Portfolio Data

Voir le CV

Analyse, transformation et visualisation de données

Tinder Match Predictor

Ce projet explore les indicateurs clés pour prédire les "matchs" sur Tinder. À partir d'un jeu de données brut, j'ai effectué une analyse exploratoire et construit des visualisations interactives pour identifier les variables les plus influentes.

Voir le projet

Voir la présentation PDF du projet

Aircraft Performance Insights

Ce notebook présente une analyse des performances des compagnies aériennes, en croisant modèles d'avions, aéroports et revenus par passager. Les données brutes ont été transformées et modélisées via dbt et Snowflake.

Voir le projet

AI Takes My Photos

Application web de génération d'images via des modèles d'IA générative (Black Forest Labs et GPT-image-1). L'interface permet de soumettre des prompts et de recevoir des images générées automatiquement.

Voir le projet

Dashboard Décisionnel Power BI

Tableau de bord interactif réalisé avec Power BI, après ingestion et transformation des données brutes via Power Query. Visualisation des KPI et indicateurs clés pour la prise de décision.

Voir le projet

© 2025 Piotr Grudzinski - Tous droits réservés

Aircraft Performance Insights

Ce notebook présente une analyse des performances des compagnies aériennes, en croisant modèles d'avions, aéroports et revenus par passager. Les données brutes ont été transformées et modélisées via dbt et Snowflake.

Voir le projet

Welcome on this dbt project on Aircraft Analytics

The goal of this project is that you bring a complete dbt configuration which will help gathering informations on Aircraft data

First of all, let's take the content of src/aircraft_db.sql and copy / paste it in a snowflake worksheet.

Be sure to select the transforming warehouse the raw database and the public schema.

run the script - it can take a few mins

check that the data is well loaded

```
In [ ]:select * from raw.public.aircraft;
In [ ]:select * from raw.public.airlines;
In [ ]:select * from raw.public.airports;
In [ ]:select * from raw.public.flight_summary_data;
```

Configure your dbt project so that it reflects

Sources of your 4 tables Staging models, facts and dimensions (at least dim_aircraft, dim_airline and dim_airport)

```
In [ ]:models/staging/sources.yml

  - name: aircraft
    database: raw
    schema: public
    tables:
      - name: aircraft
      - name: airlines
      - name: airports
      - name: individual_flights
      - name: flight_summary_data
```

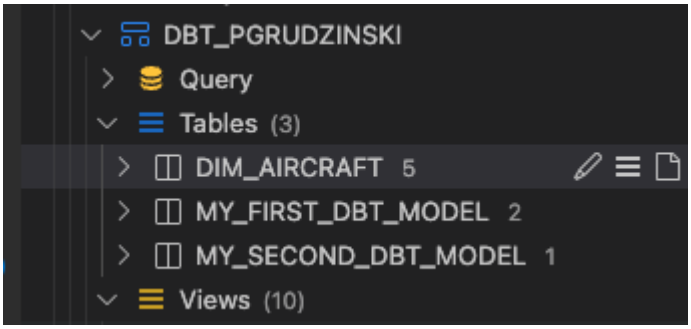
Which aircraft has flown the most?

modèle DBT de dim_aircraft

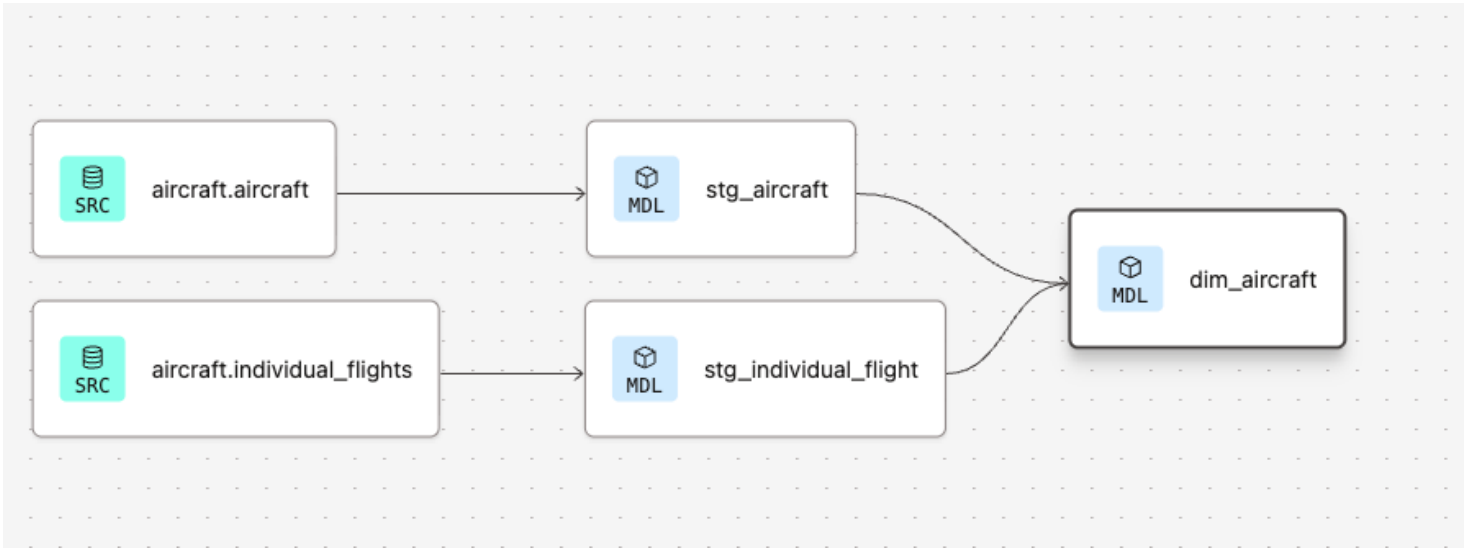
```
{{
  config(
    materialized='table',  )
}}

select
    a."Aircraft_Id",
    a."Aircraft_Type",
    COUNT(DISTINCT f."Flight_Id") AS "total_flights"
from {{ ref('stg_aircraft') }} a
join {{ ref('stg_individual_flight') }} f
on a."Aircraft_Id" = f."Aircraft_Id"
group by 1,2
```

Sur snowflake :



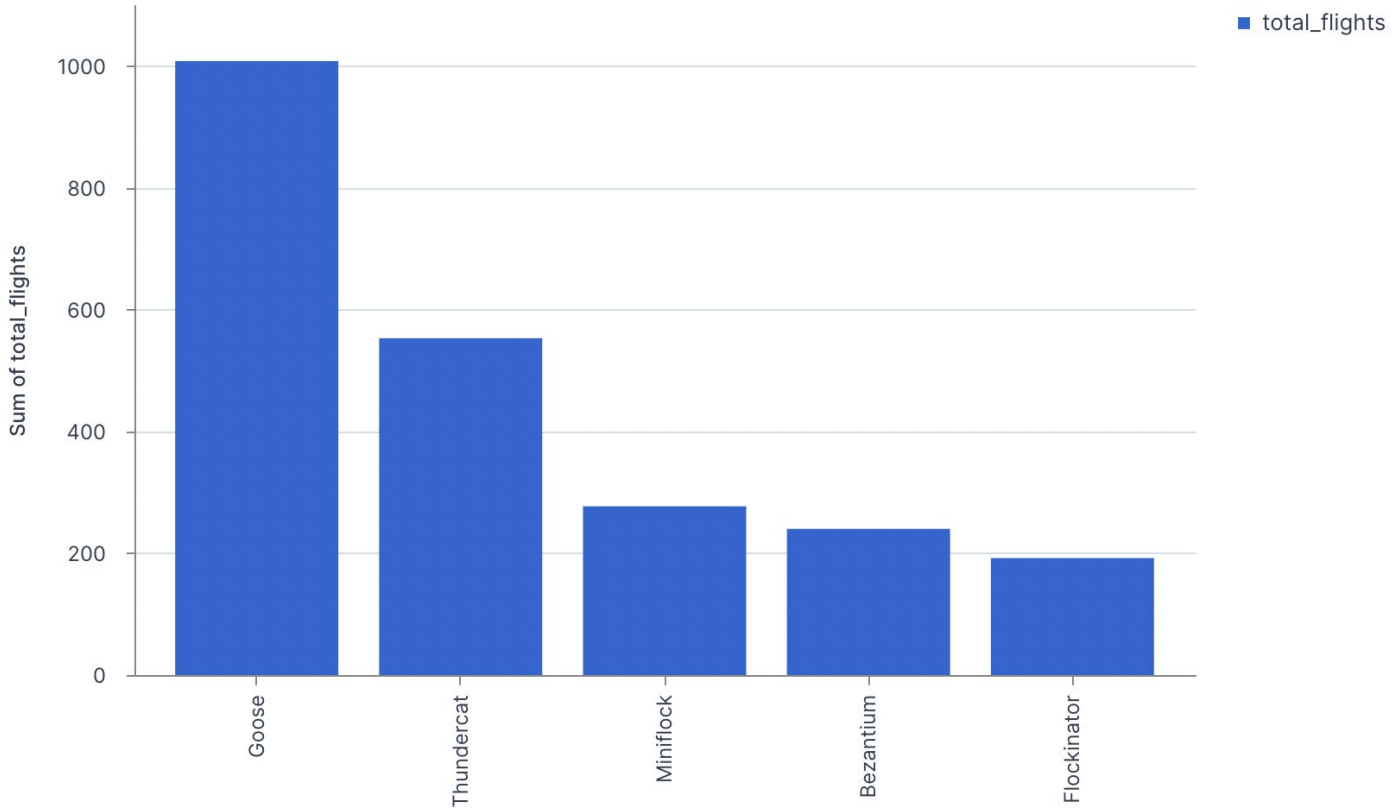
Linéage sur DBT



In [ ]:select * **from** "DIM_AIRCRAFT"

Aircraft_Id TEXT	Aircraft_Type TEXT	total_flights NUMBER(NULL)
t10	Thundercat	553
12a	Miniflock	277
g72	Goose	1008
b23	Bezantium	240
12d	Flockinator	192

Réultat sur deepnote



Which airport has transported the most passengers through it?

Calculate the number of passengers transported as the number of flights of each aircraft type, times its capacity, all together grouped by airport.

It is worth mentioning that every individual flight will transport N passengers, but that will double count the number of people because it will be attributed to both airports (departure and destination) per flight, therefore, N would be attributed to airport A and airport B.

tables intermédiaires dbt

In [ ]:
```
-- table intermédiaire "int_departure_airports" sur dbt
WITH flights AS (
    SELECT * FROM {{ ref('stg_individual_flight') }}
),
aircrafts AS (
    SELECT * FROM {{ ref('stg_aircraft') }}
),
airports AS (
    SELECT * FROM {{ ref('stg_airports') }}
)
SELECT
```

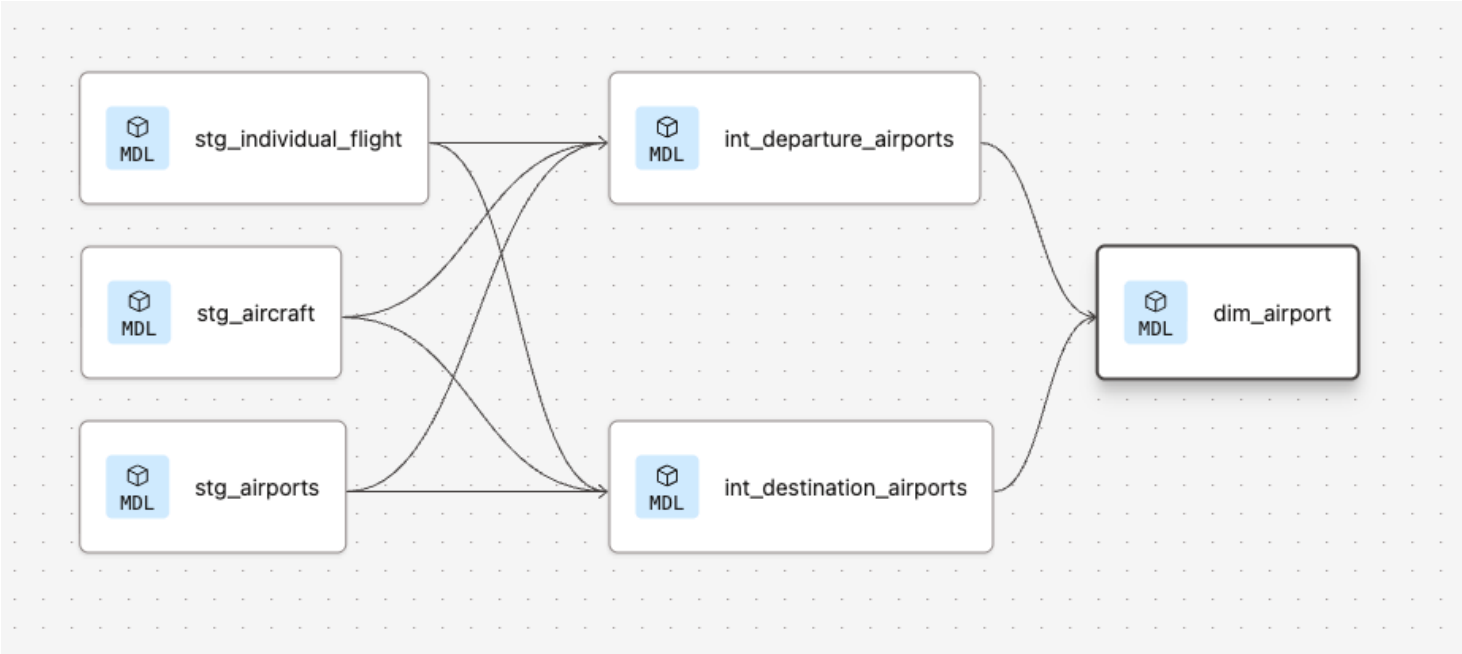
```
    f."Flight_Id",
    f."Departure_Airport_Code",
    a."Airport_Name",
    ac."Aircraft_Type",
    ac."Capacity"
FROM flights f
INNER JOIN airports a
    ON f."Departure_Airport_Code" =
a."Airport_Code"
INNER JOIN aircrafts ac
    ON f."Aircraft_Id" = ac."Aircraft_Id"
-- table intermédiaire "int_destination_airports"
sur dbt
WITH flights AS (
    SELECT * FROM {{ ref('stg_individual_flight') }}
),
aircrafts AS (
    SELECT * FROM {{ ref('stg_aircraft') }}
),
airports AS (
    SELECT * FROM {{ ref('stg_airports') }}
)
SELECT
    f."Flight_Id",
    f."Destination_Airport_Code",
    a."Airport_Name",
    ac."Aircraft_Type",
    ac."Capacity"
FROM flights f
INNER JOIN airports a
    ON f."Destination_Airport_Code" =
a."Airport_Code"
INNER JOIN aircrafts ac
    ON f."Aircraft_Id" = ac."Aircraft_Id"
```

tables dim.airports sur dbt

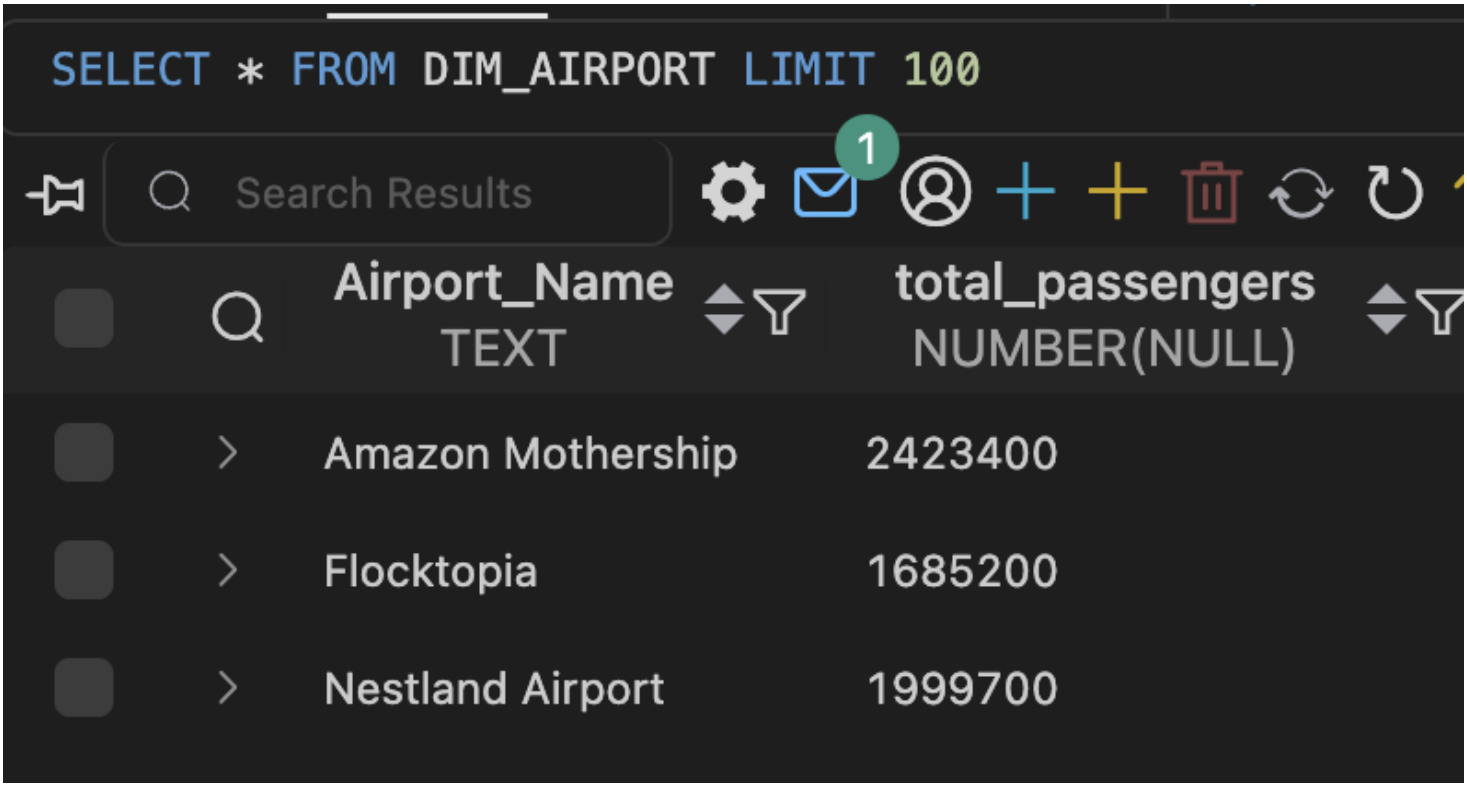
```
In [ ]:{{
      config(
    materialized='table',  )
    }}
    WITH departure_passengers AS (
        SELECT
            "Flight_Id",
            "Departure_Airport_Code",
            NULL AS "Destination_Airport_Code",
            "Airport_Name",
            "Capacity"
        FROM {{ ref('int_departure_airports') }}
    ),
    destination_passengers AS (
        SELECT
            "Flight_Id",
            "Destination_Airport_Code",
```

```
        NULL AS "Departure_Airport_Code",
        "Airport_Name",
        "Capacity"
    FROM {{ ref('int_destination_airports') }}
),
temp_total_passengers AS (
    SELECT
        "Flight_Id",
        COALESCE("Departure_Airport_Code",
"Destination_Airport_Code") AS "Airport_Code",
        "Airport_Name",
        "Capacity"
    FROM departure_passengers
    UNION ALL
    SELECT
        "Flight_Id",
        COALESCE("Departure_Airport_Code",
"Destination_Airport_Code") AS "Airport_Code",
        "Airport_Name",
        "Capacity"
    FROM destination_passengers
),
aggregated_passengers AS (
    SELECT
        "Airport_Name",
        SUM("Capacity") AS "total_passengers"
    FROM temp_total_passengers
    GROUP BY "Airport_Name"
)
SELECT
    "Airport_Name",
    "total_passengers"
FROM aggregated_passengers
ORDER BY "Airport_Name"
```

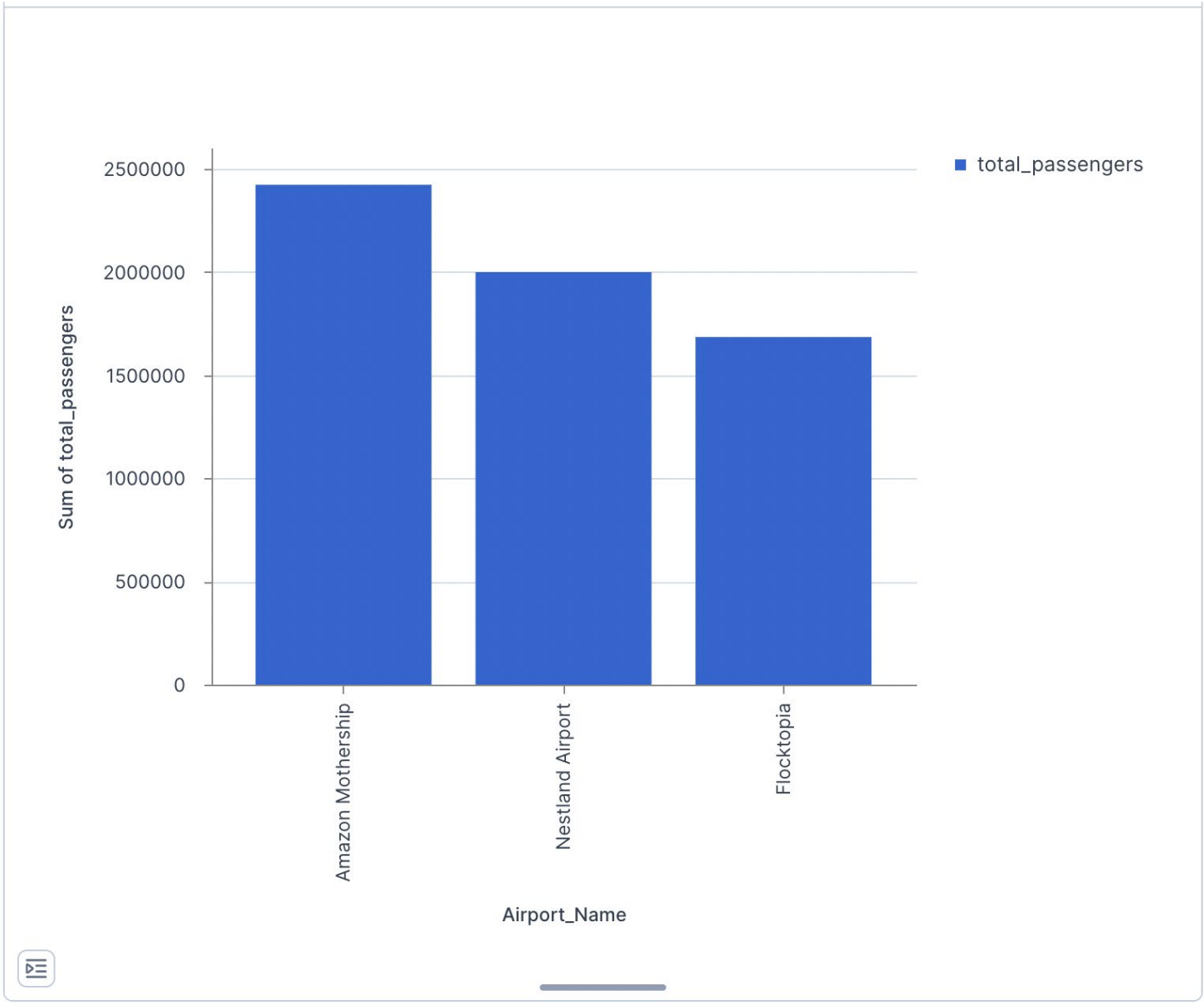
Linéage sr DBT



Résultat sur snowflake



Réultat sur deepnote



What was the best year for Revenue Passenger-Miles for each airline?

The fields that track international values have frequent null values. We will consider them as zeros for simplicity but that must be accounted for before doing any kind of analysis because that will bias the results.

Process Calculate the SUM of all RPM (Domestic, International or both at the same time, so, Total), and then select the MAX per airline, which will yield us the year as well. By selecting a specific type of RPM or both we will obtain different answers, therefore all three must be provided.

Tables intermediaires

In [ ]:-- int_domestic_rpm.sql

```
WITH summary AS (
    SELECT * FROM {{
ref('stg_flitgh_summary_data') }}
```

```
),
airlines as
    (
        SELECT * FROM {{ ref('stg_airlines')}}
    )
SELECT
    s."Airline_Code",
    s."Date",
    a."Airline_Name",
    s."RPM_Domestic",
FROM summary s
INNER JOIN airlines a
    ON s."Airline_Code" = a."Airline_Code"

-- int_international_rpm.sql

WITH summary AS (
    SELECT * FROM {{
ref('stg_flitgh_summary_data') }}
),
airlines as
    (
        SELECT * FROM {{ ref('stg_airlines')}}
    )
SELECT
    s."Airline_Code",
    s."Date",
    a."Airline_Name",
    s."RPM_International",
FROM summary s
INNER JOIN airlines a
    ON s."Airline_Code" = a."Airline_Code"
```

table de dimension

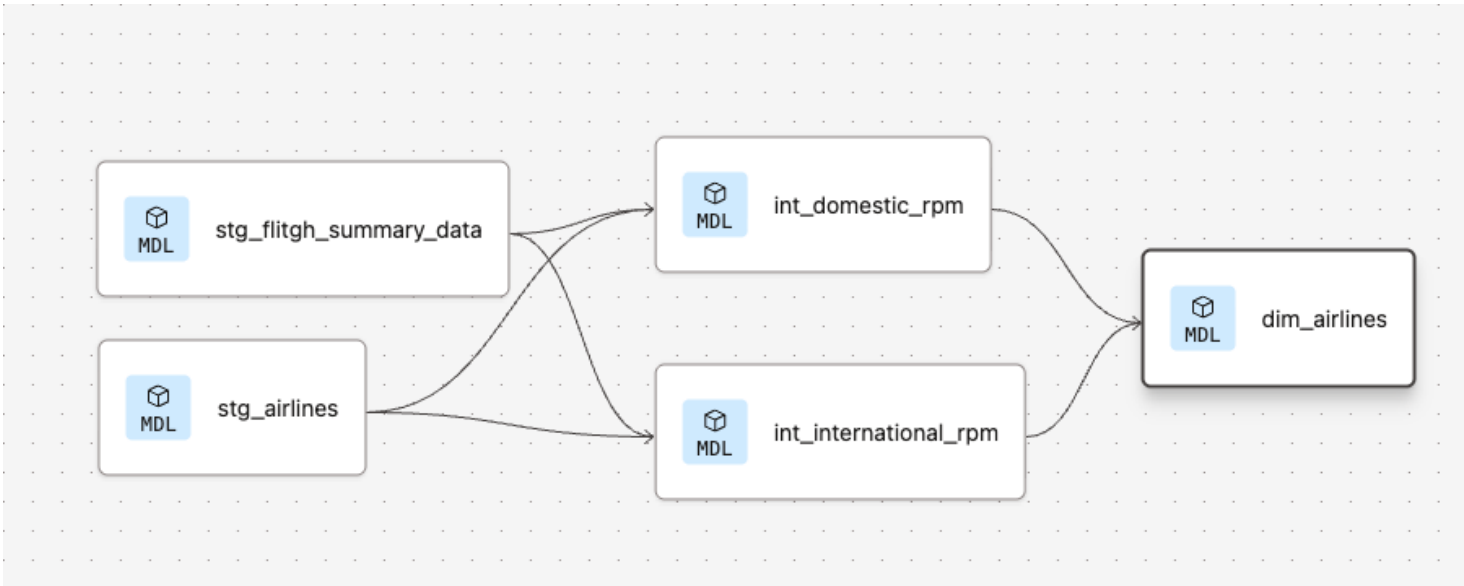
```
In [ ]:-- dim_airlines
{{
  config(
    materialized='table')
}}
WITH domestic_rpm AS (
    SELECT
        *
    FROM {{ ref('int_domestic_rpm') }}
),
international_rpm AS (
    SELECT
        *
    FROM {{ ref('int_international_rpm') }}
),
total_rpm AS (
    SELECT
    "Airline_Code",
    "Airline_Name",
```

```
    EXTRACT(YEAR FROM CAST("Date" AS DATE))
as "Year",
    "RPM_Domestic",
    NULL AS "RPM_International"
FROM domestic_rpm
UNION ALL
SELECT
    "Airline_Code",
    "Airline_Name",
    EXTRACT(YEAR FROM CAST("Date" AS DATE))
as "Year",
    NULL AS "RPM_Domestic",
    "RPM_International",
FROM international_rpm
),
aggregated_domestic_rpm1 AS (
    SELECT
        "Airline_Name",
        "Year",
        SUM("RPM_Domestic") as
"sum_domestic_rpm",
        SUM("RPM_International") as
"sum_international_rpm"
    FROM total_rpm
    GROUP BY "Airline_Name", "Year"
),
aggregated_domestic_rpm2 AS (
    SELECT
        "Airline_Name",
        "Year",
        MAX("sum_domestic_rpm") as
"max_domestic_rpm",
        MAX("sum_international_rpm") as
"max_international_rpm"
    FROM aggregated_domestic_rpm1
    GROUP BY "Airline_Name", "Year"
),
ranked_aggregated_domestic_rpm AS (
Select
    "Airline_Name",
    "Year",

"max_domestic_rpm"+"max_international_rpm"
as "max_total_sum_rpm",
    RANK() OVER
    (PARTITION BY "Airline_Name" ORDER BY
"max_total_sum_rpm" DESC) AS
"Rank_max_total_rpm",
FROM aggregated_domestic_rpm2
WHERE "max_international_rpm" IS NOT NULL
)
SELECT
    *
FROM ranked_aggregated_domestic_rpm
```

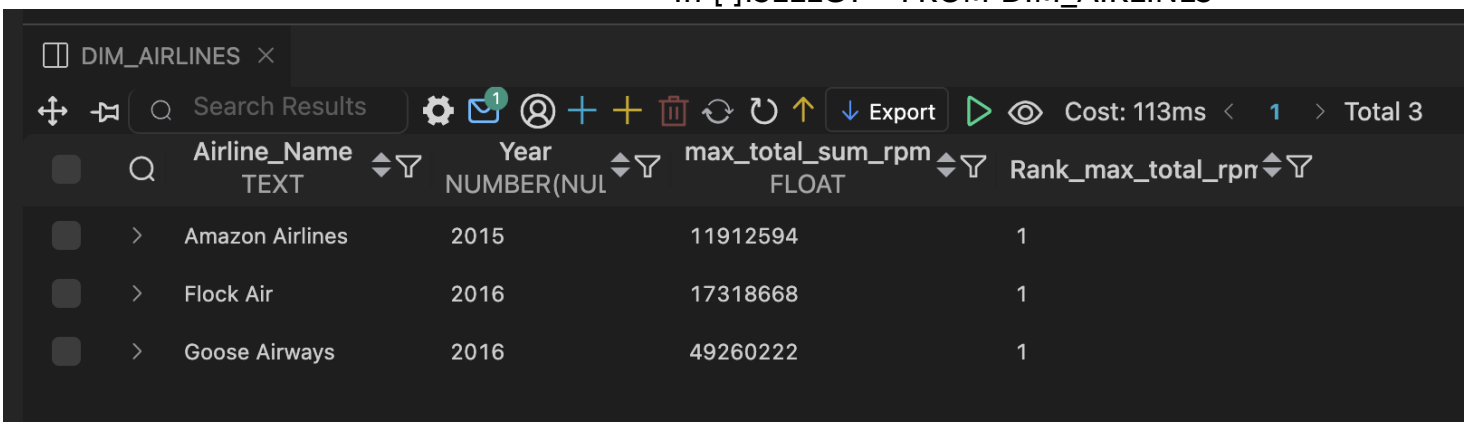
WHERE "Rank_max_total_rpm" = 1

Linéage sur DBT



Résultat sur snowflake

In [ ]:SELECT * FROM DIM_AIRLINES



What was the best year for growth for each airline?

We have ASM and RPM, as well the number of flights Domestic and for some flights International, per Airline and Airport. To list some of the possible ways of evaluate growth, we have: Decrease the difference between ASM - RPM, therefore captivating more passengers out of the ones that they had the capacity for. Increase on ASM, therefore increasing the fleet or the size of the planes, or the number of flights. Increase in number of flights alone.

For simplicity and because it seems to be the one providing the best answer, we will use ASM as our growth indicator, and the more ASM an airline has over time, the more we will say they grew.

To calculate the growth we will take the AVG(ASM_Domestic) per Airline per Year.

tables intermédiaires

In [ ]:

```
-- int_domestic_ASM

WITH summary AS (
    SELECT * FROM {{
ref('stg_flitgh_summary_data') }}
),
airlines as
    (
        SELECT * FROM {{ ref('stg_airlines')}}
    )
SELECT
    s."Airline_Code",
    s."Date",
    a."Airline_Name",
    s."ASM_Domestic",
FROM summary s
INNER JOIN airlines a
    ON s."Airline_Code" = a."Airline_Code"
```

table de dimension

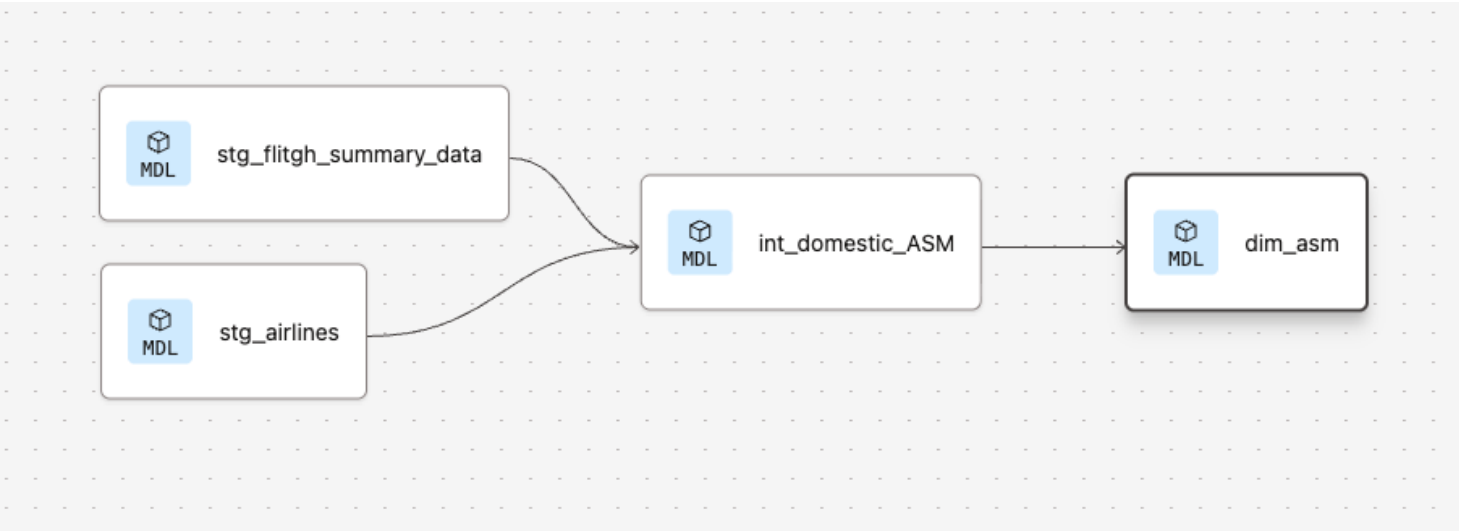
In [ ]:

```
{{
  config(
    materialized='table')
}}
WITH domestic_asm AS (
    SELECT
    "Airline_Code",
    "Airline_Name",
    EXTRACT(YEAR FROM CAST("Date" AS DATE))
as "Year",
    "ASM_Domestic"
    FROM {{ ref('int_domestic_ASM') }}
),
aggregated_domestic_asm AS (
    SELECT
        "Airline_Name",
        "Year",
        AVG("ASM_Domestic") as
"avg_domestic_asm"
    FROM domestic_asm
    GROUP BY "Airline_Name", "Year"
),
lagged_domestic_asm AS (
SELECT
    "Airline_Name",
    "Year",
    "avg_domestic_asm",
    LAG("avg_domestic_asm") OVER(ORDER BY
```

```
"Airline_Name", "Year") AS "one_year_before",
    "avg_domestic_asm" -
LAG("avg_domestic_asm") OVER(ORDER BY
"Airline_Name", "Year") AS "yearly_change",
    (("avg_domestic_asm" -
LAG("avg_domestic_asm") OVER(ORDER BY
"Airline_Name", "Year")) * 100 /
    NULLIF(LAG("avg_domestic_asm")
OVER(PARTITION BY "Airline_Name" ORDER BY
"Year"), 0)) AS "yearly_perc_change"
FROM aggregated_domestic_asm
),
ranked_lagged_asm AS (
SELECT
    "Airline_Name",
    "Year",
    "avg_domestic_asm",
    "one_year_before",
    "yearly_change",
    "yearly_perc_change",
    RANK() OVER
    (PARTITION BY "Airline_Name" ORDER BY
"yearly_perc_change" DESC) AS
"Rank_yearly_perc_change"
FROM lagged_domestic_asm
WHERE "yearly_perc_change" IS NOT NULL

)
SELECT
*
FROM ranked_lagged_asm
WHERE "Rank_yearly_perc_change" = 1
```

Linéage sur DBT



Résultat sur snowflake

In [ ]:select * **from** DIM_ASM

Airline_Name TEXT	Year NUMBER(NUL	avg_domestic_asm NUMBER(NULL)	one_year_before	yearly_change	yearly_perc_change	Rank_yearly_perc_cl
Amazon Airlines	2012	283974.916667	194470.666667	89504.250000	46.024550403410	1
Flock Air	2016	427255.416667	368195.583333	59059.833334	16.040342689441	1
Goose Airways	2010	885522.777778	733840.361111	151682.416667	20.669674864620	1

NETFLIX ANALYZER

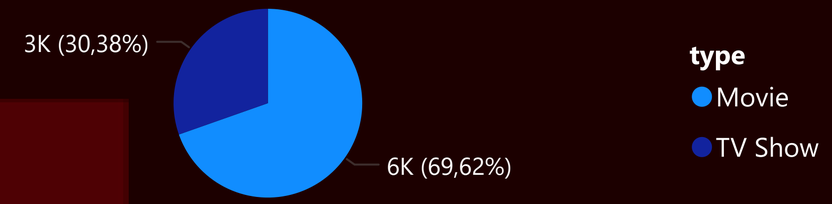
Nombre de Pays

123

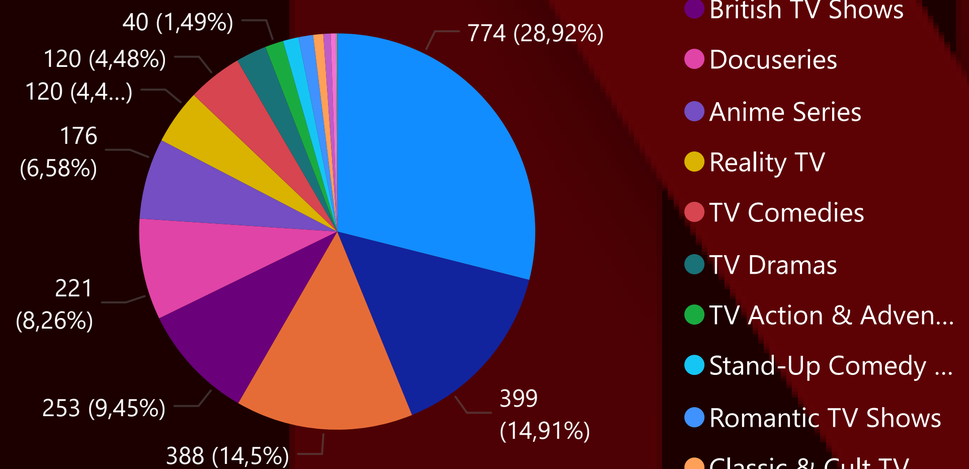
Nbre de shows

8,809K

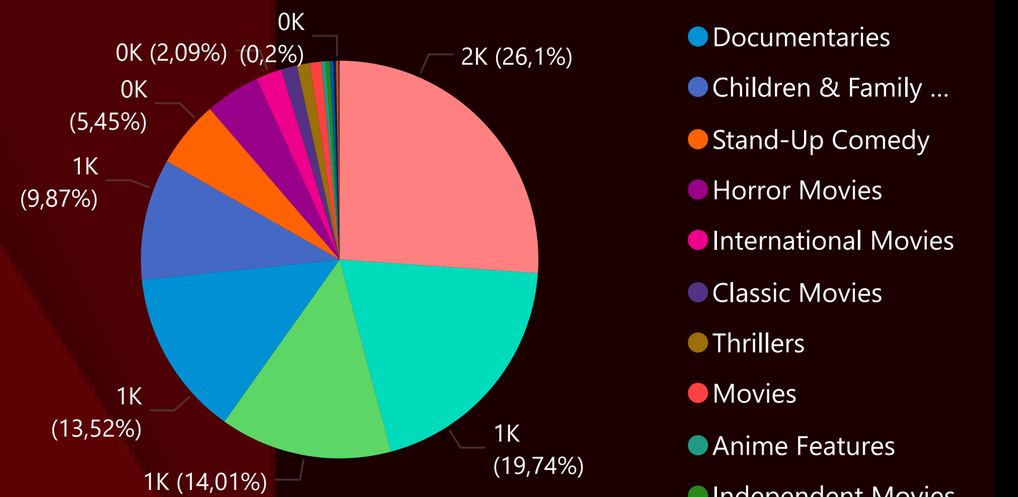
Nombre total de shows par type



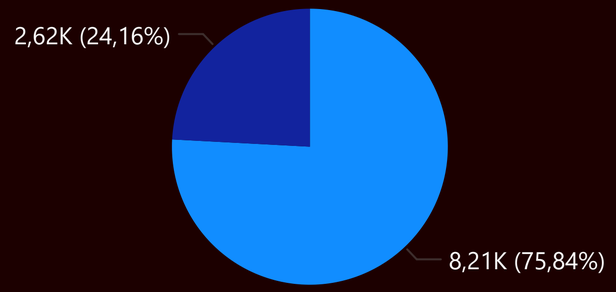
Répartitions des séries par genre principal



Répartition des films par genre principal



Proportion par TOP 10 pays



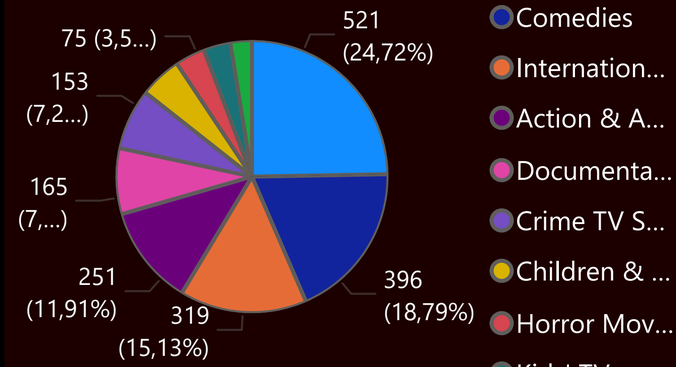
Catégorie
- TOP 10 des pays
- Le reste des pays

Type: Tout

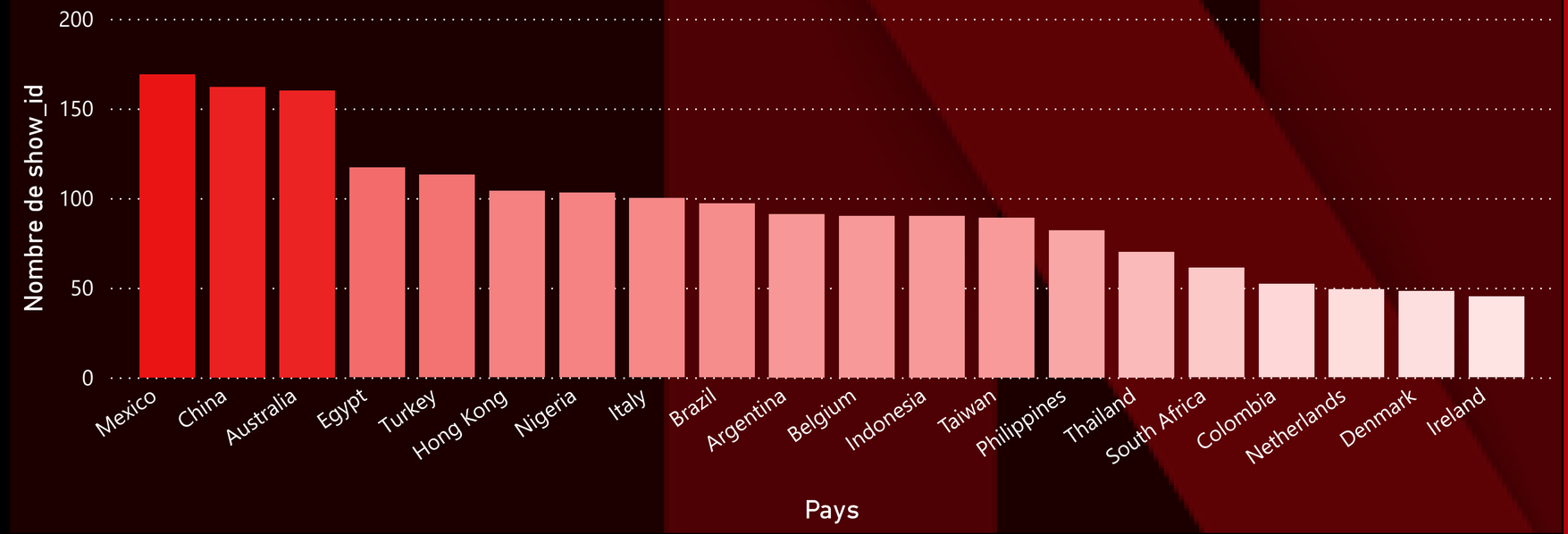
Catégorie: Le reste des pays

Region: Tout

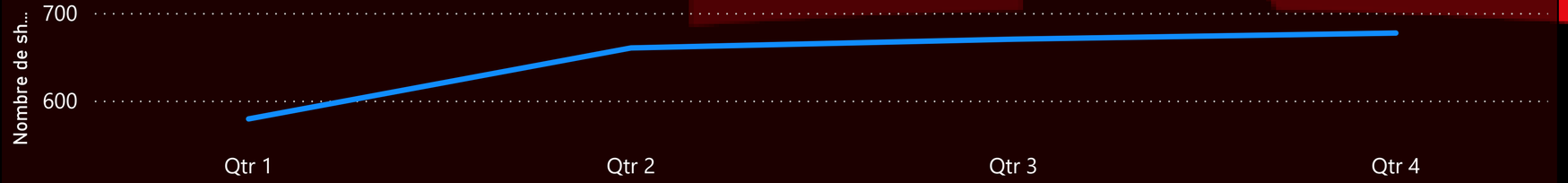
Les 10 genres dominant par pays



Nombre de titres par pays



Périodes de sorties (toutes années > 2016)



1. Les 10 pays (sur 123) les plus fournis en catalogues totalisent les 3 quart des titres (et les USA, 40% à eux seuls)
2. (films totalité) Les Drames sont toujours le genre dominant, sauf en Afrique où c'est la Comédie
3. (films hors le top 10) ca a etre plutot les films Criminels (Mexique) ou d'Action et d'Aventure (Chine) qui vont être le genre dominant
4. (Séries) genre généraliste "show international" dominant
5. Saisonalité des sorties : plûtot semestre 2 en Afrique et aux Amériques, semestre 3 en Asie et en Océanie, semestre 4 en Europe.

Projet “Tinder”

Jedha FSDA

Contexte

Suite à l’organisation d’un speed dating, un sondage a été effectué auprès des participants.

Le but ? Essayer de juguler la baisse du nombre de matchs qui a été constaté sur Tinder.

Il en ressort un ensemble de données qui peuvent nous permettre de répondre à la question : qu’est ce qui fait que les personnes se plaisent mutuellement ?

Comment faire ? en récoltant ces données, en les analysant et obtenant des visualisations pertinentes.

Le principe

on confronte le nombre de match (colonne "match") avec des indicateurs pertinents issues des réponses au questionnaire.

On en dégage des possibles influences ou inférences.

quels indicateurs garder / exclure ?

- Exclus d'office : "attr1_1" et suivantes (attractivités recherchés chez l'autre), trop qualitatif pour être analysé

- le goal → **Oui mais…**
- l'âge → **Non**
- L'income → **Oui mais…**
- les préférences → **OK**

RGPD

Dans la mesure où les données de Speed+Dating+Data.csv ont été collectées en dehors de l'UE, les données personnelles peuvent être traitée

Dictionnaires de données

Même si pas très structuré, le dictionnaire de données donne des infos précieuses sur la signification des colonnes, permettant ainsi de lire correctement les résultats des analyses

La démarche

On confronte le nombre de match (colonne "match") avec des indicateurs pertinents issues des réponses au questionnaire. On en dégage des possibles influences ou inférences

Corrélation entre le goal et les matches

Le nombre de match dépend-il du "goal"
que l'on a lorsqu'on se rend au speed dating ?
colonne : "goal"
"What is your primary goal in participating in this event?
 "Seemed like a fun night out=1
 To meet new people=2
 To get a date=3
 Looking for a serious relationship=4
 To say I did it=5
 Other=6"

Il est possible de répondre à cette question mais avec précaution.
Pourquoi ?

Corrélation entre le goal et les matches

Sanity check 1 : 79 lignes de "goal" sont vides, il faudrait les remplir avec le mode (utile pour les valeurs déclaratives).

Sanity check 2 : La variabilité est médiocre. L'écart type est de 1.4 pour une moyenne de 2,12 sur des valeurs allant de 1 à 6 : il se situe à 12% par rapport à la moyenne.

– > cet indicateur est donc à manier avec précaution pour cette raison

```
#remplir les cellules vides avec le mode récupéré
df_filled = df1.fillna({'goal': mode_value})
df1 = df_filled
```

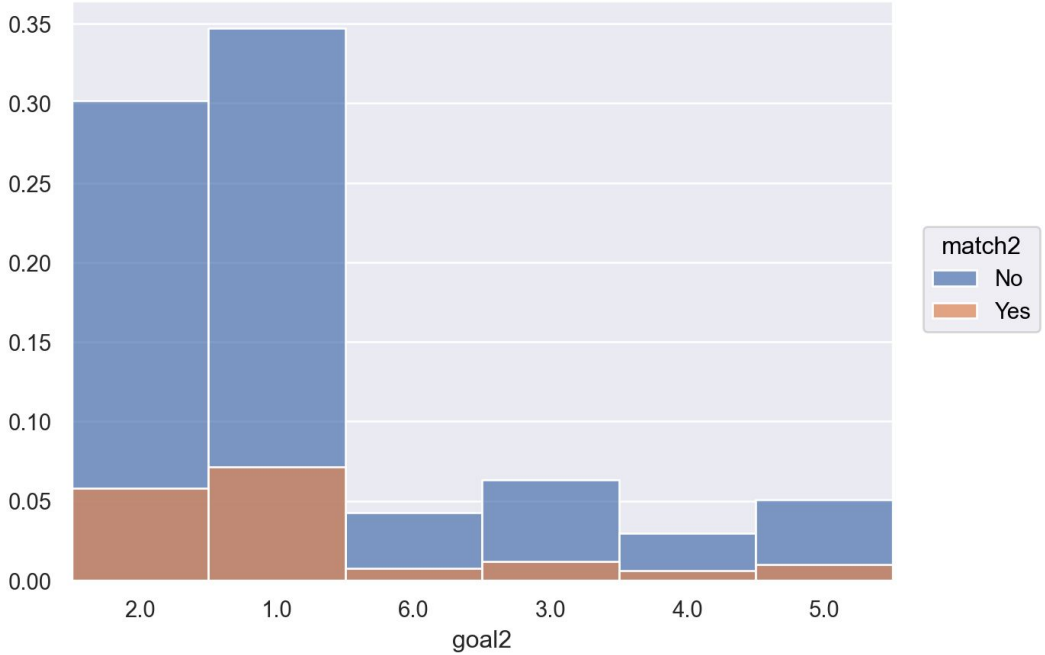
```
#Sanity check #2 : vérifier la bonne variabilité des données
df2.describe()
```

[33] ✓ 0.0s

	match	goal
count	8378.000000	8299.000000
mean	0.164717	2.122063
std	0.370947	1.407181
min	0.000000	1.000000
25%	0.000000	1.000000
50%	0.000000	2.000000
75%	0.000000	2.000000
max	1.000000	6.000000

Corrélation entre le goal et les matches

#Dataviz qui montre la relation entre les matchs et les réponses de "Goal"
#Utilisation de SO pour les histogrammes, plus lisibles

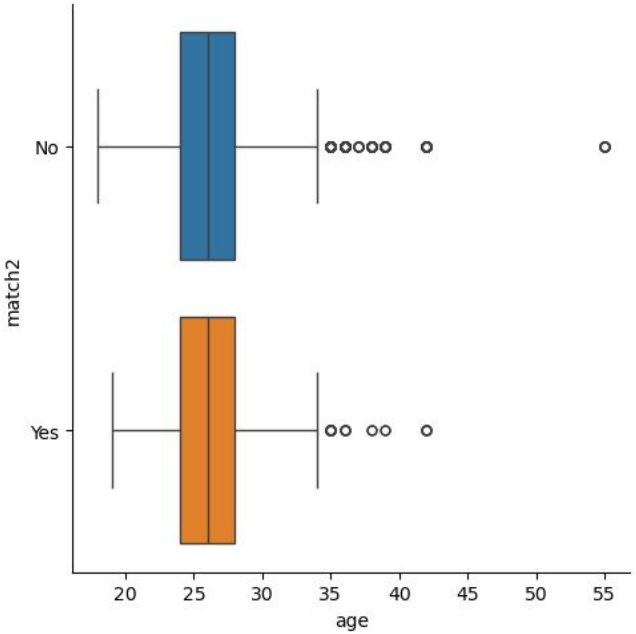


Conclusion : la corrélation entre les matchs et la valeur 1 de Goal "Seemed like a fun night out" est la plus forte. Cette prédiction est à nuancer vu la médiocre variabilité.

Corrélation avec l’âge

#la variabilité est trop faible (trop de gens ont le même âge), et l'écart type (5,45 % avec les outliers, 8,8 sans ceux là) est trop réduit par rapport à la moyenne

```
count    8283.000000
mean       26.358928
std         3.566763
min        18.000000
25%        24.000000
50%        26.000000
75%        28.000000
max        55.000000
Name: age, dtype: float64
```



Conclusion : par sa très faible variabilité , cet indicateur sera exclu des prédictions.

Corrélation avec le revenu (“income”)

#Sanity check 1 : complétude des données. La moitié des données sont manquantes (4279 sur 8378)

On peut quand même se baser sur la population renseignée si les données manquantes ne sont pas liés à la variable cible (le match).

```
#Vérification des opérations de conversion en float et de l'exclusion des NaN
df1IncomeNotNull["income"]
```

[27] ✓ 0.0s

```
0       69487.00
1       69487.00
2       69487.00
3       69487.00
4       69487.00
          ...
8351    55138.00
8352    55138.00
8353    55138.00
8354    55138.00
8355    55138.00
Name: income, Length: 4279, dtype: object
```

Voyons si c‘est le cas !

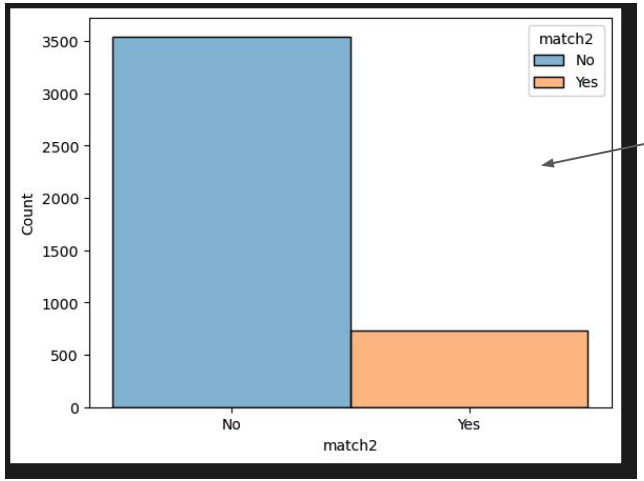
Si on démontre que les matchs sont équivalents dans la population renseignée et dans la population non renseignée, alors cet indicateur sera ok.

Corrélation avec le revenu (“income”)

On voit que les match YES et les match NO sont comparables de manière satisfaisante

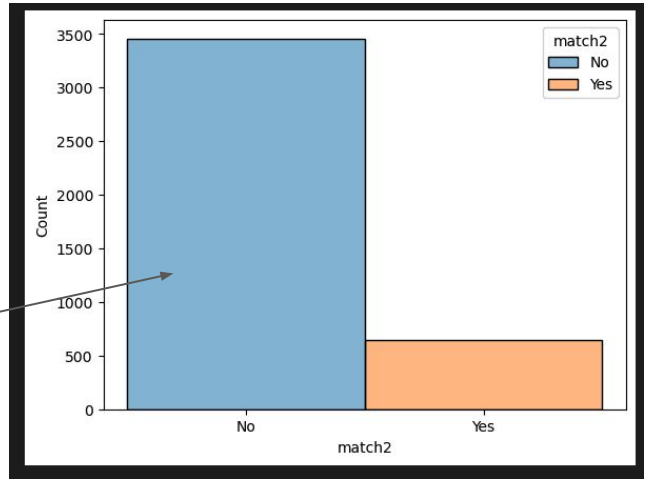
La différence de YES est de 14 %, ceux des NO de 2,5 %

	countMatchYesIncomeNotNull	countMatchNoIncomeNotNull	countMatchYesIncomeNull	countMatchNoIncomeNull	percent_diff_Yes	percent_diff_No
0	644	3455	736	3543	-14.285714	-2.547033



Population non renseignée

Population renseignée



Corrélation avec le revenu (“income”)

#Sanity check 2 : variabilité. Sur 4279 income renseigné, seuls 261 sont uniques, le reste a la même valeur. Cela reste suffisant pour dresser des tendances

```
totalcountincome = df1IncomeNotNull["income"].count()
uniquecountincome = df1IncomeNotNull["income"].nunique()
uniquecountincomeduplicated = totalcountincome - uniquecountincome
print(totalcountincome)
print(uniquecountincome)
print(uniquecountincomeduplicated)
```

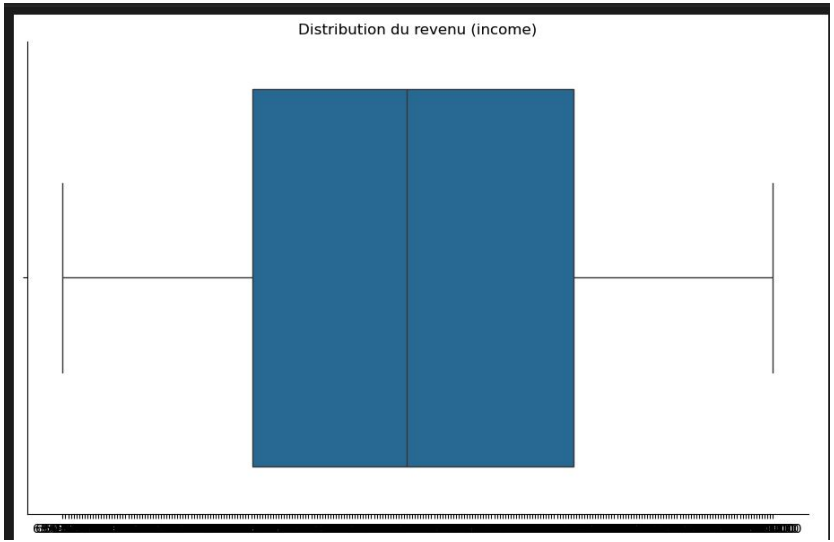
```
4279
261
4018
```

Voyons maintenant si ces valeurs sont suffisamment bien distribués !

Corrélation avec le revenu (“income”)

Variabilité suffisante : écart type qui représente 38 % de la moyenne, distribution relativement étalée

```
...    count      4279.000000
       mean      44887.606450
       std       17206.920962
       min        8607.000000
       25%       31516.000000
       50%       43185.000000
       75%       54303.000000
       max      109031.000000
       Name: income, dtype: float64
```



Et aucun outliners !
On va pouvoir voir la corrélation entre match et income.
Mais comment le faire efficacement ?

Corrélation avec le revenu (“income”)

#Création de catégories d'income pour une meilleure lisibilité (on passe du numérique au catégoriel)

```
df1IncomeNotNullUnique["income"] = df1IncomeNotNullUnique["income"].astype("float64")
df1IncomeNotNullUnique2 = df1IncomeNotNullUnique
df1IncomeNotNullUnique2["income"] = df1IncomeNotNullUnique2["income"].apply(
    lambda x: "low" if x <= 31000
              else "mediumlow" if 31000 < x <= 43000
              else "mediumhigh" if 43000 < x <= 54000
              else "high"
)
```

```
#Filtre sur march = yes
maskNotNullMatched = (df1IncomeNotNull2["match2"] == "Yes")
```

[48] ✓ 0.0s

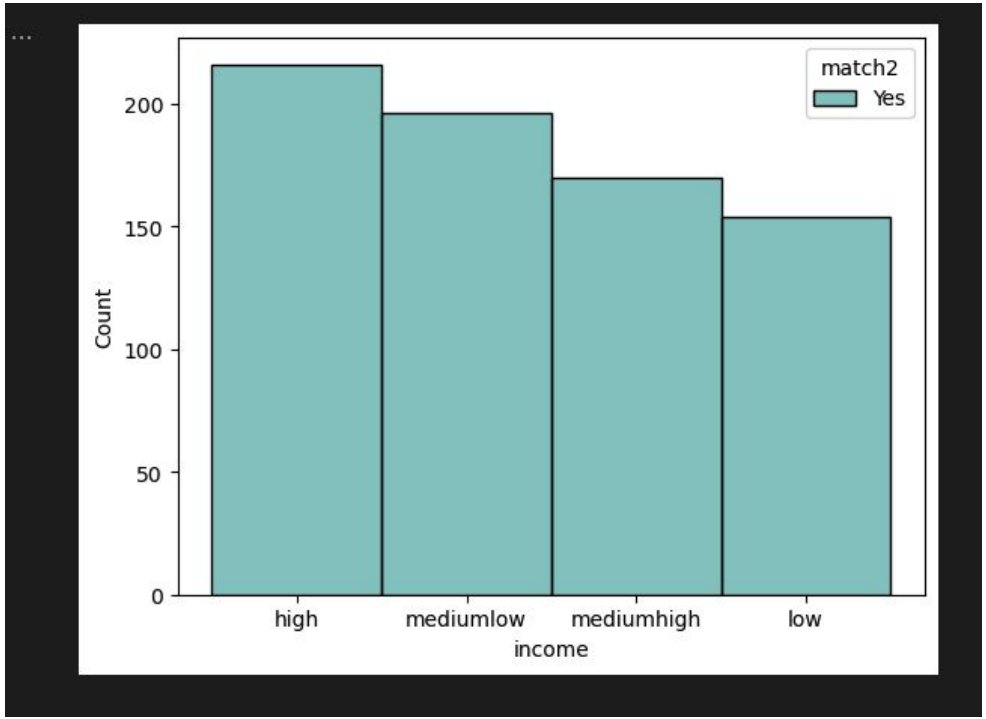
```
#Vérification du filtrage
df1IncomeNotNull2Matched = df1IncomeNotNull2.loc[maskNotNullMatched]
df1IncomeNotNull2Matched["match2"]
```

[49] ✓ 0.0s

#On focus sur les “match = yes” (4 fois moins nombreux) pour une échelle lisible

Corrélation avec le revenu (“income”)

#Création d’un histplot pour voir la corrélation



Conclusion : **les revenus les plus élevés sont plus susceptibles d'obtenir un match que les revenus faibles Les revenus moyens sont intermédiaires**

Corrélation avec les préférences d'activités

#De quoi parle-t-on ?

- des colonnes 50:67 du dataset.

12. How interested are you in the following activities, on a scale of 1-10?
sports: Playing sports/ athletics
tvsports: Watching sports
excersice: Body building/exercising
dining: Dining out
museums: Museums/galleries
art: Art
hiking: Hiking/camping
gaming: Gaming
clubbing: Dancing/clubbing
reading: Reading
tv: Watching TV
theater: Theater
movies: Movies
concerts: Going to concerts
music: Music
shopping: Shopping
yoga: Yoga/meditation

#que cherche t on à savoir :

Quelle relation entre le nombre de match et les activités préférées?

Est ce que cet indicateur est fiable ?

Corrélation avec les préférences d'activités

#Sanity check 1

- vérif de la complétude.
  Seuls 79 lignes contiennent des valeurs NaN, soit 0.94% des données
  Marginal sur l'inférence, la complétude est donc satisfaisante

- Sanity check 2 : pas d'analyse à faire sur la variabilité, les données récoltées sont subjectives et non démographiques .

Indicateur OK !

Corrélation avec les préférences d'activités

#Arrangement des données (filtre : match = yes)

- je n'aime pas ou pas beaucoup = 0 (si score entre 1 et 5),
- j'aime bien ou beaucoup = 1 (si score entre 6 e t 10)

```
for col in df1activities.columns:
    df1activities[col] = df1activities[col].apply(lambda x: 0 if 1 <= x <= 5 else 1)
```

#on crée un dataset de count de chaque match par activité distribué par préférence 0 ou 1

```
results = []
for col in df1activities.columns[:-1]:  # Exclure la colonne "match"
    count_0 = len(df1activities[(df1activities[col] == 0) & (df1activities["match"] == 1)])
    count_1 = len(df1activities[(df1activities[col] == 1) & (df1activities["match"] == 1)])
    results.append({"Activité": col, "0": count_0, "1": count_1})

results_df = pd.DataFrame(results)
```

	Activité	0	1
0	sports	458	922
1	tvsports	842	538
2	exercise	489	891
3	dining	154	1226
4	museums	292	1088
5	art	352	1028
6	hiking	562	818
7	gaming	982	398
8	clubbing	466	914
9	reading	143	1237
10	tv	693	687
11	theater	368	1012
12	movies	131	1249
13	concerts	308	1072
14	music	138	1242
15	shopping	614	766
16	yoga	813	567

Corrélation avec les préférences d'activités

Besoin de matplotlib et de numpy pour répondre au besoin : avec notamment "np arange" pour extraire les valeurs d'une seule colonne et en faire un array affiché en X et "bottom" pour pouvoir empiler 2 colonnes sur un seul histogramme :

```
x = np.arange(len(results_df["Activité"]))

ax.bar(x, results_df["0"], width=bar_width, color="#ff9999", label="N'aime pas (0)")
ax.bar(x, results_df["1"], bottom=results_df["0"], width=bar_width, color="#66b3ff", label="Aime (1)")

ax.set_xticks(x)
ax.set_xticklabels(results_df["Activité"], rotation=45, ha="right")
ax.set_ylabel("Nombre de matchs")
ax.set_title("Impact de la préférence pour une activité sur le nombre de matchs")
ax.legend()

plt.tight_layout()
plt.show()
```

Corrélation avec les préférences d'activités