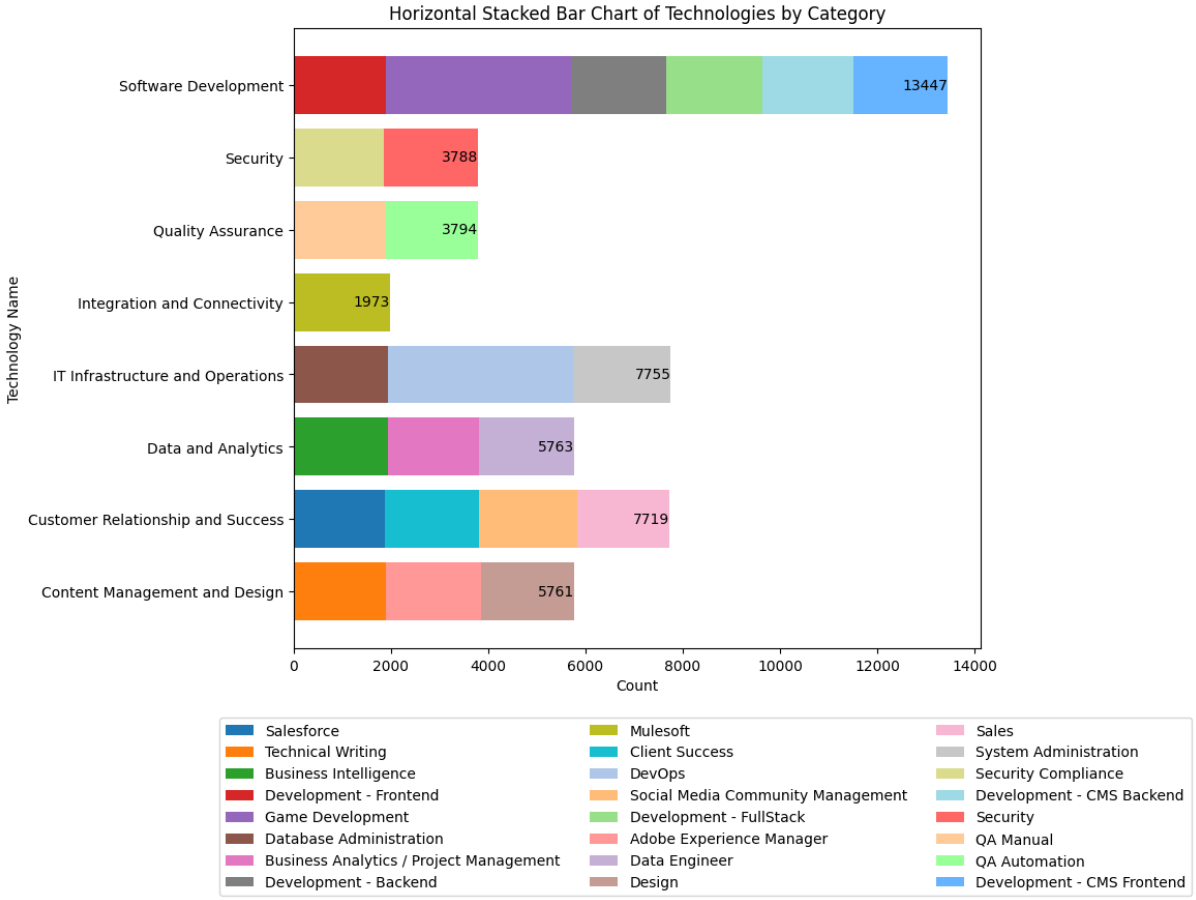
Applied that, change from:

```
technology
Game Development                          3818
DevOps                                    3808
Social Media Community Management         2028
System Administration                     2014
Mulesoft                                  1973
Development - Backend                     1965
Development - FullStack                   1961
Adobe Experience Manager                  1954
Data Engineer                             1951
Security                                  1936
Development - CMS Frontend                1934
Business Intelligence                     1934
Database Administration                   1933
Client Success                            1927
Design                                    1906
QA Manual                                 1902
Technical Writing                         1901
QA Automation                             1892
Sales                                     1890
Development - Frontend                    1887
Development - CMS Backend                 1882
Business Analytics / Project Management   1878
Salesforce                                1874
Security Compliance                       1852
Name: count, dtype: int64
```

To:

```
technology_category
Software Development                13447
IT Infrastructure and Operations     7755
Customer Relationship and Success    7719
Data and Analytics                   5763
Content Management and Design        5761
Quality Assurance                    3794
Security                             3788
Integration and Connectivity         1973
Name: count, dtype: int64
```

Seeing in a better way:



This grouping reduces the number of categories from 24 to 8. In which Software Development is the technological area with more applicants and Integration and Connectivity where Mulesoft technology is the least applied.

3. Focus in emails repeated. Is there a reason?

Explore if there is a rationale (knowing its random origin) of emails repeated and if it can give us an insight.

There are 165 records in which emails have been repeated at least 2 times.

Although the data has random origin, it's interesting 159 (96.36%) of 165 had changed technology at the time they applied.

4. Just a matter of time

Checking the `applicant_date` column, there are:

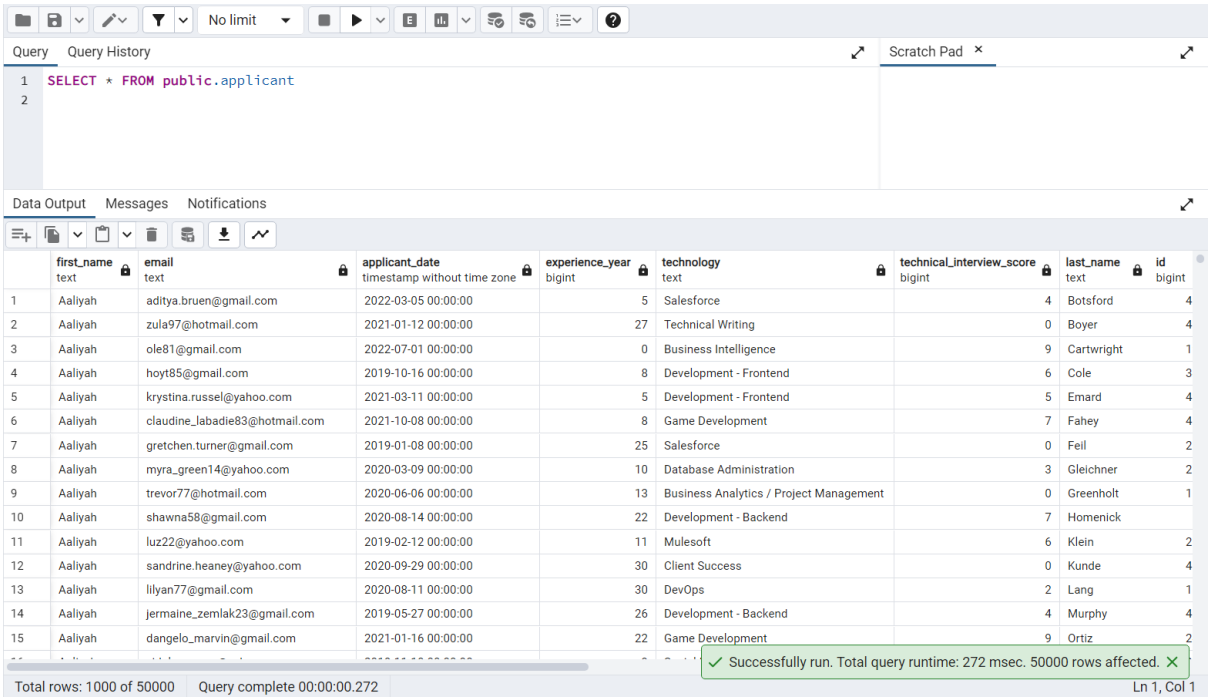
1. By Year, the quantity are:

```
applicant_year
2020    11237
2018    11061
2021    11051
2019    11009
2022     5642
Name: count, dtype: int64
```

2. By month:

```
applicant_month_name
May          4752
March        4649
January      4618
June         4566
April        4542
February     4285
July         3996
October      3813
August       3783
December     3777
September    3610
November     3609
Name: count, dtype: int64
```

Load this dataframe to applicant table in PostgreSQL

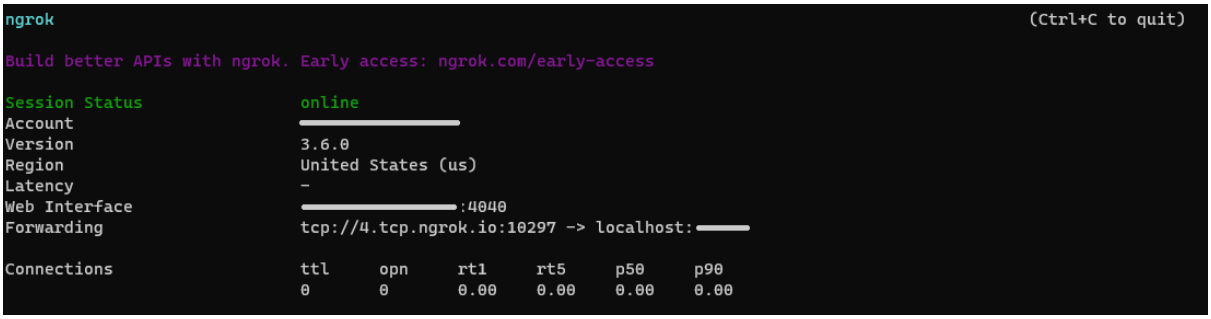


	first_name text	email text	applicant_date timestamp without time zone	experience_year bigint	technology text	technical_interview_score bigint	last_name text	id bigint
1	Aaliyah	aditya.bruen@gmail.com	2022-03-05 00:00:00	5	Salesforce	4	Botsford	4
2	Aaliyah	zula97@hotmail.com	2021-01-12 00:00:00	27	Technical Writing	0	Boyer	4
3	Aaliyah	ole81@gmail.com	2022-07-01 00:00:00	0	Business Intelligence	9	Cartwright	1
4	Aaliyah	hoyt85@gmail.com	2019-10-16 00:00:00	8	Development - Frontend	6	Cole	3
5	Aaliyah	krystina.russel@yahoo.com	2021-03-11 00:00:00	5	Development - Frontend	5	Emard	4
6	Aaliyah	claudine_labadie83@hotmail.com	2021-10-08 00:00:00	8	Game Development	7	Fahey	4
7	Aaliyah	gretchen.turner@gmail.com	2019-01-08 00:00:00	25	Salesforce	0	Feil	2
8	Aaliyah	myra_green14@yahoo.com	2020-03-09 00:00:00	10	Database Administration	3	Gleichner	2
9	Aaliyah	trevor77@hotmail.com	2020-06-06 00:00:00	13	Business Analytics / Project Management	0	Greenholt	1
10	Aaliyah	shawna58@gmail.com	2020-08-14 00:00:00	22	Development - Backend	7	Homenick	
11	Aaliyah	luz22@yahoo.com	2019-02-12 00:00:00	11	Mulesoft	6	Klein	2
12	Aaliyah	sandrine.heaney@yahoo.com	2020-09-29 00:00:00	30	Client Success	0	Kunde	4
13	Aaliyah	lilyan77@gmail.com	2020-08-11 00:00:00	30	DevOps	2	Lang	1
14	Aaliyah	jermaine_zemlak23@gmail.com	2019-05-27 00:00:00	26	Development - Backend	4	Murphy	4
15	Aaliyah	dangelo_marvin@gmail.com	2021-01-16 00:00:00	22	Game Development	9	Ortiz	2

Loading of `applicant` database in `etl_workshop_first`.

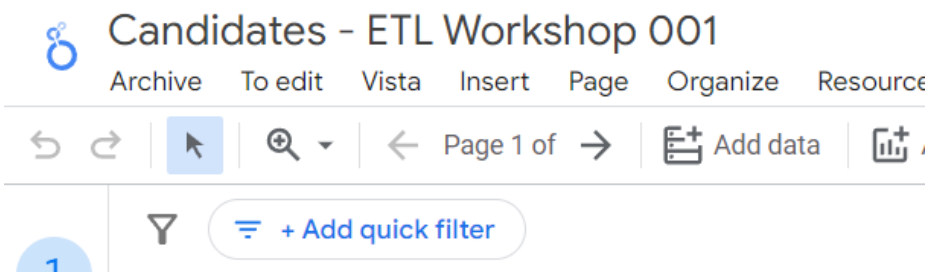
7. **Local PostgreSQL:** Create the *TCP tunnel* to connect *Local PostgreSQL* to *Looker Studio*.

   After install *Ngrok*, I execute it to get the following:

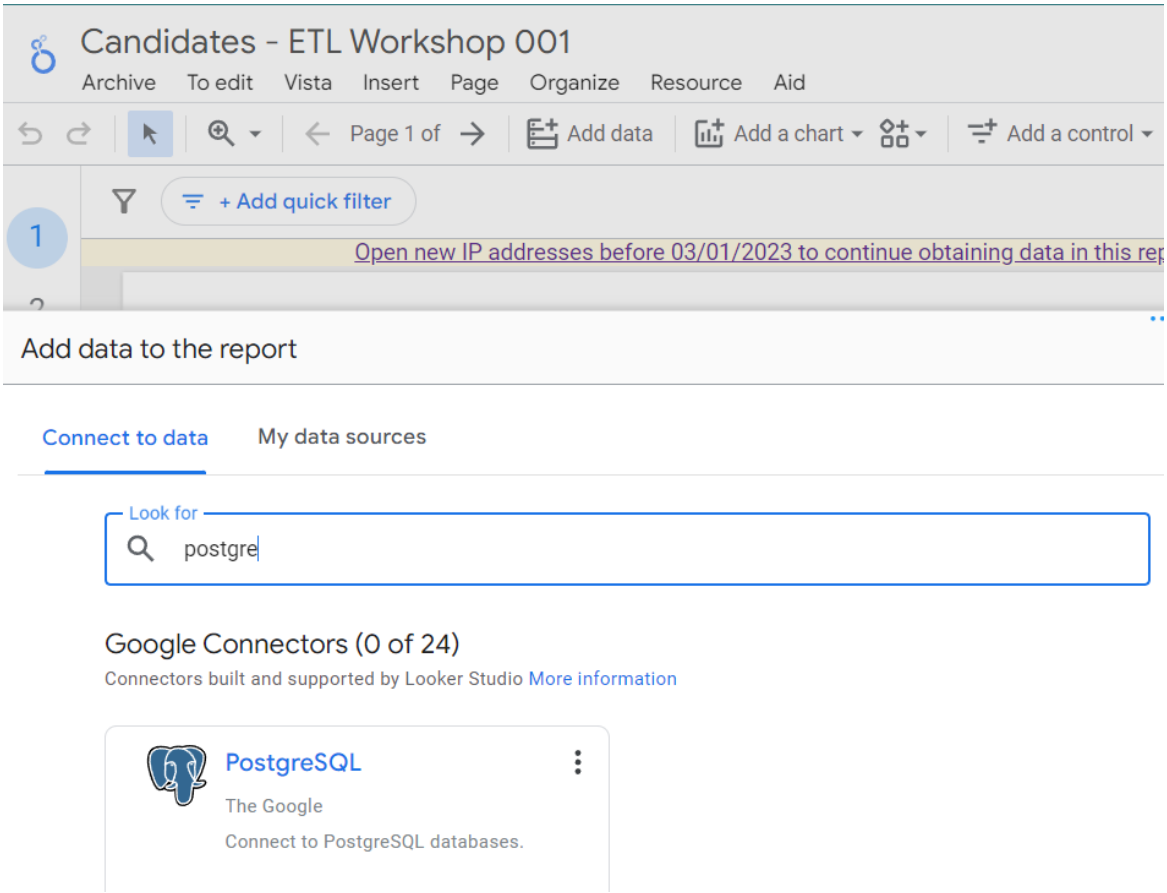


According the example, my host is `4.tcp.ngrok.io` and my port `10297` to put in "*database authentication*" in PostgreSQL connector of *Looker Studio*.

8. Now, I go to Looker Studio to look the dashboard created with these data.
   a. Click on *Add data* option:



   b. Choose PosgreSQL Connector:

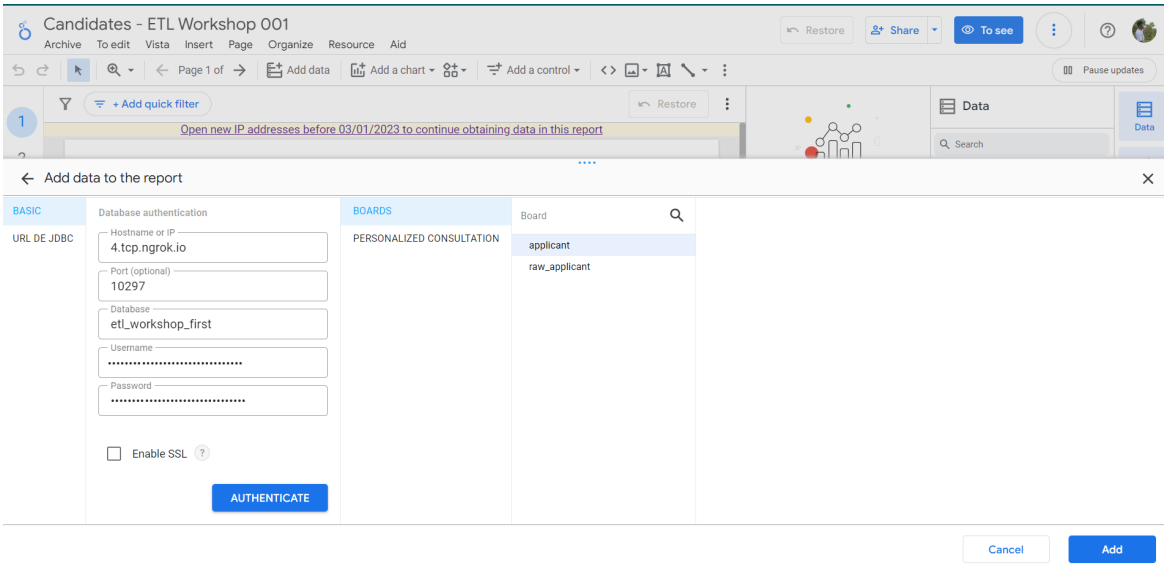


   c. **Local PostgreSQL:** Use the credentials to connect and choose *applicant* table of the database.

   Using my example credentials:

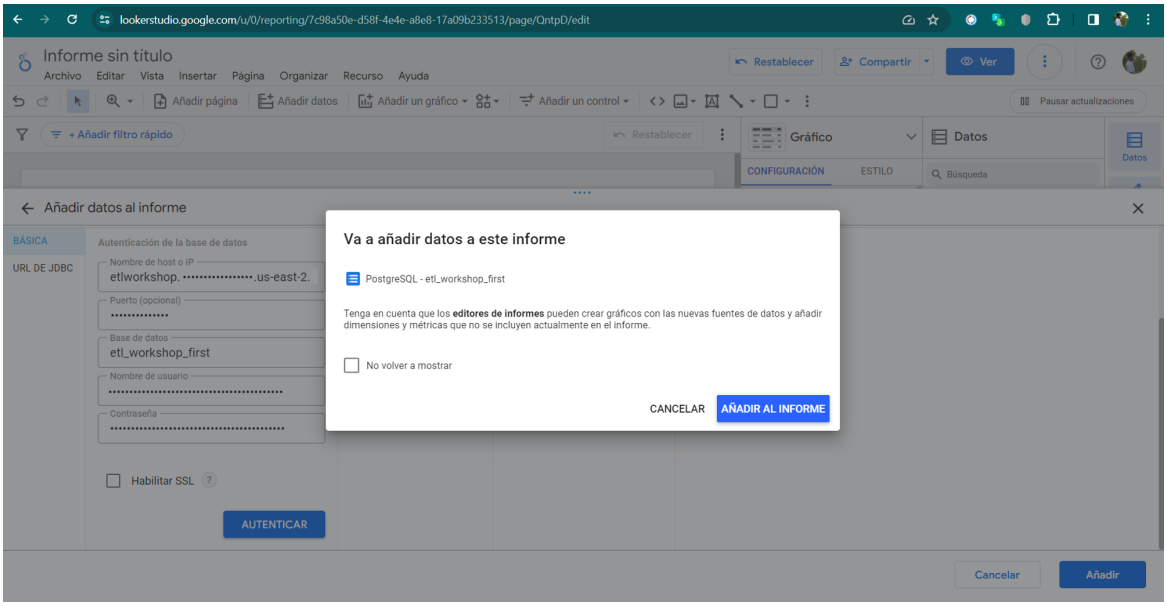
```
host = 4.tcp.ngrok.io
port = 10297
```

```
database = etl_workshop_first
user = --------
password = --------
```



Now, I have the data sync with Looker Studio.

d. **PostgreSQL in AWS RDS:** I connect Looker Studio with **etl_workshop_first** database in AWS RDS:

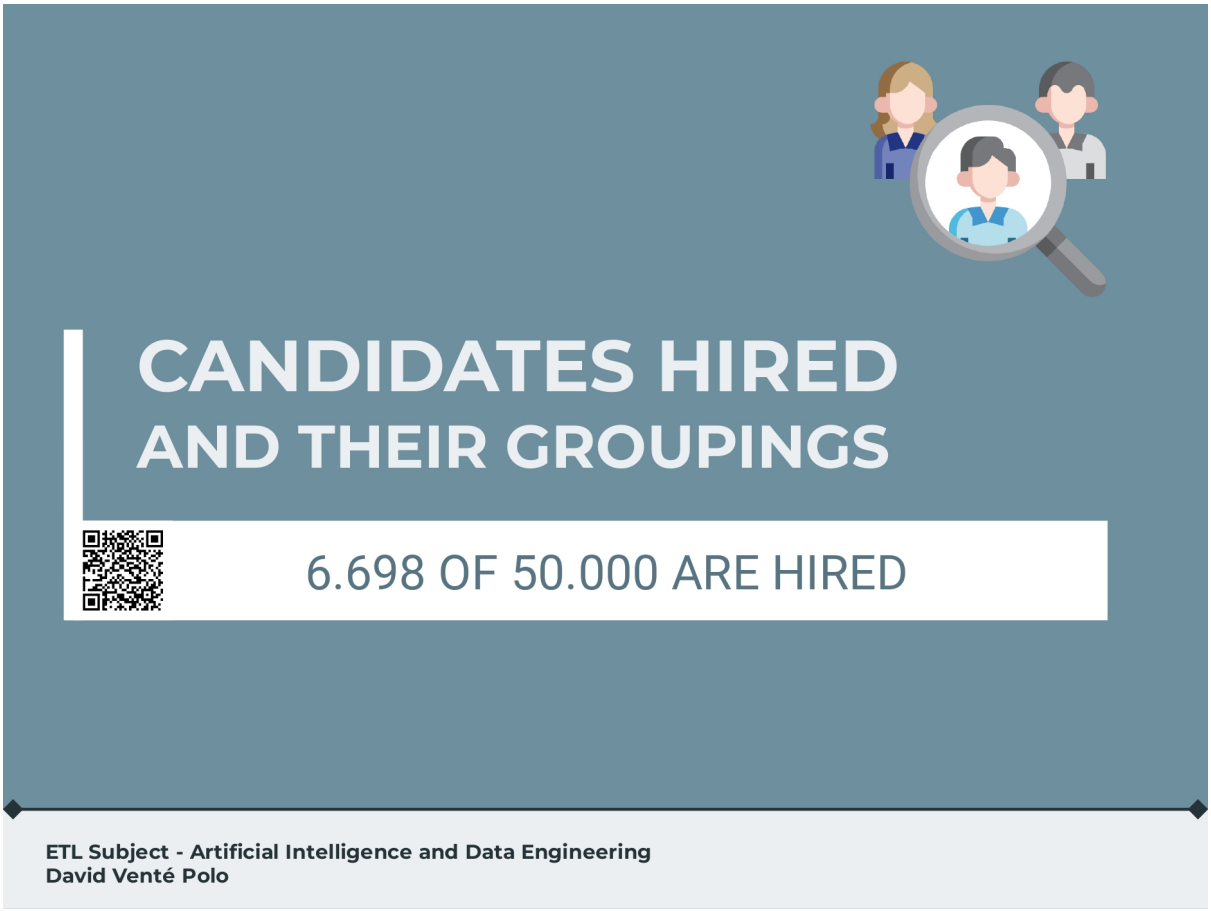


Results

Link of the dashboard: Dashboard in Looker Studio.

I create a dashboard in Looker Studio with 5 slides:

1. Home



First slide of Dashboad.

Here, it's of which subject the dashboard will have, the number of candidates hired over the totals; the owner and the subject.

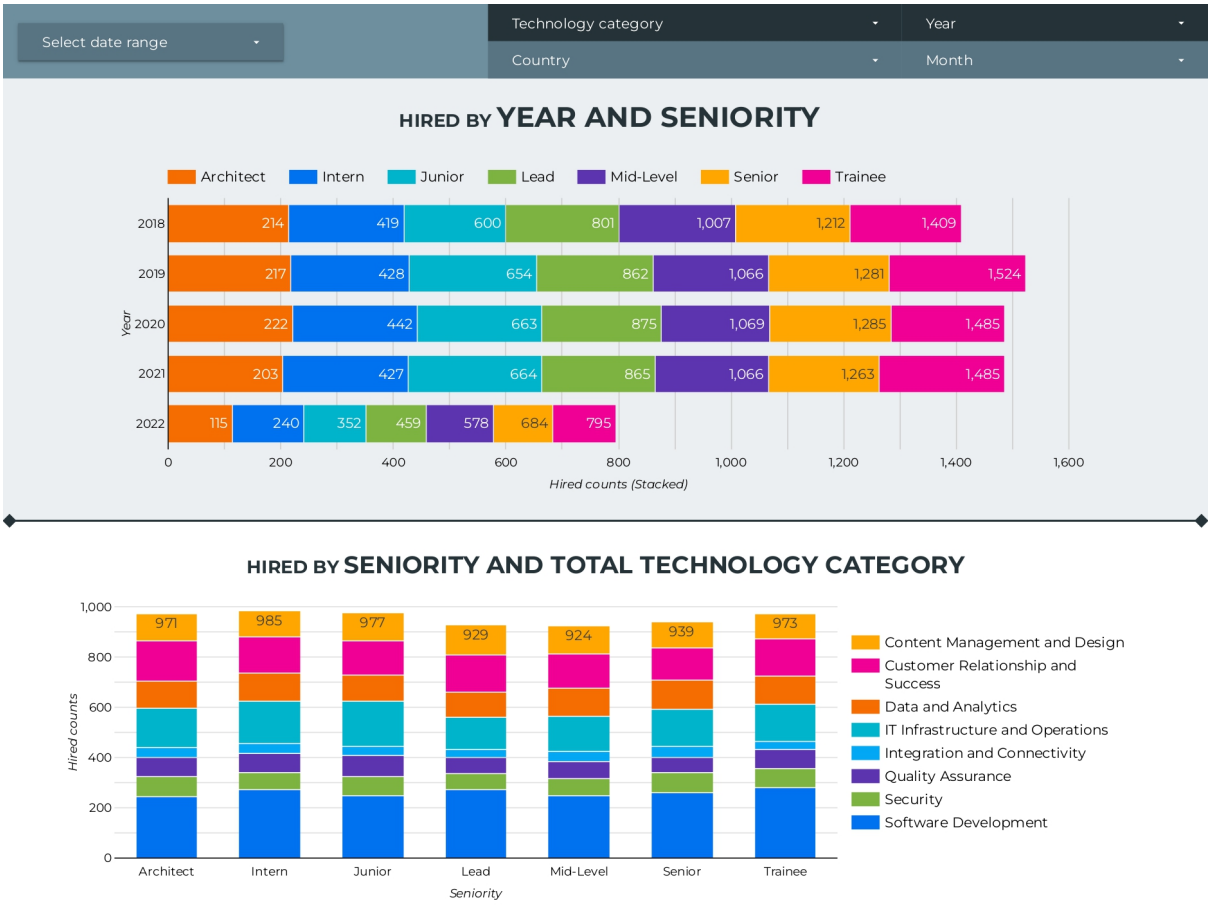
2. Hires by technology

Second slide of Dashboard.

The top chart above shows the number of applications for each technology grouped into the 8 previously mentioned. As observed, Software Development is the greasted.

The left chart shows the number of applications by technology. The 54.9% segment corresponds to **others** due to the limit of segments allowed by Looker Studio in this type of graph.

3. Hires by year & seniority



In this, there are two charts.

The top chart is a horizontal bar chart by number of applicants per year discriminating per year by seniority. It's interesting that over the years, roles are being filled and there is less and less staff turnover.

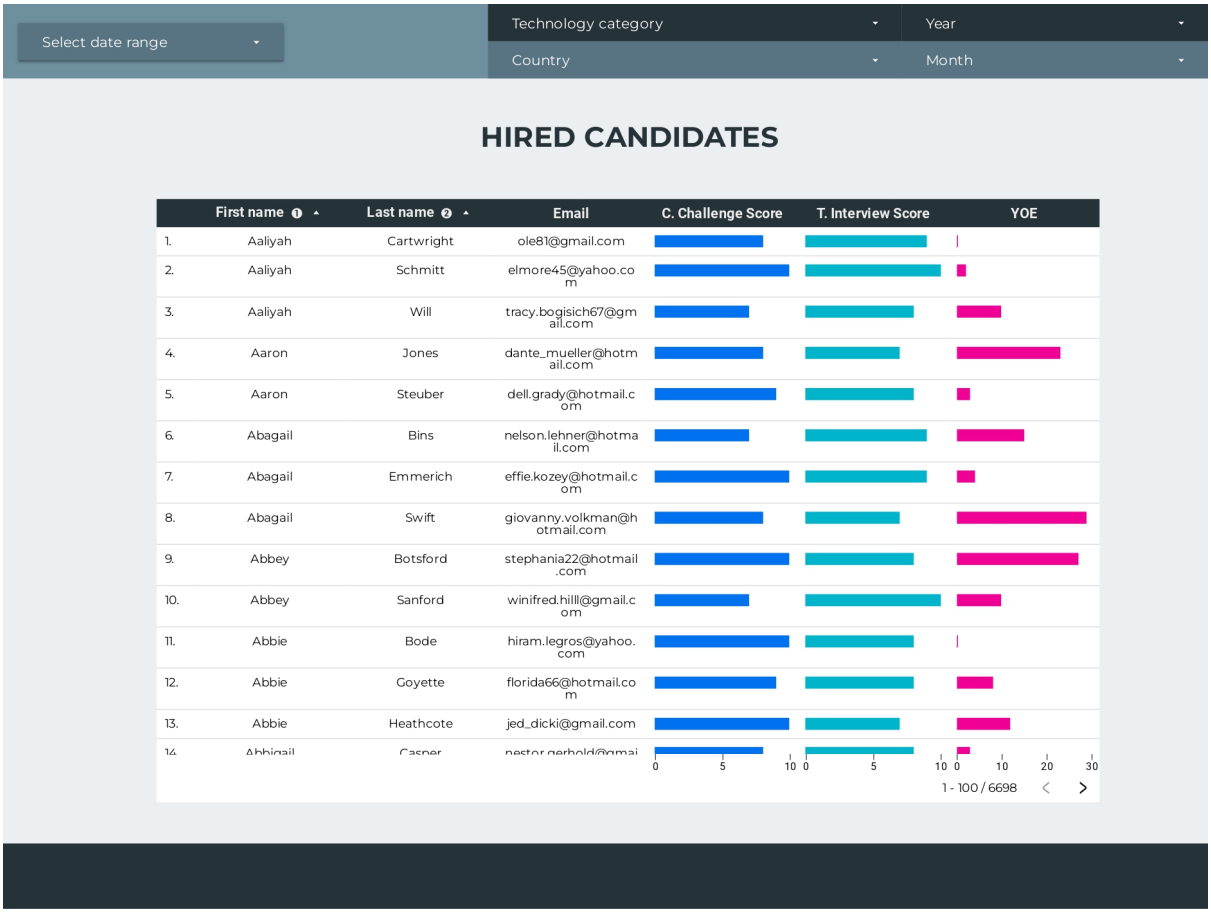
The bottom chart is a bar chart by number of applicants per seniority discriminating per technology category. It appers that the market in which candidates can apply is almost evenly distributed in terms of seniority (knowing its random origin).

4. Hires by country over years

Here, there are two line graphs, the upper one shows the number of records per country for each year filtered to show only Brazil, Colombia, Ecuadro and USA. And the lower one shows the total number of records for all countries by years.

Interestingly, knowing the source of the data, is that 2020 represents a general slowdown or stagnation in hiring.

5. Table of hired candidates



The table provide three bar chart columns, two of scores and one of experience. Curiously, there are people with low experience who have achieved a high score in both score columns.

Conclusions

The process is successfully completed with different tools in which AWS RDS and Ngrok are new to me, and Looker Studio very little previously. Although it's a pity that they were randomly generated data, because the analysis could lend itself to be much deeper, finding correlations and using whisker plots for some variables.

Throughout the process, some insights about this exploration and conclusions:

- Column removed due to its assigning automately by SQLAlchemy:
  - _sa_instance_state
- 10 columns to 16 columns loaded in **applicant** table in the database.
- 24 technologies were grouped into 8.
- 165 candidates have submitted a minimum of two applications.
- 13.4% of candidates are will be hired.
- As we can see, even randomly generated:
  - The software technology has more movement.
  - Despite critical times such as the pandemic that affected the whole world, it did not affect hiring.

- The company is either growing a lot or the employee turnover is gigantic.

References

- David Venté (2024, february 15). ETL Workshop 001 - IA & Data Engineering in Gist. https://gist.github.com/dventep/579f1646c6d6011e4e8314fb85482eba.
- David Venté (2024, february 15). workshop001_etl_education. https://github.com/dventep/workshop001_etl_education.
- David Venté (2024, february 15). Candidates - ETL Workshop 001. https://lookerstudio.google.com/reporting/7c98a50e-d58f-4e4e-a8e8-17a09b233513.
- Codigofacilito (2020, october 5). Mini Curso SQLAlchemy con Python 1 Crear Modelo. https://www.youtube.com/watch?v=XSAjQDM8ZS4
- Codigofacilito (2020, october 15). Mini Curso SQLAlchemy con Python 2 Persistir objetos. https://www.youtube.com/watch?v=110TZCWlOjo.
- Codigofacilito (2020, october 16). Mini Curso SQLAlchemy con Python 3 Consultas. https://youtu.be/T4MgHCiHwEY?si=twZbAkgUODQaWq9s
- Codigofacilito (2020, october 5). Mini Curso SQLAlchemy con Python 1 Crear Modelo.
- AWS. Connection to an Amazon RDS DB instance. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_CommonTasks.Connect.html.
- AWS. Connection to a DB instance running the PostgreSQL database engine. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToPostgreSQLInstance.html.