In [120…

```python
from queries import *
from utils import *

# QUERIES API AIRLBAS
"""
test = getFlightAirlabs('LH2001')
test_response = test.print_request()
print(test_response)
"""

# QUERIES API LUFTHANSA

test = getFlightInfoLuf('LH2057', '2022-07-28')
save_result = test.get_result().json()
print(save_result)


test = getFlightInfoLuf('LH2057', '2022-07-26')
save_result2 = test.get_result().json()
print(save_result2)

"""
test = getFlightInfoByRouteLuf('JNB', 'FRA', '2022-07-2')
test.get_result()
"""
```

```
{'FlightInformation': {'Flights': {'Flight': {'Departure': {'AirportCode': 'HAM', 'Scheduled': {'Date': '2022-0
7-28', 'Time': '12:15'}, 'Actual': {'Date': '2022-07-28', 'Time': '12:35'}, 'Terminal': {'Name': '2', 'Gate':
'A17'}, 'Status': {'Code': 'DP', 'Description': 'Flight Departed'}}, 'Arrival': {'AirportCode': 'MUC', 'Schedul
ed': {'Date': '2022-07-28', 'Time': '13:30'}, 'Estimated': {'Date': '2022-07-28', 'Time': '13:51'}, 'Actual':
{'Date': '2022-07-28', 'Time': '13:51'}, 'Terminal': {'Name': '2', 'Gate': 'G04'}, 'Status': {'Code': 'DL', 'De
scription': 'Flight Delayed'}}, 'MarketingCarrierList': {'MarketingCarrier': {'AirlineID': 'OU', 'FlightNumbe
r': '5457'}}, 'OperatingCarrier': {'AirlineID': 'LH', 'FlightNumber': '2057'}, 'Equipment': {'AircraftCode': '3
20'}, 'Status': {'Code': 'DL', 'Description': 'Flight Delayed'}}}, 'Meta': {'@Version': '1.0.0', 'Link': [{'@Hr
ef': 'https://api.lufthansa.com/v1/operations/customerflightinformation/LH2057/2022-07-28', '@Rel': 'self'},
{'@Href': 'https://api.lufthansa.com/v1/mds-references/airports/{airportCode}', '@Rel': 'related'}]}}}
{'FlightInformation': {'Flights': {'Flight': {'Departure': {'AirportCode': 'HAM', 'Scheduled': {'Date': '2022-0
7-26', 'Time': '12:15'}, 'Actual': {'Date': '2022-07-26', 'Time': '13:04'}, 'Terminal': {'Name': '2', 'Gate':
'C16'}, 'Status': {'Code': 'DP', 'Description': 'Flight Departed'}}, 'Arrival': {'AirportCode': 'MUC', 'Schedul
ed': {'Date': '2022-07-26', 'Time': '13:30'}, 'Actual': {'Date': '2022-07-26', 'Time': '14:10'}, 'Terminal':
{'Name': '2', 'Gate': 'K05'}, 'Status': {'Code': 'LD', 'Description': 'Flight Landed'}}, 'MarketingCarrierLis
t': {'MarketingCarrier': {'AirlineID': 'OU', 'FlightNumber': '5457'}}, 'OperatingCarrier': {'AirlineID': 'LH',
'FlightNumber': '2057'}, 'Equipment': {'AircraftCode': '32N'}, 'Status': {'Code': 'LD', 'Description': 'Flight
Landed'}}}, 'Meta': {'@Version': '1.0.0', 'Link': [{'@Href': 'https://api.lufthansa.com/v1/operations/customerf
lightinformation/LH2057/2022-07-26', '@Rel': 'self'}, {'@Href': 'https://api.lufthansa.com/v1/mds-references/ai
rports/{airportCode}', '@Rel': 'related'}]}}}
```

Out[120]:

```
"\ntest = getFlightInfoByRouteLuf('JNB', 'FRA', '2022-07-2')\ntest.get_result()\n"
```

In [137…

```python
# récupeartion des données à partir du json brut save_result à insérer dans les table sql

# pour la table id_flight
airlineid = save_result['FlightInformation']['Flights']['Flight']['OperatingCarrier']['AirlineID']

flightnumber = save_result['FlightInformation']['Flights']['Flight']['OperatingCarrier']['FlightNumber']

flight_number_concat = airlineid + flightnumber


# pour la table info_departure

time_dep_prev = save_result['FlightInformation']['Flights']['Flight']['Departure']['Scheduled']['Time']
time_dep_real = save_result['FlightInformation']['Flights']['Flight']['Departure']['Actual']['Time']
date_dep_prev = save_result['FlightInformation']['Flights']['Flight']['Departure']['Scheduled']['Date']
date_dep_real = save_result['FlightInformation']['Flights']['Flight']['Departure']['Actual']['Date']
on_time_or_delayed = save_result['FlightInformation']['Flights']['Flight']['Status']['Description']
status = save_result['FlightInformation']['Flights']['Flight']['Departure']['Status']['Description']
airport_code = save_result['FlightInformation']['Flights']['Flight']['Departure']['AirportCode']
terminal_dep = save_result['FlightInformation']['Flights']['Flight']['Departure']['Terminal']['Gate']


# pour la table info_arrival


time_arr_prev = save_result['FlightInformation']['Flights']['Flight']['Arrival']['Scheduled']['Time']
time_arr_real = save_result['FlightInformation']['Flights']['Flight']['Arrival']['Actual']['Time']
date_arr_prev = save_result['FlightInformation']['Flights']['Flight']['Arrival']['Scheduled']['Date']
date_arr_real = save_result['FlightInformation']['Flights']['Flight']['Arrival']['Actual']['Date']
airport_code = save_result['FlightInformation']['Flights']['Flight']['Arrival']['AirportCode']
terminal_arr = save_result['FlightInformation']['Flights']['Flight']['Arrival']['Terminal']['Gate']
```

In [138…

```python
# récupeartion des données à partir du json brut save_result2 à insérer dans les table sql

# pour la table id_flight
airlineid2 = save_result2['FlightInformation']['Flights']['Flight']['OperatingCarrier']['AirlineID']

flightnumber2 = save_result2['FlightInformation']['Flights']['Flight']['OperatingCarrier']['FlightNumber']

flight_number_concat2 = airlineid2 + flightnumber2


# pour la table info_departure

time_dep_prev2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['Scheduled']['Time']
time_dep_real2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['Actual']['Time']
date_dep_prev2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['Scheduled']['Date']
date_dep_real2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['Actual']['Date']
on_time_or_delayed2 = save_result2['FlightInformation']['Flights']['Flight']['Status']['Description']
status2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['Status']['Description']
airport_code2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['AirportCode']
terminal_dep2 = save_result2['FlightInformation']['Flights']['Flight']['Departure']['Terminal']['Gate']


# pour la table info_arrival


time_arr_prev2 = save_result2['FlightInformation']['Flights']['Flight']['Arrival']['Scheduled']['Time']
time_arr_real2 = save_result2['FlightInformation']['Flights']['Flight']['Arrival']['Actual']['Time']
date_arr_prev2 = save_result2['FlightInformation']['Flights']['Flight']['Arrival']['Scheduled']['Date']
date_arr_real2 = save_result2['FlightInformation']['Flights']['Flight']['Arrival']['Actual']['Date']
airport_code2 = save_result2['FlightInformation']['Flights']['Flight']['Arrival']['AirportCode']
terminal_arr2 = save_result2['FlightInformation']['Flights']['Flight']['Arrival']['Terminal']['Gate']
```

In [177…

```python
# init de la BDD sql


from sqlalchemy import Table, Column, Boolean, Time, DateTime, Integer, String, ForeignKey, MetaData, create_en

engine = create_engine('sqlite:///real_time.db', echo=True)
meta = MetaData()
```

In [178…

```python
# creation d'une table date

date = Table(
    'date', meta,
    Column('id_date', Integer, primary_key=True),
    Column('date', String)

)
meta.create_all(engine)
```

```
2022-07-28 13:42:11,103 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:11,104 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("date")
2022-07-28 13:42:11,104 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:11,105 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("date")
2022-07-28 13:42:11,106 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:11,106 INFO sqlalchemy.engine.Engine
CREATE TABLE date (
        id_date INTEGER NOT NULL,
        date VARCHAR,
        PRIMARY KEY (id_date)
)


2022-07-28 13:42:11,107 INFO sqlalchemy.engine.Engine [no key 0.00032s] ()
2022-07-28 13:42:11,109 INFO sqlalchemy.engine.Engine COMMIT
```

In [179…

```python
# creation de la table id_flight


id_flight = Table(
    'id_flight', meta,
    Column('id_flight', Integer, primary_key=True),
    Column('flight_number', String),
    Column('date', Integer, ForeignKey("date.id_date"))
)
meta.create_all(engine)
```

```
2022-07-28 13:42:13,861 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:13,862 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("date")
2022-07-28 13:42:13,862 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:13,862 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("id_flight")
2022-07-28 13:42:13,863 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:13,863 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("id_flight")
2022-07-28 13:42:13,863 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:13,864 INFO sqlalchemy.engine.Engine
CREATE TABLE id_flight (
        id_flight INTEGER NOT NULL,
        flight_number VARCHAR,
        date INTEGER,
        PRIMARY KEY (id_flight),
        FOREIGN KEY(date) REFERENCES date (id_date)
)


2022-07-28 13:42:13,865 INFO sqlalchemy.engine.Engine [no key 0.00038s] ()
2022-07-28 13:42:13,866 INFO sqlalchemy.engine.Engine COMMIT
```

In [180…

```python
# création de la table info_departure


info_departure = Table(
    'info_departure', meta,
    Column('id_info_departure', Integer, primary_key=True),
    Column('time_dep_prev', String),
    Column('time_dep_real', String),
    Column('date_dep_prev', String),
    Column('date_dep_real', String),
    Column('on_time_or_delayed', String),
    Column('status', String),
    Column('airport_code', String),
    Column('terminal_dep', String),
    Column('flight_number', Integer, ForeignKey("id_flight.id_flight")),
    Column('date', Integer, ForeignKey("date.id_date"))


)
meta.create_all(engine)
```

```
2022-07-28 13:42:15,362 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:15,363 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("date")
2022-07-28 13:42:15,363 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:15,364 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("id_flight")
2022-07-28 13:42:15,364 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:15,365 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("info_departure")
2022-07-28 13:42:15,365 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:15,365 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("info_departure")
2022-07-28 13:42:15,365 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:15,366 INFO sqlalchemy.engine.Engine
CREATE TABLE info_departure (
        id_info_departure INTEGER NOT NULL,
        time_dep_prev VARCHAR,
        time_dep_real VARCHAR,
        date_dep_prev VARCHAR,
        date_dep_real VARCHAR,
        on_time_or_delayed VARCHAR,
        status VARCHAR,
        airport_code VARCHAR,
        terminal_dep VARCHAR,
        flight_number INTEGER,
        date INTEGER,
        PRIMARY KEY (id_info_departure),
        FOREIGN KEY(flight_number) REFERENCES id_flight (id_flight),
        FOREIGN KEY(date) REFERENCES date (id_date)
)


2022-07-28 13:42:15,366 INFO sqlalchemy.engine.Engine [no key 0.00026s] ()
2022-07-28 13:42:15,368 INFO sqlalchemy.engine.Engine COMMIT
```

In [181…

```python
# création de la table info_arrival


info_arrival = Table(
    'info_arrival', meta,
    Column('id_info_departure', Integer, primary_key=True),
    Column('time_arr_prev', String),
    Column('time_arr_real', String),
    Column('date_arr_prev', String),
    Column('date_arr_real', String),
    Column('airport_code', String),
    Column('terminal_arr', String),
    Column('flight_number', Integer, ForeignKey("id_flight.id_flight")),
    Column('date', Integer, ForeignKey("date.id_date"))

)
meta.create_all(engine)
```

```
2022-07-28 13:42:17,281 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:17,282 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("date")
2022-07-28 13:42:17,282 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:17,283 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("id_flight")
2022-07-28 13:42:17,283 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:17,284 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("info_departure")
2022-07-28 13:42:17,284 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:17,285 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("info_arrival")
2022-07-28 13:42:17,286 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:17,286 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("info_arrival")
2022-07-28 13:42:17,286 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-07-28 13:42:17,287 INFO sqlalchemy.engine.Engine
CREATE TABLE info_arrival (
        id_info_departure INTEGER NOT NULL,
        time_arr_prev VARCHAR,
        time_arr_real VARCHAR,
        date_arr_prev VARCHAR,
        date_arr_real VARCHAR,
        airport_code VARCHAR,
        terminal_arr VARCHAR,
        flight_number INTEGER,
        date INTEGER,
        PRIMARY KEY (id_info_departure),
        FOREIGN KEY(flight_number) REFERENCES id_flight (id_flight),
        FOREIGN KEY(date) REFERENCES date (id_date)
)


2022-07-28 13:42:17,288 INFO sqlalchemy.engine.Engine [no key 0.00033s] ()
2022-07-28 13:42:17,289 INFO sqlalchemy.engine.Engine COMMIT
```

In [182…

```python
# insertion des données à partir du json dans la table id_flight

values_id_flight = [(1, flight_number_concat, 1),(2, flight_number_concat2, 2)]

with engine.connect() as connection:
    # début de la transaction
    with connection.begin() as transaction:
        # on tente d'éxécuter une transaction
        try:
            # On indique le format d'un tuple de cette table
            markers = ','.join('?' * len(values_id_flight[0]))

            # On utilise le langage SQL en format texte où markers est le format d'un tuple
            ins = 'INSERT OR REPLACE INTO {tablename} VALUES ({markers})'

            # On précise ce format particulier grâce à la fonction membre format
            ins = ins.format(tablename=id_flight.name, markers=markers)

            # Enfin on peut utiliser les tuples créés en éxécutant la commande SQL
            connection.execute(ins, values_id_flight)
        # si la transaction échoue
        except:
            transaction.rollback()
            raise
        # si la transaction réussit
        else:
            transaction.commit()
```

```
2022-07-28 13:42:20,409 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:20,410 INFO sqlalchemy.engine.Engine INSERT OR REPLACE INTO id_flight VALUES (?,?,?)
2022-07-28 13:42:20,410 INFO sqlalchemy.engine.Engine [raw sql] [(1, 'LH2057', 1), (2, 'LH2057', 2)]
2022-07-28 13:42:20,411 INFO sqlalchemy.engine.Engine COMMIT
```

In [183…

```python
# essai requête sql à partir des données que l'on a insérées dans la table id_flight

with engine.connect() as connection:
    results = connection.execute("SELECT * FROM id_flight;")
    print(results.fetchall())
```

```
2022-07-28 13:42:22,333 INFO sqlalchemy.engine.Engine SELECT * FROM id_flight;
2022-07-28 13:42:22,333 INFO sqlalchemy.engine.Engine [raw sql] ()
[(1, 'LH2057', 1), (2, 'LH2057', 2)]
```

In [184…

```python
# insertion des données à partir du json dans la table info_departure

values_info_departure = [(1, time_dep_prev, time_dep_real, date_dep_prev, date_dep_real, on_time_or_delayed, st

with engine.connect() as connection:
    # début de la transaction
    with connection.begin() as transaction:
        # on tente d'éxécuter une transaction
        try:
            # On indique le format d'un tuple de cette table
            markers = ','.join('?' * len(values_info_departure[0]))

            # On utilise le langage SQL en format texte où markers est le format d'un tuple
            ins = 'INSERT OR REPLACE INTO {tablename} VALUES ({markers})'

            # On précise ce format particulier grâce à la fonction membre format
            ins = ins.format(tablename=info_departure.name, markers=markers)

            # Enfin on peut utiliser les tuples créés en éxécutant la commande SQL
            connection.execute(ins, values_info_departure)
        # si la transaction échoue
        except:
            transaction.rollback()
            raise
        # si la transaction réussit
        else:
            transaction.commit()
```

```
2022-07-28 13:42:23,335 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:23,336 INFO sqlalchemy.engine.Engine INSERT OR REPLACE INTO info_departure VALUES
(?,?,?,?,?,?,?,?,?,?,?)
2022-07-28 13:42:23,337 INFO sqlalchemy.engine.Engine [raw sql] [(1, '12:15', '12:35', '2022-07-28', '2022-07-2
8', 'Flight Delayed', 'Flight Departed', 'MUC', 'A17', 1, 1), (2, '12:15', '13:04', '2022-07-26', '2022-07-26',
'Flight Landed', 'Flight Departed', 'MUC', 'C16', 1, 2)]
2022-07-28 13:42:23,337 INFO sqlalchemy.engine.Engine COMMIT
```

In [185…

```python
# essai requete sur table info_departure

with engine.connect() as connection:
    results = connection.execute("SELECT * FROM info_departure;")
    print(results.fetchall())
```

```
2022-07-28 13:42:24,779 INFO sqlalchemy.engine.Engine SELECT * FROM info_departure;
2022-07-28 13:42:24,780 INFO sqlalchemy.engine.Engine [raw sql] ()
[(1, '12:15', '12:35', '2022-07-28', '2022-07-28', 'Flight Delayed', 'Flight Departed', 'MUC', 'A17', 1, 1),
(2, '12:15', '13:04', '2022-07-26', '2022-07-26', 'Flight Landed', 'Flight Departed', 'MUC', 'C16', 1, 2)]
```

In [186…

```python
# insertion des données à partir du json dans la table info_arrival



values_info_arrival = [(1, time_arr_prev, time_arr_real, date_arr_prev, date_arr_real, airport_code, terminal_a

with engine.connect() as connection:
    # début de la transaction
    with connection.begin() as transaction:
        # on tente d'éxécuter une transaction
        try:
            # On indique le format d'un tuple de cette table
            markers = ','.join('?' * len(values_info_arrival[0]))

            # On utilise le langage SQL en format texte où markers est le format d'un tuple
            ins = 'INSERT OR REPLACE INTO {tablename} VALUES ({markers})'

            # On précise ce format particulier grâce à la fonction membre format
            ins = ins.format(tablename=info_arrival.name, markers=markers)

            # Enfin on peut utiliser les tuples créés en éxécutant la commande SQL
            connection.execute(ins, values_info_arrival)
        # si la transaction échoue
        except:
            transaction.rollback()
            raise
        # si la transaction réussit
        else:
            transaction.commit()
```

```
2022-07-28 13:42:25,837 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:25,838 INFO sqlalchemy.engine.Engine INSERT OR REPLACE INTO info_arrival VALUES
(?,?,?,?,?,?,?,?,?)
2022-07-28 13:42:25,838 INFO sqlalchemy.engine.Engine [raw sql] [(1, '13:30', '13:51', '2022-07-28', '2022-07-2
8', 'MUC', 'G04', 1, 1), (2, '13:30', '14:10', '2022-07-26', '2022-07-26', 'MUC', 'K05', 1, 2)]
2022-07-28 13:42:25,839 INFO sqlalchemy.engine.Engine COMMIT
```

In [187…

```python
# essai requete sur table info_arrival

with engine.connect() as connection:
    results = connection.execute("SELECT * FROM info_arrival;")
    print(results.fetchall())
```

```
2022-07-28 13:42:27,757 INFO sqlalchemy.engine.Engine SELECT * FROM info_arrival;
2022-07-28 13:42:27,758 INFO sqlalchemy.engine.Engine [raw sql] ()
[(1, '13:30', '13:51', '2022-07-28', '2022-07-28', 'MUC', 'G04', 1, 1), (2, '13:30', '14:10', '2022-07-26', '20
22-07-26', 'MUC', 'K05', 1, 2)]
```

In [188…

```python
# insertion donnees dans la table date


values_date = [(1, '2022-07-28'),(2, '2022-07-26')]

with engine.connect() as connection:
    # début de la transaction
    with connection.begin() as transaction:
        # on tente d'éxécuter une transaction
        try:
            # On indique le format d'un tuple de cette table
            markers = ','.join('?' * len(values_date[0]))

            # On utilise le langage SQL en format texte où markers est le format d'un tuple
            ins = 'INSERT OR REPLACE INTO {tablename} VALUES ({markers})'

            # On précise ce format particulier grâce à la fonction membre format
            ins = ins.format(tablename=date.name, markers=markers)

            # Enfin on peut utiliser les tuples créés en éxécutant la commande SQL
            connection.execute(ins, values_date)
        # si la transaction échoue
        except:
            transaction.rollback()
            raise
        # si la transaction réussit
        else:
            transaction.commit()
```

```
2022-07-28 13:42:29,101 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-07-28 13:42:29,102 INFO sqlalchemy.engine.Engine INSERT OR REPLACE INTO date VALUES (?,?)
2022-07-28 13:42:29,102 INFO sqlalchemy.engine.Engine [raw sql] [(1, '2022-07-28'), (2, '2022-07-26')]
2022-07-28 13:42:29,103 INFO sqlalchemy.engine.Engine COMMIT
```

In [189…

```python
# essai requete sur table date

with engine.connect() as connection:
    results = connection.execute("SELECT * FROM date;")
    print(results.fetchall())
```

```
2022-07-28 13:42:30,830 INFO sqlalchemy.engine.Engine SELECT * FROM date;
2022-07-28 13:42:30,830 INFO sqlalchemy.engine.Engine [raw sql] ()
[(1, '2022-07-28'), (2, '2022-07-26')]
```

In [224…

```python
# essai requete sql entre les différentes tables pour récupérer les info concernant l'aéroport de départ en foc

with engine.connect() as connection:
    results = connection.execute("SELECT * FROM (SELECT * FROM id_flight WHERE flight_number = 'LH2057') AS A L
    print(results.fetchall())
```

```
2022-07-28 14:30:09,453 INFO sqlalchemy.engine.Engine SELECT * FROM (SELECT * FROM id_flight WHERE flight_numbe
r = 'LH2057') AS A LEFT JOIN (SELECT * FROM info_departure WHERE info_departure.date_dep_prev = '2022-07-28') A
S B ON A.id_flight = B.flight_number WHERE B.flight_number IS NOT NULL;
2022-07-28 14:30:09,453 INFO sqlalchemy.engine.Engine [raw sql] ()
[(1, 'LH2057', 1, 1, '12:15', '12:35', '2022-07-28', '2022-07-28', 'Flight Delayed', 'Flight Departed', 'MUC',
'A17', 1, 1)]
```

In [231…

```python
# essai requete sql entre les différentes tables pour récupérer les info concernant l'aéroport de départ en foc

with engine.connect() as connection:
    results = connection.execute("SELECT * FROM (SELECT * FROM id_flight WHERE flight_number = 'LH2057') AS A L
    print(results.fetchall())
```

```
2022-07-28 14:44:07,400 INFO sqlalchemy.engine.Engine SELECT * FROM (SELECT * FROM id_flight WHERE flight_numbe
r = 'LH2057') AS A LEFT JOIN (SELECT * FROM info_arrival WHERE info_arrival.date_arr_prev = '2022-07-28') AS B
ON A.id_flight = B.flight_number WHERE B.flight_number IS NOT NULL;
2022-07-28 14:44:07,401 INFO sqlalchemy.engine.Engine [raw sql] ()
[(1, 'LH2057', 1, 1, '13:30', '13:51', '2022-07-28', '2022-07-28', 'MUC', 'G04', 1, 1)]
```

In [245…

```python
# essai sql to dataframe avec la table id_flight uniquement

import pandas as pd
import sqlite3

conn = sqlite3.connect('real_time.db')

sql_query = pd.read_sql_query ('''
                               SELECT
                               *
                               FROM id_flight
                               ''', conn)

df = pd.DataFrame(sql_query)
df.head()
```

Out[245]:

	id_flight	flight_number	date
0	1	LH2057	1
1	2	LH2057	2

In [244…

```python
# essai requete sql entre les différentes tables pour récupérer les info concernant l'aéroport de départ en foc
# essai de cette requete sql to dataframe

conn = sqlite3.connect('real_time.db')

sql_query = pd.read_sql_query ('''
                               SELECT * FROM (SELECT * FROM id_flight WHERE flight_number = 'LH2057') AS A LEFT
                               ''', conn)

df = pd.DataFrame(sql_query)
df.head()
```

Out[244]:

	id_flight	flight_number	date	id_info_departure	time_arr_prev	time_arr_real	date_arr_prev	date_arr_real	airport_code	terminal_arr	flig
0	1	LH2057	1	1	13:30	13:51	2022-07-28	2022-07-28	MUC	G04	