

# Ejercicio 1

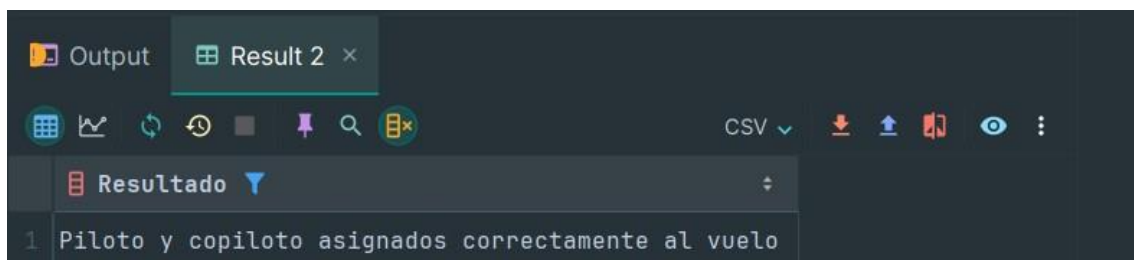
Descripción: *Para resolver el primer problema lo que hice fue crear un procedimiento el cual se le añade 2 DNI y un código de un vuelo, declaro 3 variables una para la comprobación de si es piloto o copiloto y si el vuelo existe, cada variable la compruebo con una consulta, el vuelo si no existe se cancela el procedimiento con un mensaje de error, del mismo modo comprobamos que el piloto y copiloto existen.*

*Después insertamos los datos en la tabla piloto y en la tabla tiene asignamos la id del vuelo tanto al piloto como al copiloto.*

*Por último damos fin al procedimiento*

## Ejemplos (Capturas):

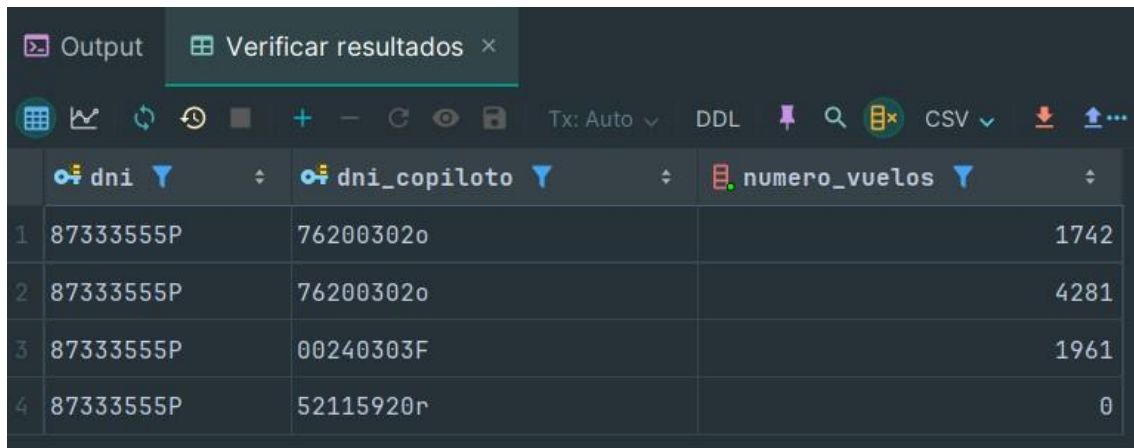
Ejemplo 1 – Prueba exitosa del procedimiento



Ejemplo 2 – Prueba de fallo del procedimiento



Ejemplo 3 – Verificamos los resultados

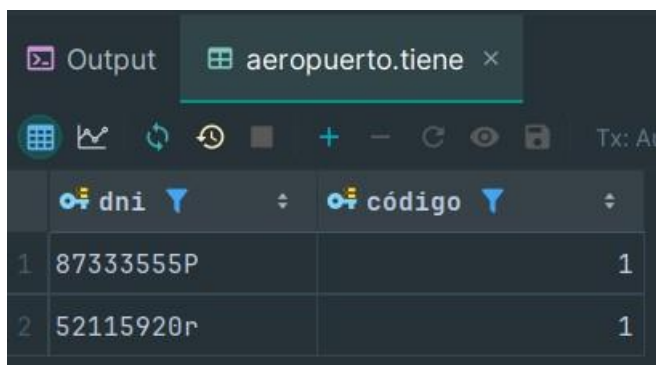


Output Verificar resultados x

Tx: Auto DDL CSV

	dni	dni_copiloto	numero_vuelos
1	87333555P	76200302o	1742
2	87333555P	76200302o	4281
3	87333555P	00240303F	1961
4	87333555P	52115920r	0

En la imagen se muestra los datos de cuantos vuelos a realizado con cada copiloto, el ultimo al ser nuevo tiene 0 vuelos realizados.



Output aeropuerto.tiene x

Tx: Auto

	dni	código
1	87333555P	1
2	52115920r	1

En la imagen se muestra que tanto el piloto como el copiloto están correctamente asignados al vuelo con el código 1.

## Ejercicio 2

Descripción: *Para este ejercicio creo una función la cual se le da un código de vuelo y un precio base (el cual estimamos que cuesta cada billete) y nos devuelve el cálculo final.*

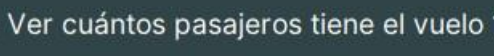
*Para ello declaramos una variable para los pasajeros y otra para los ingresos, el numero de pasajeros lo asignamos con una consulta que cuenta los pasajeros que están asignados a cada vuelo. Si el numero de pasajeros es mayor que 0 entonces modificamos la*

variable de ingresos para que sea el resultado de multiplicar el numero de pasajeros por el precio estimado que le hemos puesto a la función, si el numero de pasajeros es 0 se le asigna un 0 a la variable.

Por ultimo la función devuelve la variable de ingresos.

### Ejemplos (Capturas):

Ejemplo 1 – Contaos cuantos pasajeros tiene un vuelo en concreto.



Ver cuántos pasajeros tiene el vuelo 14

CSV

`COUNT(*)`

1	3
---	---

Ejemplo 2 – Calculamos el ingreso estimado del vuelo poniendo un precio base de cada billete en 150€

The screenshot shows the Jupyter Notebook interface. The 'Output' tab is active, displaying a table with the title 'Calcular ingresos con precio base de 150'. The table has one column, 'Ingresos\_estimados', and one row with the value '450.00'.

	Ingresos_estimados
1	450.00

Ejemplo 3 – Repetimos la prueba anterior para un vuelo que no tiene pasajeros asignados

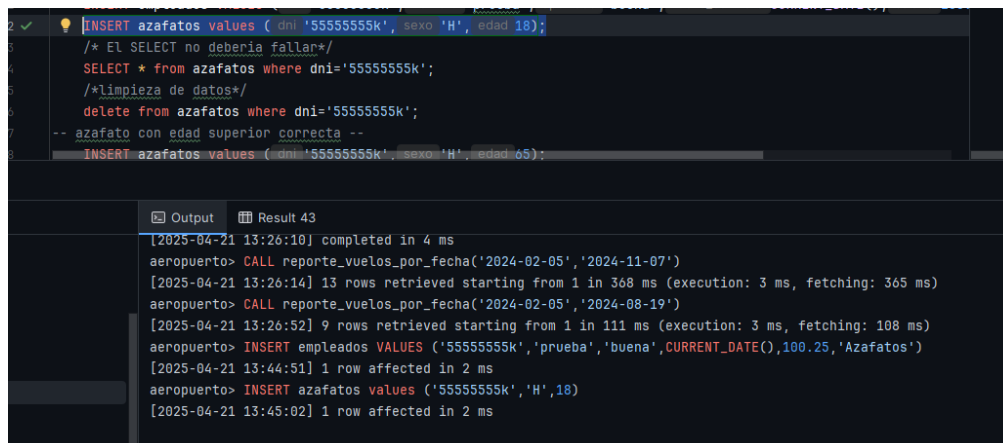
A screenshot of the Google Data Studio interface. The top bar shows the 'Output' tab selected, with a table titled 'Probar con vuelo sin...código sin pasajeros'. Below the title bar is a toolbar with various icons for table, chart, refresh, undo, redo, zoom, search, and other functions. The main area displays a table with one column named 'Ingresos\_estimados' and one row containing the value '0.00'.

## Ejercicio 3

Descripción: Para este ejercicio creo un trigger/disparador en el que se verifican las edades de los azafatos para que cumplan los requisitos que son: que sean mayores o iguales a 18 años y menores e iguales a 65 cancelando la operación de insertar un nuevo azafato si no se cumplen los requisitos gracias a que cuando no se cumplen los requisitos se lanza una excepción que cancela en INSERT.

### Ejemplos (Capturas):

Ejemplo 1 – con un valor correcto se inserta sin ningún tipo de problema



```
1 INSERT azafatos values ( dni '55555555k', sexo 'H', edad 18);
2
3 /* EL SELECT no deberia fallar*/
4 SELECT * from azafatos where dni='55555555k';
5
6 /*limpieza de datos*/
7 delete from azafatos where dni='55555555k';
8
9 -- azafato con edad superior correcta --
10 INSERT azafatos values (dni '55555555k', sexo 'H', edad 65);
```

Output Result 43

```
[2025-04-21 13:26:10] completed in 4 ms
aeropuerto> CALL reporte_vuelos_por_fecha('2024-02-05','2024-11-07')
[2025-04-21 13:26:14] 13 rows retrieved starting from 1 in 368 ms (execution: 3 ms, fetching: 365 ms)
aeropuerto> CALL reporte_vuelos_por_fecha('2024-02-05','2024-08-19')
[2025-04-21 13:26:52] 9 rows retrieved starting from 1 in 111 ms (execution: 3 ms, fetching: 108 ms)
aeropuerto> INSERT empleados VALUES ('55555555k','prueba','buena',CURRENT_DATE(),100.25,'Azafatos')
[2025-04-21 13:44:51] 1 row affected in 2 ms
aeropuerto> INSERT azafatos values ('55555555k','H',18)
[2025-04-21 13:45:02] 1 row affected in 2 ms
```

Verificación de que se ha insertado-

```

34 ✓ SELECT * from azafatos where dni='55555555k';
35 /*limpieza de datos*/
36 delete from azafatos where dni='55555555k';
37 -- azafato con edad superior correcta --
38 INSERT azafatos values (dni '55555555k', sexo 'H', edad 65);
39 /*El SELECT no deberia mostrar los datos anteriormente creados*/
40 SELECT * from azafatos where dni='55555555k';
41 /*limpieza de datos*/
42 delete from azafatos where dni='55555555k';
43 -- azafato con edad incorrecta inferior --
44 INSERT azafatos values (dni '55555555k', sexo 'H', edad 17);

```

Output **aeropuerto.azafatos**

	dni	sexo	edad
1	55555555k	H	18

Ejemplo 2 – con un valor incorrecto lanza una excepción y no se inserta

```

INSERT azafatos values (dni '55555555k', sexo 'H', edad 17);
/*los select no muestran nada debido a que no existen esas columnas porque el trigger las elimina y cancela la cr
SELECT * from azafatos where dni='55555555k';
azafato con edad incorrecta superior --
INSERT azafatos values (dni '55555555k', sexo 'H', edad 66);

```

(conn=44) edad no valida debe estar entre los 18 y 65

Output **aeropuerto.azafatos**

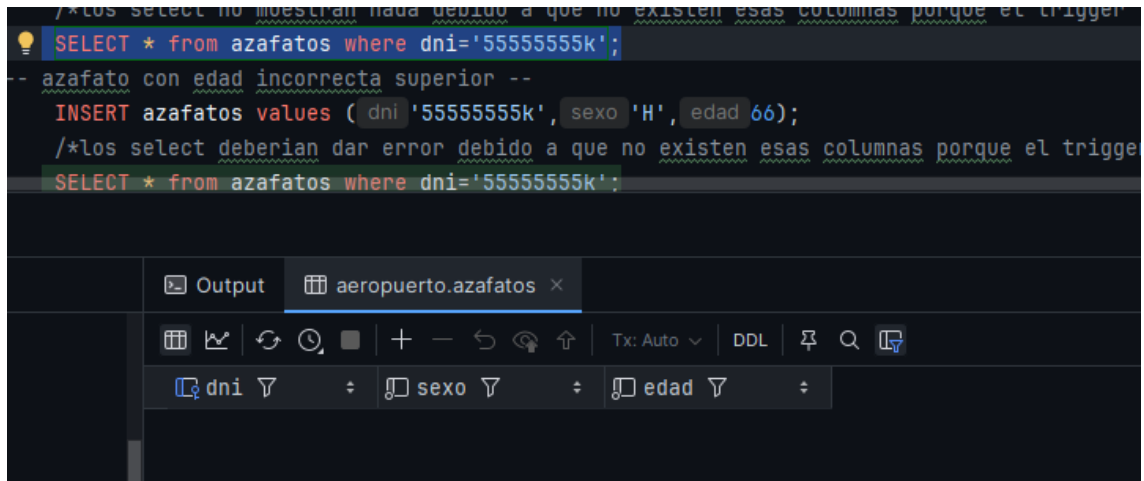
```

[2025-04-21 13:46:46] 1 row retrieved starting from 1 in 65 ms (execution: 1 ms, fetching: 64 ms)
aeropuerto> delete from azafatos where dni='55555555k'
[2025-04-21 13:47:38] 1 row affected in 2 ms
aeropuerto> INSERT azafatos values ('55555555k','H',65)
[2025-04-21 13:47:53] 1 row affected in 2 ms
aeropuerto> delete from azafatos where dni='55555555k'
[2025-04-21 13:47:59] 1 row affected in 3 ms
aeropuerto> INSERT azafatos values ('55555555k','H',17)
[2025-04-21 13:48:05] [45000][1644] (conn=44) edad no valida debe estar entre los 18 y 65

```

Verificación de que no se ha insertado-

```
/*los select no muestran nada debido a que no existen esas columnas porque el trigger
SELECT * from azafatos where dni='55555555k';
-- azafato con edad incorrecta superior --
INSERT azafatos values ( dni '55555555k', sexo 'H', edad 66);
/*los select deberian dar error debido a que no existen esas columnas porque el trigger
SELECT * from azafatos where dni='55555555k';
```



The screenshot shows a database IDE with a dark theme. The top pane is a SQL editor containing several lines of SQL code. The bottom pane is split into two sections: 'Output' and 'aeropuerto.azafatos'. The 'aeropuerto.azafatos' section displays a table with three columns: 'dni', 'sexo', and 'edad'. The table contains one row with the values '55555555k', 'H', and 66 respectively. The 'Output' section is empty.

## Ejercicio 4

Descripción: Para realizar este procedimiento se le introducen dos fechas y en base a un SELECT que se junta con la tabla lugar como lo y ld (lugar origen, lugar destino), usando un left join con embarcar debido a que estas tablas pueden tener datos no directamente relacionados y por último se filtra todo con un where para que los datos seleccionados sean los que están entre las fechas anteriormente introducidas

### Ejemplos (Capturas):

Ejemplo 1 – con un valor correcto se inserta sin ningún tipo de problema

CALL reporte\_vuelos\_por\_fecha( fecha\_inicio '2024-02-05', fecha\_fin '2024-05-19');

```

/* 5. Función: Verificar Disponibilidad de Pista
Descripción: Una función que verifica si una pista en un aeropuerto (lugar) está disponible para un horario específico.
Requisitos cubiertos:
Parámetros: codigo_lugar, hora_salida, hora_llegada.
Cursores: Consultar vuelos existentes en ese lugar.
Cancelación: Retornar falso si hay conflicto.*/

```

	codigo	Fecha	Número_Vuelo	Hora_Salida	Hora_Llegada	Origen	
1	2	2024-05-12	PT433	06:19:01	20:08:22	Centro, United States	Pal
2	4	2024-04-29	Q01170	18:40:39	18:08:48	Centro, United States	Mor
3	5	2024-02-05	ZC1946	15:31:45	11:46:11	Bella Vista, United States	Mor
4	7	2024-03-26	VD3799	07:40:15	08:39:37	Caballito, United States	Cat
5	11	2024-04-21	ZQ6849	08:40:50	10:49:48	Parque Patricios, United States	Cal

5 rows

Verificación de que se han seleccionado las columnas correctas-

```

190 ✓ SELECT * FROM vuelos WHERE fecha BETWEEN('2024-02-05')AND '2024-05-19';
191
192 /* 5. Función: Verificar Disponibilidad de Pista
193 Descripción: Una función que verifica si una pista en un aeropuerto (lugar) está disponible para un horario específico.
194 Requisitos cubiertos:
195 Parámetros: codigo_lugar, hora_salida, hora_llegada.
196 Cursores: Consultar vuelos existentes en ese lugar.
197 Cancelación: Retornar falso si hay conflicto.*/
198
199
200
201

```

	código	fecha	numero_vuelo	hora_salida	hora_llegada	tipo	código_dest
1	2	2024-05-12	PT433	06:19:01	20:08:22	Internacional	
2	4	2024-04-29	Q01170	18:40:39	18:08:48	Internacional	
3	5	2024-02-05	ZC1946	15:31:45	11:46:11	Internacional	
4	7	2024-03-26	VD3799	07:40:15	08:39:37	Nacional	
5	11	2024-04-21	ZQ6849	08:40:50	10:49:48	Internacional	

## Ejercicio 5

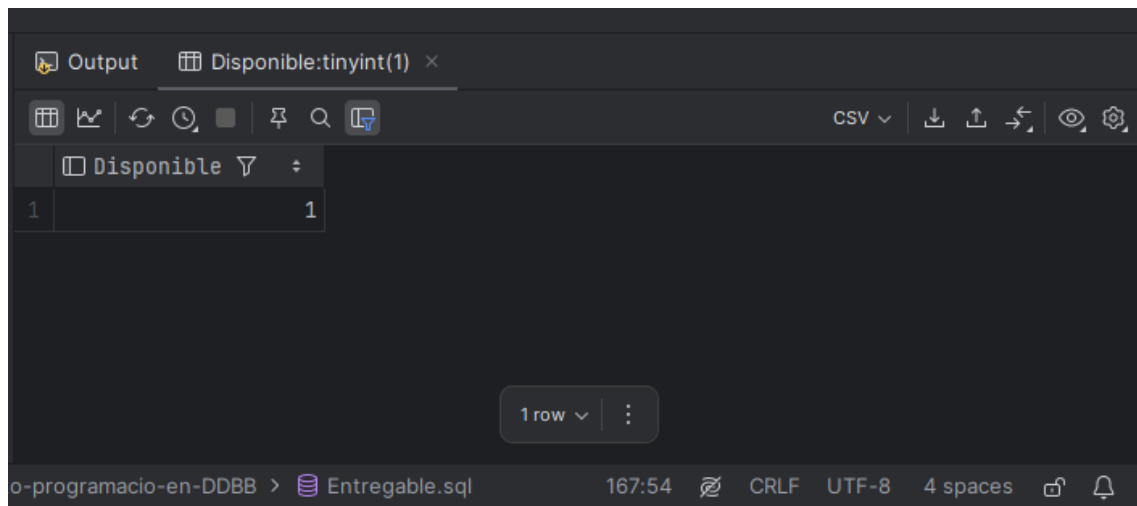
Descripción: Para este resolver este ejercicio donde necesitamos verificar si una pista está disponible creamos una función con los parámetros: código\_lugar, hora\_salida, hora\_llegada, donde el resultado sea un booleano que indique si está o no disponible.

Para ello declaramos una variable disponibilidad para comprobar si esta disponible mediante una consulta, comparando el origen y destino con el lugar, y que la hora de salida sea menor a la de llegada y viceversa.

El resultado en caso de disponible será (1) y (0) si está ocupada.

## Ejemplos (Capturas):

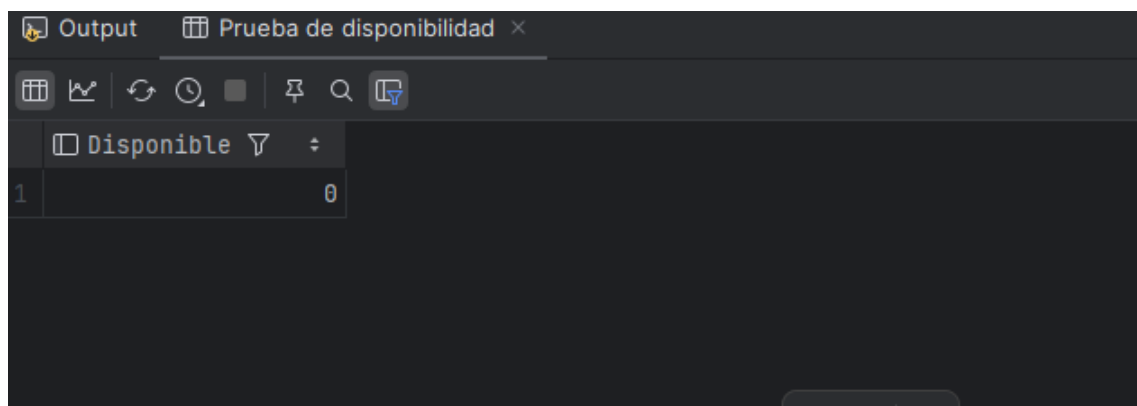
Ejemplo de pista disponible.



The screenshot shows a database query result in a dark-themed IDE. The title bar indicates the query is 'Disponibilidad:tinyint(1)'. The result is displayed in a table with one column labeled 'Disponibilidad' and one row containing the value '1'. The status bar at the bottom shows the file path 'o-programacio-en-DDBB > Entregable.sql' and various settings like '167:54', 'CRLF', 'UTF-8', and '4 spaces'.

Disponibilidad
1

Ejemplo de pista no disponible.



The screenshot shows a database query result in a dark-themed IDE. The title bar indicates the query is 'Prueba de disponibilidad'. The result is displayed in a table with one column labeled 'Disponibilidad' and one row containing the value '0'. The status bar at the bottom shows the file path 'o-programacio-en-DDBB > Entregable.sql' and various settings like '167:54', 'CRLF', 'UTF-8', and '4 spaces'.

Disponibilidad
0



## Ejercicio 6

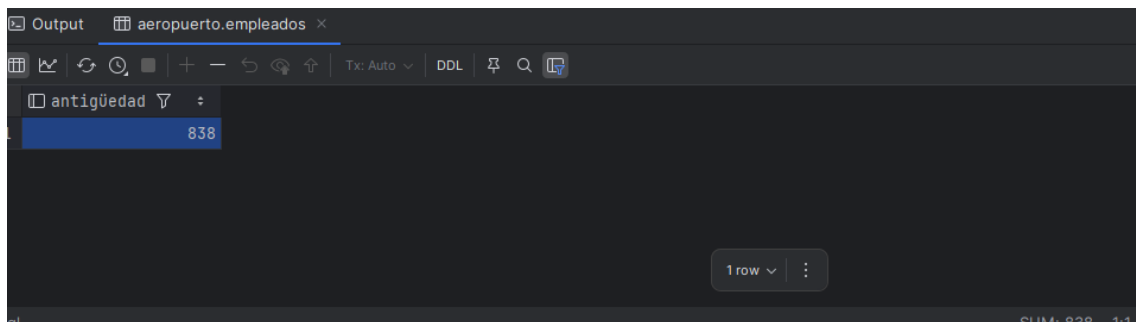
Descripción: Para resolver este ultimo ejercicio donde necesitamos actualizar la antigüedad de los empleados. Usaremos dos triggers para ello.

### Ejemplos (Capturas):

ejemplo

```
2025-04-18 18:48:30] completed in 9 ms
aeropuerto> UPDATE empleados SET fecha_contrato = '2023-01-01' WHERE dni = '87333555P'
2025-04-18 18:48:35] 1 row affected in 5 ms
```

La antigüedad del empleo que se modifio antes



antigüedad
838

## Conclusiones

Conclusiones del uso de PL/SQL en el proyecto

Ventajas: Las ventajas son que permiten una mayor facilidad en la inserción, actualización de datos al igual que permite un mayor control y filtrado de datos para evitar la introducción de datos erróneos y de paso avisa por medio de mensajes Excepciones personalizadas en que falla, también permite una mayor facilidad en la búsqueda de datos, con ejemplos como el ejercicio 2 y 4 que permiten calcular

las ganancias de los vuelos o hacer un reporte de los vuelos cursados entre ciertas fechas, evitar reutilizar consultas. Y por último permite ocultar por parte de la base de datos los nombres originales de las columnas usadas en las distintas tablas evitando los ataques de SQL injection dando un extra en la seguridad.

Limitaciones: Es un lenguaje extremadamente limitado, dificultando e imposibilitando ciertas operaciones al igual que la creación de scripts complejos, esto puede llegar a forzar el depender a futuro de medios externos para el manejo de la base de datos (Database Managers) .