

# Informe Grupo 25 - Laboratorio 5

César Luis Moreno González - 201924899

Gustavo Adolfo Tabima Luque - 201914586

Juan Carlos Eduardo Nunes Ariza 202010826

## 1. Documentación de los pasos a seguir en el laboratorio:

Con el archivo de preprocesamiento se crean las tablas, se insertan, luego los datos de las filas y lo insertado en la base de datos (con la coherencia que se espera) se usan las siguientes Querys para poblar las tablas vacías que hicimos, con eso se hace la tabla de hechos y con las foreign Keys se unen los datos y con eso ya se tendría el funcionamiento de laboratorio 5

## Creación de la base de datos y del rol

The screenshot shows the PGAdmin interface with a PostgreSQL database connection. The left sidebar displays the database structure, including the 'public' schema and the 'stockitem' table. The main window shows a query executed in the 'public.stockitem' database:

```
1 SELECT * FROM public.stockitem
2 ORDER BY stock_item_key ASC
```

The query result is displayed in the 'Data Output' tab, showing 5 rows of data. The columns are: stock\_item\_key, stock\_item, color, selling\_package, buying\_package, brand, size\_val, lead\_time\_days, quantity\_per\_outw, is\_chiller\_stock, and tax\_rate.

| stock_item_key | stock_item                 | color        | selling_package | buying_package | brand        | size_val | lead_time_days | quantity_per_outw | is_chiller_stock | tax_rate |
|----------------|----------------------------|--------------|-----------------|----------------|--------------|----------|----------------|-------------------|------------------|----------|
| 0              | Unknown                    | Not Provided | Not Provided    | Not Provided   | Not Provided | [null]   | 0              | 0                 | false            | 0.0      |
| 1              | Void fill 400 L bag (WH... | Not Provided | Each            | Each           | Not Provided | [null]   | 14             | 10                | false            | 14.0     |
| 2              | Void fill 300 L bag (WH... | Not Provided | Each            | Each           | Not Provided | [null]   | 14             | 10                | false            | 14.0     |
| 3              | Void fill 200 L bag (WH... | Not Provided | Each            | Each           | Not Provided | [null]   | 14             | 10                | false            | 14.0     |
| 4              | Void fill 100 L bag (WH... | Not Provided | Each            | Each           | Not Provided | [null]   | 14             | 10                | false            | 14.0     |





```
EXPLORE
AIRFLOW DOCKER
  _pycache_
  utils
    _pycache_
    crear_tablas.py
    ETL.py
    file_util.py
    insert_queries.py
  data
    dimension_city_no_procesados.csv
    dimension_city.csv
    dimension_customer_no_procesados.csv
    dimension_customer.csv
    dimension_date_no_procesados.csv
    dimension_date.csv
    dimension_employee_no_procesados.csv
    dimension_employee.csv
    dimension_stock_item_no_procesados.csv
    dimension_stock_item.csv
    fact_order_no_procesados.csv
    fact_order.csv
  logs
  models
  plugins
  .env
  docker-compose.yml

dags > utils > ETL.py
1 # Utilidades de airflow
2 from airflow.models import DAG
3 from airflow.providers.postgres.operators.postgres import PostgresOperator
4 from airflow.utils.task_group import TaskGroup
5 from airflow.operators.python_operator import PythonOperator
6
7 # Utilidades de python
8 from datetime import datetime
9
10 # Funciones ETL
11 from utils.crear_tablas import crear_tablas, eliminar_tablas
12 from utils.file_util import procesar_datos
13 from utils.insert_queries import *
14
15 default_args = {
16     'owner': 'Estudiante',
17     'email_on_failure': False,
18     'email': ['estudiante@unandes.edu.co'],
19     'start_date': datetime(2022, 5, 5) # inicio de ejecución
20 }
21
22 with DAG(
23     "ETL",
24     description="ETL",
25     schedule_interval="@daily", # ejecución diaria del DAG
26     default_args=default_args,
27     catchup=False) as dag:
28
29     # task: 1 - preprocesamiento
30     preprocesamiento = PythonOperator(
31         task_id="preprocesamiento",
32         python_callable=procesar_datos
33     )
34
35     # task: 2 crear las tablas en la base de datos postgres
36     eliminar_tablas_db = PostgresOperator(
37         task_id="eliminar_tablas_en_postgres",
38         postgres_conn_id="postgres_localhost", # Nótese que es el mismo ID definido en la conexión Postgres de la interfaz de Airflow
39         sql=eliminar_tablas()
40     )
41
42     crear_tablas_db = PostgresOperator(
43         task_id="crear_tablas_en_postgres",
44         postgres_conn_id="postgres_localhost",
45         sql=crear_tablas()
46     )
47
48     preprocesamiento >> eliminar_tablas_db >> crear_tablas_db
```

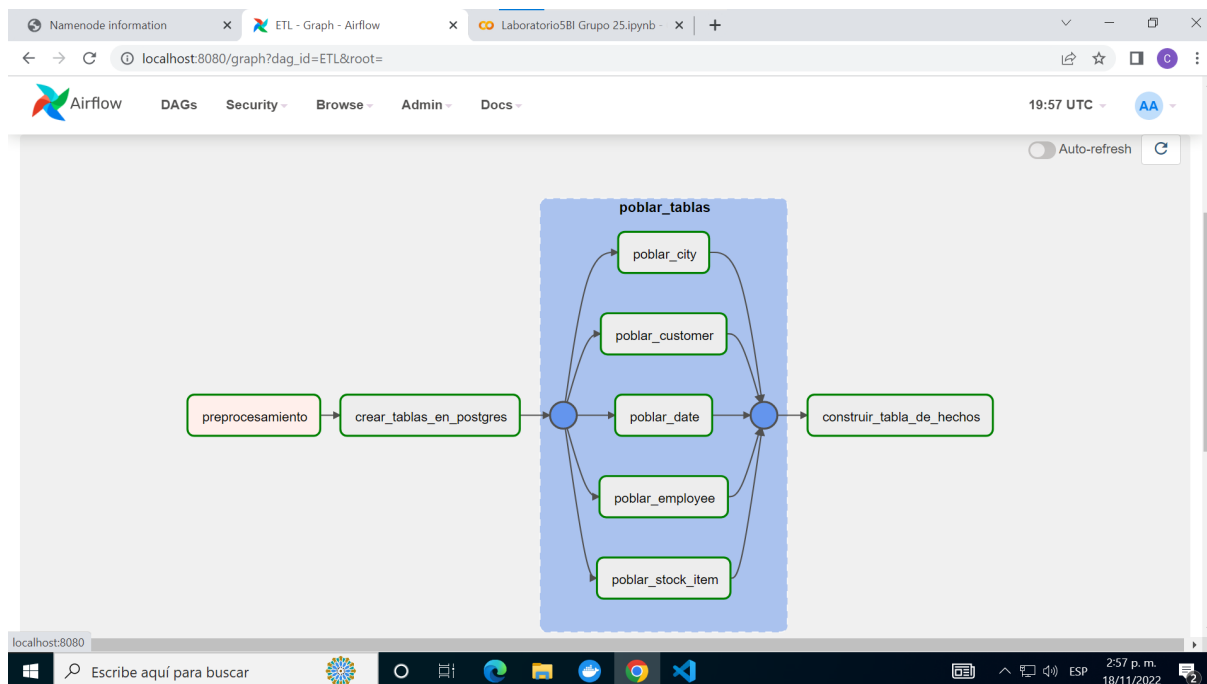
```
EXPLORE
AIRFLOW DOCKER
  _pycache_
  utils
    _pycache_
    crear_tablas.py
    ETL.py
    file_util.py
    insert_queries.py
  data
    dimension_city_no_procesados.csv
    dimension_city.csv
    dimension_customer_no_procesados.csv
    dimension_customer.csv
    dimension_date_no_procesados.csv
    dimension_date.csv
    dimension_employee_no_procesados.csv
    dimension_employee.csv
    dimension_stock_item_no_procesados.csv
    dimension_stock_item.csv
    fact_order_no_procesados.csv
    fact_order.csv
  logs
  models
  plugins
  .env
  docker-compose.yml

dags > utils > file_util.py
1 import pandas as pd
2
3 def cargar_datos(nombre):
4     df = pd.read_csv("../data/" + nombre + ".csv", sep=',', encoding='latin1', index_col=False)
5     return df
6
7 def guardar_datos(df, nombre):
8     df.to_csv("../data/" + nombre + ".csv", encoding='latin1', sep=',', index=False)
9
10 def leer_datos():
11
12     ## Dimension city
13     city = pd.read_csv("http://bigdata-cluster4-01.virtual.uniandes.edu.co:58070/webhdfs/v1/user/monitorbi/datalake01/dimension_city.csv?op=OPEN&user.name=cursob125", sep=',', encoding='latin1')
14     guardar_datos(city, "dimension_city_no_procesados")
15
16     ## Dimension Customer
17     customer = pd.read_csv("http://bigdata-cluster4-01.virtual.uniandes.edu.co:58070/webhdfs/v1/user/monitorbi/datalake01/dimension_customer.csv?op=OPEN&user.name=cursob125", sep=',', encoding='latin1')
18     guardar_datos(customer, "dimension_customer_no_procesados")
19
20     ## Dimension Date
21     date = pd.read_csv("http://bigdata-cluster4-01.virtual.uniandes.edu.co:58070/webhdfs/v1/user/monitorbi/datalake01/dimension_date.csv?op=OPEN&user.name=cursob125", sep=',', encoding='latin1')
22     guardar_datos(date, "dimension_date_no_procesados")
23
24     ## Dimension Employee
25     employee = pd.read_csv("http://bigdata-cluster4-01.virtual.uniandes.edu.co:58070/webhdfs/v1/user/monitorbi/datalake01/dimension_employee.csv?op=OPEN&user.name=cursob125", sep=',', encoding='latin1')
26     guardar_datos(employee, "dimension_employee_no_procesados")
27
28     ## Dimension Stock Item
29     stock_item = pd.read_csv("http://bigdata-cluster4-01.virtual.uniandes.edu.co:58070/webhdfs/v1/user/monitorbi/datalake01/dimension_stock_item.csv?op=OPEN&user.name=cursob125", sep=',', encoding='latin1')
30     guardar_datos(stock_item, "dimension_stock_item_no_procesados")
31
32     ## Fact Table
33     fact_order = pd.read_csv("http://bigdata-cluster4-01.virtual.uniandes.edu.co:58070/webhdfs/v1/user/monitorbi/datalake01/fact_order.csv?op=OPEN&user.name=cursob125", sep=',', encoding='latin1')
34     guardar_datos(fact_order, "fact_order_no_procesados")
35
36 def procesar_datos():
37
38     # Llamar a la función auxiliar de leer datos
39     leer_datos()
40
41     ## Dimension city:
42     # Datos vacíos procesados.
43     # Datos sin errores
```

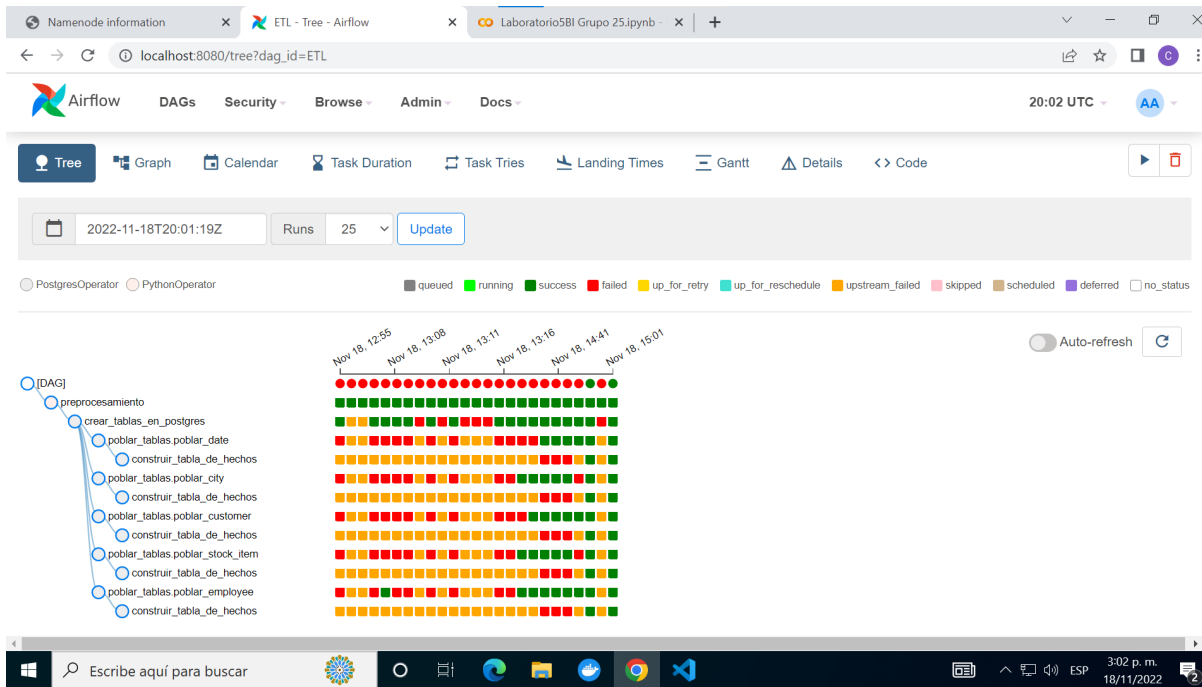
```
1 dag = DAG('utils',  
2  
3  
4  
5 ince, Country, Continent, Sales_Territory, Region, Subregion, Latest_Recorded_Population) VALUES "  
6  
7  
8  
9 City)\',\'(row.State_Province)\',\'(row.Country)\',\'(row.Continent)\',\'(row.Sales_Territory)\',\'(row.Region)\',\'(row.Subregion)\',\'(row.Latest_Recorded_Population));\n"  
10  
11  
12  
13 r, Bill_To_Customer, Category, Buying_Group, Primary_Contact, Postal_Code) VALUES "  
14  
15  
16  
17 row.Customer)\',\'(row.Bill_To_Customer)\',\'(row.Category)\',\'(row.Buying_Group)\',\'(row.Primary_Contact)\',\'(row.Postal_Code));\n"  
18  
19  
20  
21 r, Day_val, Month_val, Short_Month, Calendar_Month_Number, Calendar_Year, Fiscal_Month_Number, Fiscal_Year) VALUES "  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

## 2. Capturas de pantalla de los resultados:

### 2.1. Grafo de ejecución:



## 2.2.Árbol:



## 2.3.Base de datos Posgres:

### 3. Sentencias SQL de las consultas:

#### Tabla city

The screenshot shows the pgAdmin interface. The left pane shows the database structure with a tree view. The right pane shows a SQL query: `SELECT * FROM public.city ORDER BY city_key ASC`. The bottom pane shows the query results in a table format.

| city_key [PK] integer | city character varying (150) | state_province character varying (150) | country character varying (150) | continent character varying (150) | sales_territory character varying (150) | region character varying (150) | subregion character varying (150) | latest_recorded_population integer |
|-----------------------|------------------------------|--|---------------------------------|-----------------------------------|---|--------------------------------|-----------------------------------|------------------------------------|
| 1                     | Carrollton                   | New York                               | United States                   | North America                     | Mideast                                 | Americas                       | Northern America                  | 0                                  |
| 2                     | Carrollton                   | Virginia                               | United States                   | North America                     | Southeast                               | Americas                       | Northern America                  | 4574                               |
| 3                     | Carrollton                   | Illinois                               | United States                   | North America                     | Great Lakes                             | Americas                       | Northern America                  | 2484                               |
| 4                     | Carrollton                   | Missouri                               | United States                   | North America                     | Plains                                  | Americas                       | Northern America                  | 3784                               |
| 5                     | Carrollton                   | Ohio                                   | United States                   | North America                     | Great Lakes                             | Americas                       | Northern America                  | 3241                               |
| 6                     | Carrollton                   | Kentucky                               | United States                   | North America                     | Southeast                               | Americas                       | Northern America                  | 3938                               |
| 7                     | Carrollton                   | Georgia                                | United States                   | North America                     | Southeast                               | Americas                       | Northern America                  | 24388                              |
| 8                     | Carrollton                   | Alabama                                | United States                   | North America                     | Southeast                               | Americas                       | Northern America                  | 1019                               |
| 9                     | Carrollton                   | Mississippi                            | United States                   | North America                     | Southeast                               | Americas                       | Northern America                  | 190                                |
| 10                    | Carrollton                   | Texas                                  | United States                   | North America                     | Southwest                               | Americas                       | Northern America                  | 119097                             |
| 11                    | Carrollton Manor             | Maryland                               | United States                   | North America                     | Mideast                                 | Americas                       | Northern America                  | 0                                  |

## Tabla data\_table

Browser

public.date\_table/WWIDWGrupo25/postgres@PostgreSQL 15

public.city/WW...

public.date\_table/WWIDWGrupo25/postgres@PostgresSQL 15

public.date\_table/WWIDWGrupo25/postgres@PostgresSQL 15

Dashboard Properties SQL Statistics Dependencies Processes

public.city/WW...

public.date\_table/WWIDWGrupo25/postgres@PostgresSQL 15

Query Query History

1 SELECT \* FROM public.date\_table

2 ORDER BY date\_key ASC

Data Output Messages Notifications

|    | date_key [PK] date | day_number integer | day_val integer | month_val character varying (20) | shortMonth character varying (10) | calendar_month_number integer | calendar_year integer | fiscal_month_number integer | fiscal_year integer |      |
|----|--------------------|--------------------|-----------------|----------------------------------|-----------------------------------|-------------------------------|-----------------------|-----------------------------|---------------------|------|
| 1  | 2013-01-01         | 1                  | 1               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 2  | 2013-01-02         | 2                  | 2               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 3  | 2013-01-03         | 3                  | 3               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 4  | 2013-01-04         | 4                  | 4               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 5  | 2013-01-05         | 5                  | 5               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 6  | 2013-01-06         | 6                  | 6               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 7  | 2013-01-07         | 7                  | 7               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 8  | 2013-01-08         | 8                  | 8               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 9  | 2013-01-09         | 9                  | 9               | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 10 | 2013-01-10         | 10                 | 10              | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 11 | 2013-01-11         | 11                 | 11              | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |
| 12 | 2013-01-12         | 12                 | 12              | January                          | Jan                               |                               | 1                     | 2013                        | 3                   | 2013 |

## Tabla customer

Admin

public.customer/WWIDWGrupo25/postgres@PostgreSQL 15

Query

```
1 SELECT * FROM public.customer
2 ORDER BY customer_key ASC
```

Data Output

|    | customer_key [PK] integer | customer character varying (150) | bill_to_customer character varying (150) | category character varying (150) | buying_group character varying (150) | primary_contact character varying (150) | postal_code integer |
|----|---------------------------|----------------------------------|--|----------------------------------|--------------------------------------|---|---------------------|
| 1  | 1                         | Talispin Toys (Head Of...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Waldemar Fissar                         | 90410               |
| 2  | 2                         | Talispin Toys (Dynamit...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Lorena Cindric                          | 90216               |
| 3  | 3                         | Talispin Toys (Peoples...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Bhaargar Rambhatia                      | 90205               |
| 4  | 4                         | Talispin Toys (Medicin...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Daniel Roman                            | 90152               |
| 5  | 5                         | Talispin Toys (Gaspert...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Johanna Hulting                         | 90298               |
| 6  | 6                         | Talispin Toys (Jessie...         | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Blissajet Thakur                        | 90261               |
| 7  | 7                         | Talispin Toys (Frankw...         | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Kalidas Nader                           | 90761               |
| 8  | 8                         | Talispin Toys (Bow Mar...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Kanti Kotadia                           | 90484               |
| 9  | 9                         | Talispin Toys (Netcong...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Somtu Aalto                             | 90129               |
| 10 | 10                        | Talispin Toys (Wimbled...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Siddhartha Parkar                       | 90061               |
| 11 | 11                        | Talispin Toys (Default...        | Talispin Toys (Head Of...                | Novelty Shop                     | Talispin Toys                        | Elnaz Javan                             | 90185               |

Tabla Employee

Admin interface showing the 'employee' table structure and query results.

Query:

```
SELECT * FROM public.employee
ORDER BY employee_key ASC
```

Data Output:

| employee_key [PK] integer | employee character varying (150) | preferred_name character varying (150) | is_salesperson boolean |
|---------------------------|----------------------------------|--|------------------------|
| 1                         | 1 Lily Code                      | Lily                                   | true                   |
| 2                         | 2 Isabella Rupp                  | Isabella                               | false                  |
| 3                         | 3 Ethan Onstow                   | Ethan                                  | false                  |
| 4                         | 4 Amy Trett                      | Amy                                    | true                   |
| 5                         | 5 Jai Shand                      | Jai                                    | false                  |
| 6                         | 6 Anthony Grosse                 | Anthony                                | true                   |
| 7                         | 7 Taj Shand                      | Taj                                    | true                   |
| 8                         | 8 Hudson Hollinworth             | Hudson                                 | true                   |
| 9                         | 9 Jack Potter                    | Jack                                   | true                   |
| 10                        | 10 Piper Koch                    | Piper                                  | false                  |

Tabla fact\_order

Admin interface showing the 'fact\_order' table structure and query results.

Query:

```
SELECT * FROM public.fact_order
ORDER BY order_key ASC
```

Data Output:

| order_key [PK] integer | city_key integer | customer_key integer | stock_item_key integer | order_date_key date | picked_date_key date | salesperson_key integer | picker_key integer | package character varying (50) | quantity integer | unit_price numeric | tax_rate numeric | total_excluding_tax numeric | tax_amount numeric |
|------------------------|------------------|----------------------|------------------------|---------------------|----------------------|-------------------------|--------------------|--------------------------------|------------------|--------------------|------------------|-----------------------------|--------------------|
| 1                      | 1                | 91                   | 179                    | 533 2014-02-05      | 2013-03-17           | 135                     | 122                | S                              | 805              | 237.77             | 61               | 2245.39                     | 305.4              |
| 2                      | 2                | 83                   | 2                      | 631 2015-11-17      | 2013-07-19           | 2                       | 133                | S                              | 76               | 721.71             | 59               | 6585.7                      | 135.12             |
| 3                      | 3                | 47                   | 390                    | 174 2015-04-29      | 2015-11-03           | 128                     | 75                 | S                              | 585              | 2866.71            | 9                | 8113.37                     | 792.11             |
| 4                      | 4                | 8                    | 218                    | 157 2015-04-04      | 2014-12-03           | 84                      | 191                | S                              | 878              | 4671.07            | 39               | 9109.56                     | 668.65             |
| 5                      | 5                | 94                   | 167                    | 235 2015-01-26      | 2015-01-11           | 91                      | 197                | S                              | 583              | 1950.68            | 24               | 9277.41                     | 66.55              |
| 6                      | 6                | 3                    | 376                    | 124 2014-02-10      | 2013-06-05           | 135                     | 181                | S                              | 727              | 775.09             | 13               | 3839.04                     | 552.8              |
| 7                      | 7                | 10                   | 328                    | 44 2014-02-07       | 2016-07-29           | 105                     | 11                 | S                              | 679              | 4081.01            | 37               | 8618.58                     | 770.8              |
| 8                      | 8                | 54                   | 325                    | 629 2013-10-23      | 2014-07-01           | 56                      | 56                 | M                              | 80               | 1852.75            | 6                | 2961.56                     | 871.17             |
| 9                      | 9                | 64                   | 263                    | 357 2014-01-29      | 2015-01-18           | 184                     | 25                 | S                              | 777              | 295.73             | 41               | 9975.19                     | 410.59             |
| 10                     | 10               | 87                   | 236                    | 607 2015-08-10      | 2014-06-25           | 71                      | 200                | XL                             | 881              | 2036.81            | 22               | 8045.06                     | 628.18             |
| 11                     | 11               | 63                   | 191                    | 132 2016-08-12      | 2015-04-22           | 40                      | 203                | S                              | 454              | 2308.12            | 3                | 3785.4                      | 204.46             |

tabla Stock\_terms



The screenshot shows the PgAdmin interface with a PostgreSQL database schema on the left and a query result in the center. The schema includes tables like 'city', 'customer', 'date\_table', 'employee', 'fact\_order', 'stockitem', and 'trigger\_functions'. The query result is a table with 12 columns and 12 rows of data.

|    | stock_item_key<br>[PK] integer | stock_item<br>character varying (200) | color<br>character varying (50) | selling_package<br>character varying (50) | buying_package<br>character varying (50) | brand<br>character varying (50) | size_val<br>character varying (50) | lead_time_days<br>integer | quantity_per_order<br>integer | is_chiller_stock<br>boolean | tax_rate<br>numeric |
|----|--------------------------------|---------------------------------------|---------------------------------|---|--|---------------------------------|------------------------------------|---------------------------|-------------------------------|-----------------------------|---------------------|
| 1  | 0                              | Unknown                               | Not Provided                    | Not Provided                              | Not Provided                             | Not Provided                    | [null]                             | 0                         | 0                             | false                       | 0.0                 |
| 2  | 1                              | Void fill 400 L bag (WHI...           | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 10                            | false                       | 14.0                |
| 3  | 2                              | Void fill 300 L bag (WHI...           | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 10                            | false                       | 14.0                |
| 4  | 3                              | Void fill 200 L bag (WHI...           | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 10                            | false                       | 14.0                |
| 5  | 4                              | Void fill 100 L bag (WHI...           | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 10                            | false                       | 14.0                |
| 6  | 5                              | Air cushion machine (B...             | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 20                        | 1                             | false                       | 20.0                |
| 7  | 6                              | Air cushion film 200m...              | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 1                             | false                       | 14.0                |
| 8  | 7                              | Air cushion film 200m...              | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 1                             | false                       | 14.0                |
| 9  | 8                              | Large replacement bla...              | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 10                            | false                       | 14.0                |
| 10 | 9                              | Small 5mm replaceme...                | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 10                            | false                       | 14.0                |
| 11 | 10                             | Packing knife with met...             | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 5                             | false                       | 14.0                |
| 12 | 11                             | Packing knife with met...             | Not Provided                    | Each                                      | Each                                     | Not Provided                    | [null]                             | 14                        | 5                             | false                       | 14.0                |

#### 4. Preguntas del laboratorio:

##### 4.1.Explique a fondo los siguientes conceptos de airflow:

- Task: Es la unidad básica de ejecución en Airflow. Las tareas se organizan en DAG y luego se establecen dependencias ascendentes y descendentes entre ellas para expresar el orden en el que deben ejecutarse.  
<https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html>
- Operator: Los operadores son los componentes básicos de los DAG Airflow. Contienen la lógica de cómo se procesan los datos en una canalización. Cada tarea en un DAG se define instanciando un operador. Hay muchos tipos diferentes de operadores disponibles en Airflow. Algunos operadores, como las funciones de Python, ejecutan código general proporcionado por el usuario, mientras que otros operadores realizan acciones muy específicas, como transferir datos de un sistema a otro. <https://docs.astronomer.io/learn/what-is-an-operator>
- DAG: Un DAG, o un gráfico acíclico dirigido, es una colección de todas las tareas que desea ejecutar, organizadas de una manera que refleja sus relaciones y dependencias. Un DAG se define en un script de Python, que representa la estructura de los DAG (tareas y sus dependencias) como código. Por ejemplo, un DAG simple podría constar de tres tareas: A, B y C. Podría decir que A tiene que ejecutarse correctamente antes de que B pueda ejecutarse, pero C puede

ejecutarse en cualquier momento. Podría decir que la tarea A se agota después de 5 minutos y B se puede reiniciar hasta 5 veces en caso de que falle. También podría decir que el flujo de trabajo se ejecutará todas las noches a las 10:00 p. m., pero no debería comenzar hasta una fecha determinada.

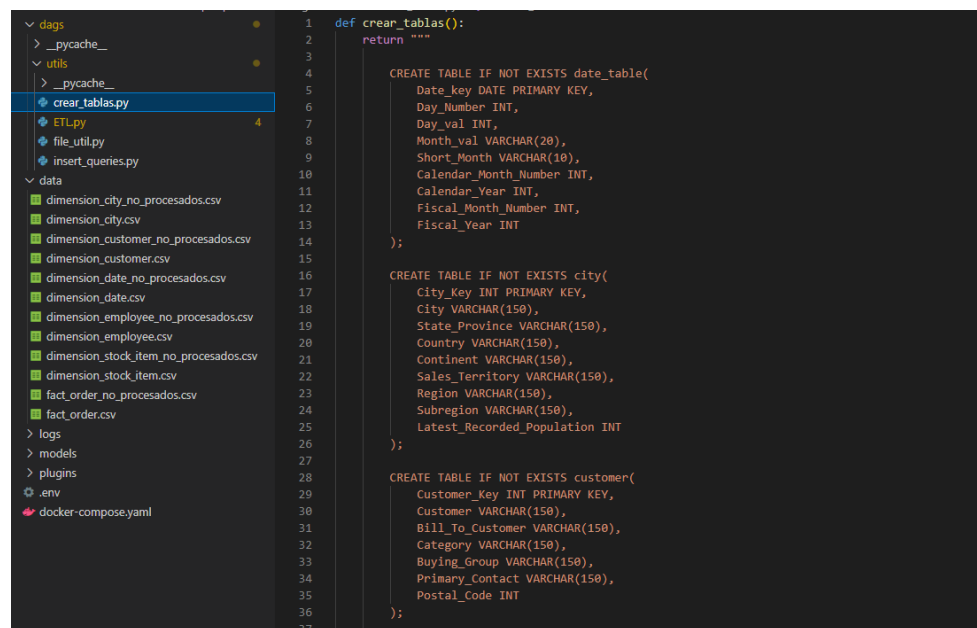
[https://airflow.apache.org/docs/apache-airflow/1.10.12/concepts.html#:~:text=In%20Airflow%2C%20a%20DAG%20%E2%80%93%20or,an%20their%20dependencies\)%20as%20code](https://airflow.apache.org/docs/apache-airflow/1.10.12/concepts.html#:~:text=In%20Airflow%2C%20a%20DAG%20%E2%80%93%20or,an%20their%20dependencies)%20as%20code).

4.2. ¿Por qué para la columna de día se utiliza el nombre “day\_val” y no “day”?

Se utiliza porque para hacer una búsqueda sobre la columna no se hace con DATE o un día sino con valor que permite encontrar el día.

¿De dónde se obtiene la información sobre las columnas que hay que crear en la tabla?

Se encuentra en el archivo crear\_tabla.py



```
1 def crear_tablas():
2     return ""
3
4     CREATE TABLE IF NOT EXISTS date_table(
5         Date_Key DATE PRIMARY KEY,
6         Day_Number INT,
7         Day_Val INT,
8         Month_Val VARCHAR(20),
9         Short_Month VARCHAR(10),
10        Calendar_Month_Number INT,
11        Calendar_Year INT,
12        Fiscal_Month_Number INT,
13        Fiscal_Year INT
14    );
15
16    CREATE TABLE IF NOT EXISTS city(
17        City_Key INT PRIMARY KEY,
18        City VARCHAR(150),
19        State_Province VARCHAR(150),
20        Country VARCHAR(150),
21        Continent VARCHAR(150),
22        Sales_Territory VARCHAR(150),
23        Region VARCHAR(150),
24        Subregion VARCHAR(150),
25        Latest_Recorded_Population INT
26    );
27
28    CREATE TABLE IF NOT EXISTS customer(
29        Customer_Key INT PRIMARY KEY,
30        Customer VARCHAR(150),
31        Bill_To_Customer VARCHAR(150),
32        Category VARCHAR(150),
33        Buying_Group VARCHAR(150),
34        Primary_Contact VARCHAR(150),
35        Postal_Code INT
36    );
37
```

4.3. ¿Por qué es necesario un flujo de ejecución de las tareas en Airflow?

Es necesario para mostrar cómo se debe ejecutar cada una de las tareas y el orden de las mismas porque es el encargado de convertir los ficheros de texto plano, extraer su información y almacenarla en la base de datos.

4.4. ¿Qué ajustes habría que hacer a este proceso de ETL si se trata de un ETL incremental, donde previamente hay datos cargados en la bodega de datos?

La diferencia entre los datos de origen y de destino se carga a través del proceso ETL en el almacén de datos. Hay 2 tipos de cargas incrementales,

según el volumen de datos que esté cargando; transmisión de carga incremental y carga incremental por lotes.

La carga incremental es más rápida que una carga completa. El principal inconveniente de este tipo de carga es la mantenibilidad. A diferencia de una carga completa, con una carga incremental no se puede volver a ejecutar la carga completa si hay un error. Además de esto, los archivos deben cargarse en orden, por lo que los errores agravarán el problema a medida que otros datos se ponen en cola. Entonces los cambios es que la carga de datos se haría mediante el procesamiento continuo, lo que significa que a medida que las aplicaciones cliente escriben datos en la fuente de datos, los datos se tratan, se transforman y se guardan en la bodega de datos de destino. Al mismo tiempo la falta de posibilidad para poder subir de nuevo todo, debido a un error.

*4.5. ¿Al revisar lo entregado por el grupo previo de consultores, se evidencia que no se está trabajando de forma apropiada con las llaves primarias, ya que se tienen las del sistema transaccional como PK, es así cómo se decide mantener esas llaves y renombrarlas con el sufijo \_T y crear las propias de la bodega de datos con el sufijo \_DWH. Estas últimas son consecutivas. ¿Qué se debe hacer para que este cambio sea efectivo? Muestre un ejemplo para una dimensión y su efecto en la tabla de hechos.*

*Se cambia las columnas que tenemos como pk y le ponemos el sufijo T, creo otra columna que tenga el mismo nombre la columna principal, con el sufijo DWH y luego se dice que esta columna sea el PK y le quito esa característica a la que tiene sufijo T. Luego lo que cambia en las demás tablas que hacen referencia,*

*Así que ahora las FK de la fact table no van a ser los IDS de la forma transaccional sino que van a ser los campos que tienen como sufijo DWH.*

*Por ejemplo, si tengo una fact table que tiene como PK y FK una referencia a la tabla date table, y la PK de esta tabla es un ID, lo que cambiaría al aplicar este cambio es que ahora la referencia a date table no va a ser un id sino la fecha en sí que ahora en la PK de la tabla.*

*4.6. ¿A nivel de la dimensión Fecha, se detecta que la llave primaria no es un entero que represente la fecha (AAAAMMDD), si no un atributo de tipo DATE, cómo corregirá este error?*

*Se utilizarían los campos con números, existen unos campos que tienen cada una de las partes de la fecha, a esos campos es a los que se hará referencia en la fact table para solucionar el error. Otra forma de solucionar el error podría ser parsear la*

*fecha a un entero y crear otra columna con el resultado para que se haga referencia a esta columna como PK de la tabla. , esos números son a los que se hace referencia en la fact table*