|  |
| --- |
| UAM EPS |
| Memoria Practica1 |
| Bases de Datos |

|  |
| --- |
| Luis Fernández Fernández y Nicolás Ruiz Martínez  10-25-2023 |

****­­­

Índice

[Claves primarias y extranjeras de cada tabla: 1](#_Toc149135812)

[Diagrama del modelo relacional 2](#_Toc149135813)

[Query 1: 3](#_Toc149135814)

[Query 2: 4](#_Toc149135815)

[Query 3: 5](#_Toc149135816)

[Query 4: 6](#_Toc149135817)

[Query 5: 7](#_Toc149135818)

[Query 6: 7](#_Toc149135819)

## Claves primarias y extranjeras de cada tabla:

aircrafts\_data (**aircraft\_code**, model)

airports\_data (**airport\_code**, airport\_name, city, coordinates, timezone)

boarding\_passes (**ticket\_no**  ticket\_flights. ticket\_no, **flight\_id** ticket\_flights.flight\_id, boarding\_no, seat\_no)

bookings (**book\_ref**, book\_date, total\_amount)

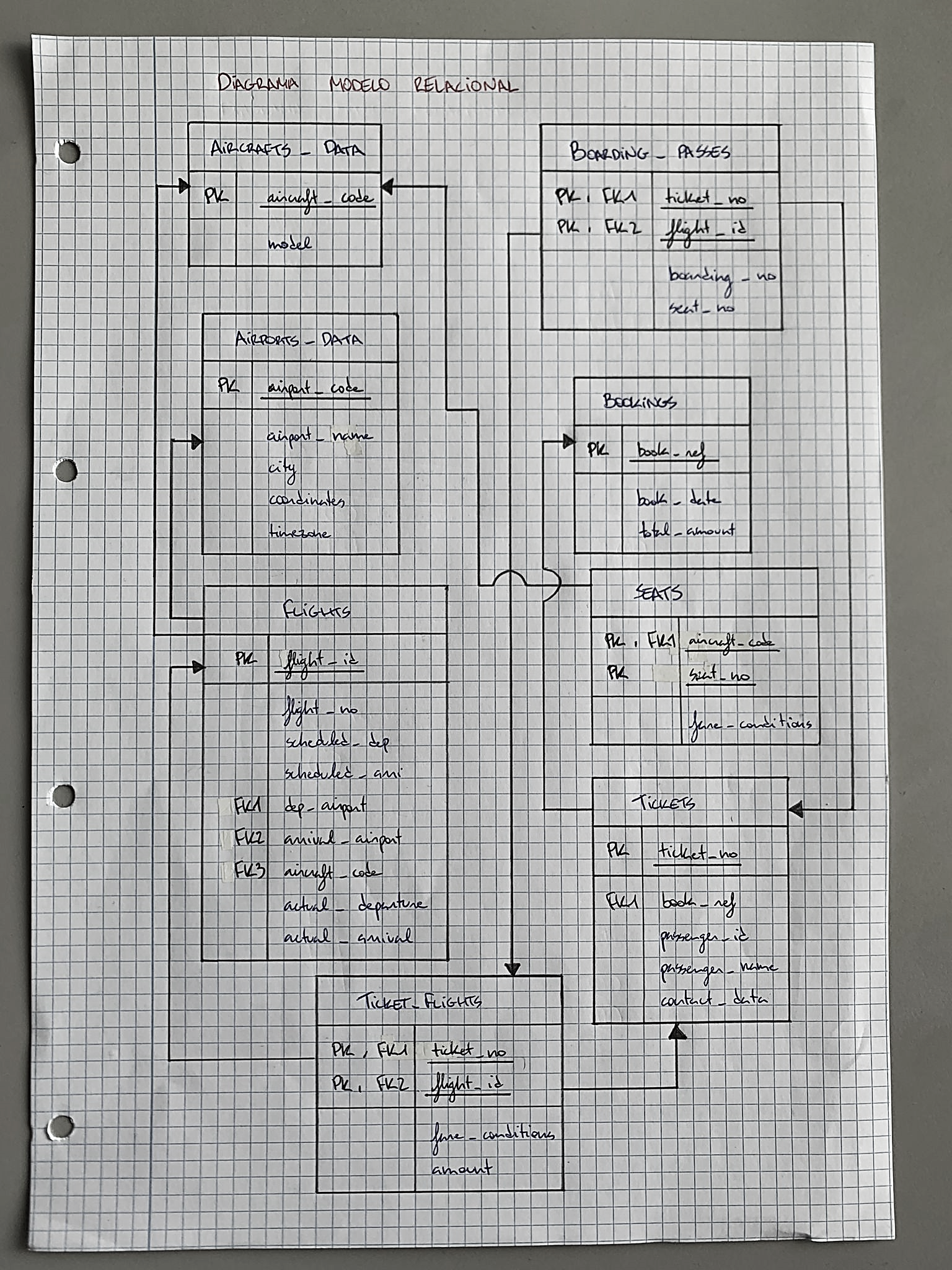
flights (**flight\_id**, flight\_no, scheduled\_departure, scheduled\_arrival, departure\_airport  airports\_data.airport\_code, arrival\_airport  airports\_data.airport\_code, aircraft\_code  aircrafts\_data.aircraft\_code, actual\_departure, actual\_arrival)

seats (**seat\_no**, **aircraft\_code** aircrafts\_data.aircraft\_code, fare\_conditions)

ticket\_flights (**ticket\_no** tickets.ticket\_no, **flight\_id** flights.flight\_id, fare\_conditions, amount)

tickets (**ticket\_no**, book\_ref bookings.book\_ref, passenger\_id, passenger\_name, contact\_data)

## Diagrama del modelo relacional



Todas las querys se han obtenido en formato html (y texto) para una mejor visualización de la información usando el comando \H en psql para cambiar la salida a dicho formato.

El makefile realiza la query de manera normal (cargandola en formato texto en el log ) y después la vuelve a realizar pero la guarda en formato html y no la muestra en la terminal.

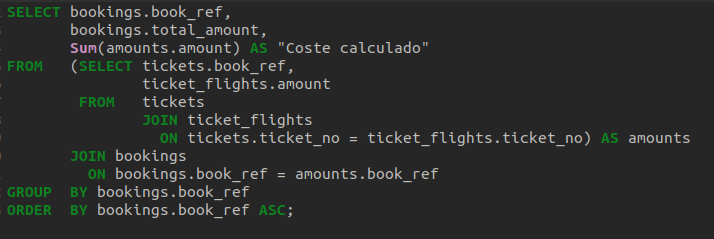
## Query 1:

Lista el número de reservas (bookings) que contengan un billete de ida y vuelta agrupadas por aeropuerto. Un billete es de ida y vuelta si el aeropuerto de origen del primer vuelo es idéntico al aeropuerto de destino del último vuelo. El resultado mostrará dos columnas con el código del aeropuerto de salida, así como el número de reservas que contienen billetes de ida y vuelta para ese aeropuerto. Igualmente, el resultado se ordenará de forma ascendente usando el atributo (departure airport).

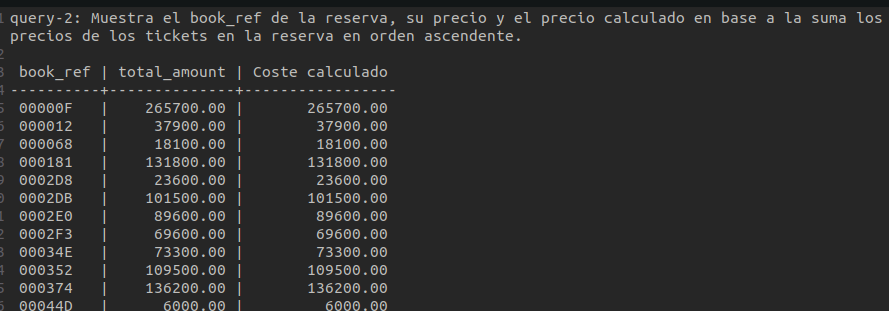
No se ha hecho.

## Query 2:

El precio de una reserva (booking.total amount) puede ser calculado a partir del precio de cada vuelo (ticket flights.amount). Crea una consulta que para cada reserva muestre el precio de la misma calculado a partir del valor almacenado en ticket flights.amount. La salida debe mostrar tres columnas conteniendo los atributos booking.book ref, booking.total amount y el valor calculado por vuestra consulta. Ordena la consulta de forma ascendiente usando el atributo bookigs.book ref.



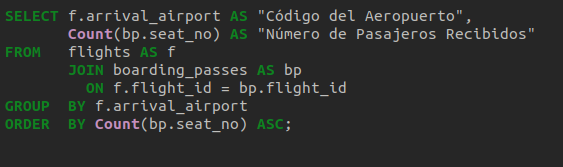
Se ha implementado de manera anidada, el select del from permite obtener los atributos book\_ref y el coste de cada ticket de la unión de las tablas tickets y ticket\_flights. Esta tabla se une con bookings, donde se encuentra el coste total de la reserva. Todo esto se agrupa y ordena de forma ascendente por book\_ref. En el select principal se obtiene el book\_ref, el coste por reserva y la suma de los costes de cada ticket agrupados por reserva(book\_ref).



## 

## Query 3:

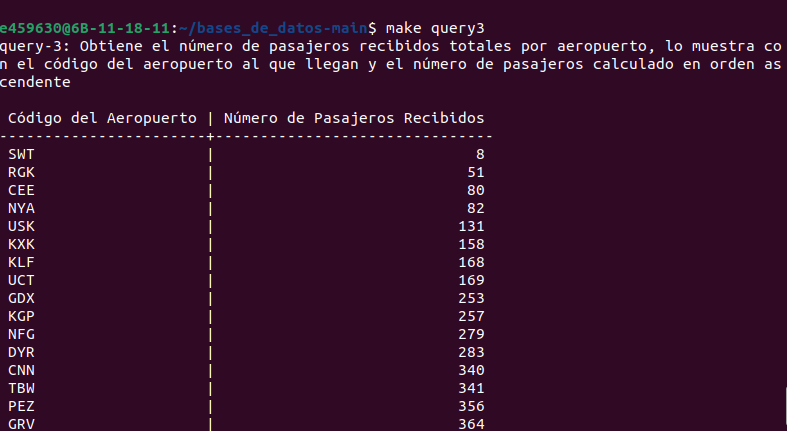
Crea una consulta que muestre por pantalla el código de aeropuerto y el número de pasajeros recibidos, y ordena la misma de forma ascendiente por el número de pasajeros recibidos. Se considera que un pasajero ha volado al aeropuerto si se ha emitido una tarjeta de embarque. Una tarjeta de embarque para un vuelo del aeropuerto X al Y incrementa en 1 el número de pasajeros recibidos en el aeropuerto Y .



Se ha implementado haciendo una unión de flights y boarding\_passes, agrupándolo por el aeropuerto de llegada, y haciendo un count en cada grupo del número de asientos(seat\_no).

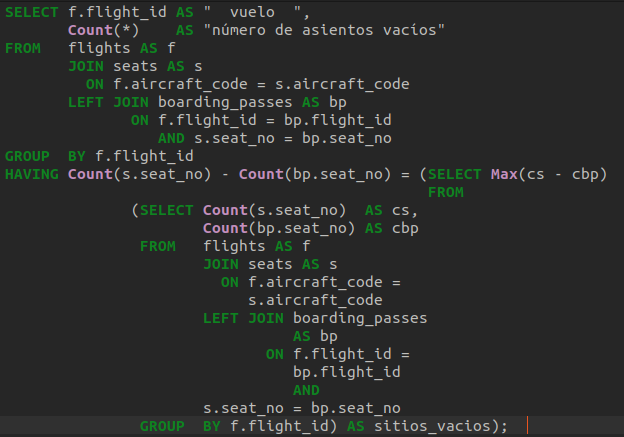
Esto se ha ordenado por la cuenta de los asientos de forma ascendente.

Al final muestra el código del aeropuerto de llegada y el número de asientos(llenos) de los aviones que llegan a dicho aeropuerto, lo cual equivale a decir el número de pasajeros recibidos.



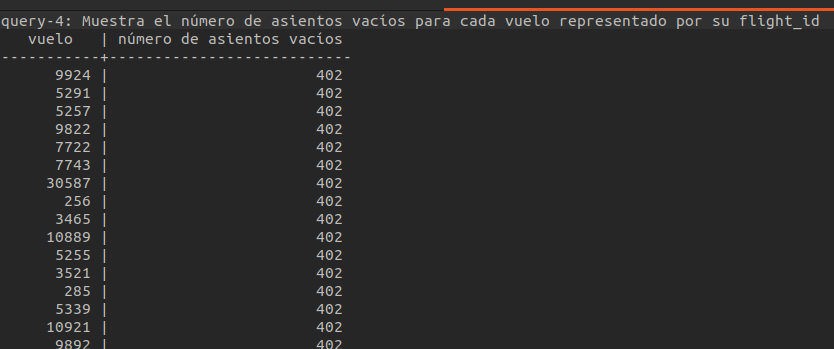
## Query 4:

Vuelo con más asientos vacíos. Muestra en la salida el atributo flight id y el número de asientos vacíos. En caso de empate muestra todos los vuelos que hayan empatado.

Se ha implementado haciendo un left join de manera que la tabla boarding\_passes no tenga la prioridad y por tanto en los casos que el seat\_no de esta tabla no pueda añadirse al join se pondrá null y al contarla contribuirá como 0 asientos. Al unir flights y seats se busca obtener los asientos que se pueden llenar de cada avión, al restarle a esto el número de asientos no nulos de boarding\_passes se están contando los asientos que no han sido ocupados en cada avión (ya que se agrupa por flight\_id).

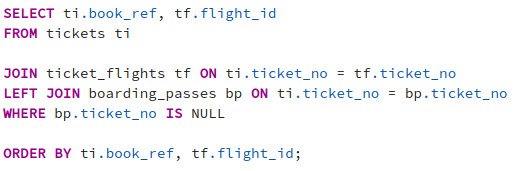
En la parte inferior al Having se da la condición de que el número de asientos no ocupados de cada avión(Count(s.seat\_no)-Count(bp.seat\_no)) sea igual al máximo número de asientos vacíos contado(SELECT Max(cs-cbp) FROM) en el select de abajo(que hace exactamente lo mismo que el FROM y GROUP BY de arriba).

Se muestra con el select principal el flight\_id de cada vuelo y la cuenta de asientos vacíos de este. Mostrando solo los que tengan el máximo número de asientos vacíos por la condición del GROUP BY HAVING.

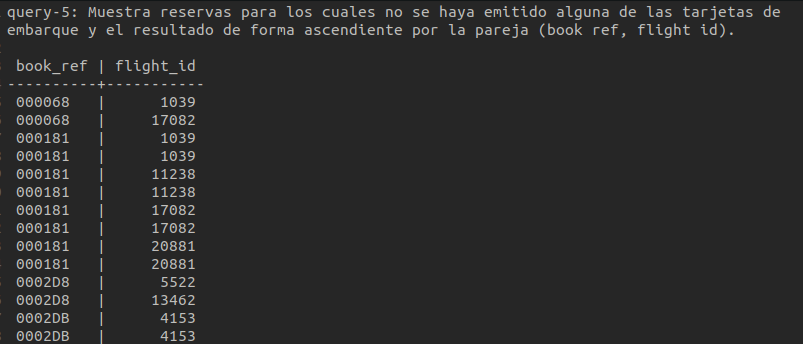


## Query 5:

Muestra reservas para los cuales no se haya emitido alguna de las tarjetas de embarque. La salida debe mostrar los atributos book ref y flight id. Ordenar el resultado de forma ascendente por la pareja (book ref, flight id).

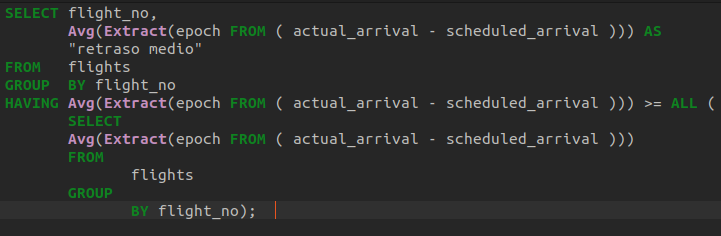


Se ha implementado haciendo uso de 1 join y 1 left join. El primero join combina los datos de las tablas tickets y ticket\_flights, a través de la condición ticket\_no, que ha de ser igual en ambas tablas . De esta manera conseguimos establecer una relación entre los tickets y los vuelos asociados a esos tickets. Después, con el left join, incluimos todos los registros de la tabla tickets con los registros que coincidan en la tabla boarding\_passes, estableciendo así la relación entre ambas tablas. Otra vez, la condición es que ese ticket\_no sea igual en ambas tablas. Ahora viene la condición WHERE bp.ticket\_no IS null, la cual es clave para conseguir el resultado deseado por el enunciado. Al coger únicamente los registros donde no hay una coincidencia en la tabla boarding\_passes, de esta manera sabemos que NO se han emitido tarjetas de embarque para dichos vuelos. Después, mediante el order by, ordenamos la salida justo como se nos indica.



## Query 6:

Por definición dos vuelos (flights) realizan el mismo trayecto si su número de vuelo (flight no) es el mismo. Se solicita el trayecto donde la media de los retrasos acumulados sea mayor. Se define retraso como la diferencia entre los valores actual arrival - scheduled arrival. Mostrar en la línea de salida flight no y el retraso medio. En caso de empate deben aparecer todos los trayectos con mayor retraso.



Se ha implementado haciendo uso de Extract epoch FROM con lo que se ha obtenido la diferencia actual arrival - scheduled arrival en segundos; a eso se le ha realizado un Avg para obtener la media acumulada.

Se han agrupado los valores de las medias en función de flight\_no y se muestran aquellas tuplas, de flight\_on y la media de sus retrasos, que tengan una media mayor o igual a los demás trayectos(flight\_no).

De esta manera se obtienen las que mayor media de retrasos tienen.

