Universidad del Rosario
Escuela de Ingeniería, Ciencia y Tecnología
Matemáticas Aplicadas y Ciencias de la Computación
Ingeniería de datos
Nelson Sánchez Sánchez
Alejandro Vega, Juan Diego Martínez, Laura Hernández
Semestre 1, 2022

Introducción

Muchas veces es complicado conseguir información relacionada con vuelos aéreos específicos, como hallar los datos del avión o de la aerolínea, de tal manera, se decidió crear el proyecto, el cual sirve para conseguir toda la información relacionada con la aerolínea, el avión, el aeropuerto, el vuelo, entre otros; en un solo lugar y de forma sencilla.

Fase de análisis

En esta fase se nombrará a detalle cada requerimiento de información que el sistema debe cumplir para ser funcional y lograr el objetivo del proyecto.

El sistema debe ser capaz de dar información la siguiente información, con relación a los aviones, se debe poder hallar la marca del avión, el modelo y la aerolínea a la que pertenece, por otra parte, en relación a los pasajeros, debe proporcionar el nombre, apellido, la nacionalidad y la fecha de nacimiento de cada uno, además, también debe ser posible obtener el número de vuelo y del ticket, la fecha de salida y de llegada y el aeropuerto del vuelo de los pasajeros.

Fase de diseño

En esta fase se nombrará las tablas de la base de datos, sus columnas, los tamaños, los tipos de datos y las restricciones de dominio y de integridad.

Antes de iniciar con las tablas, cabe destacar que para las columnas de tipo varchar, todas tienen un límite de caracteres de 80, por lo tanto, cuando se mencione este tipo de dato, este será su límite. Como primera tabla tenemos marcas, la cual tiene como columnas id_marca que es de tipo integer y es el primary key, y nombre que es de tipo varchar.

Luego está la tabla pasajeros, que tiene como columnas id_pasajero de tipo integer y es la primary key, nombre, apellido y nacionalidad que todas son de tipo varchar, y fecha_nacimiento que es de tipo date.

Sigue la tabla modelos, que tiene a las columnas id_modelo de tipo integer y es la primary key, nombre de tipo varchar, y marca de tipo integer, además, marca es un foreign key que referencia la tabla marcas y la columna id_marca.

Ahora está la tabla aerolíneas, con las columnas id_aerolinea de tipo integer y es la primary key, y nombre de tipo varchar.

Luego se tiene la tabla aeropuertos que tiene como columnas id_aeropuerto de tipo integer y es la primary key, y nombre y ciudad de tipo varchar.

Sigue la tabla aviones que tiene las siguientes columnas, id_avion de tipo integer y es la primary key, id_modelo e id_aerolinea ambas de tipo integer, además, las dos son foreign keys, id_aerolinea referencia a la tabla aerolínea y la columna id_aerolinea, e id_modelo referencia la tabla modelos y la columna id_modelo.

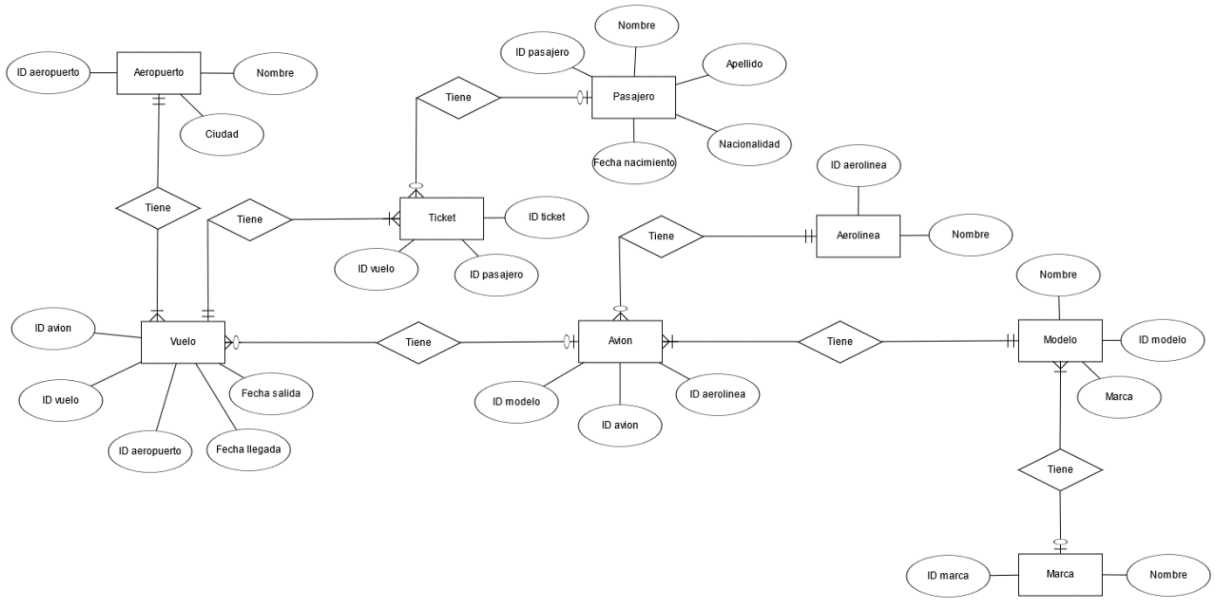
Ahora viene la tabla vuelo, con las columnas id_vuelo de tipo varchar y es la primary key, id_avion e id_aeropuerto ambas de tipo integer, y fecha_llegada y fecha_salida ambas de tipo timestamp, además, id_avion y id_aeropuerto son foreign keys, id_avion referencia a

la tabla aviones y la columna id_avion, e id_aeropuerto referencia la tabla aeropuertos y la Columba id_aeropuerto.

Por último, esta la tabla tickets, con las columnas id_ticket de tipo integer y es la primary key, id_pasajero de tipo integer, e id_vuelo de tipo varchar, además, id_avion e id_aeropuerto son foreign keys, id_avion referencia la tabla aviones y la columna id_avion, e id_aeropuerto referencia la tabla aeropuertos y la columna id_aeropuerto.

Diseño conceptual

Modelo Entidad/Relación:

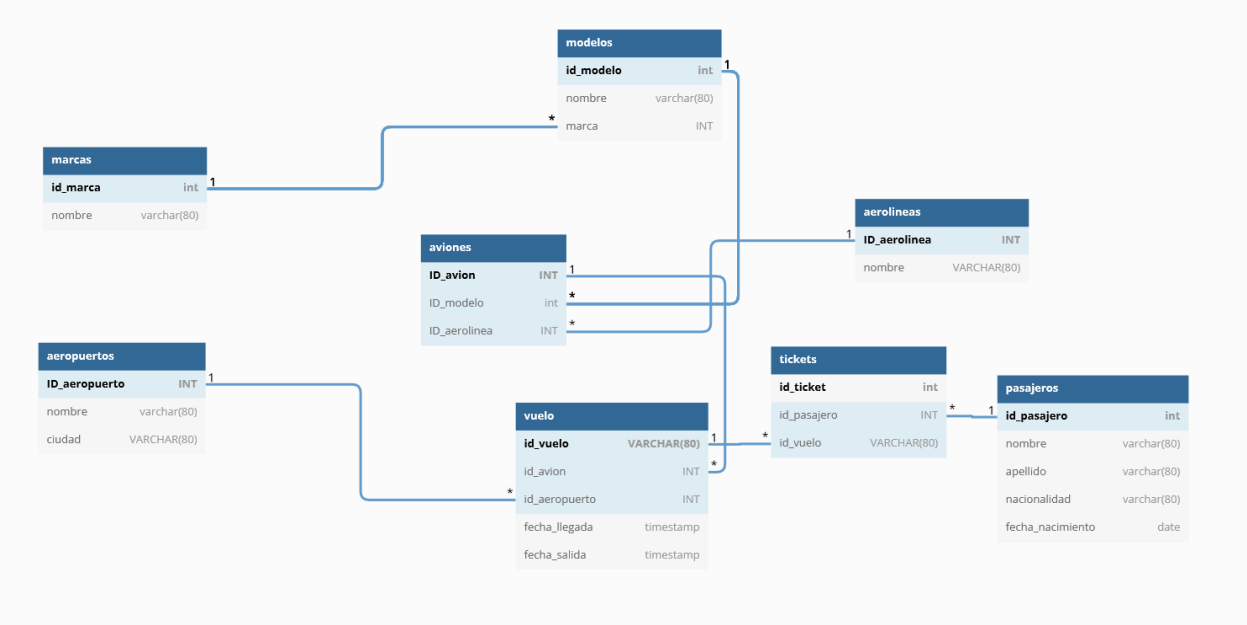


En el modelo E/R anterior se puede ver las tablas descritas anteriormente y sus columnas, en este caso, los rectángulos son las tablas, los óvalos son las columnas, y los rombos son la relación que tienen entre si las tablas, también se puede ver qué tipo de relación tienen, como se presencia, la tabla vuelo y aeropuerto tienen una relación de muchos vuelos obligatoriamente a un aeropuerto, la tabla vuelo y ticket tienen una relaciona de muchos tickets obligatoriamente a un vuelo, ticket y pasajero tienen una relación de muchos tickets

opcionalmente a un pasajero, vuelo y avión tienen una relación de muchos vuelos opcionalmente a un avión, avión y aerolínea tiene una relación de muchos aviones opcionalmente a una aerolínea, avión y modelo tienen una relación de muchos aviones obligatoriamente a un modelo y modelo y marca tienen una relación de muchos modelos obligatoriamente a una marca.

Diseño Lógico

Esquema de base de datos relacional:



En este esquema se puede presenciar el diagrama E/R adaptado a una base de datos relacional, también se pueden ver las tablas con sus nombres, sus columnas con su tipo de dato, además, se puede ver que las columnas con el nombre en negrita son las primary keys, y, por otra parte, también se pueden apreciar las relaciones entre las tablas, en este caso, el asterisco (*) significa muchos.

Fase de construcción

En esta fase se busca cambiar de un modelo de datos abstracto a la implementación de la base de datos incluyendo todo lo relacionado con el DML y DDL.

DDL:

```
CREATE TABLE marcas(
    id_marca int PRIMARY KEY,
    nombre varchar(80)
);

CREATE TABLE pasajeros(
    id_pasajero int PRIMARY KEY,
    nombre varchar(80),
    apellido varchar(80),
    nacionalidad varchar(80),
    fecha_nacimiento date
);

CREATE TABLE aerolineas(
    ID_aerolinea INT PRIMARY KEY,
    nombre VARCHAR(80)
);

CREATE TABLE aeropuertos(
    ID_aeropuerto INT PRIMARY KEY,
    nombre varchar(80),
    ciudad VARCHAR(80)
);
```

```
CREATE TABLE modelos(
    id_modelo int PRIMARY KEY,
    nombre varchar(80),
    marca INT,
    FOREIGN KEY (marca) REFERENCES marcas(id_marca)
);

CREATE TABLE aviones(
    ID_avion INT PRIMARY KEY,
    ID_modelo int,
    ID_aerolinea INT,
    FOREIGN KEY(ID_aerolinea) REFERENCES aerolineas(ID_aerolinea),
    FOREIGN KEY(ID_modelo) REFERENCES modelos(id_modelo)
);
CREATE TABLE vuelo(
    id_vuelo VARCHAR(80) PRIMARY KEY,
    id_avion INT,
    id_aeropuerto INT,
    fecha_llegada timestamp,
    fecha_salida TIMESTAMP,
    FOREIGN KEY (id_avion) REFERENCES aviones(ID_avion),
    FOREIGN KEY (id_aeropuerto) REFERENCES aeropuertos(ID_aeropuerto)
);

CREATE TABLE tickets(
    id_ticket int PRIMARY KEY,
    id_pasajero INT,
    id_vuelo VARCHAR(80),
    FOREIGN KEY (id_pasajero) REFERENCES pasajeros(id_pasajero),
    FOREIGN KEY (id_vuelo) REFERENCES vuelo(id_vuelo)
);
```

Como se puede ver, ya están creadas todas las tablas con sus columnas, los tipos de datos, y las restricciones, además, estas también están en 3NF, ya que en ninguna columna pueden haber duplicados ni múltiples valores, cada tabla solo tiene un primary key, y tampoco hay dependencia transitiva.

DML:

Para la inserción de información se llevó a cabo mediante archivos csv de la siguiente manera:

```
COPY aeropuertos FROM '/Applications/PostgreSQL 14/Aeropuerto.csv' HEADER CSV DELIMITER ';';
COPY marcas FROM '/Applications/PostgreSQL 14/marcas.csv' HEADER CSV DELIMITER ';';
COPY modelos FROM '/Applications/PostgreSQL 14/modelos.csv' HEADER CSV DELIMITER ';';
COPY aerolineas FROM '/Applications/PostgreSQL 14/aerolineas.csv' HEADER CSV DELIMITER ';';
COPY pasajeros FROM '/Applications/PostgreSQL 14/pasajeros.csv' HEADER CSV DELIMITER ';';
COPY aviones FROM '/Applications/PostgreSQL 14/aviones.csv' HEADER CSV DELIMITER ';';
COPY vuelo FROM '/Applications/PostgreSQL 14/vuelos.csv' HEADER CSV DELIMITER ';';
COPY tickets FROM '/Applications/PostgreSQL 14/tickets.csv' HEADER CSV DELIMITER ';';
```

Utilizando estas inserciones se realizaron las siguientes roles, usuarios y consultas:

Roles:

```
CREATE ROLE Tecnico_administrativo with login;
GRANT SELECT,INSERT,DELETE,UPDATE ON ALL TABLES IN SCHEMA PUBLIC TO Tecnico_administrativo with GRANT OPTION;
```

```
CREATE ROLE Tecnico_Operaciones_aeroportuarias with login;
GRANT SELECT,INSERT,DELETE,UPDATE on public.tickets,public.vuelo,public.pasajeros to
Tecnico_Operaciones_aeroportuarias;

CREATE ROLE Auxiliar_tierra with login;
GRANT SELECT ON public.tickets,public.vuelo,public.pasajeros to Auxiliar_tierra;
```

Usuarios:

```
CREATE USER TOA1 with login PASSWORD 'TOA1' IN ROLE Tecnico_Operaciones_aeroportuarias;

CREATE USER Tecni_admi1 PASSWORD 'tecadmi' IN ROLE Tecnico_administrativo;

CREATE USER aux1 PASSWORD 'aux1' IN ROLE Auxiliar_tierra;
```

Consultas:

Todos los modelos de los aviones:

```
CREATE VIEW Modelos_aviones AS
SELECT F.id_avion,X.nombre marca, F.referencia, F.id_modelo,F.aerolinea
FROM
(SELECT M.id_avion,A.nombre aerolinea,M.nombre referencia,M.marca id_marca, M.id_modelo
FROM
(SELECT *
FROM modelos M
NATURAL JOIN aviones A) M
INNER JOIN
aerolineas A
on A.id_aerolinea = M.id_aerolinea) F
INNER JOIN marcas X
ON F.id marca = X.id marca;
```

Todos los nombres de las aerolíneas:

```
CREATE VIEW nom_aerolineas
AS
SELECT nombre FROM Aerolineas;
```

Datos del pasajero, tickets de vuelo y aeropuerto:

```
CREATE VIEW Pasajeros_vuelos AS

SELECT p.id_pasajero,p.id_ticket, P.nombre nombre_pasajero, P.apellido apellido_pasajero, P.id_vuelo,
V.id_aeropuerto, V.nombre aeropuerto,V.fecha_salida, V.fecha_llegada  --V.nombre,V.
FROM
(SELECT *
FROM vuelo
NATURAL JOIN aeropuertos) AS V
INNER JOIN
(SELECT *
FROM pasajeros
NATURAL JOIN tickets) AS P
ON V.id_vuelo = P.id_vuelo;
```

Información de vuelo de los pasajeros y el id, nombre, apellido y nacionalidad:

```
CREATE VIEW nacionalidad_pasajeros AS
SELECT p.id_pasajero,p.id_ticket, P.nombre nombre_pasajero, P.apellido apellido_pasajero, P.nacionalidad
FROM
(SELECT *
FROM vuelo
NATURAL JOIN aeropuertos) AS V
INNER JOIN
(SELECT *
FROM pasajeros
NATURAL JOIN tickets) AS P
ON V.id_vuelo = P.id_vuelo;
```

La aerolínea con más vuelos:

```
SELECT P.nombre, Count(*), dense_rank() over(order by count(nombre)
        desc) as s_rank
FROM
(SELECT *
FROM aerolineas A
INNER JOIN aviones V
ON A.id_aerolinea = V.id_aerolinea) AS P
INNER JOIN vuelo F
on P.id_avion = F.id_avion
GROUP BY P.nombre;
```

Pasajeros que más han viajado:

```
SELECT id_pasajero,nombre_pasajero, count(*) num_tickets, DENSE_RANK() OVER (ORDER BY count(nombre_pasajero)
DESC) as rank
FROM Pasajeros_vuelos
GROUP BY  id_pasajero,nombre_pasajero
```

Jerarquía del número de pasajeros por nacionalidad:

```
SELECT nacionalidad, count(*),  DENSE_RANK() OVER (ORDER BY count(nacionalidad) DESC) as rank
FROM nacionalidad_pasajeros
GROUP BY nacionalidad
```

Almacenar la información de los cambios efectuados al número de vuelo de un pasajero y la fecha en la que se realizó el cambio:

```
drop table if exists tr_1;
create table tr_1(
   id_ticket int PRIMARY KEY,
   id_pasajero INT,
   id_vuelo_anterior VARCHAR(80),
   id_vuelo_nuevo VARCHAR(80),
   fecha_cambio timestamp
);

drop function if exists f_tr_1();
CREATE FUNCTION  f_tr_1() returns Trigger
as
$$
begin
insert into tr_1(id_ticket,id_pasajero, id_vuelo_anterior,id_vuelo_nuevo,fecha_cambio)
   values(old.id_ticket, old.id_pasajero, old.id_vuelo,new.id_vuelo, now());
return new;
End
$$
Language plpgsql;

drop trigger if exists Tr_1 on  tickets;
create trigger Tr_1 before update on tickets
for each row
execute procedure f_tr_1();

select *
from tickets;
update tickets set id_vuelo='WN4239' where id_pasajero=2;

SELECT *
FROM tr_1
```

Edad de los pasajeros:

```
CREATE OR REPLACE FUNCTION  edad_viajero (id_viajero integer)
RETURNS integer
AS $$
    DECLARE edad INT;
BEGIN
edad:=(SELECT EXTRACT (YEAR FROM AGE(fecha_nacimiento) )
FROM pasajeros
WHERE id_pasajero = id_viajero);

return edad;
end;
$$
language 'plpgsql';

SELECT * FROM edad_viajero(1);
```

Almacenar la información de los cambios efectuados a la hora de llegada de los vuelos y evidenciar el retraso desde la última fecha en horas:

```
drop table if exists tr_2;
create table tr_2(
   id_vuelo VARCHAR(80),
   id_avion INT,
   id_aeropuerto INT,
   fecha_llegada timestamp,
   anterior_fecha_llegada timestamp,
   retraso int

);
 drop trigger if exists Tr_2 on  vuelo;
drop function if exists f_tr_2();
CREATE FUNCTION  f_tr_2() returns Trigger
as
$$
begin
insert into tr_2(id_vuelo,id_avion , id_aeropuerto,fecha_llegada,anterior_fecha_llegada,retraso)
   values(old.id_vuelo, old.id_avion, old.id_aeropuerto,new.fecha_llegada,old.fecha_llegada,EXTRACT('hour'
FROM new.fecha_llegada)-EXTRACT('hour' FROM old.fecha_llegada));
return new;
End
$$
Language plpgsql;


create trigger Tr_2 before update on vuelo
for each row
execute procedure f_tr_2();


update vuelo set fecha_llegada='2022-04-06 23:30:00' where id_vuelo='WN4239';


select * from tr_2;
```

Cantidad de vuelos que llegan por hora:

```
SELECT EXTRACT('hour' FROM fecha_llegada ) hora_llegada, COUNT(*) total
FROM Vuelo
GROUP BY EXTRACT('hour' FROM fecha_llegada )
ORDER BY hora_llegada ASC;
```

Cantidad de tickets por vuelo:

```
SELECT id_vuelo, COUNT(*) num_tickets
FROM tickets
GROUP BY id_vuelo;
```

Cantidad de aviones por aerolínea:

```
SELECT nombre, COUNT(*)
FROM aviones
NATURAL JOIN aerolineas
GROUP BY nombre;
```

Además, también se realizó la conexión con Python de la siguiente manera:

```
import psycopg2


class Connection:

    def __init__(self):
        self.connection = None

    def openConnection(self):
        try:
            self.connection = psycopg2.connect(user="postgres",
                                               password="Juanita5544",
                                               database="proyecto_ing",
                                               host="localhost",
                                               port="5432")
        except Exception as e:
            print(e)

    def closeConnection(self):
        self.connection.close()
```

Con el objetivo de utilizar dash de la siguiente forma:

```
import dash
from dash import dcc
from dash import html
import plotly.express as px
import pandas as pd
from Connection_p import Connection
import plotly.graph_objects as go
import proyecto  as sql

app = dash.Dash(__name__)

con = Connection()
con.openConnection()
# queries
query_1 = pd.read_sql_query(sql.vuelo_tickets(), con.connection)
query_2 = pd.read_sql_query(sql.vuelo_hora(), con.connection)
query_3 = pd.read_sql_query(sql.num_aviones(), con.connection)
query_4 = pd.read_sql_query(sql.nacionalidad(), con.connection)
query_5 = pd.read_sql_query(sql.destino(), con.connection)
query_6 = pd.read_sql_query(sql.modelos(), con.connection)
query_7 = pd.read_sql_query(sql.nacionalidad_asc(), con.connection)
con.closeConnection()
```

```
#Dataframes
aeropuertos = pd.DataFrame(query_1, columns=["id_vuelo","num_tickets"])
vuelos_hora= pd.DataFrame(query_2, columns=["hora_llegada", "total"])
num_aviones= pd.DataFrame(query_3, columns=["aerolinea", "num_aviones"])
nacionalidad= pd.DataFrame(query_4, columns=["nacionalidad", "total"])
nacionalidad_asc= pd.DataFrame(query_7, columns=["nacionalidad", "total"])
destino= pd.DataFrame(query_5, columns=["ciudad", "total"])
modelos= pd.DataFrame(query_6, columns=["nombre", "total"])
print(nacionalidad)

#Crear las gráficas
fig=px.bar(aeropuertos,x="id_vuelo",y="num_tickets",title="Número de tiquetes por vuelo")
fig_2=px.line(vuelos_hora,x="hora_llegada",y="total",title="Número de vuelos que llegan por hora")
fig_3=px.bar(num_aviones,x="aerolinea",y="num_aviones",title="Número de aviones por aerolinea")
fig_5=px.bar(nacionalidad, x='nacionalidad',y="total",title="Flujo de turistas por nacionalidad")
#fig_4 = px.scatter(nacionalidad, x="nacionalidad", y="total",size="total",title="Nacionalidad con mayor flujo de turistas")
fig_4 = px.scatter_geo(destino, locations="ciudad", size="total",projection="natural earth",title="Mapa mundi")
fig_6=px.pie(modelos, values='total', names='nombre',title="Modelos de aviones")

fig_7=px.scatter_geo(destino, locations="ciudad", size="total",projection="natural earth",title="Mapa mundi")


#actualizar los estilos de las gráficas
fig.update_traces(marker_color='rgb(199, 0, 57)', marker_line_color='rgb(144, 12, 63)',
                  marker_line_width=1.5, opacity=0.6)
fig_3.update_traces(marker_color='rgb(199, 0, 57)', marker_line_color='rgb(144, 12, 63)',
                  marker_line_width=1.5, opacity=0.6)

fig.update_layout(paper_bgcolor = "#D7EDFA ", font = {'color': "black", 'family': "Times New Roman"})
fig_2.update_layout(paper_bgcolor = "#BDD5E4 ", font = {'color': "black", 'family': "Times New Roman"})
fig_3.update_layout(paper_bgcolor = "#D7EDFA", font = {'color': "black", 'family': "Times New Roman"})
fig_4.update_layout(paper_bgcolor = "#D7EDFA", font = {'color': "black", 'family': "Times New Roman"})
fig_5.update_layout(paper_bgcolor = "#D7EDFA", font = {'color': "black", 'family': "Times New Roman"})
fig_6.update_layout(paper_bgcolor = "#BDD5E4", font = {'color': "black", 'family': "Times New Roman"})
app.layout = html.Div(children=[
            html.Div('Base de datos aérea',style={ 'family': "Times New Roman", 'fontSize': 50,'text-align':"center"}
            ),

            dcc.Graph(
            id='mapa',
            figure=fig_7,
            ),

            dcc.Graph(
            id='modelos',
            figure=fig_6,
            ),


            dcc.Graph(
            id='tiquetes',
            figure=fig,
            ),

            dcc.Graph(
            id='llegadas',
            figure=fig_2,
            ),

            dcc.Graph(
            id='aviones',
            figure=fig_3,
            ),dcc.Dropdown( id = 'menu',
            options = [
                {'label':'Top 10 nacionalidades ascendentemente', 'value':'asc' },
                {'label': 'Top 10 nacionalidades descendentemente', 'value':'desc'},

                ],
            style={'backgroundColor':"#e4e4e6;"}
            ),
            html.Button('Ver', id='b_ver', n_clicks=0, style={'backgroundColor':'#D7EDFA'}),
            dcc.Graph(
            id='nacionalidad',
            ),
                                ],
        style={'backgroundColor':'#BDD5E4'})
@app.callback(
```

```
@app.callback(
        dash.dependencies.Output("nacionalidad","figure"),
        dash.dependencies.Input("menu","value"),
        dash.dependencies.Input('b_ver', 'n_clicks'),

    )
def update_graph(value,n_clicks):
    df=nacionalidad
    if n_clicks!=0:
        if value is not None:
            if value=="asc":
                df=nacionalidad_asc

            print(df)

    fig_5=px.bar(df, x='nacionalidad',y="total",title="Flujo de turistas por nacionalidad")
    fig_5.update_layout(paper_bgcolor = "#BDD5E4", font = {'color': "black", 'family': "Times New Roman"})
    fig_5.update_traces(marker_color='rgb(199, 0, 57)', marker_line_color='rgb(144, 12, 63)',
                        marker_line_width=1.5, opacity=0.6)
    return fig_5

if __name__ == "__main__":
    app.run_server(debug=True)
```

Para graficar la siguiente consulta la cual nos da toda la información relacionada a los aeropuertos:

```
def vuelo_hora():
    return """  SELECT EXTRACT('hour' FROM fecha_llegada ) hora_llegada, COUNT(*) total
                FROM Vuelo
                GROUP BY EXTRACT('hour' FROM fecha_llegada )
                ORDER BY hora_llegada ASC;"""
def vuelo_tickets():
    return """SELECT id_vuelo, COUNT(*) num_tickets
              FROM tickets
              GROUP BY id_vuelo;"""
def num_aviones():
    return """  SELECT nombre as aerolinea, COUNT(*) num_aviones
                FROM aviones
                NATURAL JOIN aerolineas
                GROUP BY nombre;"""
def nacionalidad_asc():
    return """SELECT nacionalidad , COUNT(*) total
                FROM pasajeros
                NATURAL JOIN tickets GROUP BY nacionalidad ORDER BY total asc LIMIT 20"""
def nacionalidad():
    return """SELECT nacionalidad , COUNT(*) total
                FROM pasajeros
                NATURAL JOIN tickets GROUP BY nacionalidad ORDER BY total desc LIMIT 20"""
def destino():
    return"""SELECT ciudad,COUNT(*) total
            FROM vuelo natural JOIN aeropuertos group by ciudad;

    """
def modelos():
    return """
            SELECT nombre,COUNT(*) total
            FROM  aviones natural JOIN modelos group by nombre;"""
```