Projet Final - Administration de Bases de Données Relationnelles

NOM : ADRAOUI
PRÉNOM : ALI
Classe : Cybersécurité
Année : B3
Date de rendu : 26/11/2025

1. Introduction

1.1. Présentation du projet Ynov-Air

Le projet Ynov-Air consiste en l'administration et l'amélioration d'un système de réservation de vols aériens. L'objectif est de mettre en pratique les compétences acquises en administration de bases de données relationnelles.

1.2. Fonctionnalité choisie et justification

Fonctionnalité choisie : Nous avons ajouté une fonctionnalité qui permet l'annulation des réservations demandé tout en justifiant la raison de l'annulation.

Justification : Dans certains cas, des clients sont obligés d'annuler leur réservation à cause d'un problème quelconque. On peut aussi appliquer le CRUD depuis l'ORM de l'application.

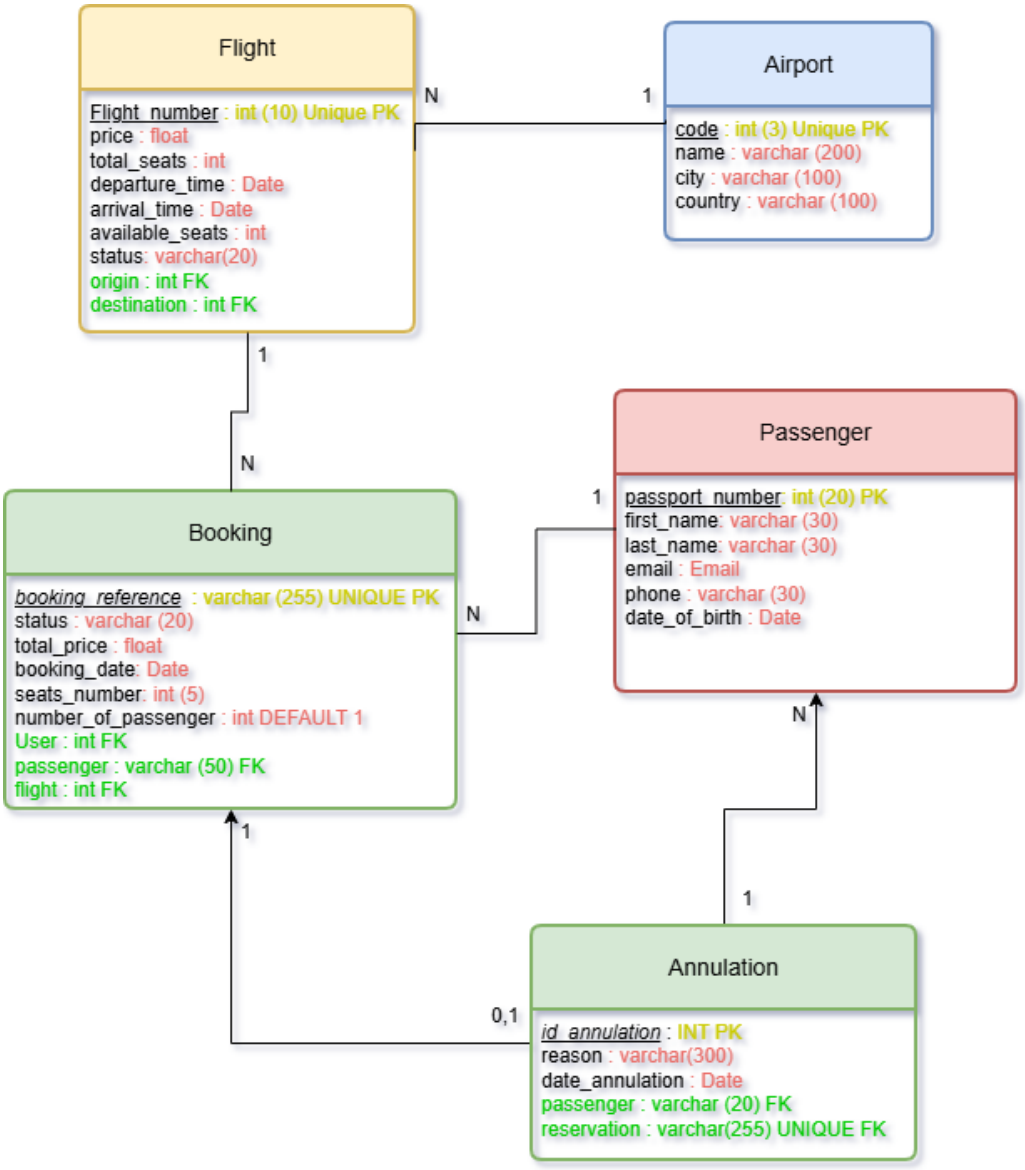
1.3. Objectifs personnels

Mon objectif principal dans ce projet est de développer une application web avec Django capable de gérer des données, afin de renforcer mes compétences en conception de modèles, en manipulation de bases de données et en mise en place d'un CRUD.

2. Modélisation de la Base de Données

2.1. Schéma relationnel mis à jour (MCD/MLD)

Voici le schéma relationnel qui définit chacune des entités ainsi que leurs relations



2.2. Explication des choix de conception

J'ai ajouté la table Annulation afin de permettre la gestion des réservations annulées par les clients

2.3. Dictionnaire de données

Table	Colonne	Type de Donnée	Contraintes
Annulation	Id_annulation	INT	PRIMARY KEY
	Date_annulation	DATE	-
	Reason	VARCHAR (20)	NOT NULL
	Passenger	VARCHAR (20)	FOREIGH KEY DEFAULT 1
	Booking	VARCHAR (255)	FOREIGH KEY

3. Scripts SQL de Création

3.1. Création des tables et des relation

```python
class Annulation (models.Model):
    date_annulation = models.DateField(
        auto_now_add=True,
        null=True,
        verbose_name="Date d'annulation",
    )
    reason = models.CharField(
        max_length=400,
        blank=False,
        null=False,
        verbose_name="Raison"
        )
    booking = models.ForeignKey(
        Booking,
        default= None,
        on_delete=models.SET_NULL,
        null=True,
        verbose_name="Reservation",
        )

    passenger = models.ForeignKey(
        Passenger,
        on_delete=models.CASCADE,
        default=1,
        verbose_name="Passager")

    class Meta:
        verbose_name = "Annulation"
        verbose_name_plural = "Annulations"
```

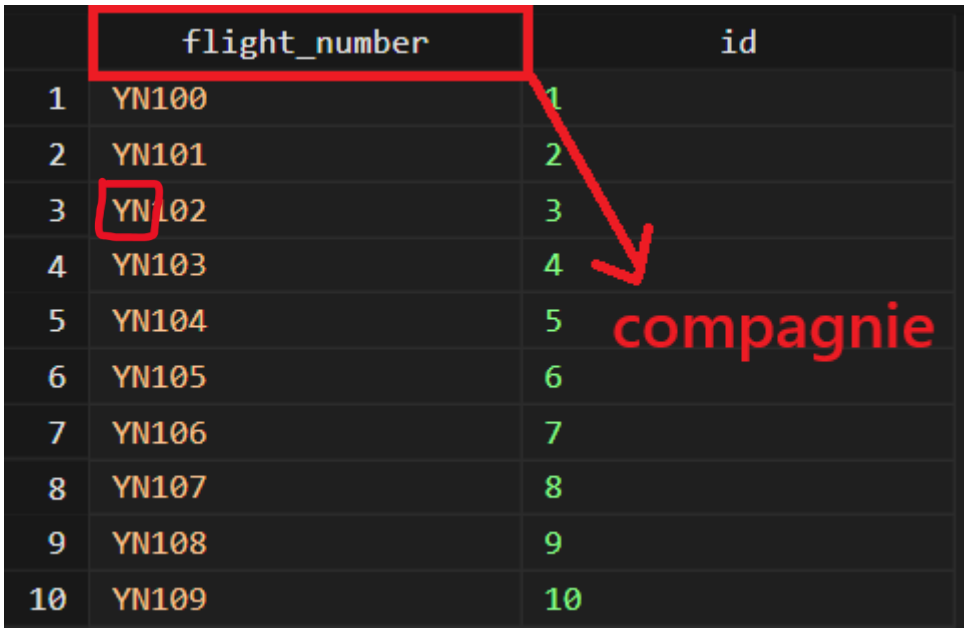
5. Requêtes SQL Documentées

5.1. Catégorie 1 : Requêtes de Base

1) Liste des vols d'une compagnie spécifique pour une date donnée

Chaque vol opéré par une compagnie possède un `flight_number` qui sert à l'identifier de manière unique au sein de cette compagnie

Exemple :

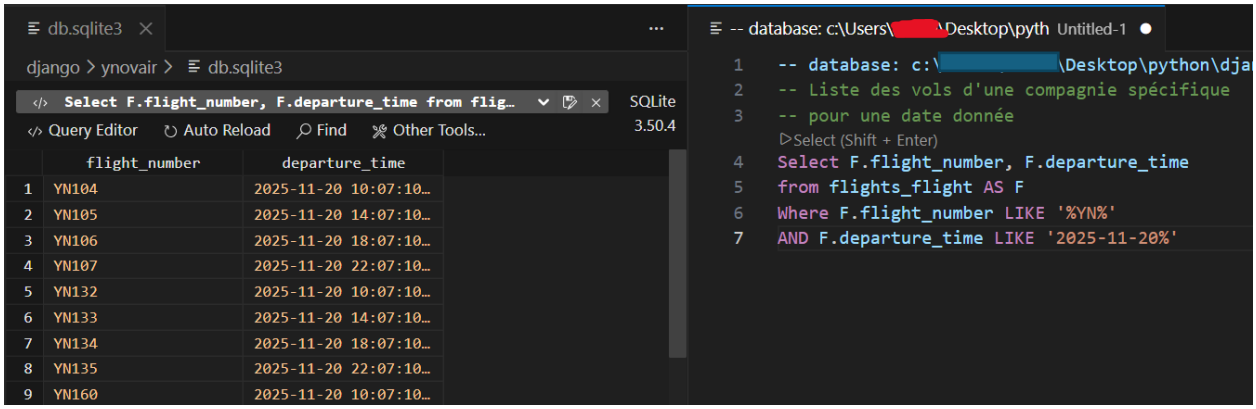


	flight_number	id
1	YN100	1
2	YN101	2
3	YN102	3
4	YN103	4
5	YN104	5
6	YN105	6
7	YN106	7
8	YN107	8
9	YN108	9
10	YN109	10

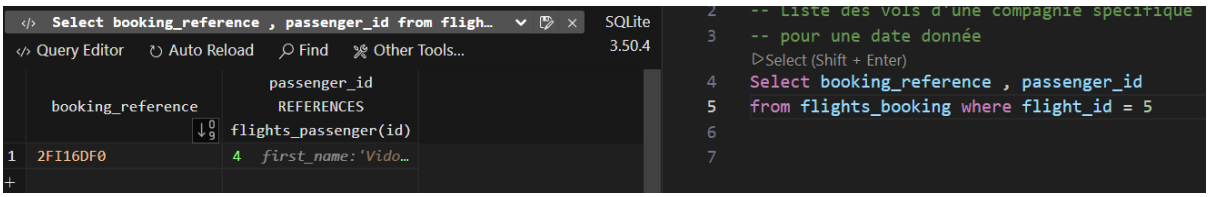
Voila le resultat :

```
-- Liste des vols d'une compagnie spécifique
-- pour une date donnée
▷Select (Shift + Enter)
Select F.flight_number, F.departure_time
from flights_flight AS F
Where F.flight_number LIKE '%compagnie%'
AND F.departure_time = 'temps_voulu'
```

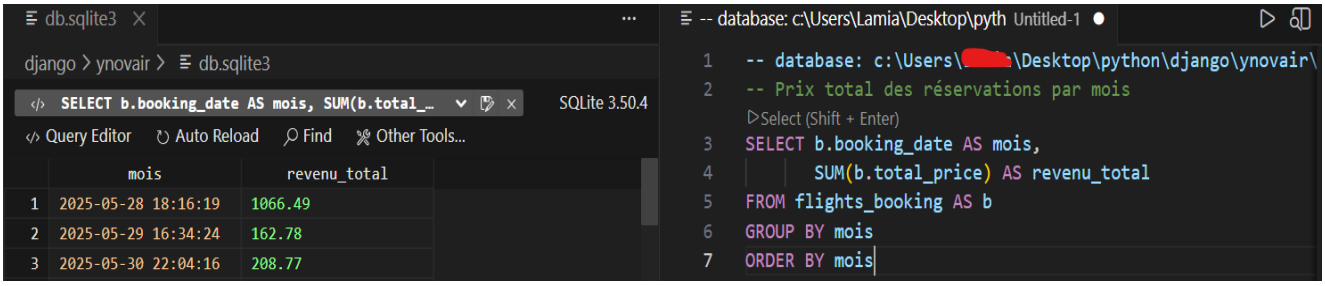
Exemple :



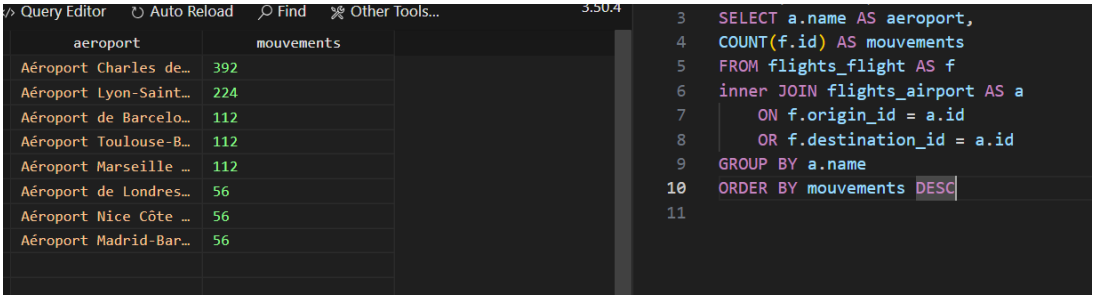
2) Passagers ayant réservé un vol spécifique :



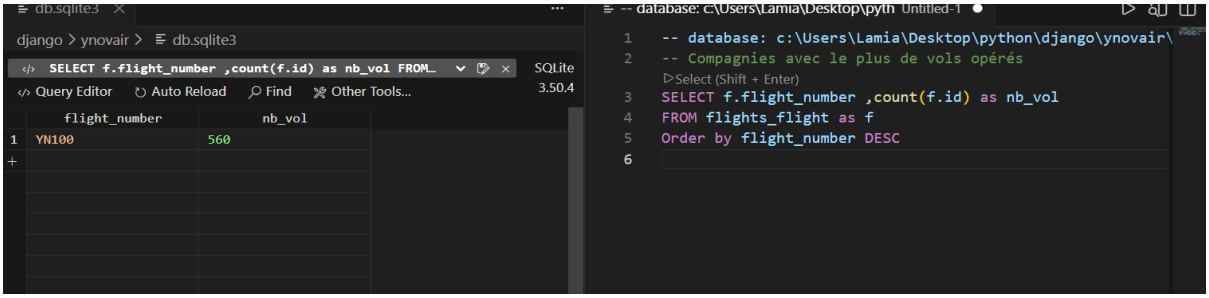
3) Prix total des réservations par mois :



4) Passagers ayant réservé un vol spécifique :

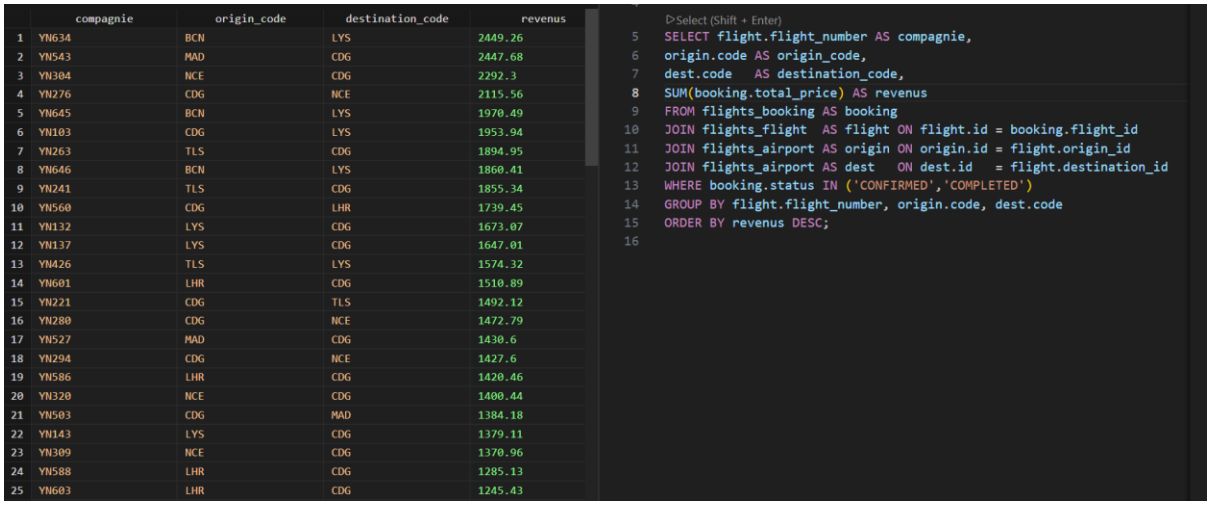


5) Compagnies avec le plus de vols opérés



5.2. Catégorie 2 : Requêtes avec Jointures Complexes

1) Revenus par compagnie et par route



	compagnie	origin_code	destination_code	revenus
1	YN634	BCN	LYS	2449.26
2	YN543	MAD	CDG	2447.68
3	YN304	NCE	CDG	2292.3
4	YN276	CDG	NCE	2115.56
5	YN645	BCN	LYS	1970.49
6	YN103	CDG	LYS	1953.94
7	YN263	TLS	CDG	1894.95
8	YN646	BCN	LYS	1860.41
9	YN241	TLS	CDG	1855.34
10	YN560	CDG	LHR	1739.45
11	YN132	LYS	CDG	1673.07
12	YN137	LYS	CDG	1647.01
13	YN426	TLS	LYS	1574.32
14	YN601	LHR	CDG	1510.89
15	YN221	CDG	TLS	1492.12
16	YN280	CDG	NCE	1472.79
17	YN527	MAD	CDG	1430.6
18	YN294	CDG	NCE	1427.6
19	YN586	LHR	CDG	1420.46
20	YN320	NCE	CDG	1400.44
21	YN503	CDG	MAD	1384.18
22	YN143	LYS	CDG	1379.11
23	YN309	NCE	CDG	1370.96
24	YN588	LHR	CDG	1285.13
25	YN603	LHR	CDG	1245.43

2) Occupation moyenne des vols par destination

	destination	taux
1	BCN	35.80357142857143
2	CDG	35.80374149659863
3	LHR	30.811507936507937
4	LYS	34.814682539682536
5	MAD	31.21865079365079
6	MRS	37.19583333333333
7	NCE	50.24166666666666
8	TLS	30.355753968253968

5.3. Catégorie 3 : Requêtes Analytiques

1) Top 10 des routes les plus rentables

	origin_code	destination_code	revenus
1	BCN	LYS	8091.7
2	CDG	NCE	6588.62
3	LHR	CDG	6439.14
4	LYS	CDG	5920.01
5	NCE	CDG	5382.13
6	LYS	BCN	4812.87
7	CDG	LHR	4609.13
8	MAD	CDG	4577.389999999999
9	LYS	MRS	4160.46
10	CDG	LYS	4015.16

```sql
SELECT origin.code AS origin_code,
       dest.code   AS destination_code,
       SUM(booking.total_price) AS revenus
FROM flights_booking AS booking
JOIN flights_flight  AS flight ON flight.id = booking.flight_id
JOIN flights_airport AS origin ON origin.id = flight.origin_id
JOIN flights_airport AS dest   ON dest.id   = flight.destination_id
WHERE booking.status IN ('CONFIRMED','COMPLETED')
GROUP BY origin_code, destination_code
ORDER BY revenus DESC
LIMIT 10
```

2) Analyse de la saisonnalité des réservations

	date	nb_reservations	revenus
1	2025-05-28 18:16:19	1	1066.49
2	2025-05-31 21:10:30	1	521.95
3	2025-06-03 09:53:37	1	1574.32
4	2025-06-03 16:09:56	1	2447.68
5	2025-06-12 23:28:20	1	813.53
6	2025-06-13 02:49:53	1	412.1
7	2025-06-13 16:45:52	1	217.08
8	2025-06-15 04:45:53	1	1223.11
9	2025-06-15 09:56:06	1	333.33
10	2025-06-20 18:39:55	1	193.91
11	2025-06-23 04:07:21	1	595.45
12	2025-06-23 04:52:49	1	579.74
13	2025-06-23 09:37:32	1	414.81
14	2025-06-24 16:43:51	1	470.43
15	2025-06-27 19:54:29	1	1673.07
16	2025-06-27 23:41:10	1	1245.43
17	2025-06-30 05:47:37	1	456.14
18	2025-06-30 18:41:46	1	1023.96

```sql
SELECT booking.booking_date AS date,
       COUNT(*) AS nb_reservations,
       SUM(booking.total_price) AS revenus
FROM flights_booking AS booking
WHERE booking.status IN ('CONFIRMED','COMPLETED')
GROUP BY date
ORDER BY date;
```

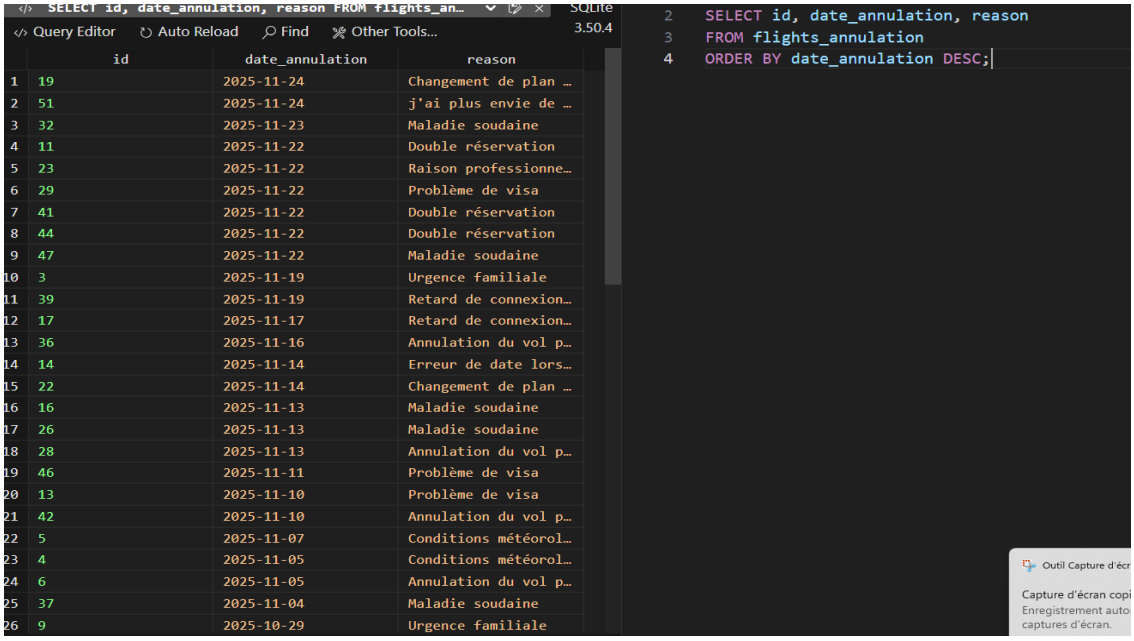
3) Taux de remplissage moyen par type d'avion

	type_avion	taux_remplissage_moyen
1	200	0.36360139860139856
2	180	0.3577690972222222
3	250	0.3491891891891892
4	150	0.34761229314420805

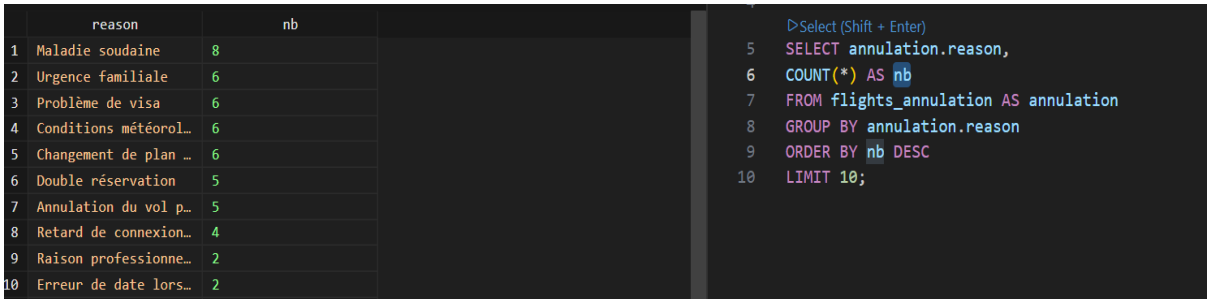
```sql
SELECT flight.total_seats AS type_avion,
AVG( (flight.total_seats - flight.available_seats) * 1.0 / flight.total_seats)
AS taux_remplissage_moyen
FROM flights_flight AS flight
GROUP BY type_avion
ORDER BY taux_remplissage_moyen DESC
```

5.4. Catégorie 4 : Requêtes Liées à la nouvelle Fonctionnalité

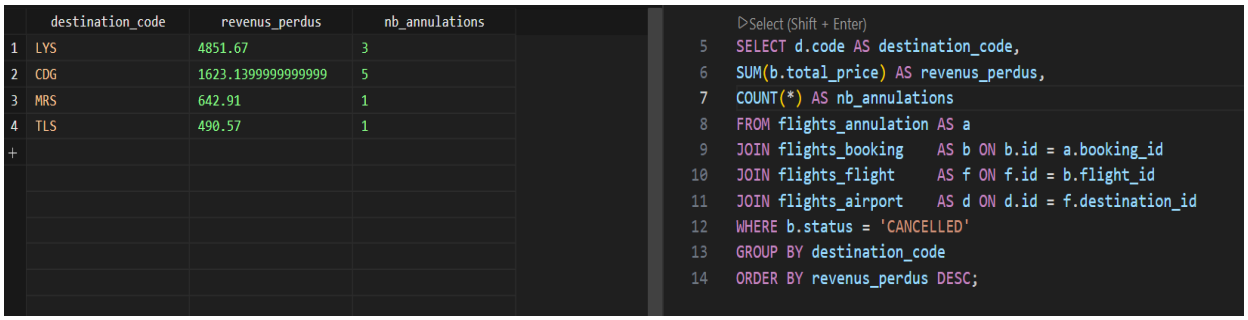
1) Liste des annulations (triées par date)



2) Motifs d'annulation les plus fréquents



3) Revenus perdus par destination (sur réservations annulées)



6. Documentation de la Nouvelle Fonctionnalité

6.1. Description détaillée

La table ***Annulation*** permet de conserver une trace précise de chaque réservation annulée. Elle indique quel passager a annulé, quelle réservation est concernée, à quelle date l'annulation a eu lieu, ainsi que la raison fournie par le passager.

6.2. Cas d'usage

Il peut être utilisé pour avoir un historique précis des annulations pour faire des statistiques, comprendre les motifs les plus fréquents et vérifier facilement ce qui s'est passé en cas de contrôle ou de demande d'information.

6.3. Règles de gestion

- Chaque annulation doit être associée à un passager
- Une annulation peut être liée à une réservation, mais ce n'est pas obligatoire
- Le motif d'annulation est obligatoire
- La date d'annulation est fixée automatiquement
- Créer une annulation entraîne un changement d'état de la réservation
- Le système ne libère pas automatiquement les sièges lors d'une annulation via la vue
- Une réservation ne doit normalement pas être annulée plusieurs fois
- Une annulation doit toujours contenir des informations fiables pour l'audit

6.4. Contraintes d'intégrité

- **reason** est obligatoire et doit contenir un texte non vide (max 400 caractères)
- **date_annulation** est automatiquement renseignée à la création et ne doit pas être modifiée ensuite
- **id** est une clé primaire auto-incrémentée garantissant l'unicité de chaque annulation

Conclusion

La mise en place de la table **Annulation** a permis d'ajouter au projet Ynov-Air une traçabilité complète des réservations annulées, en enregistrant pour chaque action qui a annulé, quelle réservation est concernée, à quelle date et pour quel motif. Cette fonctionnalité répond pleinement aux besoins de suivi, de reporting et d'audit, tout en assurant l'intégrité des données grâce à des règles cohérentes : motif obligatoire, date générée automatiquement, lien obligatoire avec le passager et conservation de l'historique même si la réservation est supprimée. Le développement a tout de même présenté quelques difficultés, notamment la gestion des relations entre Booking et Annulation, l'intégrité référentielle (SET NULL / CASCADE) et l'absence initiale de libération automatique des sièges, ce qui a nécessité une clarification de la logique métier. Parmi les améliorations possibles, il serait pertinent d'appeler systématiquement la méthode booking.cancel() pour centraliser le comportement d'annulation, d'empêcher plusieurs annulations sur une même réservation. cette table apporte un véritable plus au fonctionnement du système en garantissant une gestion fiable, propre et historique des annulations.