

Système décisionnel de Gestion des offres d'Hotels

-Rapport-

Encadré par : **Pr. Imade BENELALLAM**

Réalisé par :

- ENNAJI AYOUB
- ELHOURI MOHAMED EL MEHDI

Filière : DSE

Semestre : S4

TABLE DES MATIERES

I.	SCHEMAS DE L'ENTREPOT DE DONNEES	7
1-	ANALYSE DES OFFRES.....	8
2-	LA TABLE FAIT	8
3-	MODEL LOGIQUE DE DONNEES :	9
II.	CREATION ET ALIMENTATIONS DES TABLES.....	9
	CREATION ET ALIMENTATIONS DES TABLES SURS MYSQL WORKBENCH	9
III.	APIFY	10
IV.	WEB SCRAPING.....	10
V.	VISUALISATION DE DAGS.....	12
1-	LOAD_FILE :	13
2-	GET_DISTINCT_VALUES :	14
3-	GROUP BY :	15
4-	JOIN :	16
5-	DELETE_TMP_FILES	17
VI.	VISUALISATION DES DONNEES SUR POWERBI	18

LISTE DES FIGURES

FIGURE 15: PYRAMIDE DE LA MODELISATION DIMENSIONNELLE -----	7
FIGURE 16: TABLE FAIT -----	8
FIGURE 17 : MLD -----	9
FIGURE 1: LOGO D'APIFY-----	10
FIGURE 2: EXTRACTION DES DONNEES-----	10
FIGURE 3: APERÇU DES DONNEES JSON -----	11
FIGURE 4: VISUALISATION DES DAGS 1 -----	12
FIGURE 5: VISUALISATION DES DAGS 2 -----	12
FIGURE 6: FONCTION « LOAD_FILE » -----	13
FIGURE 7: FONCTION « GET_DISTINCT_VALUES » -----	14
FIGURE 8FONCTION « GROUP BY » -----	15
FIGURE 9: FONCTION « JOIN » -----	16
FIGURE 10: FONCTION « DELETE_TMP_FILES » -----	17
FIGURE 11: NOMBRE TOTAL DES OFFRES PAR TYPE DE CHAMBRE -----	18
FIGURE 12: NOMBRE DES OFFRES PAR REGION -----	19
FIGURE 13: REGION AVEC EVALUATION MOYENNE, MAXIMALE ET MINIMALE -----	19
FIGURE 14: NOMBRE DE PLACE PAR TYPE D'ETABLISSEMENT-----	20

INTRODUCTION

Au Maroc Le tourisme a été parmi les secteurs les plus touchés par la crise sanitaire et par les mesures de restrictions des déplacements qui ont été adoptées pour limiter la propagation du virus.

En effet, les arrivées aux postes frontières ont connu une baisse drastique et les nuitées enregistrées aux établissements d'hébergement ont suivi la même tendance avec une régression de -72%.

Après la réouverture des frontières les responsables du secteur touristique, notamment le ministère du tourisme, pourrait demander l'acquisition un système décisionnel permettant de gérer les offres des hôtels.

D'où l'idée de réaliser un Système décisionnel de gestions des offres des hôtels. Afin de suivre les offres proposées par les établissements d'hébergement touristique.

Ce projet s'inscrit dans le cadre de l'élément de module Data Warehouse & Business Analytics. Il est réalisé en utilisant Airflow, MySQL Workbench, Apify et Power BI.

I. Schémas de l'entrepôt de données

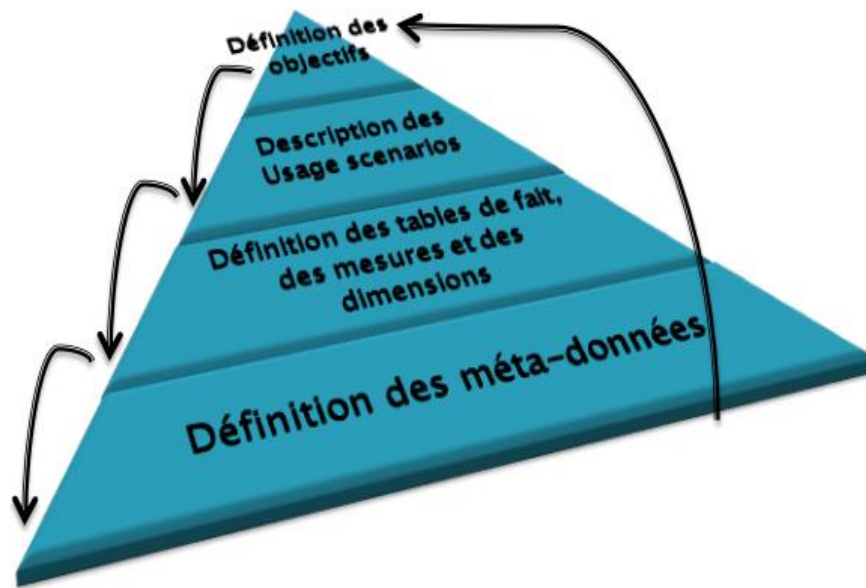


Figure 1: pyramide de la modélisation dimensionnelle

❖ **Nous avons suivi les étapes suivantes vues lors du cours :**

- i. Définir l'objectif métier afin d'identifier les sujets et d'adapter l'entrepôt aux besoins.
- ii. Déterminer les scénarios d'utilisation qui vont nous permettre de déterminer comment les utilisateurs finaux vont interagir avec le Data Warehouse.
- iii. Définir le schéma conceptuel à partir du sujet et des questions métiers.

1- Analyse des offres

	ANALYSE DES RESERVATIONS
Objectif métier	Suivre les offres proposées par les établissements d'hébergement touristique pour améliorer l'infrastructure touristique.
Sujet	Les offres
Scenarios d'utilisation	<ul style="list-style-type: none"> - Evaluer le nombres des offres par région, type de chambre, capacité et types d'hôtels. - Evaluer (rating) les établissements par région, type de chambre, capacité types d'hôtels.

2- La table fait

```
##### TABLE FAIT #####
query = '''SELECT
a.id AS address_id,
p.id AS place_type_id,
r.id AS room_type_id,
n.id AS number_of_persons_id,
t.number_of_offers, t.average_rating, t.max_rating, t.min_rating
FROM tmp_df t
INNER JOIN d_addresses_df a on t.postal_code = a.postal_code
INNER JOIN d_place_types_df p on t.place_type = p.place_type
INNER JOIN d_room_types_df r on t.room_type = r.room_type
INNER JOIN d_number_of_persons_df n on t.number_of_persons = n.number
'''
fact_table_df = sqldf(query,locals())
display(fact_table_df)
```

	address_id	place_type_id	room_type_id	number_of_persons_id	number_of_offers	average_rating	max_rating	min_rating
0	27	6	1213	1	82	9.060976	9.9	8.3
1	27	6	1192	1	71	9.070423	9.8	8.2
2	27	6	509	1	41	9.151220	9.9	8.3
3	27	6	1290	1	38	9.252632	10.0	8.3
4	27	6	788	1	27	9.188889	9.9	8.0
...
3808	27	10	306	3	1	9.200000	9.2	9.2

Figure 2: table fait

3- Model logique de données :

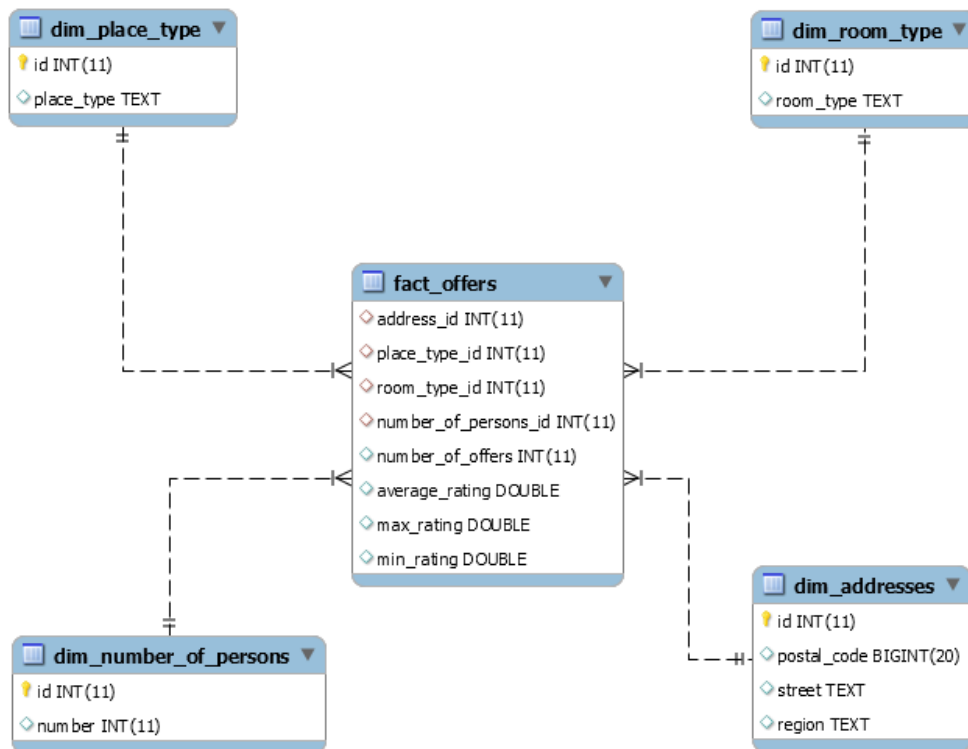


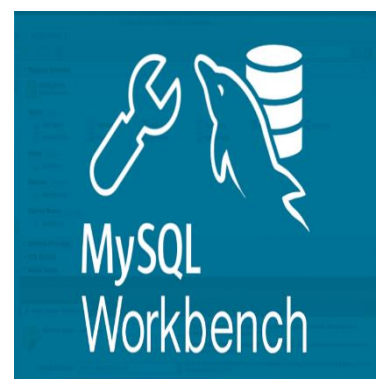
Figure 3 : MLD

II. Création et alimentations des tables

Création et alimentations des tables surs MySQL Workbench

MySQL Workbench est un logiciel de gestion de base de données MySQL, il permet de gérer des tables (ajout, modification, suppression) à travers une interface graphique simple d'usage.

Les tables ont été créés à partir des fichier JSON que nous avons générer à partir d'Airflow. Ensuite, ces fichiers ont été importés sur MySQL Workbench. Puis nous avons établi les relations entre les différentes tables.



III. Apify



Figure 4: Logo d'Apify

C'est un robot web hébergé qui permet à toute personne possédant des compétences élémentaires en programmation d'extraire des données structurées de n'importe quel site web. Contrairement au scraping web par pointer-cliquer, Apify fonctionne sur des sites web modernes de plus en plus complexes et dynamiques.

Il peut être utilisé par une large gamme d'utilisateurs, Quiconque a besoin de données provenant du web, qu'il s'agisse d'un étudiant, un journaliste, une start-up ou une grande entreprise.

IV. Web scraping

Il s'agit d'explorer des sites Web, d'en extraire des données structurées et de les exporter dans des formats tels qu'Excel, CSV ou JSON.

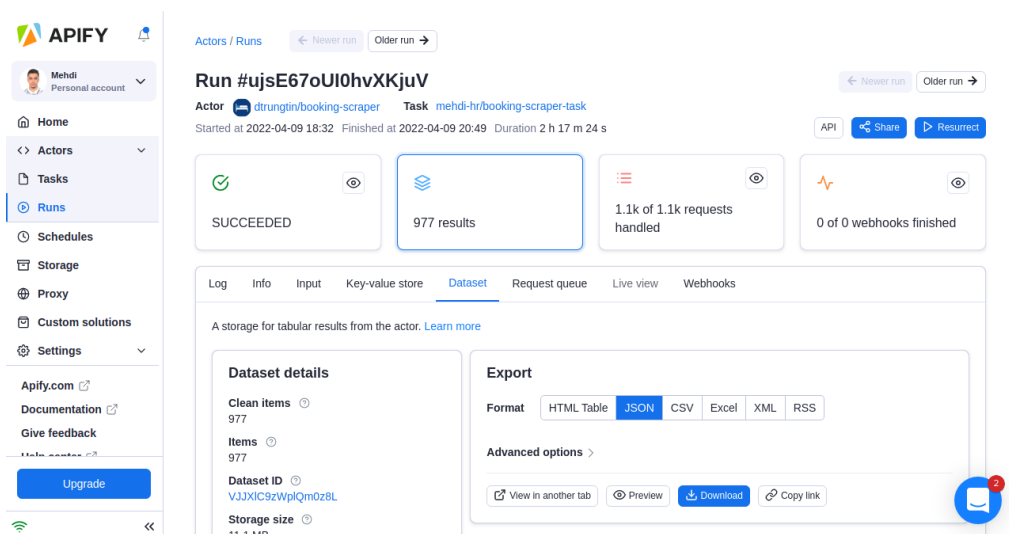
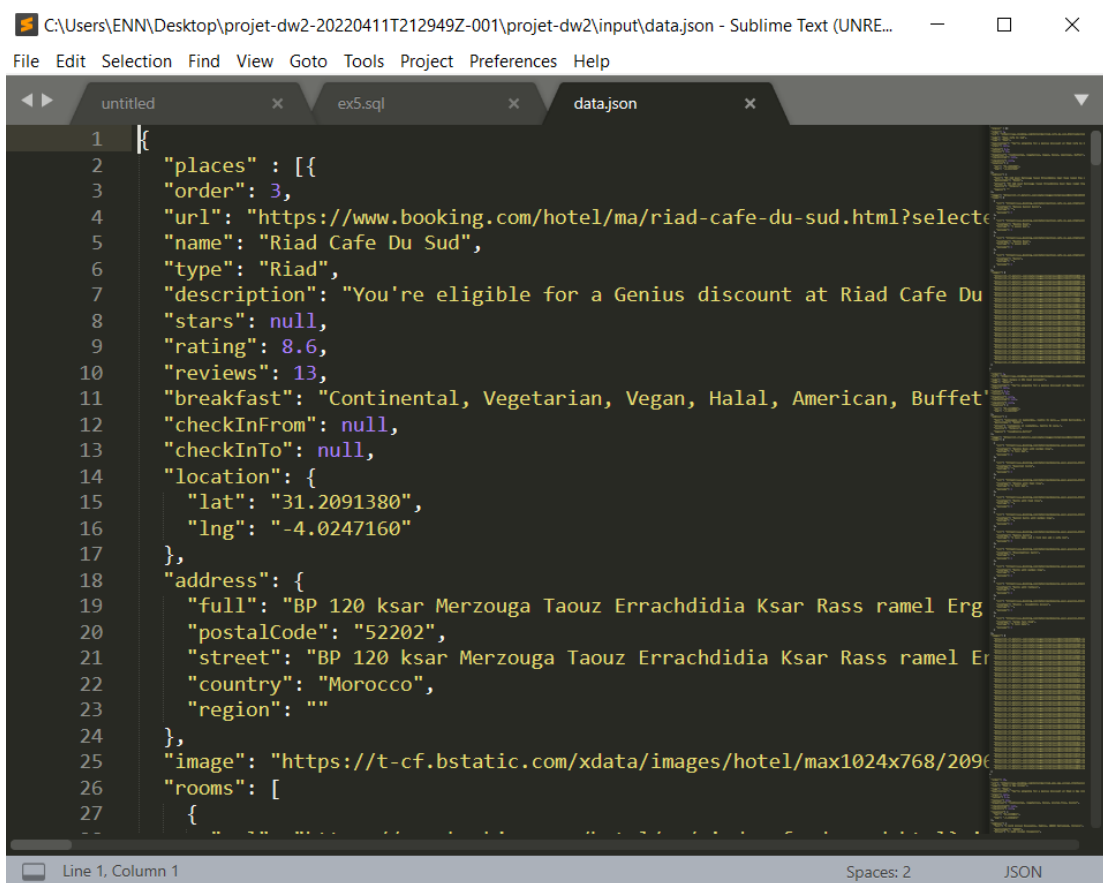


Figure 5: extraction des données

- ❖ D'abord nous avons choisis l'acteur dont on a besoin dans le volet Actors sur le menu gauche. Nous avons choisi celui responsables de la réservation des hôtels après avoir précisé la région (MAROC), afin de récupérer les données des établissements d'hébergement marocains qui font l'objet de notre système.
- ❖ Ensuite nous avons téléchargé les données JSON.



```
1 {
2   "places" : [{
3     "order": 3,
4     "url": "https://www.booking.com/hotel/ma/riad-cafe-du-sud.html?selecte
5     "name": "Riad Cafe Du Sud",
6     "type": "Riad",
7     "description": "You're eligible for a Genius discount at Riad Cafe Du
8     "stars": null,
9     "rating": 8.6,
10    "reviews": 13,
11    "breakfast": "Continental, Vegetarian, Vegan, Halal, American, Buffet'
12    "checkInFrom": null,
13    "checkInTo": null,
14    "location": {
15      "lat": "31.2091380",
16      "lng": "-4.0247160"
17    },
18    "address": {
19      "full": "BP 120 ksar Merzouga Taouz Errachdidia Ksar Rass ramel Erg
20      "postalCode": "52202",
21      "street": "BP 120 ksar Merzouga Taouz Errachdidia Ksar Rass ramel Er
22      "country": "Morocco",
23      "region": ""
24    },
25    "image": "https://t-cf.bstatic.com/xdata/images/hotel/max1024x768/2096
26    "rooms": [
27      {
```

Figure 6: aperçu des données JSON

Le fichier contient l'ensemble des informations relatives à chaque hôtel : adresse, type d'hôtel, type de chambre, capacité...etc.

V. Visualisation de dags

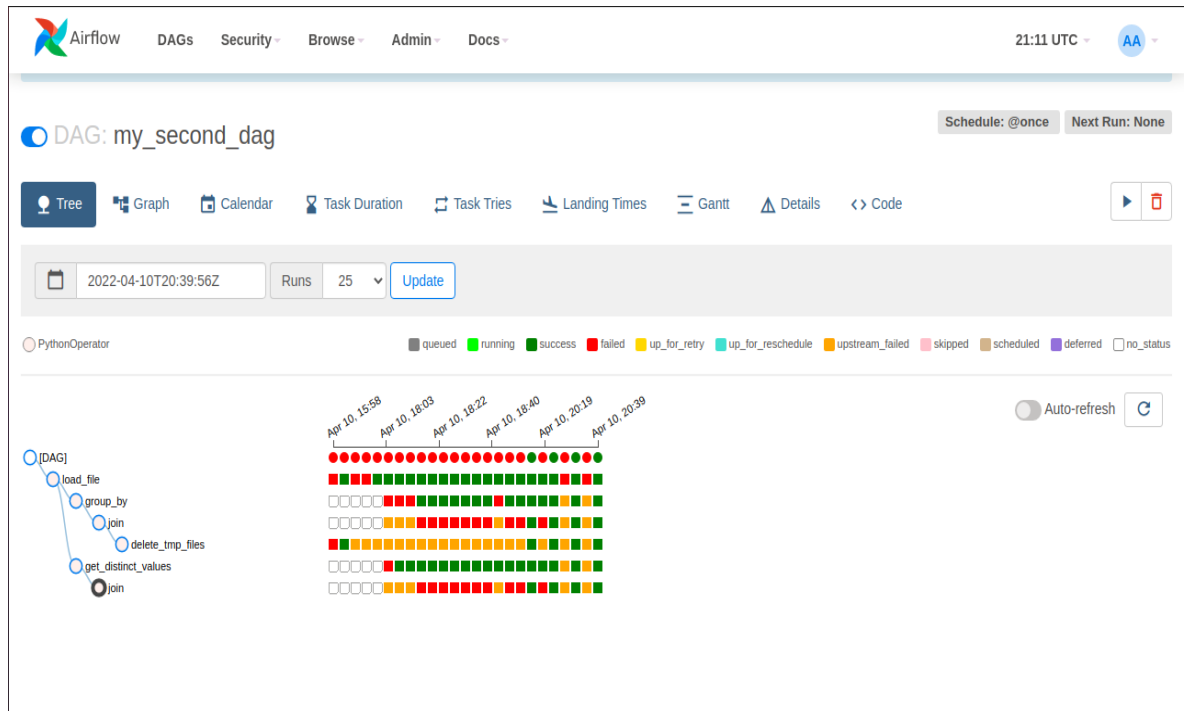


Figure 7: visualisation des Dags 1

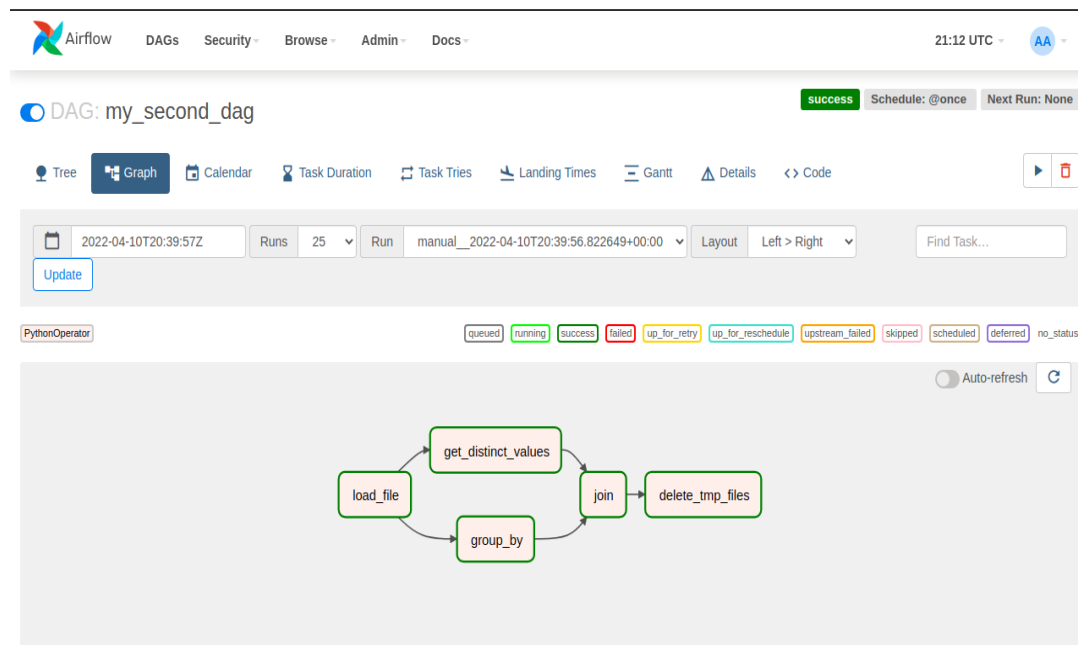


Figure 8: visualisation des Dags 2

On remarque la présence de 5 étapes qui sont :

- 1- **Load_file** : permet d'importer les données depuis le dossier input.

```
def _load_file():  
    with open(INPUT) as f:  
        data = json.load(f)  
  
        places = data['places']  
  
        #cleaning, deleting useless data  
        id = 1  
        for place in places:  
            del place['url']  
            del place['name']  
            del place['description']  
            del place['stars']  
            del place['reviews']  
            del place['breakfast']  
            del place['checkInFrom']  
            del place['checkInTo']  
            del place['location']  
            del place['image']  
            del place['images']  
            del place['order']  
            place['id'] = id  
            id = id + 1  
  
        with open(TMP + '/1.json', 'w+') as e:  
            json.dump(places,e)
```

Figure 9: fonction « load_file »

2- Get_distinct_values :

Elle permet d'extraire les valeurs distinctes afin d'alimenter les tables dimensions par la suite.

```
1
2 def _get_distinct_values():
3
4     with open(TMP + '/1.json') as f:
5         places = json.load(f)
6         distinct_place_types = list(set([i['type'] for i in places]))
7         distinct_postal_codes = list(set([i['address']['postalCode'] for i in places]))
8
9         distinct_addresses = []
10        for i in distinct_postal_codes:
11            address = dict()
12            for j in places:
13                if j['address']['postalCode'] == i:
14                    address['postal_code'] = j['address']['postalCode']
15                    address['street'] = j['address']['street']
16                    address['region'] = j['address']['region']
17                    address['place_id'] = j['id']
18                    distinct_addresses.append(address)
19                    break
20
21
22        all_rooms = []
23        for i in places:
24            for j in i['rooms']:
25                j['place_id'] = i['id']
26                del j['url']
27                del j['bedType']
28                all_rooms.append(j)
29
30        distinct_room_types = list(set([i['roomType'] for i in all_rooms]))
31        distinct_number_of_persons = list(set([i['persons'] for i in all_rooms]))
32
33        with open(TMP + '/distinct_place_types.json', 'w+') as e:
34            json.dump(distinct_place_types,e)
35
36        with open(TMP + '/distinct_addresses.json', 'w+') as e:
37            json.dump(distinct_addresses,e)
38
39        with open(TMP + '/distinct_postal_codes.json', 'w+') as e:
40            json.dump(distinct_postal_codes,e)
41
42        with open(TMP + '/distinct_room_types.json', 'w+') as e:
43            json.dump(distinct_room_types,e)
44
45        with open(TMP + '/distinct_number_of_persons.json', 'w+') as e:
46            json.dump(distinct_number_of_persons,e)
47
```

Figure 10: fonction « get_distinct_values »

3- Group by :

❖ S'exécute simultanément avec la fonction précédente.

Permet d'extraire des données regroupées par : type d'établissement, code postal, type de chambre, capacité de la chambre.

```
1 def _group_by():
2     with open(TMP + '/1.json') as f:
3         places = json.load(f)
4         before_groupby = []
5         for place in places:
6             for room in place['rooms']:
7                 line = dict()
8                 line['place_type'] = place['type']
9                 line['postal_code'] = place['address']['postalCode']
10                line['room_type'] = room['roomType']
11                line['number_of_persons'] = room['persons']
12                line['rating'] = place['rating']
13                before_groupby.append(line)
14
15        df = pd.DataFrame(before_groupby)
16        query = '''SELECT number_of_persons, postal_code, room_type, place_type,
17        COUNT(*) AS number_of_offers,
18        AVG(rating) AS average_rating,
19        MAX(rating) AS max_rating,
20        MIN(rating) AS min_rating
21        from df
22        GROUP BY place_type, postal_code, room_type, number_of_persons
23        ORDER BY number_of_offers DESC
24        '''
25        tmp_df = sqldf(query,locals())
26
27        tmp_dict = json.loads(tmp_df.to_json(orient="records"))
28        with open(TMP + '/tmp_view.json', 'w+') as e:
29            json.dump(tmp_dict,e)
```

Figure 11 fonction « group by »

4- Join :

Permet d'effectuer des jointures afin de créer la table fait à partir des tables dimensions :

```
1 def _join():
2
3     distinct_place_types = []
4     distinct_room_types = []
5     distinct_number_of_persons = []
6
7     with open(TMP + '/distinct_place_types.json') as f:
8         distinct_place_types = json.load(f)
9
10    with open(TMP + '/distinct_addresses.json') as f:
11        distinct_addresses = json.load(f)
12
13    with open(TMP + '/distinct_room_types.json') as f:
14        distinct_room_types = json.load(f)
15
16    with open(TMP + '/distinct_number_of_persons.json') as f:
17        distinct_number_of_persons = json.load(f)
18
19    tmp = dict()
20    with open(TMP + '/tmp_view.json') as f:
21        tmp = json.load(f)
22
23
24    ### tmp df
25
26    tmp_df = pd.DataFrame.from_dict(tmp)
27
28    ##### TABLES DIMENSIONS #####
29
30    d_addresses_df = pd.DataFrame(distinct_addresses)
31    d_addresses_df.insert(loc=0, column='id', value=np.arange(len(d_addresses_df)))
32    d_addresses_df.drop('place_id', inplace=True, axis=1)
33
34    d_place_types_df = pd.DataFrame(distinct_place_types)
35    d_place_types_df.insert(loc=0, column='id', value=np.arange(len(d_place_types_df)))
36    d_place_types_df.columns = ['id', 'place_type']
37
38    d_room_types_df = pd.DataFrame(distinct_room_types)
39    d_room_types_df.insert(loc=0, column='id', value=np.arange(len(d_room_types_df)))
40    d_room_types_df.columns = ['id', 'room_type']
41
42    d_number_of_persons_df = pd.DataFrame(distinct_number_of_persons)
43    d_number_of_persons_df.insert(loc=0, column='id',
44                                  value=np.arange(len(d_number_of_persons_df)))
45    d_number_of_persons_df.columns = ['id', 'number']
46
47    #display(d_addresses_df)
48
49    ##### TABLE FAIT #####
50    query = '''SELECT
51    a.id AS address_id,
52    p.id AS place_type_id,
53    r.id AS room_type_id,
54    n.id AS number_of_persons_id,
55    t.number_of_offers, t.average_rating, t.max_rating, t.min_rating
56    FROM tmp_df t
57    INNER JOIN d_addresses_df a on t.postal_code = a.postal_code
58    INNER JOIN d_place_types_df p on t.place_type = p.place_type
59    INNER JOIN d_room_types_df r on t.room_type = r.room_type
60    INNER JOIN d_number_of_persons_df n on t.number_of_persons = n.number
61    '''
62    fact_table_df = sqldf(query, locals())
63    fact_table_df.to_csv(OUTPUT + '/offers.csv')
64    tmp_dict = json.loads(fact_table_df.to_json(orient="records"))
65    with open(OUTPUT + '/offers.json', 'w+') as e:
66        json.dump(tmp_dict, e)
```

Figure 12: fonction « Join »

5- delete_tmp_files

Permet de supprimer les fichiers temporaires.

```
1
2 def _delete_tmp_files():
3     shutil.rmtree(TMP)
4
5
6 with DAG("my_second_dag", start_date=datetime(2022, 4, 3),
7         schedule_interval="@once", catchup=False ) as dag:
8
9     load_file = PythonOperator(
10         task_id = "load_file",
11         python_callable = _load_file
12     )
13
14
15     get_distinct_values = PythonOperator(
16         task_id = "get_distinct_values",
17         python_callable = _get_distinct_values
18     )
19
20     group_by = PythonOperator(
21         task_id = "group_by",
22         python_callable = _group_by
23     )
24
25     join = PythonOperator(
26         task_id = "join",
27         python_callable = _join
28     )
29
30
31     delete_tmp_files = PythonOperator(
32         task_id = "delete_tmp_files",
33         python_callable = _delete_tmp_files
34     )
35
36     load_file >> get_distinct_values >> join >> delete_tmp_files
37
38     load_file >> group_by >> join >> delete_tmp_files
39
```

Figure 13: fonction « delete_tmp_files »

VI. Visualisation des données sur POWERBI

Microsoft Power BI est une solution d'analyse de données de Microsoft. Il permet de créer des visualisations de données personnalisées et interactives avec une interface suffisamment simple pour que les utilisateurs finaux créent leurs propres rapports et tableaux de bord



Il suffit de se connecter à notre base de données mysql pour générer les tableaux de bords suivants :

room_type	Count of number_of_offers
	1
1000 Parfumes Double Room	1
4-Bed Female Dormitory Room	1
4-Bed Mixed Dormitory Room	1
6-Bed Mixed Dormitory Room	1
Aawa Double Room	10
Abla Suite	1
Abricot Suite	2
Adam Twin Room	1
Adarissa Suite	1
ADELAIDE ROOM	1
Africa Double Room	6
Africa Standard Twin Room	1
African Suite	1
Agerzam - Standard Twin Room with Private Bathroom	1
Ahlam Suite	1
Aicha Suite	1
Aicha Triple Room	1
Aida Suite	1
Air Standard Double Room	1
Total	3812

Figure 14: nombre total des offres par type de chambre

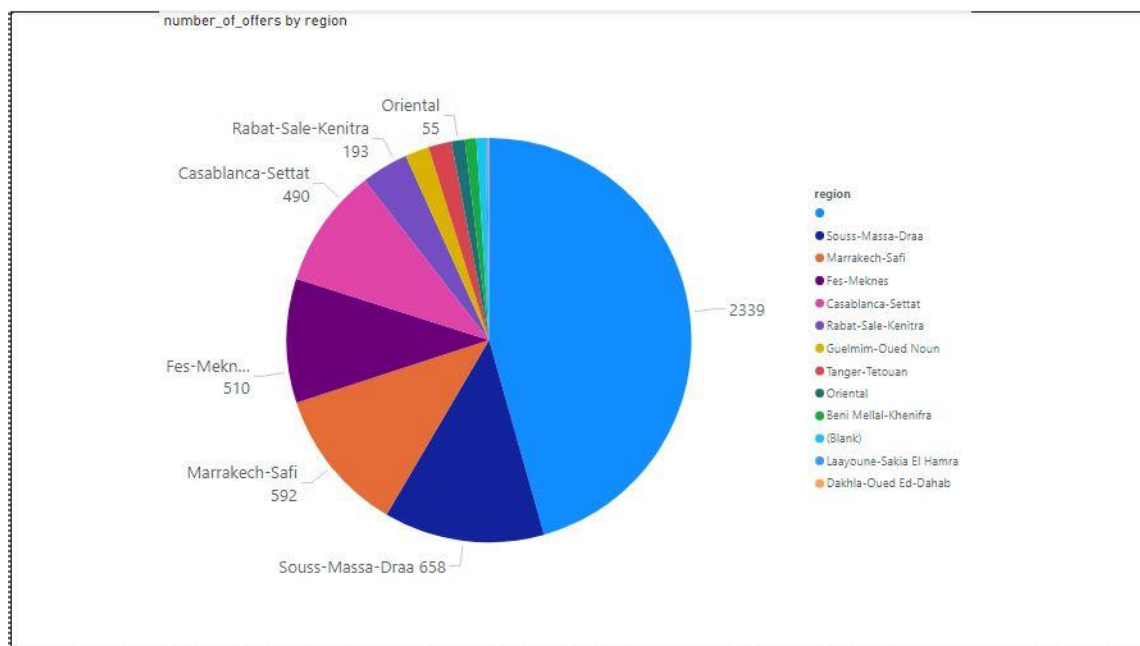


Figure 15: nombre des offres par région

region	Average of average_rating	Max of max_rating	Min of min_rating
Tanger-Tetouan	8,77	10,00	8,00
Souss-Massa-Draa	8,95	10,00	8,00
Rabat-Sale-Kenitra	9,08	9,70	8,10
Oriental	8,97	9,70	8,00
Marrakech-Safi	8,87	10,00	8,00
Laayoune-Sakia El Hamra	8,40	8,40	8,40
Guelmim-Oued Noun	9,12	9,70	8,20
Fes-Meknes	8,77	9,80	8,00
Dakhla-Oued Ed-Dahab	8,80	8,80	8,80
Casablanca-Settat	8,80	9,90	8,00
Beni Mellal-Khenifra	8,67	9,40	7,90
	9,04	10,00	8,00
	9,09	10,00	8,20
Total	8,94	10,00	7,90

Figure 16: région avec évaluation moyenne, maximale et minimale

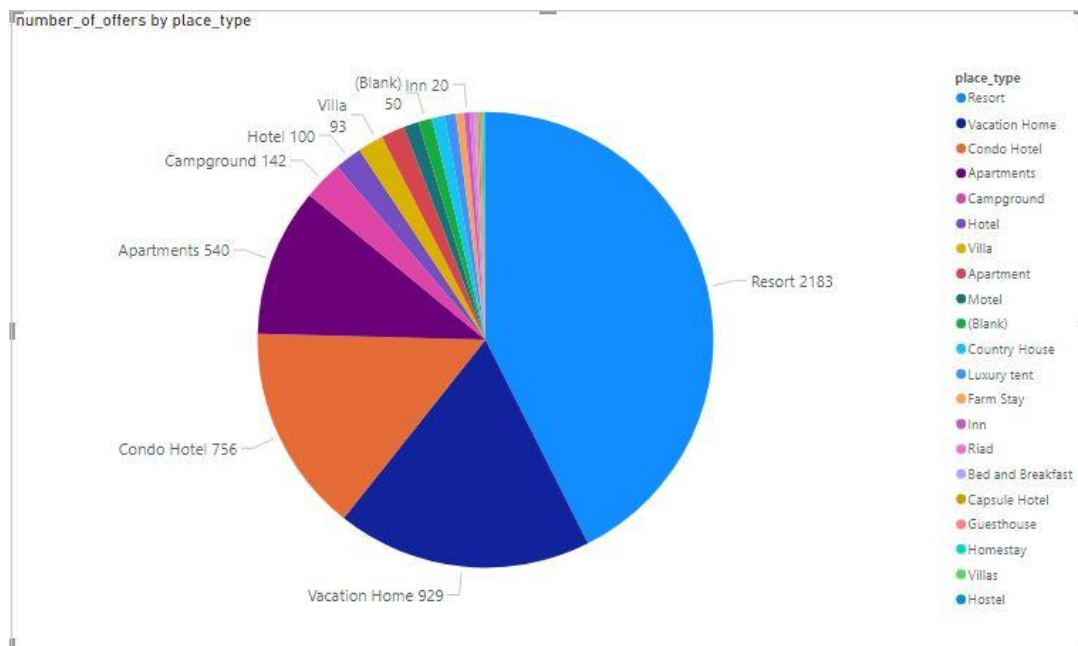


Figure 17: nombre de place par type d'établissement

CONCLUSION

La réalisation de ce travail a été très bénéfique et très enrichissante. Elle nous a permis d'utiliser des technologies nouvelles tel que Apify, Airflow, MySQL Workbench...

Cette solution s'avère être très utile pour la gestion des offres des hôtels. Afin de permettre aux responsables du secteur touristique de suivre l'ensemble des offres proposées par les établissements d'hébergement touristique.

Ce système pourrait être amélioré au fur et à mesure de l'avancement du cours afin de produire une solution plus efficace.