ESTRUCTURA DE COMPUTADORES

PRÁCTICA 1

—

Rafael Dominguez Saez

Andrés Alonso De Pool Alcántara

Pareja 4

Grupo 1262

.

1. Análisis de la base de datos

1.1. Claves primarias y extranjeras

flights(flight_id, flight_no, scheduled_departure, scheduled_arrival,
departure_airport→airports_data.airport_code,
arrival_airport→airports_data.airport_code, status,
aircraft_code→aircrafts_data.aircraft_code, actual_departure, actual_arrival)

airports_data(airport_code, airport_name, city, coordinates, timezone)

aircrafts_data(aircraft_code, model, range)

seats(**aircraft_code**→aircrafts_data.aircraft_code, seat_no, fare_conditions)

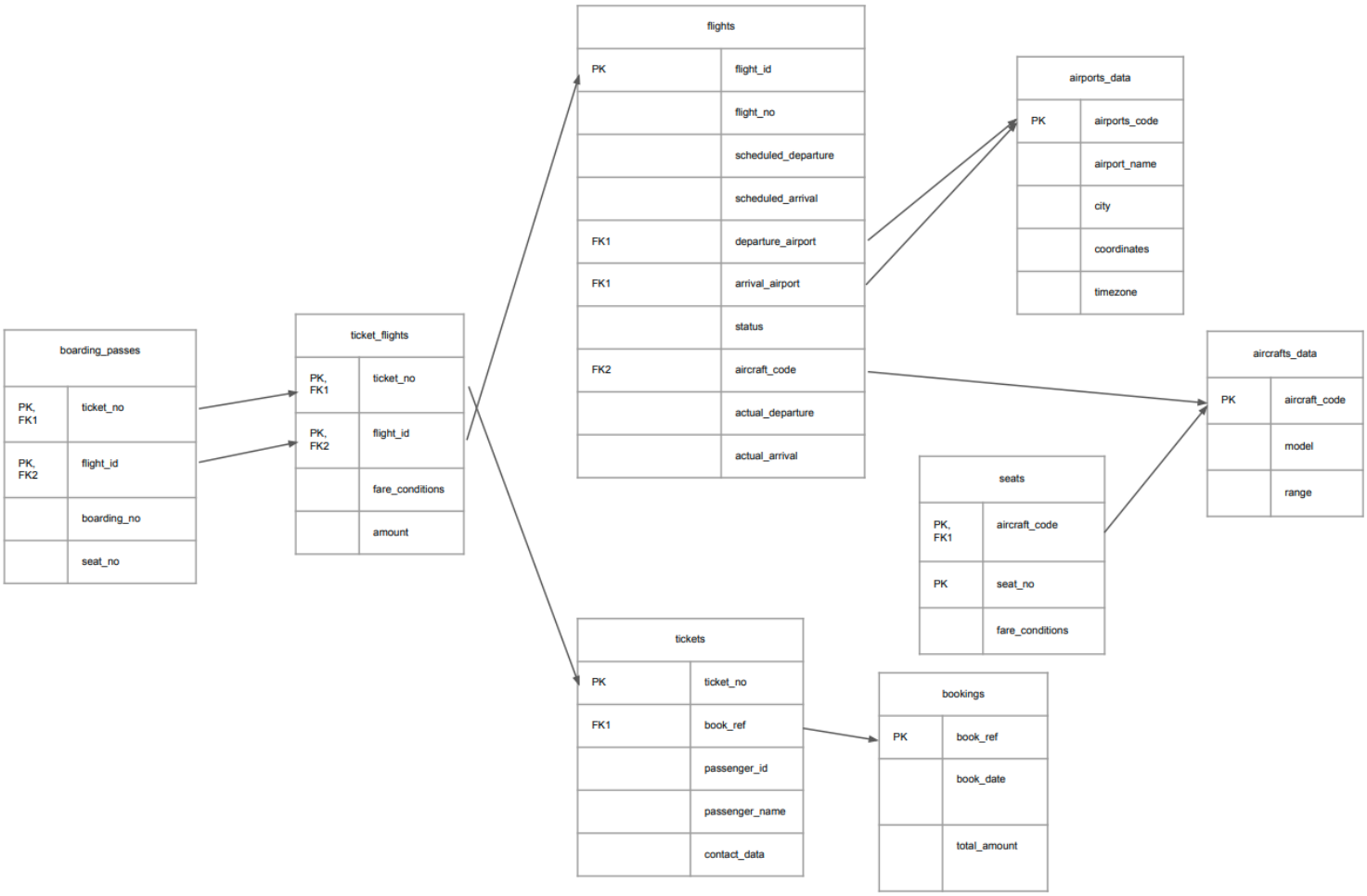
bookings(book_ref, book_date, total_amount)

tickets(ticket_no, **book_ref**→bookings.book_ref, passenger_id, passenger_name,
contact_data)

ticket_flights(**ticket_no**→tickets.ticket_no, **flight_id**→flights.flight_id,
fare_conditions, amount)

boarding_passes(**ticket_no**→ticket_flights.ticket_no,
flight_id→ticket_flights.flight.id, boarding_no, seat_no)

1.2. Diagrama del modelo relacional



2. Consultas

2.1. Query 1

En esta query la hemos realizado creando dos tablas nuevas, **VIAJE_IDA** que guarda de todos los vuelos su *departure_airport,* el *scheduled_arrival* y el *id del pasajero*, y **VIAJE_VUELTA** que guarda de todos todos los vuelos su *arrival_airport*, el *scheduled_departures* y el *id del pasajero.* A continuación, juntamos las tablas y comprobamos que los aeropuertos de ida como de vuelta son los mismos, que los horarios no son incompatibles, y que el id del pasajeros son iguales en ambos vuelos.

```
with VIAJE_IDA as
   (
       select
           f.departure_airport, t.passenger_id, f.scheduled_arrival
       from
           tickets t join ticket_flights tf on t.ticket_no = tf.ticket_no
           join flights f on tf.flight_id = f.flight_id
   ),
   VIAJE_VUELTA as
   (
       select
           f.arrival_airport, t.passenger_id, f.scheduled_departure
       from
           tickets t join ticket_flights tf on t.ticket_no = tf.ticket_no
           join flights f on tf.flight_id = f.flight_id
   )

select
   vi.departure_airport, count(*)
from
   viaje_ida vi, viaje_vuelta vv
   where
       vi.departure_airport = vv.arrival_airport
       and vi.passenger_id = vv.passenger_id and vi.scheduled_arrival < vv.scheduled_departure
group by
   vi.departure_airport
order by
   vi.departure_airport;
```

2.2. Query 2

Para este apartado hemos comprobado que la de todos los *amount* de **TICKET_FLIGHTS** es el mismo valor que el *total_amount* de **BOOKINGS** uniendo ambas tablas con **TICKETS** y comprobando el resultado

```
select
   b.book_ref, b.total_amount, sum(tf.amount) as total_amount_calculated
from
   bookings b join tickets t on b.book_ref = t.book_ref
   join ticket_flights tf on t.ticket_no = tf.ticket_no
group by
   b.book_ref
order by
   b.book_ref
```

2.3. Query 3

En esta query hemos contado el número total de pasajeros que llegan a X aereopuerto, para ello, hemos comprobado que el **FLIGHTS** su *arrival_airport* es igual a *airport_code* del **AIRPORTS_DATA**, luego, hemos comprobado que el *flight_id* del **FLIGHTS** y del **BOARDING_PASSES** son el mismo.

```
SELECT
   a.airport_code, count(*) as total_passengers
FROM
   airports_data a JOIN flights f ON a.airport_code=f.arrival_airport
   join boarding_passes bp on bp.flight_id=f.flight_id
group by
   a.airport_code
order by
   total_passengers;
```

2.4. Query 4

En este apartado, hemos creado varias tablas, la primera llamada **A_AVION_T** obtiene los asientos totales de cada aeronave, la segunda llamada **A_AVION_OC** que obtiene los asientos ocupados de cada *flight_id* y *aircrafts_code*, la tercera **A_AVION_AV** utiliza las anteriores para calcular los asientos vacíos. Luego con **A_AVION_AV_MAS_VACIO** obtuvimos el máximo de la anterior tabla, y ahora comparamos de **A_AVION_AV** con el máximo y hacemos una tabla solo con esos.

```
with A_AVION_T(aircrafts_code, totalasientos) as
   (
       select
           ad.aircraft_code, count(*) as totalasioentos
       from
           aircrafts_data ad join seats s on ad.aircraft_code=s.aircraft_code
       group by
           ad.aircraft_code
   ),
   A_AVION_OC(flight_id, aircrafts_code, ocupadosasientos) as
   (
       select
           f.flight_id, f.aircraft_code, count(*) as ocupadosasientos
       from
           flights f join ticket_flights tf on f.flight_id=tf.flight_id
       group by
           f.flight_id, f.aircraft_code
   ),
   A_AVION_AV as
   (
       select
           aoc.flight_id, (ato.totalasientos - aoc.ocupadosasientos) as asientos_vacios
       from
           A_AVION_T ato, A_AVION_OC aoc
       where
           ato.aircrafts_code= aoc.aircrafts_code
       group by
           aoc.flight_id, ato.totalasientos, aoc.ocupadosasientos
   ),
   A_AVION_AV_MAS_VACIO as
   (
       select
           *
       from
           a_avion_av av
       order by
           av.asientos_vacios desc limit 1
   )

   select
       av.flight_id, av.asientos_vacios
   from
       A_AVION_AV av join A_AVION_AV_MAS_VACIO avmv on av.asientos_vacios = avmv.asientos_vacios
```

```
   order by av.asientos_vacios;
```

2.5. Query 5

Esta query en **book_ref_with_tickets** lo que hace es agrupar en una tabla de cada *book_ref* su *ticket_no* y de cada uno de esos su *flight_id*. Luego como solo queremos aquellos *ticket_no* que no estén incluidos en los **boarding_passes** hacemos un left join cuando los *ticket_no* de **bording_passes** que sean nulos. Por lo que coge aquellos del conjunto de los **ticket_flight** que no estén en **boarding_passes**.Y representamos lo que queremos.

```
with book_ref_with_tickets(book_ref, ticket_no, flight_id) as
   (
       select
           t.book_ref , t.ticket_no ,tf.flight_id
       from
           tickets t, ticket_flights tf
       where
           t.ticket_no = tf.ticket_no
       group by
           t.ticket_no, tf.flight_id
   )

select
   distinct brwt.book_ref, brwt.flight_id
from
   book_ref_with_tickets brwt left join boarding_passes bp
on
   bp.ticket_no = brwt.ticket_no
where
   bp.ticket_no is null
order by
   brwt.book_ref asc, brwt.flight_id asc;
```

2.6. Query 6

En esta query hemos primado agrupado las tuplas con las que iba a trabajar, esto lo hicimos en **flight_no_and_id**, agrupando *flight_no*, *actual_arrival*, *scheduled_arrival*. Ahora usando esa tabla en **flight_delays**, calculamos para cada *flight_no* sus retrasos. Seguidamente en **flight_average_delay** se calcula el retraso medio de cada *flight_no*, a través de la función AVG(). Luego al final en **flight_average_delay_max** hago una tabla donde obtengo el máximo de la tabla, para por último seleccionar de la tabla de **flight_average_delay** los que tengan un retraso igual al máximo calculado en anteriormente.

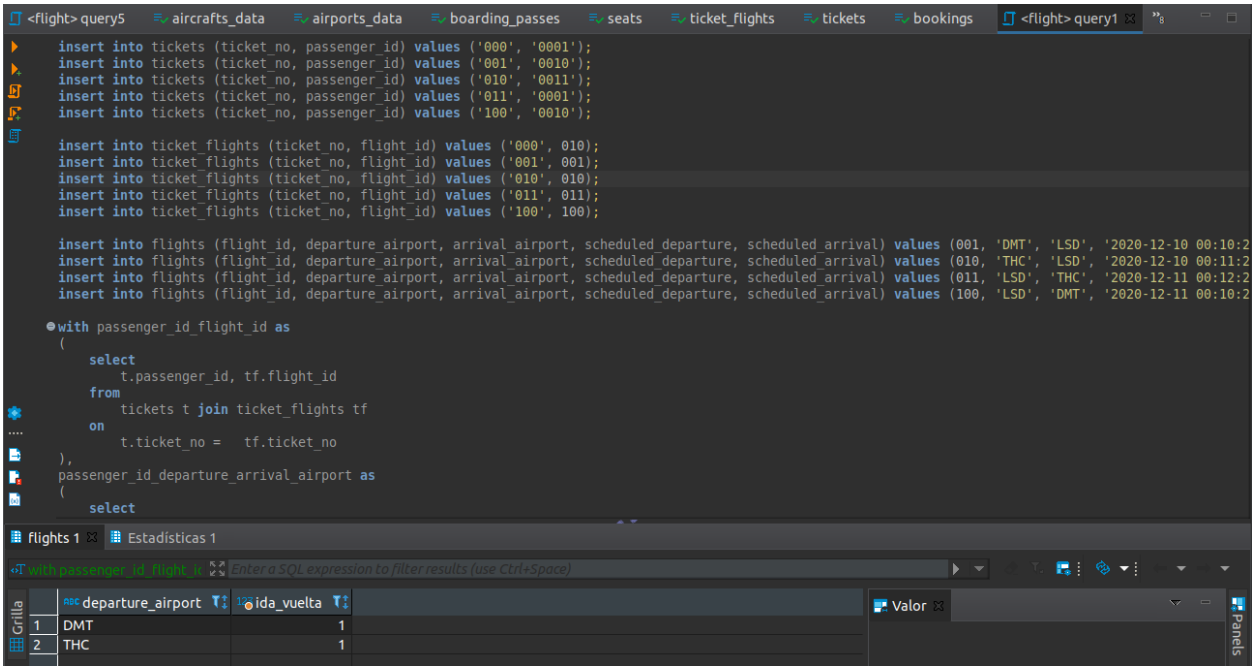
```
WITH flight_no_and_id(flight_no, actual_arrival, scheduled_arrival) as
(
   SELECT
       f.flight_no, f.actual_arrival, f.scheduled_arrival
   FROM
       flights f
   group by
       f.flight_no, f.actual_arrival, f.scheduled_arrival
),
flight_delays(flight_no, retraso) as
(
   SELECT
       fni.flight_no, (fni.actual_arrival - fni.scheduled_arrival) as retraso
   FROM
       flight_no_and_id fni
   order by
       fni.flight_no asc
),
flight_average_delay(flight_no, retraso_medio) as
(
   SELECT
       fd1.flight_no, AVG(fd1.retraso) as retraso_medio
   FROM
       flight_delays fd1
   GROUP BY
       fd1.flight_no
),
flight_average_delay_max(flight_no, retraso_medio_max) as
(
   SELECT
       fd.flight_no, fd.retraso_medio as retraso_medio_max
   FROM
       flight_average_delay fd
   GROUP BY
       fd.flight_no, fd.retraso_medio
   ORDER BY
       fd.retraso_medio desc limit 1
),
all_flights_with_max_delay as
(
   SELECT
       fad.flight_no, fad.retraso_medio
   FROM
       flight_average_delay fad, flight_average_delay_max fadm
   WHERE
       fad.retraso_medio = fadm.retraso_medio_max
)
SELECT * FROM all_flights_with_max_delay;
```

3. Pruebas consultas

Para realizar las pruebas sobre las queries creadas modificamos la base de datos *flight.sql* dejándola tan solo con las tablas generadas sin insertar nada. Luego también para no tener que introducir datos que no necesitábamos para probar la query quitamos todas las restricciones de IS NOT NULL y algunos de los CONSTRAINTS. Ahora con esto podíamos introducir los datos antes de correr el query y ver si daban el resultado esperado.

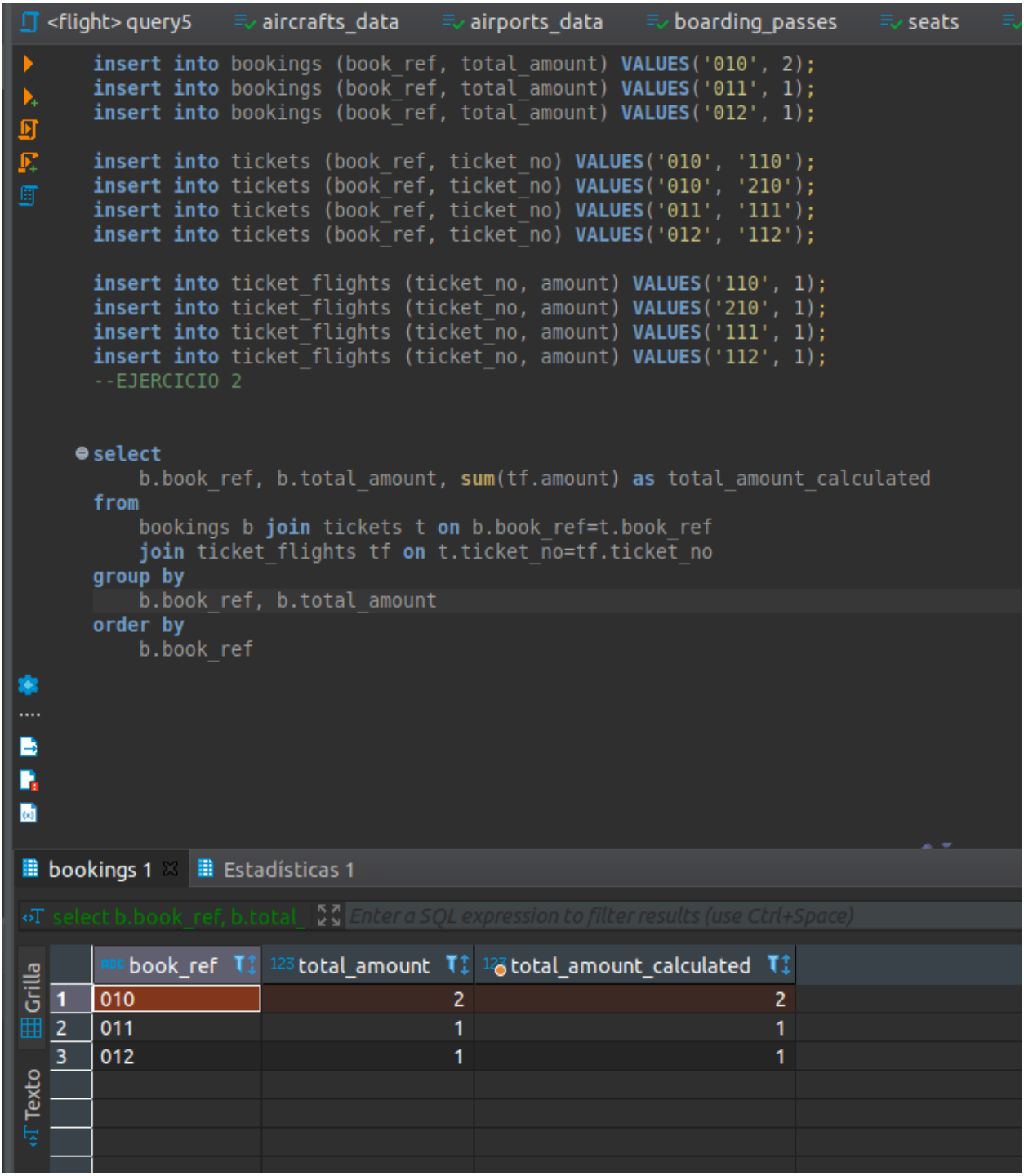
3.1. Query 1

Para esta prueba introducimos 3 pasajeros, y dos de ellos hacen recorridos de ida y vuelta pero iniciando de *departure_airport* distintos. Por lo que como era de esperarse solo tenemos como resultado los 1 reserva en dos aeropuertos distintos.



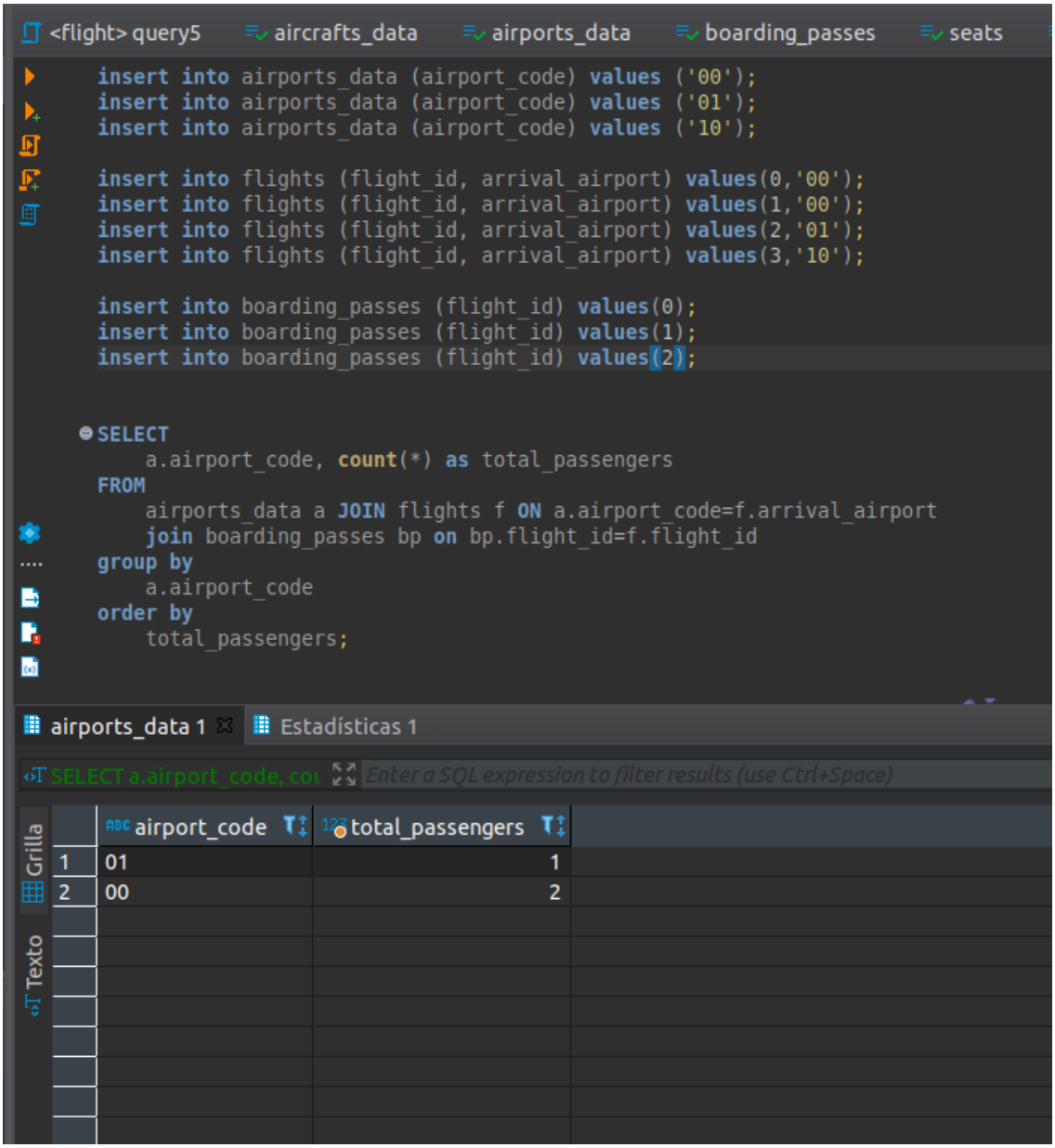
3.2. Query 2

En esta segunda query no era tan necesario probar su correcto funcionamiento porque de normal se prueba a sí misma. Pero aquí introdujimos 3 *tickets* todos de valor 1 y 3 *book_ref* donde tan solo uno de estos ha reservado dos *tickets*. Por lo que tan solo 1 de estas tiene un 2 y las otras un valor de 1.


<flight> query5
aircrafts_data
airports_data
boarding_passes
seats
insert into bookings (book_ref, total_amount) VALUES('010', 2);
insert into bookings (book_ref, total_amount) VALUES('011', 1);
insert into bookings (book_ref, total_amount) VALUES('012', 1);
insert into tickets (book_ref, ticket_no) VALUES('010', '110');
insert into tickets (book_ref, ticket_no) VALUES('010', '210');
insert into tickets (book_ref, ticket_no) VALUES('011', '111');
insert into tickets (book_ref, ticket_no) VALUES('012', '112');
insert into ticket_flights (ticket_no, amount) VALUES('110', 1);
insert into ticket_flights (ticket_no, amount) VALUES('210', 1);
insert into ticket_flights (ticket_no, amount) VALUES('111', 1);
insert into ticket_flights (ticket_no, amount) VALUES('112', 1);
--EJERCICIO 2
select
b.book_ref, b.total_amount, sum(tf.amount) as total_amount_calculated
from
bookings b join tickets t on b.book_ref=t.book_ref
join ticket_flights tf on t.ticket_no=tf.ticket_no
group by
b.book_ref, b.total_amount
order by
b.book_ref
bookings 1
Estadísticas 1
select b.book_ref, b.total_
Enter a SQL expression to filter results (use Ctrl+Space)
Grilla
Texto
book_ref
total_amount
total_amount_calculated
1 010 2 2
2 011 1 1
3 012 1 1

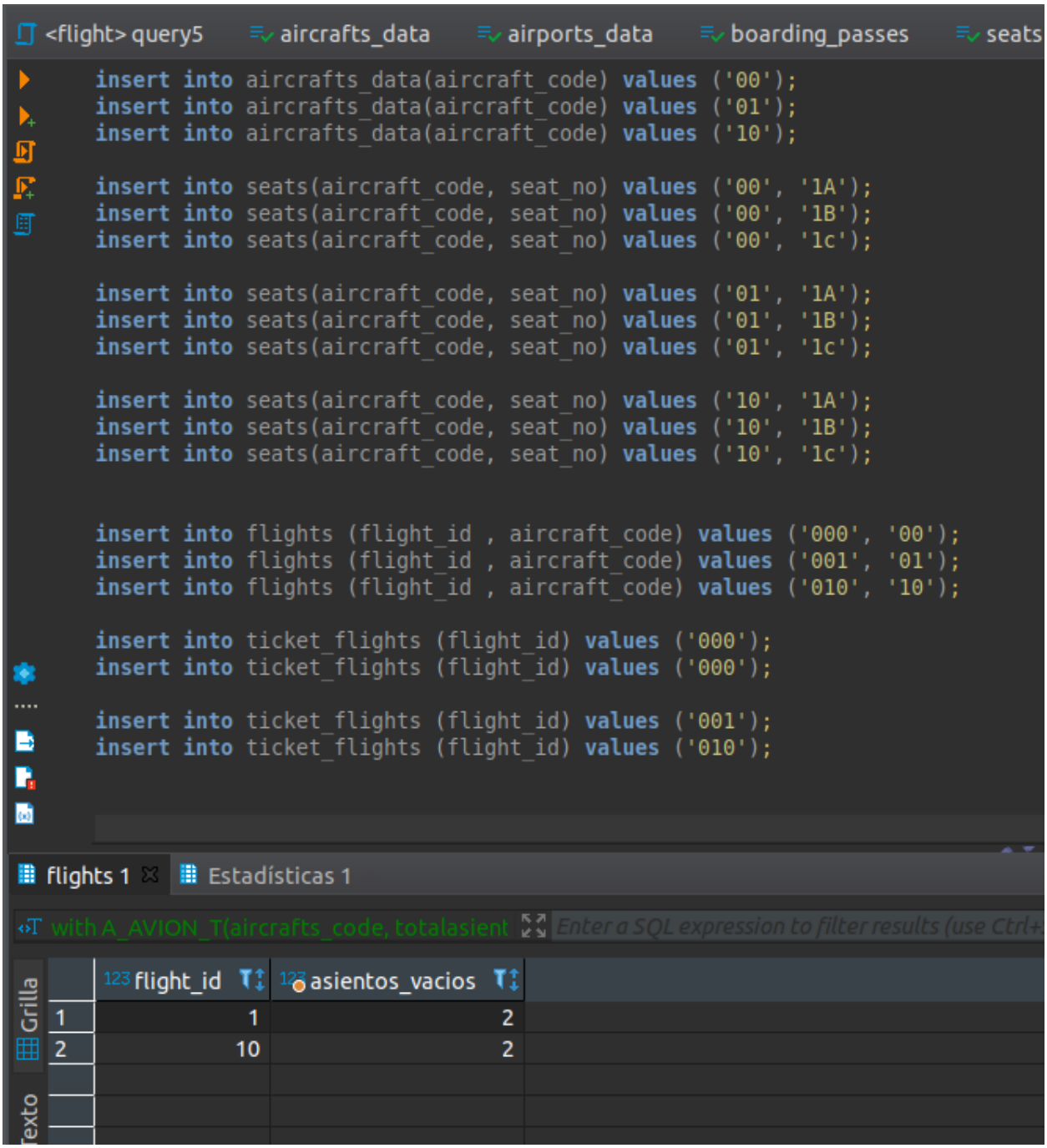
3.3. Query 3

Para esta query introducimos 3 *airport_code*, pero tan solo cree **boarding_passes** para 2 de estos y a uno le cree dos. Entonces como es de esperarse tan solo hay dos *airport_codes* y uno tiene 2 y el otro 1 *total_passengers*.



3.4. Query 4

Para el query 4 introdujimos 3 aviones todos con 3 asientos, dos de estos tan solo tienen un asiento ocupado y el otro tiene dos. Con esto probamos que muestra los que tienen más asientos vacíos independientemente de si son uno o varios.



3.5. Query 5

Para esta query creamos 4 **ticket_flights,** una *book_ref* tiene dos de estas y luego las otras están repartidas una a una, pero tan solo hicimos **boarding_passes** para 1 de estos *ticket_no* entonces como es de esperarse, observamos 3 *flight_id* que no tienen **boarding_passes** y dos pertenecen a una *book_ref*.

```
insert into ticket_flights (ticket_no, flight_id) values ('00', 00);
insert into ticket_flights (ticket_no, flight_id) values ('01', 01);
insert into ticket_flights (ticket_no, flight_id) values ('10', 10);
insert into ticket_flights (ticket_no, flight_id) values ('11', 11);

insert into tickets (ticket_no, book_ref) values ('00', 'A');
insert into tickets (ticket_no, book_ref) values ('01', 'B');
insert into tickets (ticket_no, book_ref) values ('10', 'B');
insert into tickets (ticket_no, book_ref) values ('11', 'C');

insert into boarding_passes (ticket_no) values ('00');


--Ejercicio 5

with book_ref_with_tickets(book_ref, ticket_no, flight_id) as
    (
        select
            t.book_ref , t.ticket_no ,tf.flight_id
        from
            tickets t, ticket_flights tf
        where
            t.ticket_no = tf.ticket_no
        group by
            t.ticket_no, tf.flight_id, t.book_ref
    )

select
    distinct brwt.book_ref, brwt.flight_id
```

tickets(+) 1

with book_ref_with_tickets(book_ref, ticket — Enter a SQL expression to filter results (use Ct

Grilla

Texto

	book_ref	flight_id
1	B	1
2	B	10
3	C	11

3.6. Query 6

En la última query introdujimos 3 *flight_no* con 3 tiempos distintos. Dos de estos *flight_no* tienen el mismo retraso promedio para probar que muestra que cuando hay más de un caso con el mismo retraso máximo