

# Databases: Second Semester Project

## *La Salle AIR 2020-2021*

### List of members (name and email):

- Alex Collado Garrido - [a.collado@students.salle.url.edu](mailto:a.collado@students.salle.url.edu)
- Wesley Lucas Mas - [wesley.lucas@students.salle.url.edu](mailto:wesley.lucas@students.salle.url.edu)
- Laura Fengtan Pascual López - [laurafengtan.pascual@students.salle.url.edu](mailto:laurafengtan.pascual@students.salle.url.edu)
- Guillem Roselló Christiaen - [guillem.rosello@students.salle.url.edu](mailto:guillem.rosello@students.salle.url.edu)

Date of finalisation: 30/05/2021

### Summary of tasks

Task	Subtask	Status (completed, in progress, not started, 3/5 completed...)
Requirement 1 Flights	Queries x5	Completed
	Triggers x3	Completed
	Event x1	Completed
Requirement 2 Airports	Queries x5	Completed
	Triggers x3	Completed
	Event x1	Completed
Requirement 3 Shopping	Queries x5	Completed
	Triggers x3	Completed
	Event x1	Completed
Requirement 4 Luggage	Queries x5	Completed
	Triggers x2	Completed
	Event x1	Completed
Requirement 5 Cross-module	Queries x6	Completed
Graph database Case Study 1	Generation of CSV	Completed
	Import CSV	Completed
	Queries x6	Completed
Graph database Case Study 2	Generation of CSV	Completed
	Import CSV	Completed
	Queries x6	Completed
Conclusions	Use of resources	Completed
	Lessons learned	Completed
	Future work and conclusions	Completed

# **Index**

INTRODUCTION (1 PAGE)	7
LSAIR RELATIONAL DATABASE REQUIREMENTS	9
FLIGHT MODULE	9
Requirement (Query) 1.1 Anticipating countries	9
Solution	9
Explanation	10
Query validation	10
Requirement (Query) 1.2 Passengers and turbulences	11
Solution	11
Explanation	12
Query validation	12
Requirement (Query) 1.3 Pilots and co-pilots	14
Solution	14
Explanation	14
Query validation	15
Requirement (Query) 1.4 Very old passengers cannot communicate	16
Solution	16
Explanation	16
Query validation	17
Requirement (Query) 1.5 Window passengers	20
Solution	20
Explanation	21
Query validation	21
Requirement (Trigger) 1.6 Invalid tickets	21
Solution	21
Explanation	22
Trigger validation	23
Requirement (Trigger) 1.7 Credit card criminals	25
Solution	25
Explanation	25
Trigger validation	25
Requirement (Trigger) 1.8 Cancelled flights	27
Solution	27
Explanation	28
Trigger validation	29
Requirement (Event) 1. 9 Flight statistics	30
Solution	30
Explanation	32
Trigger validation	32
AIRPORT MODULE	33
Requirement (Query) 2.1 Airlines and petrol capacity	33
Solution	33
Explanation	34
Query validation	34
Requirement (Query) 2.2 Mechanic grades	35

Solution	35
Explanation	36
Query validation	36
Requirement (Query) 2.3 Airports and mean distance routes	37
Solution	37
Explanation	38
Query validation	38
Requirement (Query) 2.4 Airlines and routes	39
Solution	39
Explanation	40
Query validation	40
Requirement (Query) 2.5 Pieces replaced	41
Solution	41
Explanation	43
Query validation	43
Requirement (Trigger) 2.6 Routes cancelled	44
Solution	44
Explanation	44
Trigger validation	44
Requirement (Trigger) 2.7 Mechanics firings	46
Solution	46
Explanation	47
Trigger validation	48
Requirement (Trigger) 2.8 Petrol updates	52
Solution	52
Explanation	53
Trigger validation	53
Requirement (Event) 2.9 Yearly maintenance costs	56
Solution	56
Explanation	56
Trigger validation	56
<b>SHOPPING MODULE</b>	61
Requirement (Query) 3.1 Local commerce	61
Solution	61
Explanation	62
Query validation	62
Requirement (Query) 3.2 Best trade relations	64
Solution	64
Explanation	65
Query validation	65
Requirement (Query) 3.3 The best restaurant	67
Solution	67
Explanation	67
Query validation	68
Requirement (Query) 3.4 Stores and restaurants	69
Solution	69

Explanation	70
Query validation	70
Requirement (Query) 3.5 Waiting areas without shop keepers	71
Solution	71
Explanation	72
Query validation	72
Requirement (Trigger) 3.6 Waiting areas shutdown	74
Solution	74
Explanation	75
Trigger validation	76
Requirement (Trigger) 3.7 Updated products	77
Solution	77
Explanation	78
Trigger validation	78
Requirement (Trigger) 3.8 Product values per m <sup>2</sup>	78
Solution	78
Explanation	79
Trigger validation	79
Requirement (Event) 3.9 Expired food	80
Solution	80
Explanation	81
Trigger validation	81
LUGGAGE MODULE	82
Requirement (Query) 4.1 Luggage handlers' rewards	82
Solution	82
Explanation	82
Query validation	82
Requirement (Query) 4.2 Luggage details	83
Solution	83
Explanation	84
Query validation	84
Requirement (Query) 4.3 Lost objects	85
Solution	85
Explanation	85
Query validation	85
Requirement (Query) 4.4 Special objects	85
Solution	85
Explanation	86
Query validation	86
Requirement (Query) 4.5 Claims	86
Solution	86
Explanation	87
Query validation	87
Requirement (Trigger) 4.6 Refunded tickets	88
Solution	88
Explanation	88

Trigger validation	89
Requirement (Trigger) 4.7 Lost object days	89
Solution	89
Explanation	90
Trigger validation	90
Requirement (Event) 4.8 Transported statistics	90
Solution	90
Explanation	93
Trigger validation	93
CROSS MODULE QUERIES	94
Requirement (Query) 5.1 Overbooked airlines	94
Solution	94
Explanation	94
Query validation	95
Requirement (Query) 5.2 Lost objects due to jet lag	96
Solution	96
Explanation	97
Query validation	97
Requirement (Query) 5.3 Flight attendant vacancy	98
Solution	98
Explanation	99
Query validation	99
Requirement (Query) 5.4 Smuggling airports	100
Solution	100
Explanation	100
Query validation	101
Requirement (Query) 5.5 Reliable plane types	101
Solution	101
Explanation	102
Query validation	102
Requirement (Query) 5.6 Employees languages	102
Solution	102
Explanation	103
Query validation	103
LSAIR GRAPH DATABASE REQUIREMENTS	104
CASE STUDY 1: AIRPLANE, AIRPORT, CITY, AND COUNTRY	104
Data migration	104
Neo4J Data model	105
Query 1	105
Solution	105
Explanation	106
Query validation	106
Query 2	107
Solution	107
Explanation	108
Query validation	108

Query 3	109
Solution	109
Explanation	110
Query validation	110
Query 4	111
Solution	111
Explanation	111
Query validation	111
Query 5	111
Solution	112
Explanation	112
Query validation	112
CASE STUDY 2: PILOTS, FLIGHT ATTENDEE, LANGUAGE, FLIGHT AND AIRPORT	113
Data migration	113
Neo4J Data model	113
Query 1	114
Solution	114
Explanation	115
Query validation	115
Query 2	115
Solution	115
Explanation	116
Query validation	116
Solution	117
Explanation	118
Query validation	118
Query 4	118
Solution	118
Explanation	119
Query validation	119
Query 5	120
Solution	120
Explanation	121
Query validation	121
Query 6	123
Solution	123
Explanation	123
Query validation	123
CONCLUSIONS	125
USE OF RESOURCES	125
LESSONS LEARNT (1 PAGE)	127
FUTURE WORK AND CONCLUSIONS (1 PAGE)	128

## 1 Introduction (1 page)

En aquesta ocasió se'ns presenta el següent context: tenim un sistema d'anàlisi i gestor de viatges anomenat LSAIR, el qual conté informació relacionada amb els viatges, aeròports, compres i equipatge (que són els quatre mòduls que componen aquest sistema). I nosaltres, com a estudiants, se'ns ha encomanat la tasca d'anar actualitzant la base de dades del sistema a través del Sistema Gestor de Base de Dades 'MySQL', tenint en compte totes les relacions existents entre les taules dels mòduls que componen el sistema, a més de representar aquestes mateixes relacions de forma gràfica a través d'una plataforma com Neo4j.

Donat aquest context que se'ns ha presentat, els objectius d'aquest projecte són no només treballar amb el SGBD 'MySQL' implementant consultes, triggers i events, sinó també fer recerca i posar en pràctica els coneixements apresos fins ara del Neo4j per a no només fer la importació de dades del MySQL al Neo4j, sinó també per a poder graficar les relacions entre les entitats importades.

I per últim, documentar tot el procés seguit des del principi del desenvolupament del projecte fins al final d'aquest projecte a través d'una memòria que es redacta en aquest precís moment.

### **Alex Collado Garrido**

El mòdul que he realitzat ha sigut el número 4, 'Luggage', i el mòdul 5, 'Cross-module'. He fet les queries, triggers i events d'aquests mòduls, i he ajudat dins del possible amb la resta de mòduls.

En la part de Neo4j, he realitzat les queries 1.4 i 1.4 del case 1.

En la memòria, he redactat totes les explicacions del mòdul 4 i de les meves querys de Neo4j, i la query 5.2 del mòdul 5.

### **Wesley Lucas Mas**

En el meu cas, el mòdul del que m'en vaig encarregar jo de fer en aquesta ocasió va ser el de 'Airports', des de les queries normals fins les dels triggers, a més de l'event assignat.

I en relació a les queries del Neo4j, he donat suport als meus companys amb la importació de informació del Case 1, específicament amb la informació a importar. Per altra banda, també m'he encarregat de realitzar dues queries tant del case 1 (1.2 i 1.3) com del case 2 (2.3 i 2.4), a més de donar/demanar suport/ajuda als meus companys quan fossi necessari.

I per últim, la redacció de la meva part de la memòria enfocada en les queries del mòdul 'Airports' i la part conjunta dels cases.

### **Laura Pascual López**

L'apartat que jo he realitzat ha estat el del mòdul dels "Flights", m'he encarregat de les consultes, triggers i event del pertinent apartat.

A més a més, he pres gran mesura en la importació de les dades del NEO4j, he col·laborat en ambdós cases per així aconseguir tota la informació necessària per fer les consultes que se'ns demana en aquesta interfície.

També he fet dues de les consultes del case 2 del NEO4j i, al igual que tots, ens hem anat ajudant entre nosaltres amb els dubtes que sorgien.

Finalment he redactat la meva part de la memòria.

### **Guillem Roselló Christiaen**

En el meu cas, m'he ocupat del mòdul de “Shopping”, he realitzat les queries, els triggers i el event d'aquest.

A part d'això he realitzat les queries dels datasets del case study 2. Queries que han servit per crear les dades que importàvem més tard en el Neo4j. També he ajudat en la creació del script d'importació de dades del mateix Case Study.

Vem tenir uns problemes amb el Case Study 1 i vaig ajudar a arreglar-lo modificant les queries per crear els CSV i el script d'importació al Neo4j.

També he realitzat 3 queries en el Neo4j, una del Case Study 1 i dues del Case Study 2.

Finalment he redactat la part de la memòria que hem pertoca i he fet les verificacions de les queries 5.1 i 5.6 i les he redactat a la memòria.

## 2 LSAIR Relational Database requirements

Per poder executar correctament totes les queries, és necessari canviar el “DBMS connection read timeout interval” de les preferències de MySQL Workbench a 90 segons.

DBMS connection read timeout interval (in seconds):

90

The maximum amount of time the query can take to return data from the DBMS. Set 0 to skip the read timeout.

### 2.1 Flight Module

#### 2.1.1 Requirement (Query) 1.1 Anticipating countries

##### 2.1.1.1 Solution

```
(SELECT 'Most anticipating', c.name AS 'Country name', AVG(datediff(flight.date, f.date_of_purchase ))*24 AS 'difference in hours', AVG(f.price) -- c.countryID, c.name, AVG(flight.date), AVG(f.date_of_purchase), AVG((flight.date) - f.date_of_purchase), AVG(f.price)
FROM flighttickets AS f
JOIN person AS pe ON pe.personID = f.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight ON flight.flightID = f.flightID
GROUP BY c.countryID
HAVING COUNT(f.passengerID) > 300
ORDER BY AVG(datediff(flight.date, f.date_of_purchase )) DESC
LIMIT 1)

UNION

(SELECT 'Less anticipating', c.name AS 'Country name', AVG(datediff(flight.date, f.date_of_purchase ))*24 AS 'difference in hours', AVG(f.price) -- c.countryID, c.name, AVG(flight.date), AVG(f.date_of_purchase), (AVG(flight.date) - AVG(f.date_of_purchase)), AVG(f.price)
FROM flighttickets AS f
JOIN person AS pe ON pe.personID = f.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight ON flight.flightID = f.flightID
GROUP BY c.countryID
HAVING COUNT(f.passengerID) > 300
ORDER BY AVG(datediff(flight.date, f.date_of_purchase ))/*(flight.date) - f.date_of_purchase*/ ASC
LIMIT 1);
```

Most anticipating	Coutry name	difference in hours	AVG(f.price)
Most anticipating	United Kingdom	1006.2143	307.7201
Less anticipating	Netherlands	945.4599	285.5637

### 2.1.1.2 Explanation

Per aquesta consulta, he hagut de realitzar un union de dos querys gairebé idèntiques. Per aquestes queries, he agrupat segons el CountryId (el país). Gràcies a això puc afegir com a condició en el having de que cada país tingui mínim 300 persones.

Finalment, ordeno segons el país que s'anticipa més (en la primera query), o menys (a la segona query). Això ho controlo a gràcies fer el AVG de les diferències de les dates entre la sortida de l'avió i les dates de compra.

A més a més, per fer que sigui el país que més s'anticipa l'ordeno per descendent i fico el límit a 1. Posteriorment faig un union de una query gairebé identica però amb la diferència principal de que es el less anticipating, fent que s'ordeni en ascendent i també tenint el límit a 1.

### 2.1.1.3 Query validation

```
SELECT datediff(flight.date, f.date_of_purchase )*24 AS 'difference in hours', f.price, flight.date, f.date_of_purchase
FROM flighttickets AS f
JOIN person AS pe ON pe.personID = f.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight ON flight.flightID = f.flightID
ORDER BY datediff(flight.date, f.date_of_purchase ) DESC;
```

difference in hours	price	date	date_of_purchase
2376	146	2011-06-18	2011-03-11

Gràcies a aquesta query, he pogut corroborar que el càlcul de la diferència de les dates en hores funciona, ja que en aquestes dues dates passen com 3 mesos considerant cada mes uns 30 dies i 24h al dia, dona 2160h. Aquesta validació ha estat necessària perquè al principi vaig fer un càlcul d'hores que no era correcte i gràcies a aquesta query, vaig aconseguir adonar-me.

```
SELECT 'Less anticipating', c.name AS 'Country name', AVG(datediff(flight.date,
f.date_of_purchase ))*24 AS 'difference in hours', AVG(f.price), COUNT(f.passengerID) AS
'People of country' -- c.countryID, c.name, AVG(flight.date), AVG(f.date_of_purchase),
( AVG(flight.date) - AVG(f.date_of_purchase)), AVG(f.price)
FROM flighttickets AS f
JOIN person AS pe ON pe.personID = f.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight ON flight.flightID = f.flightID
GROUP BY c.countryID
HAVING COUNT(f.passengerID) > 300
ORDER BY AVG(datediff(flight.date, f.date_of_purchase ))/*(flight.date) - f.date_of_purchase*/
ASC
```

Less anticipating	Country name	difference in hours	AVG(f.price)	People of country
Less anticipating	Netherlands	945.4599	285.5637	1233
Less anticipating	Japan	954.4473	321.6093	1167
Less anticipating	Morocco	956.8352	333.6551	1183
Less anticipating	Argentina	956.8993	315.8163	1192
Less anticipating	Germany	957.7042	307.5306	1129
Less anticipating	Saudi Arabia	963.0538	264.0930	1226
Less anticipating	India	964.0327	322.6994	1101
Less anticipating	France	965.0326	305.0388	1135
Less anticipating	Australia	965.1557	305.5853	1201
Less anticipating	Canada	967.0167	329.9359	1139
Less anticipating	Switzerland	969.8835	296.7022	1202
Less anticipating	China	971.3361	321.8134	1211
Less anticipating	Norway	974.3446	334.6323	1126

Podem observar que en el nombre de personnes d'un país és major a 300.

També veiem que la que té menys hores és el país de Netherlands igual que en el resultat final.

### 2.1.2 Requirement (Query) 1.2 Passengers and turbulences

#### 2.1.2.1 Solution

```
SELECT person.personID, person.name, person.surname, person.born_date
FROM flight
JOIN flighttickets AS f ON flight.flightID = f.flightID
JOIN passenger AS p ON f.passengerID = p.passengerID
JOIN person ON p.passengerID = person.personId
JOIN status ON flight.statusID = status.statusID
WHERE status.status LIKE "Strong turbulences" AND
      person.personID NOT IN (SELECT person2.personID
                               FROM flight AS fl
                               JOIN flighttickets AS ft ON fl.flightID = ft.flightID
                               JOIN passenger AS pas ON ft.passengerID = pas.passengerID
                               JOIN person AS person2 ON pas.passengerID =
                                  person2.personId
                               WHERE fl.date > flight.date) AND
      person.personID NOT IN (SELECT flt.passengerID FROM flight AS fli
                               JOIN flighttickets AS flt ON fli.flightID = flt.flightID
                               JOIN status AS sta ON fli.statusID = sta.statusID
                               WHERE sta.status LIKE "Strong turbulences"
                               GROUP BY flt.passengerID
                               HAVING COUNT(sta.status) > 1
                               ORDER BY passengerID)
;
```

#### 2.1.2.2 Explanation

Per aquesta consulta el que he fet ha estat primer buscar en la consulta general totes aquelles persones que han estat en algun avió que ha tingut “*Strong turbulences*” . Posteriorment he fet una subquery que busca totes aquelles persones que han viatjat en avió posteriorment a la data del cop que van viatjar amb moltes turbulències per així indicar que aquestes persones no apareixen com a resultat.

Per acabar, vaig fer una altre subquery que permet que si una persona viatjava en dos avions amb grans turbulències faria que també es mostra. Per evitar això, vaig realitzar aquesta última subquery, que mostra tots els ID d'aquelles persones que han estat més d'un cop en algú avió amb “*Strong turbulences*”. Al fer un NOT IN, també em permet excloure aquestes persones de la consulta per a que es compleixi el requisit que se'ns demana.

#### 2.1.2.3 Query validation

```

SELECT person.personID, person.name, person.surname, person.born_date, COUNT(DISTINCT flight.flightID), status.status
FROM flight
JOIN flighttickets AS f ON flight.flightID = f.flightID
JOIN passenger AS p ON f.passengerID = p.passengerID
JOIN person ON p.passengerID = person.personID
JOIN status ON flight.statusID = status.statusID
WHERE status.status LIKE "Strong turbulences"
GROUP BY person.personID;

```

personID	name	surname	born_date	COUNT(DISTINCT flight.flightID)	status
1718	Brandon	Gowanson	1965-10-18	1	Strong turbulences
1730	Ambros	Habble	1936-07-26	2	Strong turbulences

Amb aquesta consulta he buscat a dues persones que hagin estat en avions de “Strong turbulences” per així poder buscar la informació pertinent

```

SELECT person.personID, person.name, person.surname, person.born_date, status.status, flight.date
FROM flight
JOIN flighttickets AS f ON flight.flightID = f.flightID
JOIN passenger AS p ON f.passengerID = p.passengerID
JOIN person ON p.passengerID = person.personID
JOIN status ON flight.statusID = status.statusID
WHERE person.personID = 1718;

```

	personID	name	surname	born_date	status	date
▶	1718	Brandon	Gowanson	1965-10-18	Strong turbulences	2016-03-06

El primer de tots podem observar que només ha viatjat un cop en avió per tant hauria de trobar-se a la query original

personID	name	surname	born_date
1718	Brandon	Gowanson	1965-10-18

I efectivament, en la query original, es troba en Brandon.

	personID	name	surname	born_date	status	date
▶	1730	Ambros	Habble	1936-07-26	Perfect	1979-05-14
	1730	Ambros	Habble	1936-07-26	Strong turbulences	1989-08-02
	1730	Ambros	Habble	1936-07-26	Strong turbulences	2018-09-15

La segona persona, Ambros, veiem que va assistir a dos Strong turbulences abans de un viatge perfecte d’avió. En aquest cas, la segona query que he realitzat hauria de fer que no aparegués aquesta persona a la consulta.

personID	name	surname	born_date
57137	Amber	holmes	1938-03-25
22216	Amby	Walles	1979-04-22

En la query original veiem que les úniques persones que el seu nom comença per (Amb) no són en Ambros, pel que ens mostra que funciona correctament.

Finalment he buscat una persona que complís amb el requisit de la meva primera subquery

personID	name	surname	born_date	COUNT(DISTINCT flight.flightID)	status
965	Elberta	Rechert	1989-08-03	2	Strong turbulences

personID	name	surname	born_date	status	date
965	Elberta	Rechert	1989-08-03	Strong turbulences	1990-05-22
965	Elberta	Rechert	1989-08-03	Little turbulences	2005-12-26

La Elberta ha viatjat amb avió un cop més després d'un viatge de 'Strong turbulences'. Per aquest motiu no hauria d'aparèixer en la query original

personID	name	surname	born_date
10003	Elbertina	Lamputt	1934-02-10

En el moment de buscar a la nostra query veiem que no es troba en els resultats de la consulta final. Només apareix una noia amb nom semblant però que no és la Elberta.

(Per trobar si estan o no a la query general, he afegit un filtre en el Where on apareguin les persones que el seu nom comença igual).

`AND person.name LIKE 'Elb%'`

### 2.1.3 Requirement (Query) 1.3 Pilots and co-pilots

#### 2.1.3.1 Solution

```
(SELECT pilot.flying_license, COUNT(flight.flightID) AS 'times she/he was pilot', pilot.grade
FROM pilot
JOIN flight ON flight.pilotID = pilot.pilotID
WHERE pilot.grade >= (SELECT 2+AVG(p.grade) FROM pilot AS p)
GROUP BY pilot.pilotID
HAVING COUNT(flight.flightID) < (SELECT COUNT(copilot.pilotID)
                                    FROM pilot AS copilot
                                    WHERE pilot.pilotID = copilot.copilotID
                                    GROUP BY copilotID)

ORDER BY pilot.pilotID);
```

flying_license	times she/he was pilot	grade
25-504-6465	1	9.76
08-440-1486	1	9.4
82-958-9279	1	9.62
15-741-5329	2	9.68
92-674-3016	1	9.5
73-636-9959	1	9.56
22-707-0728	1	9.54
54-988-7758	1	9.52
65-983-7024	1	9.42
05-564-1878	2	9.6
82-324-0620	2	9.56
60-173-1960	1	9.98
87-994-3423	1	9.64

#### 2.1.3.2 Explanation

Bàsicament en aquesta query el que faig és buscar tots aquells pilots que tinguin una puntuació major a (la mitjana de puntuacion de tots els pilots + 2). Per trobar aquesta mitjana, faig una breu subquery que em retorna quin és el valor d'aquesta mitjana + 2.

Després, faig un Group by per poder contar quants cops ha pilotat un avió. Gràcies a això, puc comparar els cops que ha pilotat amb l'altre subquery que em retorna els cops que ha estat copilot. Gràcies al ( $\geq$ ) indico que només vull mostrar aquells pilots que hagin estat més cops copilots que pilots.

Cal considerar que per saber quants cops ha estat copilot, s'ha de fer des de la pròpia taula de pilots, en canvi, per saber quants cops s'ha estat pilot, cal fer-ho en la taula de flights.

### 2.1.3.3 Query validation

pilotID	flying_license	grade	COUNT(flight.flightID)
134927	65-983-7024	9.42	1
135499	05-564-1878	9.6	2
135813	82-324-0620	9.56	2
136024	60-173-1960	9.98	1
136339	87-994-3423	9.64	1

Amb la mateixa query que la original però amb canvis en el select, trobem aquests exemples.

A més veiem que en tots la puntuació és alta i supera el AVG (grade) + 2

Amb les següents queries hem comprovat els resultats:

```
SELECT COUNT(copilot.pilotID) AS 'Cops_copilot', copilot.copilotID
FROM pilot AS copilot
-- WHERE pilot.pilotID = copilot.copilotID
GROUP BY copilot.copilotID
ORDER BY copilot.copilotID;
```

```
(SELECT pilot.pilotID, pilot.flying_license, pilot.grade, COUNT(flight.flightID) AS 'Times pilot'
FROM pilot
JOIN flight ON flight.pilotID = pilot.pilotID
WHERE pilot.grade >= (SELECT 2+AVG(p.grade) FROM pilot AS p)
GROUP BY pilot.pilotID
HAVING COUNT(flight.flightID)
ORDER BY pilot.pilotID);
```

En aquesta query buscavem quants cops ha estat com a copilot el pilot en concret

En aquesta query mirem quants cops ha sigut el pilot.

Aquí estan algun dels resultats:

	Cops_copilot	copilotID	pilotID	flying_license	grade	Times pilot
▶	2	134927	134927	65-983-7024	9.42	1
	Cops_copilot	copilotID				
	3	135499	135499	05-564-1878	9.6	2
	Cops_copilot	copilotID				
	3	135813	135813	82-324-0620	9.56	2

En totes aquestes imatges podem veure que 's el mateix pilot la foto de la esquerre com de la dreta perquè el seu ID coincideix.

A més a més, es pot apreciar que tots ells han estat més cops com a copilots que com pilots.

Per aquest motiu apareixen a la query original, perquè a més el veu que la puntuació ho compleix ja

2+AVG(p.grade)

que el AVG de les puntuacions és de:

SELECT 2+AVG(p.grade) FROM pilot AS p

9.399678428592708

Un exemple de que no compleixi amb lo demanat per entrar a la query seria aquest:

Cops_copilot	copilotID	pilotID	flying_license	grade	Times pilot
1	134804	134804	87-871-0491	9.98	4

Veiem que ha estat més cops de pilot que de copilot per tant no hauria d'entrar a la consulta

Result Grid	Filter Rows:	1348	Export:	Wrap Cell Content:
pilotID	flying_license	grade	COUNT(flight.flightID)	
134864	54-988-7758	9.52	1	

En el moment de buscar-lo a la query original, no apareix. Complint així els requisits.

### 2.1.4 Requirement (Query) 1.4 Very old passengers cannot communicate

#### 2.1.4.1 Solution

```

SELECT oldperson.name, oldperson.surname, oldperson.born_date
FROM passenger
JOIN person AS oldperson ON passenger.passengerID = oldperson.personID
JOIN flighttickets ON flighttickets.passengerID = oldperson.personID
WHERE (DATE_FORMAT(NOW(), '%Y') - DATE_FORMAT(oldperson.born_date, '%Y')) >= 100 AND
    NOT EXISTS (SELECT l.languageID FROM languageperson AS l
                 WHERE l.personID = oldperson.personID AND
                     EXISTS (SELECT DISTINCT languageID
                             FROM flight_attendant
                             JOIN person ON person.personID = flight_attendant.flightattendantID
                             JOIN languageperson AS lp ON lp.personID = person.personID
                             JOIN flight_flightattendant AS ff ON ff.flightattendantID =
                                 flight_attendant.flightattendantID
                             WHERE ff.flightID = flighttickets.flightID))
;

```

name	surname	born_date
Gabbi	Sparway	1920-01-01
Melisandra	Lanfranchi	1920-01-01
Astra	Bangle	1920-01-01
Philipa	Kasting	1920-01-01
Hyatt	Jersch	1920-01-02
Khalil	Meins	1920-01-03
Jacquie	Ibbett	1920-01-05
Phil	Dudmesh	1920-01-05
Aleda	Godehard...	1920-01-09
Mitchell	Wheowall	1920-01-10
Alejoa	Lenahan	1920-01-10
Olivero	Simonot	1920-01-10
Bunny	Antic	1920-01-11
Dorothy	Tremaine	1920-01-11

#### 2.1.4.2 Explanation

Per aquesta query he afegit la condició de que el la resta entre anys de quan van néixer i ara, sigui de 100 o major per així complir aquest requisit.

A més a més he afegit una condició per trobar totes aquelles persones que en el seu vol, no es poden comunicar per causa de l'idioma. Per fer això, al principi vaig fer un NOT IN amb els idiomes que parlaven les flightattendant d'aquell vol, però vaig trobar que si aquella persona parlava més d'un idioma, per tots aquells idiomes que no coincidissin, s'afegeix a la llista, fent que fos incorrecte.

Per aquest motiu, vaig decidir modificar-ho ficant el NOT EXISTS, considerant que No existeix cap idioma del passatger vell, que Existeix en els idiomes de les flight Attendant.

Aquesta doble subquery busca tots els llenguatges del passatger i mira si existeixen a la taula de la flight Attendant, si cap d'ells es troba en aquell llistat, s'afegeirà a la consulta.

#### 2.1.4.3 Query validation

M'he fixat que la majoria de resultats són de persones que no en saben cap idioma, en aquest sentit, tindria sentit la query ja que si no saben cap idioma, no es poden comunicar. Aquí mostrem alguns exemples:

personID	name	surname	born_date	flightID
23414	Gabbi	Sparway	1920-01-01	10007
23414	Gabbi	Sparway	1920-01-01	12090
73122	Melisandra	Lanfranchi	1920-01-01	12554
9791	Astra	Bangle	1920-01-01	14819
57791	Philipa	Kasting	1920-01-01	14360
57536	Hyatt	Jersch	1920-01-02	10019
949	Khalil	Meins	1920-01-03	10011
63672	Jacquie	Ibbett	1920-01-05	64
44879	Phil	Dudmesh	1920-01-05	90
44879	Phil	Dudmesh	1920-01-05	17229
5282	Aleda	Godehard...	1920-01-09	114

Aquesta és la mateixa query que la original però afegint informació en el SELECT.

Amb aquesta consulta comprovo quins idiomes parla el passatger:

```
SELECT l.languageID, l.personID FROM languageperson AS l
WHERE l.personID = 23414;
```

languageID	personID
NULL	NULL
NULL	NULL

Al principi com no estava segura, vaig fer una altre consulta que corroborés que no parla cap idioma

```
SELECT flightID, oldperson.personID, oldperson.name, oldperson.surname, oldperson.born_date, languageID, DATE_FORMAT(NOW(), '%Y'), DATE_FORMAT(oldper
FROM person AS oldperson
JOIN flighttickets ON flighttickets.passengerID = oldperson.personID
LEFT JOIN languageperson AS l ON l.personID = oldperson.personID
WHERE oldperson.personID = 23414 AND
      DATE_FORMAT(NOW(), '%Y') - DATE_FORMAT(oldperson.born_date, '%Y') >= 100
ORDER BY oldperson.personID;
```

flightID	personID	name	surname	born_date	languageID	DATE_FORMAT(NOW(), '%Y')	DATE_FORMAT(oldperson.born_date, '%Y')	DATE_FORMAT(NOW(), '%Y') - DATE_FORMAT(oldperson.born_date, '%Y')
10007	23414	Gabbi	Sparway	1920-01-01	NULL	2021	1920	101
12090	23414	Gabbi	Sparway	1920-01-01	NULL	2021	1920	101

Gràcies a un LEFT JOIN puc comprovar-ho, veig que compleix l'edat, que ha viatjat en els mateixos vols que en la query original...

```
SELECT * FROM flight_flightattendant AS ff
JOIN languageperson ON languageperson.personID = ff.flightattendantID
WHERE ff.flightID = 10007 OR ff.flightID = 12090;
```

flightID	flightAttendantID	languageID	personID
12090	141808	12	141808
12090	141808	21	141808
12090	141808	26	141808
12090	141808	30	141808
12090	141808	55	141808
12090	145943	26	145943

Respecte aquests avions veiem que les seves flightAttendant parlen aquests idiomes, però clar, com el passatger no en sap cap, no es pot comunicar.

D'aquest casos n'he trobat molíssims, he anat comprobant manualment a veure si troava algun que no fos perquè el passatger no parlés cap idioma i només en vaig trobar 1. Per aquest motiu, vaig arribar a tal punt de deixar de mira-ho manualment i canviar la query original que tenia per validar (amb més elements el SELECT)

```

SELECT DISTINCT oldperson.personID, oldperson.name, oldperson.surname, oldperson.born_date, flighttickets.flightID
FROM person AS oldperson
JOIN languageperson AS l ON l.personID = oldperson.personID
JOIN flighttickets ON flighttickets.passengerID = oldperson.personID
WHERE (DATE_FORMAT(NOW(), '%Y') - DATE_FORMAT(oldperson.born_date, '%Y')) >= 100 AND
NOT EXISTS (SELECT l.languageID FROM languageperson AS l
WHERE l.personID = oldperson.personID AND
EXISTS (SELECT DISTINCT lp.languageID
FROM flight_attendant
JOIN person ON person.personID = flight_attendant.flightattendantID
JOIN languageperson AS lp ON lp.personID = person.personID
JOIN flight_flightattendant AS ff ON ff.flightattendantID = flight_attendant.flightattendantID
WHERE ff.flightID = flighttickets.flightID))

```

Simplement vaig fer un JOIN amb languageperson per a que només em mostra aquells passatgers que sí que parlen idiomes.

(D'aquesta query va passar de tenir 1149 resultats a 288 resultats)

```

✓ 243 13:40:12 SELECT DISTINCT oldperson.personID, oldperson.name, oldperson.surname, oldperson.born_date, flighttickets.flightID FROM person ... 1149 row(s) returned
✓ 244 13:41:15 SELECT DISTINCT oldperson.personID, oldperson.name, oldperson.surname, oldperson.born_date, flighttickets.flightID FROM person ... 288 row(s) returned

```

Un cop vaig fer això vaig decidir fer alguna comprovació amb algun dels nous resultats.

personID	name	surname	born_date	flightID
4146	Vernon	Clapton	1920-04-20	1631
46589	Thorny	Cicchillo	1921-07-08	4444

languageID	personID
2	4146
21	46589

```

SELECT l.languageID, l.personID FROM languageperson AS l
WHERE l.personID = 4146 OR l.personID = 46589;

```

Un cop tenia el llenguatge de les persones vaig anar a buscar el de les flightAttendant, però vaig veure que no m'apareixia res:

```

SELECT * FROM flight_flightattendant AS ff
LEFT JOIN languageperson ON languageperson.personID = ff.flightattendantID
WHERE ff.flightID = 1631 OR ff.flightID = 4444;

```

flightID	flightAttendantID	languageID	personID
----------	-------------------	------------	----------

Aquest fet em va fer dubtar pel que vaig buscar quins flight attendants de l'avió i vaig veure que no

```

SELECT flight.flightID, flight_flightattendant.flightAttendantID FROM flight
LEFT JOIN flight_flightattendant ON flight.flightID = flight_flightattendant.flightID
WHERE flight.flightID = 1631 OR flight.flightID = 4444;

```

n'hi havien, sortia NULL en el seu ID:

flightID	flightAttendantID
1631	NULL
4444	NULL

Al igual que anteriorment vaig buscar varis de manera manual però també em sortien sense flightattendant així que com anteriorment vaig modificar la query un altre cop perquè no em sortissin els avions que no tenien flight Attendant per així poder trobar els altres i fer la correcta validació.

Cal considerar que és correcte que m'apareguin aquests resultats a la query perquè si un passatger va en un avió sense Flight Attendant, tampoc es podrà comunicar.

```

SELECT DISTINCT oldperson.personID, oldperson.name, oldperson.surname, oldperson.born_date, flighttickets.flightID
FROM person AS oldperson
JOIN languageperson AS l ON l.personID = oldperson.personID
JOIN flighttickets ON flighttickets.passengerID = oldperson.personID
WHERE (DATE_FORMAT(NOW(), '%Y') - DATE_FORMAT(oldperson.born_date, '%Y')) >= 100 AND
    NOT EXISTS (SELECT 1.languageID FROM languageperson AS l
                 WHERE l.personID = oldperson.personID AND
                 EXISTS (SELECT DISTINCT lp.languageID
                         FROM flight_attendant
                         JOIN person ON person.personID = flight_attendant.flightattendantID
                         JOIN languageperson AS lp ON lp.personID = person.personID
                         JOIN flight_flightattendant AS ff ON ff.flightattendantID = flight_attendant.flightattendantID
                         WHERE ff.flightID = flighttickets.flightID))

    AND flighttickets.flightID IN (SELECT DISTINCT ffli.flightID FROM flight_flightattendant AS ffli
/*JOIN languageperson AS lff ON ffli.flightAttendantID = lff.personID*/);

```

El canvi que vaig fer a la query de validació va ser afegir aquesta subquery en el WHERE la qual em permetia que només veiés els avions amb flightattendant. (Ara sortien 22 resultats)

Vaig tornar a fer el procés anterior:

personID	name	surname	born_date	flightID
20365	Mordy	Odhams	1920-09-08	38
48203	Elia	Pochon	1920-01-13	10017

```

SELECT l.languageID, l.personID FROM languageperson AS l
WHERE l.personID = 20365 OR l.personID = 48203;

SELECT * FROM flight_flightattendant AS ff
LEFT JOIN languageperson ON languageperson.personID = ff.flightattendantID
WHERE ff.flightID = 38 OR ff.flightID = 10017;

```

languageID	personID	flightID	flightAttendantID	languageID	personID
21	20365	38	144829	NULL	NULL
21	48203	10017	143950	NULL	NULL
58	48203				
NULL	NULL				

Aquest cop em vaig adonar que ara eren els Flight Attendant que no tenien idioma.

Aquests valors també serien correctes perquè si algun passatger va a un avió o els flight attendant no en saben cap idioma, tampoc es podran comunicar.

Finalment vaig seguir comprovant però veia també que els resultats que veia també tots eren que les flight attendant no sabien cap idioma. Per aquest motiu, vaig decidir remodificar la consulta per a que no apareguessin i trobar els flight Attendant que si que sapiguessin algún idioma.

```
(SELECT DISTINCT ffli.flightID FROM flight_flightattendant AS ffli
/*JOIN languageperson AS lff ON ffli.flightAttendantID = lff.personID*/);
```

bàsicament és el que està com a comentari.

Un cop vaig compilar, em va donar com a resultat 0 rows, per tant vaig concloure que només els avis de més de 100 anys que no s'han pogut comunicar han estat:

Aquells que no saben cap idioma

Aquells viatges on no ha hagut azafatas

Aquells viatges on les azafatas no sabien cap idioma.

### 2.1.5 Requirement (Query) 1.5 Window passengers

#### 2.1.5.1 Solution

```
SELECT DISTINCT person.name, person.surname  
FROM flighttickets  
JOIN person ON personID = flighttickets.passengerID  
JOIN checkin ON checkin.flightTicketID = flighttickets.flightTicketID  
WHERE flighttickets.business = 1 AND  
    personID NOT IN (SELECT person2.personID  
                      FROM person AS person2  
                     JOIN flighttickets AS ft ON ft.passengerID = person2.personID  
                     JOIN checkin AS checkin2 ON checkin2.flightticketID = ft.flightticketID  
                     WHERE (checkin2.seat NOT LIKE 'A') AND (checkin2.seat NOT LIKE 'F'))  
  
ORDER BY person.personID;
```

name	surname
Katrinka	Barstow
Anselma	Dioniso
Elnar	Negri
Bordy	Beaver
Mortie	Sidle
Julianna	Lepper
Mateo	Zielinski
Archibold	Outright
Doloritas	Crother
Nickie	Rosenschein
Joli	Beglin
Graeme	Carmichael
Leonidas	Sitlington
Wenda	Stigger
Bartram	Gillean
Brannon	Siddle
Marcelo	Schimann

#### 2.1.5.2 Explanation

Per fer aquesta consulta vaig afegir com a condició de que el flighttickets.business fos = 1 perquè així, implicaria que si es compleix un cop, vol dir que com a mínim ha anat en business 1 cop.

A més a més, vaig fer una subquery que m'indiqués quines de les persones no havien anat en el seient A i F, permetent que fes com un (Minus) i pogués mostrar totes aquelles persones que haguessin fet mínim 1 viatge i sempre haguessin viatjat en un seient A o F.

### 2.1.5.3 Query validation

personID	name	surname	business	seat
530	Katrinka	Barstow	1	A
885	Anselma	Dioniso	1	A
894	Elnar	Negri	1	A
1104	Bordy	Beaver	1	F
1284	Mortie	Sidle	1	F

Per fer la validació, primer vaig fer ús de la query original afegint-li elements en el SELECT

Dels següents resultats vaig fer aquesta query perquè em mostrés tota la informació necessària per validar la query

```
-- primer miro que en totes hagi estat en el seient A o F i mínim un business
SELECT person.personID, person.name, person.surname , flighttickets.business, checkin.seat
FROM flighttickets
JOIN person ON personID = flighttickets.passengerID
JOIN checkin ON checkin.flightTicketID = flighttickets.flightTicketID
WHERE person.personID = 530 OR person.personID = 885 OR person.personID = 894 OR person.personID = 1104 OR person.personID = 1284
ORDER BY person.personID;
```

personID	name	surname	business	seat
530	Katrinka	Barstow	1	A
885	Anselma	Dioniso	1	A
894	Elnar	Negri	0	A
894	Elnar	Negri	1	A
1104	Bordy	Beaver	1	F
1284	Mortie	Sidle	1	F

Amb aquesta validació vaig poder veure que tots els seients eren de tipus A i F, que mínims havien anat en business 1 cop (El Elnar ha anat 1 cop business i un altre sense business)

### **2.1.6 Requirement (Trigger) 1.6 Invalid tickets**

#### **2.1.6.1 Solution**

```
DROP TABLE IF EXISTS TicketError;
CREATE TABLE TicketError (
    ticketErrorID SERIAL,
    personId INTEGER,
    name VARCHAR(255),
    surname VARCHAR(255),
    flightID INTEGER,
    dateOffFlight DATE,
    dateOfTheTicketPurchase DATE,
    PRIMARY KEY (ticketErrorID)
);
```

```
DROP TABLE IF EXISTS FlightticketsDelete;
CREATE TABLE FlightticketsDelete (
    flightticketID INTEGER
);
```

```
DELIMITER $$

DROP TRIGGER IF EXISTS lsair.invalid_tickets $$

CREATE TRIGGER invalid_tickets AFTER INSERT ON flighttickets
FOR EACH ROW BEGIN

    IF NEW.date_of_purchase > (SELECT flight.date FROM flight WHERE flight.flightID = NEW.flightID) THEN

        INSERT INTO TicketError (personID, name, surname, flightID, dateOffFlight,
        dateOfTheTicketPurchase)
        SELECT NEW.passengerID, person.name, person.surname, NEW.flightID,
        flight.date, NEW.date_of_purchase
        FROM person
        JOIN flight
        WHERE person.personID = NEW.passengerID AND
        flight.flightID = NEW.flightID;

        INSERT INTO FlightticketsDelete(flightticketID)
        VALUES (NEW.flightticketID);

    END IF ;

END $$

DELIMITER ;
```

```
DELIMITER $$

DROP EVENT IF EXISTS DeleteandoFlighttickets $$

CREATE EVENT DeleteandoFlighttickets
ON SCHEDULE EVERY 30 SECOND
DO BEGIN
```

```

DELETE flighttickets
FROM flighttickets
JOIN flightticketsDelete ON FlightticketsDelete.flightTicketID =
flighttickets.flightTicketID;

END $$
```

#### 2.1.6.2 Explanation

En aquest trigger he creat dues taules, la primera d'elles “*Ticket Error*” és la que se’ns demana a l'enunciat. La segona taula ha estat creada com a auxiliar ja que no podem fer un delete de la mateixa taula amb la que cridem el trigger, per tant, aquesta taula ens servirà per guardar quin ID de flighttickets cal borrar.

He decidit utilitzar el ID perquè és un atribut serial, és a dir, mai es repeteix, ni encara que algun ID s’hagi borrat aquell no tornarà a aparèixer.

Després de la creació de la taula, fem el trigger que s’executarà cada cop que rebi un INSERT a la taula flighttickets. Un cop entri al trigger, es troba amb un condicional que tracta si s’ha comprat el bitllet d’avió abans de que hagi sortit l’avió o si s’ha comprat un bitllet d’un vol ja fet.

En el cas de que el dia del vol ja hagués passat, entra dins el condicional i es fa un insert a la taula de TicketError tota la informació que es demana. Posteriorment es fa un INSERT a la taula auxiliar que aquesta ens ajudarà a borrar la fila que no desitgem.

Sortint del trigger ens trobem un EVENT auxiliar que gràcies a la taula de auxiliar creada, podrem esborrar les files de flighttickets que se’ns demana a l’enunciat.

#### 2.1.6.3 Trigger validation

Per fer la validació, he creat el mateix trigger i el mateix event però amb noms diferents i amb la taula que detecta el trigger diferent. Aquesta taula és una còpia de flighttickets, té les mateixes columnes, i la mateixa informació però, aquesta no conserva les claus primàries i foranes pel que ens permet actuar sense tenir els problemes de incompatibilitat per les dependències amb les altres taules.

```
CREATE TABLE flighttickets_trigger SELECT * FROM flighttickets;
```

```

DELIMITER $$

DROP TRIGGER IF EXISTS lsair.invalid_tickets_validation $$

CREATE TRIGGER invalid_tickets_validation AFTER INSERT ON flighttickets_trigger
BEGIN

FOR EACH ROW BEGIN

    IF NEW.date_of_purchase > (SELECT flight.date FROM flight WHERE flight.flightID = NEW.flightID) THEN

        INSERT INTO TicketError (personID, name, surname, flightID, dateOfFlight, dateOfTheTicketPurchase)
        SELECT NEW.passengerID, person.name, person.surname, NEW.flightID, flight.date, NEW.date_of_purchase
        FROM person
        JOIN flight
        WHERE person.personID = NEW.passengerID AND
              flight.flightID = NEW.flightID;

        INSERT INTO FlightticketsDelete(flightticketID)
        VALUES (NEW.flightticketID);

    END IF;

END $$

DELIMITER ;

```

Per fer les següents comprovacions vaig escollir tres compres de tiquets i vaig decidir que 2 d'ells tindrien el problema de comprar bitllets d'avions que ja s'han anat, però que un altre ho compra correctament

FlightID	date	Faig que entri en el trigger (compra bitllet després de que surti l'avió)	compra bitllet	passenger
6839	2021-03-08	No	2021-03-07	755
207669	2021-03-05	Si	2021-03-07	756
429	2021-03-04	Si	2021-03-07	757

El primer insert és el que la compra és correcte, i els 2 de sota, són els que han d'entrar a TicketError

```

INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase)
VALUES (100005, 755, 6839, 100, 0, "2021-03-07");
INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase)
VALUES (100006, 756, 207669, 100, 1, "2021-03-07");
INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase)
VALUES (100007, 757, 429, 100, 0, "2021-03-07");

```

Un cop faig el INSERT, m'indica el compilador que s'ha fet els 3 INSERTS correctament:

```

546 23:04:21 INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase) V... 1 row(s) affected
547 23:04:21 INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase) V... 1 row(s) affected
548 23:04:21 INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase) V... 1 row(s) affected

```

Si ara entro en la taula de TicketError:

8 • `SELECT * FROM ticketerror;`

9

Result Grid   Filter Rows: [ ]   Edit: [ ] [ ] [ ]   Export/Import: [ ] [ ]   Wrap Cell Content: [ ]						
ticketErrorID	personId	name	surname	flightID	dateOffFlight	dateOfTheTicketPurchase
1	756	Emelia	Skoughman	20769	2021-03-05	2021-03-07
2	757	Jeddy	Polamontayne	429	1966-01-26	2021-03-07
NUL	NUL	NUL	NUL	NUL	NUL	NUL

Veiem que inserta correctament aquells que han fet la compra amb un bitllet dolent.

15 • `SELECT * FROM flightticketsDelete;`

16

Result Grid   Filter Rows: [ ]   Export: [ ]   Wrap Cell Content: [ ]	
flightticketID	
100006	
100007	

Si mirem també la taula auxiliar, veiem que es troben els ID dels flightticket que s'han d'eliminar

14 • `SELECT * FROM flighttickets_trigger WHERE flightTicketID>=100005;`

<

Result Grid   Filter Rows: [ ]   Export: [ ]   Wrap Cell Content: [ ]						
	flightTicketID	passengerID	flightID	price	business	date_of_purchase
▶	100005	755	6839	100	0	2021-03-07

Finalment comprovem que dels 3 ID que he afegit ens els inserts (100005, 100006, 100007) només apareix el que ha fet la bona compra, indicant-nos que el EVENT ha eliminat de la taula aquells que no havien d'estar a la taula. Fent així que es compleixi les condicions de l'enunciat.

### **2.1.7 Requirement (Trigger) 1.7 Credit card criminals**

#### **2.1.7.1 Solution**

```
DROP TABLE IF EXISTS CrimeSuspect;
CREATE TABLE CrimeSuspect (
    crimeSuspectID SERIAL,
    passengerId INTEGER,
    name VARCHAR(255),
    surname VARCHAR(255),
    passport VARCHAR(11),
    phone VARCHAR(20),
    PRIMARY KEY (crimeSuspectID)
);
```

```
DELIMITER $$
```

```
DROP TRIGGER IF EXISTS possible_criminal $$  
CREATE TRIGGER possible_criminal AFTER INSERT ON passenger  
FOR EACH ROW BEGIN  
  
    IF NEW.creditCard IN (SELECT creditCard FROM passenger WHERE NEW.passengerID  
    <> passengerID) THEN  
  
        INSERT INTO CrimeSuspect (passengerId, name, surname, passport, phone)  
        SELECT personID, name, surname, passport, phone_number  
        FROM person WHERE personID = NEW.passengerId;  
  
    END IF;  
  
END $$  
DELIMITER ;
```

#### **2.1.7.2 Explanation**

Primer el que fem és crear la taula que se'ns demana insertar valors en el trigger. Després creem un trigger que vigila si es fa algun INSERT a la taula passatger.

El primer que trobem en el trigger és un condicional amb una query. Aquest condicional busca si la targeta de crèdit nova que ens estan insertant es troba ja assignada a alguna persona diferent. En el cas de que si que es troba, s'ha de insertar la informació que se'ns demana a la taula que hem creat "Crime Suspect".

Considero que el INSERT era prou senzill pel que no considero que faci falta explicar els detalls.

#### **2.1.7.3 Trigger validation**

Per fer aquesta validació, he creat un trigger i unes variables idèntiques però sense els índexs (PK o FK) per així, no haver de modificar els valors reals de les taules.

```

DELIMITER $$

DROP TRIGGER IF EXISTS possible_criminal_validation $$

CREATE TRIGGER possible_criminal_validation AFTER INSERT ON passenger_trigger
FOR EACH ROW BEGIN

    IF NEW.creditCard IN (SELECT creditCard FROM passenger_trigger WHERE NEW.passengerID <> passengerID) THEN

        INSERT INTO CrimeSuspect (passengerId, name, surname, passport, phone)
        SELECT personID, name, surname, passport, phone_number
        FROM person WHERE personID = NEW.passengerId;

    END IF;

END $$

DELIMITER ;

```

```
CREATE TABLE passenger_trigger SELECT * FROM passenger;
```

Per fer la validació em vaig fer una taula d'informació on indicava quins haurien d'entrar en el condicional i ficar-se la informació en la nova taula i quins no:

Passenger_ID	Credit_card	NEW passenger_ID
512	3551506430106933	514
513	6767477861265965621	512
514	4017956377698135	513

INSERT		
Passenger_ID	Credit_card	
512	512	

Principalment, vaig agafar la taula de passenger i vaig mirar els seus valors. Després vaig decidir intercanviar els ID de les persones que tenien les targetes que anteriorment havia escollit, d'aquesta manera al fer el trigger trobaria que s'inserten persones amb targetes de crèdit ja agregades.

Finalment, vaig fer un insert on la credit card no podia coincidir amb cap altre i per tant no hauria de afegir-se a la taula

```

INSERT INTO passenger_trigger (passengerID, creditCard)
VALUES (512, 512);

INSERT INTO passenger_trigger (passengerID, creditCard)
VALUES (514, 3551506430106933);

INSERT INTO passenger_trigger (passengerID, creditCard)
VALUES (512, 6767477861265965621);

INSERT INTO passenger_trigger (passengerID, creditCard)
VALUES (513, 4017956377698135);

```

```

45 •   SELECT * FROM passenger_trigger WHERE (passengerID = 512 OR passengerID = 513 OR passengerID = 514)
46     ORDER BY passengerID;
47

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
passengerID	creditCard			
512	3551506430106933			
512	512			
512	6767477861265965621			
513	6767477861265965621			
513	4017956377698135			
514	4017956377698135			
514	3551506430106933			

Aquí podem observar les insercions noves + els valors que ja estaven anteriorment.

```

49 •   SELECT * FROM CrimeSuspect;
50

```

Result Grid						Filter Rows:	Edit:	Export/Import:
crimeSuspectID	passengerId	name	surname	passport	phone			
1	514	Vergil	Krier	788-91-7454	+60 297 680 5537			
2	512	Eustace	Poluzzi	218-96-0899	+48 362 430 8087			
3	513	Tanhyia	Skull	247-41-8624	+7 352 152 5911			
NULL	NULL	NULL	NULL	NULL	NULL			

Finalment veiem que els 3 passatgers s'han afegit correctament.

També podem observar que si busquem aquestes persones a la taula de persona, la informació coincideix.

```

48 •   SELECT * FROM person WHERE (personID = 512 OR personID = 513 OR personID = 514) ;
49

```

Result Grid										Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:	
personID	name	surname	countryID	passport	email	phone_number	born_date	sex						
512	Eustace	Poluzzi	206	218-96-0899	eustace.poluzzi@statcounter.com	+48 362 430 8087	1926-05-12	M						
513	Tanhyia	Skull	106	247-41-8624	tanhyia.skull@huffingtonpost.com	+7 352 152 5911	1994-05-21	F						
514	Vergil	Krier	171	788-91-7454	vergil.krier@wufuu.com	+60 297 680 5537	1987-02-04	M						
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL						

### **2.1.8 Requirement (Trigger) 1.8 Cancelled flights**

#### **2.1.8.1 Solution**

```
DROP TABLE IF EXISTS CancelledFlightsMails;
CREATE TABLE CancelledFlightsMails(
    cancelledFlightsMailsID SERIAL,
    flightID DOUBLE,
    personId INTEGER,
    namePerson VARCHAR(255),
    emailPerson VARCHAR(255),
    priceOfTicket FLOAT,
    isBusinessTicket INTEGER,
    comission FLOAT,
    PRIMARY KEY (cancelledflightsMailsID)
);
```

```
DROP TABLE IF EXISTS CancellationCost;
```

```
CREATE TABLE CancellationCost(
    cancellationCostID SERIAL,
    flightID DOUBLE,
    refund FLOAT,
    cancelledFlightsMailsID INTEGER,
    PRIMARY KEY (cancellationCostID)
```

```
);
```

```
DELIMITER $$
```

```
DROP TRIGGER IF EXISTS CancelledFlights $$
```

```
CREATE TRIGGER CancelledFlights BEFORE DELETE ON flight
FOR EACH ROW BEGIN
```

```
    IF OLD.date > NOW() THEN
```

```
        INSERT INTO CancelledFlightsMails (flightID, personId, namePerson,
        emailPerson, priceOfTicket, isBusinessTicket, comission)
        SELECT OLD.flightID, flighttickets.passengerID, person.name, person.email,
        flighttickets.price, flighttickets.business, DATEDIFF(flight.date, NOW())
        FROM flighttickets JOIN person ON person.personID = flighttickets.passengerID
        JOIN flight ON flight.flightID = flighttickets.flightID
        WHERE flighttickets.flightID = OLD.flightID;
```

```
        INSERT INTO CancellationCost (flightID, refund)
        SELECT OLD.flightID, SUM(POW(c.priceOfTicket,2)*(c.comission+1))
        FROM CancelledFlightsMails AS c WHERE c.flightID = OLD.flightID
        GROUP BY c.flightID;
```

```
    END IF;
```

```
END $$
```

```
DELIMITER ;
```

### 2.1.8.2 Explanation

Per aquest trigger primer s'ha creat dues taules tal i com se'ns demanava a l'enunciat, per així, poder ficar la informació dels triggers.

Un cop es detecta que es farà el delete en alguna fila de flight, el trigger salta i comprova si aquell avió que estan borrant, ja fa temps que va fer el seu viatge, o encara no ha despegat. Nosaltres hem ficat aquest condicional perquè hem considerat que no tindria sentit anular un vol si aquest ja ha estat fet.

Un cop fet això, es procedeix a fer els inserts corresponents a les taules que hem creat, la de “CancelledFlightsMails” i “CancellationCost” .

Per el primer insert per la taula dels correus només crec que cal comentar que necessitava una altre taula, la de person per poder fer les insercions correctament. A més de necessitar el trigger com a BEFORE per poder utilitzar la taula que s'esborrà la fila.

Pel INSERT del cancellation, s'ha emprat la fórmula que se'ns donava a l'enunciat, a més de fer un sumatori de totes les persones per així trobar el cost total de la cancel·lació del vol.

### 2.1.8.3 Trigger validation

Igual que en els trigger anteriors, he creat un trigger idèntic de validació però amb entitats amb els mateixos valors que les originals però sense les Pk i FK

```
DELIMITER $$  
DROP TRIGGER IF EXISTS CancelledFlights_validation $$  
CREATE TRIGGER CancelledFlights_validation BEFORE DELETE ON flight_trigger  
FOR EACH ROW BEGIN  
  
    IF OLD.date > NOW() THEN  
  
        INSERT INTO CancelledFlightsMails (flightID, personId, namePerson, emailPerson, priceOfTicket, isBusinessTicket, comission)  
        SELECT OLD.flightID, flighttickets_trigger.passengerID, person.name, person.email, flighttickets_trigger.price, flighttickets_trigger.business, DATEDIFF(flight_trigger.date, NOW()) * 10  
        FROM flighttickets_trigger JOIN person ON person.personID = flighttickets_trigger.passengerID  
        JOIN flight_trigger ON flight_trigger.flightID = flighttickets_trigger.flightID  
        WHERE flighttickets_trigger.flightID = OLD.flightID;  
  
        INSERT INTO CancellationCost (flightID, refund)  
        SELECT OLD.flightID, SUM(POW(c.priceOfTicket,2)*(c.comission+1))  
        FROM CancelledFlightsMails AS c WHERE c.flightID = OLD.flightID  
        GROUP BY c.flightID;  
  
    END IF;  
  
    CREATE TABLE flight_trigger SELECT * FROM flight;  
    CREATE TABLE flighttickets_trigger SELECT * FROM flighttickets;
```

Per aquest trigger he creat una copia de flight i de flighttickets.

un cops tenim aquestes taules creades, insertem les dades necessàries per després comprobar si el delete funciona:

```

INSERT INTO flight_trigger (flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)
VALUES (1122334455, 1122334455, 1122334455, 1122334455, "2021-05-31", "LB", 1122334455, "15:00:00", 1),
(2233445566, 2233445566, 2233445566, 2233445566, "2021-06-2", "LB", 2233445566, "15:00:00", 1);

INSERT INTO flighttickets_trigger (flightticketID, passengerID, flightID, price, business, date_of_purchase)
-- FlightID --> 1122334455
VALUES (100000, 512, 1122334455, 100, 0, "2021-04-20"),
(100001, 513, 1122334455, 100, 1, "2021-04-15"),
(100002, 514, 1122334455, 90, 0, "2021-04-12"),
(100003, 515, 1122334455, 85, 0, "2021-04-05"),
(100004, 516, 1122334455, 100, 0, "2021-04-15"),
-- flightID --> 2233445566
(100000, 600, 2233445566, 200, 0, "2021-03-20"),
(100001, 601, 2233445566, 200, 1, "2021-03-15"),
(100002, 602, 2233445566, 190, 0, "2021-03-12"),
(100003, 603, 2233445566, 185, 0, "2021-03-05"),
(100004, 604, 2233445566, 200, 0, "2021-03-15"),
(100002, 602, 2233445566, 190, 0, "2021-03-12"),
(100003, 603, 2233445566, 285, 0, "2021-05-20"),
(100004, 604, 2233445566, 300, 0, "2021-05-30");

```

Primer creem dos vols diferents amb les seves dades qualsevols.

Un cop fet això, creem varis flighttickets que han estat venuts per aquests avions inventats, fent que la ID dels avions coincideixi amb els que hem creat abans.

Un cop insertats, veiem que la inserció és correcte:

flightID	pilotID	planeID	routeID	date	gate	fuel	departure_hour	statusID
2233445566	2233445566	2233445566	2233445566	2021-06-02	LB	2233445566	15:00:00	1
1122334455	1122334455	1122334455	1122334455	2021-05-31	LB	1122334455	15:00:00	1

41 • SELECT \* FROM flighttickets\_trigger ORDER BY flightID DESC;

result Grid					Filter Rows:	Export:	Wrap Cell Content:
flightTicketID	passengerID	flightID	price	business	date_of_purchase		
100004	604	2233445566	200	0	2021-03-15		
100000	600	2233445566	200	0	2021-03-20		
100001	601	2233445566	200	1	2021-03-15		
100004	604	2233445566	300	0	2021-05-30		
100003	603	2233445566	185	0	2021-03-05		
100002	602	2233445566	190	0	2021-03-12		
100002	602	2233445566	190	0	2021-03-12		
100003	603	2233445566	285	0	2021-05-20		
100000	512	1122334455	100	0	2021-04-20		
100001	513	1122334455	100	1	2021-04-15		
100002	514	1122334455	90	0	2021-04-12		
100004	516	1122334455	100	0	2021-04-15		
100003	515	1122334455	85	0	2021-04-05		

Un cop les dades estan insertades, fem un delete per comprobar que funciona.

DELETE FROM flight\_trigger WHERE flightID = 1122334455 OR flightID = 2233445566;

```
11 • SELECT * FROM CancelledFlightsMails;
```

	cancelledFlightsMailsID	flightID	personID	namePerson	emailPerson	priceOfTicket	isBusinessTicket	comission
▶	1	1122334455	512	Eustace	eustace.poluzzi@statcounter.com	100	0	2
2	1122334455	513	Tanhyia		tanhya.skull@huffingtonpost.com	100	1	2
3	1122334455	514	Vergil		vergil.krier@wufoo.com	90	0	2
4	1122334455	515	Dreddy		dreddy.latimer@blog.com	85	0	2
5	1122334455	516	Colman		colman.loffill@t-online.de	100	0	2
8	2233445566	600	Sibeal		sibeal.orht@cbc.ca	200	0	4
9	2233445566	601	Dewitt		dewitt.blaxeland@sohu.com	200	1	4
10	2233445566	602	Brok		brok.billingsley@paypal.com	190	0	4
11	2233445566	603	Kania		kania.sworne@amazon.com	185	0	4
12	2233445566	604	Gloriane		gloriane.paddingdon@bigcartel.com	200	0	4
13	2233445566	602	Brok		brok.billingsley@paypal.com	190	0	4
14	2233445566	603	Kania		kania.sworne@amazon.com	285	0	4
15	2233445566	604	Gloriane		gloriane.paddingdon@bigcartel.com	300	0	4
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Tal i com podem veure, s'han insertat totes les dades a les diferents taules, comprovant així el funcionament del trigger.

```
12 • SELECT * FROM CancellationCost;
```

	cancellationCostID	flightID	refund
▶	1	1122334455	135975
2	2233445566	1988250	
*	HULL	HULL	HULL

### 2.1.9 Requirement (Event) 1. 9 Flight statistics

#### 2.1.9.1 Solution

```
DROP TABLE IF EXISTS DailyFlights;
```

```
CREATE TABLE DailyFlights (
```

```
    dailyFlightsID SERIAL,
```

```
    date DATE,
```

```
    numFlights INTEGER,
```

```
    PRIMARY KEY (dailyFlightsID)
```

```
);
```

```
DROP TABLE IF EXISTS MonthlyFlights;
```

```
CREATE TABLE MonthlyFlights (
```

```
    monthlyFlightsID SERIAL,
```

```
    month INTEGER,
```

```
    year INTEGER,
```

```
    avg_numFlights INTEGER,
```

```
    PRIMARY KEY (monthlyFlightsID)
```

```
);
```

```
DELIMITER $$
```

```
DROP EVENT IF EXISTS NumFlightsDay $$
```

```
CREATE EVENT NumFlightsDay
```

```
ON SCHEDULE EVERY 1 DAY
```

```
STARTS '2021-05-02 23:50:00'
```

```
COMMENT 'Afegir quants avions han volat en aquest dia'
```

```
DO BEGIN
```

```
    INSERT INTO DailyFlights(date, numFlights)
```

```
    SELECT CURDATE(), COUNT(DISTINCT flight_trigger.flightID)
```

```
    FROM flight WHERE flight.date = CURDATE();
```

```
END $$
```

```
DELIMITER ;
```

```
DELIMITER $$
```

```
DROP EVENT IF EXISTS averageFlightsMonth $$
```

```
CREATE EVENT averageFlightsMonth
```

```
ON SCHEDULE EVERY 1 MONTH
```

```
STARTS '2021-05-29 23:59:00'
```

```
COMMENT 'Afegir la mitjana d'avions que han volat en aquest mes'
```

```
DO BEGIN
```

```
    INSERT INTO MonthlyFlights(month, year, avg_numFlights)
```

```
    SELECT DATE_FORMAT(CURDATE(), '%m') , DATE_FORMAT(CURDATE(), '%Y') ,
```

```
    AVG(d.numFlights)
```

```
    FROM DailyFlights as d
```

```
    WHERE DATE_FORMAT(CURDATE(), '%m') = DATE_FORMAT(d.date, '%m') AND
```

```
    DATE_FORMAT(CURDATE(), '%Y') = DATE_FORMAT(d.date, '%Y');
```

```
END $$  
DELIMITER ;
```

### 2.1.9.2 Explanation

Per aquest event he hagut de crear dues taules noves, una que indiqui quants avions han volat aquell dia (DailyFlights), i un altre per fer la mitjana d'avions que han volat en un mes (MonthlyFlights).

Després d'això he procedit a crear els events. Per el primer, he indicat que es repeteixi diàriament i que inserti a la taula de DailyFlights la data actual i el número d'avions que han volat aquell dia.

Posteriorment creo un altre event perquè el interval de temps no és el mateix, i faig que es repeteixi un cop al mes.

En el INSERT d'aquest event del MONTH, el que es fa, és insertar quin mes és i quin any és de la data actual, i el AVG del nombre de vols que ha hagut. Per fer el càlcul d'aquesta mitjana, el que es fa, és agafar de la taula de daily flights, tots els vols que es trobin en el mateix mes i en el mateix any que actualment, i amb aquests, es fa la mitjana.

### 2.1.9.3 Trigger validation

Per fer la validació d'aquest event, he creat un igual però amb alguna taula auxiliar en comptes de les reals (flight\_trigger). A més a més, també he canviat el moment d'inici i el interval de temps de repetició per poder comprovar en el moment que funcionava.

Per el DailyFlights

Primer insertem aquesta informació en la taula, els quals serien els vols que s'han fet d'aquell dia.

```
INSERT INTO flight_trigger (flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)  
VALUES (123456789, 123456789, 123456789, 123456789, "2021-05-29", "LA", 123456789, "15:00:00", 1),  
(123456790, 123456789, 123456789, 123456789, "2021-05-29", "LA", 123456789, "15:00:00", 1),  
(123456791, 123456789, 123456789, 123456789, "2021-05-29", "LA", 123456789, "15:00:00", 1),  
(123456792, 123456789, 123456789, 123456789, "2021-05-29", "LA", 123456789, "15:00:00", 1),  
(123456793, 123456789, 123456789, 123456789, "2021-05-29", "LA", 123456789, "15:00:00", 1),  
(123456794, 123456789, 123456789, 123456789, "2021-05-29", "LA", 123456789, "15:00:00", 1);
```

```
DELIMITER $$  
DROP EVENT IF EXISTS NumFlightsDay_validation $$  
CREATE EVENT NumFlightsDay_validation  
ON SCHEDULE EVERY 3 MINUTE  
-- STARTS '2021-05-02 23:50:00'  
COMMENT 'Afegir quants avions han volat en aquest dia'  
DO BEGIN  
  
    INSERT INTO DailyFlights(date, numFlights)  
    SELECT CURDATE(), COUNT(DISTINCT flight_trigger.flightID)  
    FROM flight_trigger WHERE flight_trigger.date = CURDATE();  
  
END $$  
DELIMITER ;
```

`SELECT * FROM dailyflights;`

dailyFlightsID	date	numFlights
1	2021-05-29	6

Un cop executem el event, ens trobem que s'ha realitzat la inserció correctament. Si esperem els 3 minuts, veiem que la repetició de l'interval funciona perquè apareix 2 cops.

dailyFlightsID	date	numFlights
1	2021-05-29	6
2	2021-05-29	6

Per la taula del Monthly flights es basaria en algo semblant:

```

DELIMITER $$

DROP EVENT IF EXISTS averageFlightsMonth_validation $$

CREATE EVENT averageFlightsMonth_validation
ON SCHEDULE EVERY 2 MINUTE
-- STARTS '2021-05-29 19:42:00'
COMMENT 'Afegir la mitjana d'avions que han volat en aquest mes'
DO BEGIN

    INSERT INTO MonthlyFlights(month, year, avg_numFlights)
    SELECT DATE_FORMAT(CURDATE(), '%m') , DATE_FORMAT(CURDATE(), '%Y') , AVG(d.numFlights)
    ----
    INSERT INTO dailyflights (date, numFlights)
    VALUES ("2021-05-27", 1), (DATE_FORMAT(d.date, '%m') AND DATE_FORMAT(d.date, '%Y')), ("2021-05-26", 32),
    |("2021-05-02", 15);

    DELIMITER ;

```

monthlyFlightsID	month	year	avg_numFlights
1	5	2021	6
2	5	2021	11
NULL	NULL	NULL	NULL

Actualment tenim 2 dies iguals afegits pel que la seva mitjana seria de 6.

monthlyFlightsID	month	year	avg_numFlights
1	5	2021	6

En el cas de que jo li afageixi aquests valors al DailyFlights, farà la mitja de tots ells sigui diferent:

dailyFlightsID	date	numFlights
1	2021-05-29	6
2	2021-05-29	6
3	2021-05-27	1
4	2021-05-26	32
5	2021-05-02	15
6	2021-05-29	6

Podem apreciar que la mitjana de la taula de la dreta (numFlights) = 11, per tant, ho fa bé

## 2.2 Airport module

### 2.2.1 Requirement (Query) 2.1 Airlines and petrol capacity

#### 2.2.1.1 Solution

```

SELECT airl.name AS 'airline name', COUNT(r.routeID) AS '# routes'
FROM Airline AS airl
JOIN Plane AS pl ON airl.airlineID = pl.airlineID
JOIN Planetype AS pt ON pl.planetypeID = pt.planetypeID
JOIN Routeairline AS rairl ON airl.airlineID = rairl.airlineID
JOIN Route AS r ON rairl.routeID = r.routeID
JOIN Airport AS airp ON r.destination_airportID = airp.airportID
JOIN Airport AS airp2 ON r.departure_airportID = airp2.airportID
JOIN City AS city1 ON airp.cityID = city1.cityID
JOIN City AS city2 ON airp2.cityID = city2.cityID
JOIN Country AS c1 ON city1.countryID = c1.countryID
JOIN Country AS c2 ON city2.countryID = c2.countryID
WHERE pt.capacity < r.minimum_petrol AND c1.countryID <> c2.countryID
GROUP BY airl.airlineID;

```

	airline name	# routes
▶	Aigle Azur	320
	American Airlines	153476
	Asiana Airlines	3971
	Afriqiyah Airways	160
	Afrinat International Airlines	18
	Allegiant Air	696
	Air Togo	20
	Advance Leasing Company	64
	Adria Airways	114
	Air Europa	474
	Aegean Airlines	2314
	Air Europe	2
	Ariana Afghan Airlines	18
	Aeroflot Russian Airlines	11288
	Air France	39040
	Air Caledonie International	6
	Air Senegal International	672
	Air Namibia	184
	Air Service Gabon	28
	Aeroper	4
	Azerbaijan Airlines	167

#### 2.2.1.2 Explanation

Apart de fer les relacions adients amb els JOINs segons les taules involucrades relacionades entre si, s'havia de tenir en compte que tant els aeroports, cities i countries tant de les rutes d'entrada com de sortida fossin diferents entre elles, ja que no tindria sentit que fossin iguals, a més de que un dels requisits principals que se'n demana és que la capacitat del petroli utilitzat per a realitzar la ruta sigui major que la capacitat total de l'avió sencer. És a dir, segons es pot veure a la següent línia:

$$\text{pt.capacity} < \text{r.minimum_petrol}$$

### 2.2.1.3 Query validation

```

SELECT airl.name AS 'airline name'/*, COUNT(r.routeID) AS '# routes' */, pt.capacity AS 'Aircraft capacity', r.minimum_petrol AS 'Minimum petrol', r.departure_airportID AS 'Route departure', r.destination_airportID AS 'Route destination'
FROM Airline AS airl
JOIN Plane AS pl ON airl.airlineID = pl.airlineID
JOIN Planetype AS pt ON pl.planetypeID = pt.planetypeID
JOIN Routeairline AS rairl ON airl.airlineID = rairl.airlineID
JOIN Route AS r ON rairl.routeID = r.routeID
JOIN Airport AS airp ON r.destination_airportID = airp.airportID
JOIN Airport AS airp2 ON r.departure_airportID = airp2.airportID
JOIN City AS city1 ON airp.cityID = city1.cityID
JOIN City AS city2 ON airp2.cityID = city2.cityID
JOIN Country AS c1 ON city1.countryID = c1.countryID
JOIN Country AS c2 ON city2.countryID = c2.countryID
WHERE      pt.capacity < r.minimum_petrol AND c1.countryID <> c2.countryID
GROUP BY airl.airlineID , r.departure_airportID;

```

Aquesta query en principi funciona gràcies a la verificació de dades de la comparació de la capacitat de l'avió amb la del petroli, que es pot observar que en tot moment és més gran la del petroli que la de l'avió. I per donar una imatge més 'neta' de la verificació, vaig agrupar-ho no només segons el nombre de la aerolínea, sinó també segons la ruta d'inici. És per aquesta raó la qual no puc mostrar tots els resultats, ja que la llista és llarga, i només mostro el que arriba a ésser visible.

	airline name	Aircraft capacity	Minimum petrol	Route departure	Route destination
▶	Aigle Azur	145	3306	208	1318
	Aigle Azur	145	9696	209	1347
	Aigle Azur	145	12027	219	1318
	Aigle Azur	145	6950	220	1300
	Aigle Azur	145	11763	228	1318
	Aigle Azur	145	4566	229	1347
	Aigle Azur	145	6035	233	1351
	Aigle Azur	145	5659	291	1351
	Aigle Azur	145	23393	1022	1351
	Aigle Azur	145	7619	1239	209
	Aigle Azur	145	7515	1300	209
	Aigle Azur	145	3665	1318	209
	Aigle Azur	145	12686	1347	209
	Aigle Azur	145	19146	1351	219
	Aigle Azur	145	11847	1364	209
	Aigle Azur	145	3409	1584	1347
	Aigle Azur	145	2984	1594	1351
	Aigle Azur	145	3448	1596	1351
	Aigle Azur	145	12911	2831	1351
	Aigle Azur	145	3994	3260	1351
	Aigle Azur	145	16720	4192	1300

## 2.2.2 Requirement (Query) 2.2 Mechanic grades

### 2.2.2.1 Solution

```

SELECT CONCAT(FLOOR(mech.grade), '-', FLOOR(mech.grade) + 1) AS 'grade range',
AVG(maint.duration) AS 'duration average'
FROM mechanic AS mech
JOIN Maintenance AS maint ON mech.mechanicID = maint.mechanicID
JOIN Piecemaintenance AS pmaint ON maint.maintenanceID = pmaint.maintenanceID
JOIN Piece AS p ON pmaint.pieceID = p.pieceID
WHERE (SELECT COUNT(DISTINCT pmaint2.pieceID) FROM
      Maintenance AS maint2
      JOIN piecemaintenance AS pmaint2 ON maint2.maintenanceID = pmaint2.maintenanceID
      WHERE maint2.maintenanceID = maint.maintenanceID
      GROUP BY pmaint2.maintenanceID) < 10
GROUP BY FLOOR(mech.grade)
ORDER BY FLOOR(mech.grade) ASC;

```

	grade range	duration average
►	0-1	51.3065
	1-2	43.7366
	2-3	48.7768
	3-4	36.5139
	4-5	45.3895
	5-6	39.8911
	6-7	47.8186
	7-8	41.6261
	8-9	41.3897
	9-10	41.1456

### 2.2.2.2 Explanation

Apart de fer les relacions amb l'ús dels JOINs segons amb quines taules, també s'ha de tenir en compte un factor determinant característic d'aquesta consulta, que el nombre de peces reemplaçades durant un manteniment sigui menor de 10, el qual per a portar-ho a terme cal realitzar una subconsulta en el WHERE on li demanem un nombre de peces diferents (DISTINCT) a través d'un COUNT() on tornem a realitzar les relacions adients amb els JOINs, i verificar que sigui < 10, a més de fer un SELECT i GROUP BY del rang dels graus de la següent forma:

**CONCAT(FLOOR(mech.grade), ' - ', FLOOR(mech.grade) + 1)**

### 2.2.2.3 Query validation

```

SELECT maint.maintenanceID AS 'Maintenance ID', AVG(maint.duration) AS 'duration average',
mech.grade AS 'Mechanic grade'
, (SELECT COUNT(DISTINCT pmaint2.pieceID)
  FROM Maintenance AS maint2
  JOIN piecemaintenance AS pmaint2 ON maint2.maintenanceID = pmaint2.maintenanceID
  WHERE maint2.maintenanceID = maint.maintenanceID
  GROUP BY pmaint2.maintenanceID) AS 'Number of pieces'

```

```

FROM mechanic AS mech
JOIN Maintenance AS maint ON mech.mechanicID = maint.mechanicID
JOIN Piecemaintenance AS pmaint ON maint.maintenanceID = pmaint.maintenanceID
JOIN Piece AS p ON pmaint.pieceID = p.pieceID
WHERE (SELECT COUNT(DISTINCT pmaint2.pieceID) FROM
      Maintenance AS maint2
      JOIN piecemaintenance AS pmaint2 ON maint2.maintenanceID = pmaint2.maintenanceID
      WHERE maint2.maintenanceID = maint.maintenanceID
      GROUP BY pmaint2.maintenanceID) < 10
GROUP BY maint.maintenanceID
ORDER BY mech.grade ASC;

```

La validació d'aquesta query té sentit que funcioni ja que l'únic que hem de fer és demostrar el nombre de peces per manteniment portada a terme per cada mecànic, i això es fa sense problema amb una subconsulta a dins del SELECT com a paràmetre adicional on fem un COUNT() i portem a terme el mateix exacte procediment que vam fer amb la validació del nombre de peces < 10 amb la diferència de que ens estalviem el '< 10'. I com es pot veure, el nombre de peces sempre és < 10.

	Maintenance ID	duration average	Mechanic grade	Number of pieces
▶	969	38.0000	0.05	9
	563	20.0000	0.05	8
	203	47.0000	0.05	8
	1114	148.0000	0.35	2
	575	45.0000	0.35	4
	1080	7.0000	0.35	1
	1053	74.0000	0.35	2
	1187	23.0000	0.35	2
	417	80.0000	0.35	6
	246	30.0000	0.35	5
	528	7.0000	0.35	5
	32	54.0000	0.35	7
	613	71.0000	0.76	7
	624	64.0000	0.76	5
	795	67.0000	0.96	5
	303	55.0000	0.96	8
	510	47.0000	0.96	6
	738	69.0000	0.96	9
	1166	30.0000	0.96	1
	706	34.0000	0.96	8
	751	34.0000	0.96	5

### 2.2.3 Requirement (Query) 2.3 Airports and mean distance routes

#### 2.2.3.1 Solution

```

SELECT airp.airportID AS 'airport id', c.countryID AS 'country id', AVG(r.distance) AS 'average
distance'
FROM airport AS airp
JOIN Route AS r ON airp.airportID = r.departure_airportID
JOIN City ON airp.cityID = city.cityID
JOIN Country AS c ON city.countryID = c.countryID
GROUP BY airp.airportID
HAVING AVG(r.distance) > (SELECT AVG(r2.distance)
                            FROM airport AS airp2
                            JOIN Route AS r2 ON airp2.airportID = r2.departure_airportID
                            JOIN City AS city2 ON airp2.cityID = city2.cityID
                            JOIN Country AS c2 ON city2.countryID = c2.countryID
                            WHERE c2.countryID = c.countryID
                            GROUP BY c2.name
                            ORDER BY c2.name ASC)
ORDER BY c.countryID;

```

NOTA: Vaig clicar sobre el paràmetre ‘airport id’ per a ordenar-ho de menor a major ja de pas

	airport id	country id	average distance
▶	5	1	1422.4063
	16	3	2681.4516
	145	4	2544.6222
	155	4	2705.9865
	177	4	1402.8644
	192	4	3069.4161
	209	5	1535.2063
	219	5	1323.2857
	226	5	1311.0000
	244	7	2073.7143
	246	8	2821.4828
	268	10	2481.0000
	271	10	2728.4762
	285	12	1513.5172
	291	12	1365.9565
	296	13	1905.6923
	300	14	1971.2517
	327	15	2212.7470

#### 2.2.3.2 Explanation

La idea principal a extreure d'aquesta query és realitzar una comparació de mitjanes de distàncies fetes entre les rutes que surten d'un aeroport i la de tots els aeroports del mateix país del que surten, el qual es fa a través d'una subquery on li demanem l' ‘average’ i la comprovació de que aquest sigui menor a l'average demanat a la query principal. A més de que el segon country declarat a la subconsulta sigui el mateix country que el de la query principal per a satisfer la demanda de l'enunciat de que provingui del mateix country els aeroports de la subconsulta.

### 2.2.3.3 Query validation

```

SELECT airp.airportID AS 'airport id', c.countryID AS 'country id', AVG(r.distance) AS 'average
distance routes departing from airport', (SELECT AVG(r2.distance)
FROM airport AS airp2
JOIN Route AS r2 ON airp2.airportID = r2.departure_airportID
JOIN City AS city2 ON airp2.cityID = city2.cityID
JOIN Country AS c2 ON city2.countryID = c2.countryID
WHERE c2.countryID = c.countryID
GROUP BY c2.name
ORDER BY c2.name ASC) AS
average_distance_routes_departing_from_the_same_country
FROM airport AS airp
JOIN Route AS r ON airp.airportID = r.departure_airportID
JOIN City ON airp.cityID = city.cityID
JOIN Country AS c ON city.countryID = c.countryID
GROUP BY airp.airportID
HAVING AVG(r.distance) > average_distance_routes_departing_from_the_same_country
ORDER BY c.countryID;

```

	airport id	country id	average distance routes departing from airport	average_distance_routes_departing_from_the_same_country
▶	5	1	1422.4063	1057.4850
	16	3	2681.4516	1879.5918
	145	4	2544.6222	1315.6014
	155	4	2705.9865	1315.6014
	177	4	1402.8644	1315.6014
	192	4	3069.4161	1315.6014
	209	5	1535.2063	1175.2805
	219	5	1323.2857	1175.2805
	226	5	1311.0000	1175.2805
	244	7	2073.7143	1967.5333
	246	8	2821.4828	2609.2188
	740	10	7401.0000	1041.4425

Es verifica la validació d'aquesta query com a funcional ja que li demanem que ens mostri l'average demandat a la subquery però indicat al SELECT de la query principal, i tal i com es pot veure, el valor sempre serà menor en comparació a la mitjana de la distància de les rutes que fan 'departure' des del mateix país on es localitza l'aeroport principal que analitzem a la query principal.

## 2.2.4 Requirement (Query) 2.4 Airlines and routes

### 2.2.4.1 Solution

```

SELECT airl.name AS 'airline name', airl.airlineID AS 'airline id', c3.name AS 'country name' ,
MAX(r.time) AS 'longest route duration'
FROM Airline AS airl
JOIN Routeairline AS rairl ON airl.airlineID = rairl.airlineID
JOIN Route AS r ON rairl.routeID = r.routeID
JOIN Airport AS airp ON r.destination_airportID = airp.airportID
JOIN Airport AS airp2 ON r.departure_airportID = airp2.airportID
JOIN City AS city1 ON airp.cityID = city1.cityID
JOIN City AS city2 ON airp2.cityID = city2.cityID
JOIN Country AS c ON city1.countryID = c.countryID
JOIN Country AS c2 ON city2.countryID = c2.countryID
JOIN Country AS c3 ON c3.countryID = airl.countryID
WHERE airl.active LIKE 'Y' AND NOT c.name LIKE 'Spain' AND NOT c2.name LIKE 'Spain'
AND r.time IS NOT NULL
GROUP BY airl.airlineID
ORDER BY MAX(r.time) ASC;

```

	airline name	airline id	country name	longest route duration
▶	China United Airlines	1769	Germany	00:37:48
	Air Caledonie International	139	Wallis and Futuna	00:45:33
	Atlantic Airways	881	Norway	00:49:14
	Air Seychelles	319	Seychelles	00:50:27
	Air Antilles Express	11741	Guadeloupe	00:54:27
	Apache Air	19016	Kyrgyzstan	00:57:16
	Cyprus Turkish Airlines	1943	Kyrgyzstan	00:57:16
	Air Florida	882	Kyrgyzstan	00:57:16
	Skybservice Airlines	4496	Ghana	00:58:48
	Air Sinai	442	Egypt	01:00:02
	Air Tindi	68	Canada	01:00:12
	Seaborne Airlines	4335	Virgin Islands	01:00:39
	Proflight Commuter Services	4066	Zambia	01:01:54
	Hex'Air	2757	France	01:02:00
	Air Bosna	132	Bosnia and Herz...	01:02:39
	Air Iceland	896	Iceland	01:02:53
	Salmon Air	10776	Haiti	01:03:08
	Pacific East Asia Cargo Air...	3857	Congo (Brazzaville)	01:03:31
	LSM Airlines	14061	Philippines	01:04:14
	Excel Airways	2264	Albania	01:04:16
	Tiara Air	5002	Netherlands Antil...	01:05:02
	Monolian International Ai	16615	Mexico	01:05:17

### 2.2.4.2 Explanation

Apart de fer les relacions amb l'ús dels JOINs segons amb quines taules, també s'ha de tenir en compte varis aspectes. Un d'ells el fet de que hem de crear varies cities (city1 i city2) per a diferenciar la ruta d'origen de la de destí, a més de crear una city adicional (city3) per a englobar ambdues. I per últim, comprovar que la airline estigui activa o no, i que el país d'origen d'on entra o surt, sigui Espanya en aquest cas/context, a través de la clàusula WHERE.

#### 2.2.4.3 Query validation

```

SELECT airl.name AS 'airline name', airl.airlineID AS 'airline id', c3.name AS 'country name' ,
MAX(r.time) AS 'longest route duration', airl.active AS 'status'
FROM Airline AS airl
JOIN Routeairline AS rairl ON airl.airlineID = rairl.airlineID
JOIN Route AS r ON rairl.routeID = r.routeID
JOIN Airport AS airp ON r.destination_airportID = airp.airportID
JOIN Airport AS airp2 ON r.departure_airportID = airp2.airportID
JOIN City AS city1 ON airp.cityID = city1.cityID
JOIN City AS city2 ON airp2.cityID = city2.cityID
JOIN Country AS c ON city1.countryID = c.countryID
JOIN Country AS c2 ON city2.countryID = c2.countryID
JOIN Country AS c3 ON c3.countryID = airl.countryID
WHERE airl.active LIKE 'Y' AND NOT c.name LIKE 'Spain' AND NOT c2.name LIKE 'Spain'
AND r.time IS NOT NULL
GROUP BY airl.airlineID
ORDER BY MAX(r.time) ASC;

```

Més enllà de indicar que em demostri si les aerolínies estan actives o no, en general les dades que es demanen a la consulta en general són bastant descriptives ja de per sí i dóna detall del seu bon funcionament. Una altra forma que es pot comprovar de forma ‘indirecta’ és mirant la taula de resultats i comprovar que no hi ha cap país que sigui igual a ‘Spain’. Però més enllà d’aquests detalls, no trobo una altra forma de justificar aquesta consulta més de lo mencionat fins ara.

	airline name	airline id	country name	longest route duration	status
▶	China United Airlines	1769	China	00:37:48	Y
	Air Caledonie International	139	France	00:45:33	Y
	Atlantic Airways	881	Faroe Islands	00:49:14	Y
	Air Seychelles	319	Seychelles	00:50:27	Y
	Air Antilles Express	11741	Guadeloupe	00:54:27	Y
	Apache Air	19016	United States	00:57:16	Y
	Cyprus Turkish Airlines	1943	Turkey	00:57:16	Y
	Air Florida	882	United States	00:57:16	Y
	Skyservice Airlines	4496	Canada	00:58:48	Y
	Air Sinai	442	Egypt	01:00:02	Y
	Air Tindi	68	Canadian Te...	01:00:12	Y
	Seaborne Airlines	4335	United States	01:00:39	Y
	Proflight Commuter Services	4066	Zambia	01:01:54	Y
	Hex'Air	2757	France	01:02:00	Y
	Air Bosna	132	Bosnia and H...	01:02:39	Y
	Air Iceland	896	Iceland	01:02:53	Y
	Salmon Air	10776	United States	01:03:08	Y
	Pacific East Asia Cargo Air...	3857	Philippines	01:03:31	Y
	LSM Airlines	14061	Russia	01:04:14	Y
	Excel Airways	2264	United Kingdom	01:04:16	Y
	Tiara Air	5002	Aruba	01:05:02	Y
	Mongolian International Ai...	16615	Mongolia	01:05:17	v

## 2.2.5 Requirement (Query) 2.5 Pieces replaced

### 2.2.5.1 Solution

```

SELECT *, COUNT(piecemaintenance.maintenanceID)
FROM piecemaintenance
JOIN maintenance ON piecemaintenance.maintenanceID = maintenance.maintenanceID
JOIN piece ON piecemaintenance.pieceID = piece.pieceID
GROUP BY piecemaintenance.pieceID, maintenance.planeID, piece.cost
HAVING (COUNT(piecemaintenance.maintenanceID)*piece.cost) >
(SELECT (SUM(piece.cost)/2) FROM piecemaintenance AS pm
 JOIN maintenance AS m ON pm.maintenanceID = m.maintenanceID
 JOIN piece AS p ON pm.pieceID = p.pieceID
 WHERE maintenance.planeID = m.planeID AND
       piece.pieceID <> p.pieceID
 GROUP BY maintenance.planeID)
 AND COUNT(piecemaintenance.maintenanceID) > 1
ORDER BY COUNT(piecemaintenance.maintenanceID) DESC;

```

	maintenanceID	pieceID	maintenanceID	duration	planeID	mechanicID	date	pieceID	name	cost	COUNT(piecemaintenance.maintenanceID)
▶	5	10	5	67	2791	141	1997-02-05	10	Rudder	328046	5
	5	15	5	67	2791	141	1997-02-05	15	Wheel	410761	4
	836	1	836	10	3516	156	1994-07-30	1	Aileron	610271	4
	162	8	162	70	4111	38	2003-08-30	8	Fuselage	980785	4
	87	8	87	35	2301	111	2011-03-26	8	Fuselage	980785	4
	177	13	177	12	2791	151	1994-12-08	13	Struts	314633	3
	47	9	47	22	1311	347	1999-01-15	9	Horizontal Stabilizer	388341	3
	365	5	365	3	3864	214	1992-05-17	5	Empennage	174893	3
	104	7	104	16	2059	67	2008-12-25	7	Flap	761033	3
	330	7	330	34	4825	262	1972-12-03	7	Flap	761033	3
	133	5	133	30	2111	112	2000-10-19	5	Empennage	174893	3
	251	1	251	34	3647	69	1979-11-18	1	Aileron	610271	3
	224	1	224	72	3131	93	1994-06-09	1	Aileron	610271	3
	162	11	162	70	4111	38	2003-08-30	11	Slat	349968	3
	87	15	87	35	2301	111	2011-03-26	15	Wheel	410761	3
	198	14	198	61	1516	165	1999-11-01	14	Vertical Stabilizer	333445	3
	130	4	130	49	3960	353	2009-09-13	4	Elevator	766652	3
	5	5	67	2791	141	1997-02-05	5	Empennage	174893	3	
	838	5	838	13	3516	101	2004-03-19	5	Empennage	174893	3
	33	7	33	36	3261	716	2006-06-22	7	Antenna	600145	3

Una anotació que vull fer és que en aquest cas s'ens demanava mostrar per pantalla només els paràmetres: Plane id, Piece name, # Pieces replaced

La query que he escrit abans representa la idea executada tal i com voliem. El problema és que al limitar els paràmetres a mostrar, ens sortien errors relacionats amb la no adició d'aquestes en clàusules com el GROUP BY o al SELECT. La consulta quedava així al principi:

```

SELECT maintenance.planeID, piece.name, COUNT(piecemaintenance.maintenanceID) FROM
piecemaintenance
JOIN maintenance ON piecemaintenance.maintenanceID = maintenance.maintenanceID
JOIN piece ON piecemaintenance.pieceID = piece.pieceID
GROUP BY piece.pieceID, maintenance.planeID
HAVING (COUNT(piecemaintenance.maintenanceID))*piece.cost >
(SELECT (SUM(p.cost))/2 FROM piecemaintenance AS pm
 JOIN maintenance AS m ON pm.maintenanceID = m.maintenanceID
 JOIN piece AS p ON pm.pieceID = p.pieceID
 WHERE maintenance.planeID = m.planeID AND
       piece.pieceID <> p.pieceID
 GROUP BY maintenance.planeID)
 ORDER BY COUNT(piecemaintenance.maintenanceID) DESC;

```

```
✖ 21 13:33:04 SELECT maintenance.planeID, piece.name, COUNT(pi... Error Code: 1054. Unknown column 'piece.cost' in 'h...
```

I si feiem els canvis pertinents per a corregir aquests errors, ens sortia el problema següent:

```
SELECT maintenance.planeID, piece.name, COUNT(picemaintenance.maintenanceID) FROM
picemaintenance
JOIN maintenance ON piuemaintenance.maintenanceID = maintenance.maintenanceID
JOIN piece ON piuemaintenance.pieceID = piece.pieceID
GROUP BY piece.pieceID, maintenance.planeID, piece.cost
HAVING (COUNT(piuemaintenance.maintenanceID))*piece.cost >
(SELECT (SUM(p.cost))/2 FROM piuemaintenance AS pm
JOIN maintenance AS m ON pm.maintenanceID = m.maintenanceID
JOIN piece AS p ON pm.pieceID = p.pieceID
WHERE maintenance.planeID = m.planeID AND
piece.pieceID <> p.pieceID
GROUP BY maintenance.planeID)
ORDER BY COUNT(piuemaintenance.maintenanceID) DESC;
```

```
✖ 22 13:34:22 SELECT maintenance.planeID, piece.name, ... Error Code: 1062. Duplicate entry '2262' for key '<group_key>'
```

El qual suposem que aquest problema es troba realment en la base de dades i no en la consulta així. Vam provar d'altres formes, però és que la forma correcta que hem trobat de fer-ho és com la que es va indicar al principi del tot, que és fent un select de tots els paràmetres relacionats amb piuemaintenance a més del COUNT de manteniments.

#### 2.2.5.2 *Explanation*

Apart de fer les relacions amb l'ús dels JOINs segons amb quines taules, la cosa a remarcar amb aquesta consulta és comprovar que el cost del canvi de peces sigui més de la meitat del total de peces canviades a l'avió fent servir una subconsulta a dins de la clàusula del HAVING on tornem a fer les relacions adients amb JOINs.

#### 2.2.5.3 *Query validation*

```
SELECT *, COUNT(piuemaintenance.maintenanceID)
FROM piuemaintenance
JOIN maintenance ON piuemaintenance.maintenanceID = maintenance.maintenanceID
JOIN piece ON piuemaintenance.pieceID = piece.pieceID
GROUP BY piuemaintenance.pieceID, maintenance.planeID, piece.cost
HAVING (COUNT(piuemaintenance.maintenanceID))*piece.cost > (SELECT
(SUM(piece.cost)/2) FROM piuemaintenance AS pm
JOIN maintenance AS m ON pm.maintenanceID = m.maintenanceID
JOIN piece AS p ON pm.pieceID = p.pieceID
WHERE maintenance.planeID = m.planeID AND
piece.pieceID <> p.pieceID
GROUP BY maintenance.planeID)
AND COUNT(piuemaintenance.maintenanceID) > 1
ORDER BY COUNT(piuemaintenance.maintenanceID) DESC;
```

Com ja vaig explicar abans a la secció 2.2.5.1., la query que vaig escriure allà equival igual per a la de validació, la qual funciona perquè les dades addicionals que demanem mostrar per pantalla, dóna validesa al procediment recalcat a la query escrita.

	maintenanceID	pieceID	maintenanceID	duration	planeID	mechanicID	date	pieceID	name	cost	COUNT(pieceMaintenance.maintenanceID)
▶	5	10	5	67	2791	141	1997-02-05	10	Rudder	328046	5
	5	15	5	67	2791	141	1997-02-05	15	Wheel	410761	4
	836	1	836	10	3516	156	1994-07-30	1	Aileron	610271	4
	162	8	162	70	4111	38	2003-08-30	8	Fuselage	980785	4
	87	8	87	35	2301	111	2011-03-26	8	Fuselage	980785	4
	177	13	177	12	2791	151	1994-12-08	13	Struts	314633	3
	47	9	47	22	1311	347	1999-01-15	9	Horizontal Stabilizer	388341	3
	365	5	365	3	3864	214	1992-05-17	5	Empennage	174893	3
	104	7	104	16	2059	67	2008-12-25	7	Flap	761033	3
	330	7	330	34	4825	262	1972-12-03	7	Flap	761033	3
	133	5	133	30	2111	112	2000-10-19	5	Empennage	174893	3
	251	1	251	34	3647	69	1979-11-18	1	Aileron	610271	3
	224	1	224	72	3131	93	1994-06-09	1	Aileron	610271	3
	162	11	162	70	4111	38	2003-08-30	11	Slat	349968	3
	87	15	87	35	2301	111	2011-03-26	15	Wheel	410761	3
	198	14	198	61	1516	165	1999-11-01	14	Vertical Stabilizer	333445	3
	130	4	130	49	3960	353	2009-09-13	4	Elevator	766652	3
	5	5	5	67	2791	141	1997-02-05	5	Empennage	174893	3
	838	5	838	13	3516	101	2004-03-19	5	Empennage	174893	3
	33	2	23	36	2791	716	2006-06-22	7	Antenna	600145	3

## 2.2.6 Requirement (Trigger) 2.6 Routes cancelled

### 2.2.6.1 Solution

```
DROP TABLE IF EXISTS RoutesCancelled;
CREATE TABLE RoutesCancelled(
    destination_name VARCHAR(50), -- JOIN Route, Airport, City
    origin_name VARCHAR(50), -- JOIN Route, Airport, City
    num_airlines INT, -- COUNT(airline.airlineID)
    date_route_deletion DATE -- NOW()
);

DELIMITER $$

DROP TRIGGER IF EXISTS routes_cancelled $$

CREATE TRIGGER routes_cancelled
    BEFORE DELETE ON Route
    FOR EACH ROW
BEGIN
    INSERT INTO RoutesCancelled(destination_name, origin_name, num_airlines,
date_route_deletion)
        SELECT c.name, c1.name, COUNT(airl.airlineID), NOW() FROM Airline AS airl
        JOIN RouteAirline AS rairl ON airl.airlineID = rairl.airlineID
        JOIN Route AS r ON rairl.routeID = r.routeID
        JOIN Airport AS airp ON r.departure_airportID = airp.airportID
        JOIN Airport AS airp1 ON r.destination_airportID = airp1.airportID
        JOIN City AS c ON c.cityID = airp.cityID
        JOIN City AS c1 ON c1.cityID = airp1.cityID
        WHERE r.routeID = old.routeID;

    DELETE FROM RouteAirline WHERE routeID = old.routeID;
END $$

DELIMITER ;
```

### 2.2.6.2 Explanation

La forma de funcionar que té aquest trigger és bastant directe i entendible a primera vista. Per una banda, considerant que se'ns està demanant principalment “cancel·lar” info d'una taula, farem servir la clàusula BEFORE DELETE per a adelantar-nos'en i afegir la informació a una altra taula anomenada RoutesCancelled abans de que s'elimini la info més endavant. de forma permanent. A més d'això, i en relació al que deia referenciava, se'ns demana a continuació eliminar la informació de la taula provinent RouteAirline, on després de fer l'adició d'aquesta informació a una altra taula, aquí moments després haurem de fer un DELETE FROM de la info. de la taula RouteAirline sempre i quan l'ID de la ruta sigui la mateixa que l'antiga, al igual que de la mateixa forma quan fem el procés de cancel·lació.

### 2.2.6.3 Trigger validation

NOTA: Per a la validació he creat taules auxiliars iguals a les de Route i RouteAirline, anomenades respectivament RouteTrigger i RouteAirline. Per tant, a la query original que he penjat adalt a la solució, no conté aquestes taules perquè en principi haurien d'actuar per a les taules exportades del projecte. Però per raons que detallaré d'ara en endavant, la validació l'he feta en comptes amb les taules auxiliars Trigger en comptes de les normals del projecte. I així, per a la resta de triggers.

(Creació de taules auxiliars per a estalviar-nos' en eliminar informació de les taules oficials i evitar exportar el projecte múltiples vegades perquè en el meu cas el meu portàtil no va ben bé del tot i triga sense exagerar 45 minuts en exportar el projecte sencer de nou.

Aquestes taules contenen tota la informació de Route i RouteAirline respectivament)

CREATE TABLE routeTrigger

SELECT \* FROM Route;

CREATE TABLE routeAirlineTrigger

SELECT \* FROM RouteAirline;

(Comprovem que la informació s'ha copiat correctament)

SELECT \* FROM routeTrigger;

routeID	destination	departure	distance	minimum_petrol	time
1	3	1	795	6876	00:47:51
2	4	1	688	5684	00:37:53
3	2	1	957	4099	00:51:52
4	5	1	818	7834	01:02:40
5	1	2	222	8110	00:54:47
6	3	2	356	8123	01:04:56
7	4	2	112	7479	00:49:24
8	6	2	292	5831	00:41:09
9	1	3	124	9313	00:41:31
10	4	3	545	2158	00:59:14
11	2	3	451	4551	00:41:39
12	3425	3	523	8369	00:42:29
13	5	3	260	10679	00:51:32
14	3428	3	533	9082	00:44:17
15	3427	3	885	9298	00:56:47
16	6	3	924	12429	01:11:39
17	3415	4	965	13318	01:37:51
18	1	4	155	6612	00:50:48
19	3	4	481	11364	00:53:30
20	3418	4	937	13740	01:02:00

(Execució de la creació de la taula RoutesCancelled i del Trigger prèviament establertes)

(Comprovem l'existència de la nova taula RoutesCancelled)

SELECT \* FROM RoutesCancelled;

	destination_name	origin_name	num_airlines	date_route_deletion

(Eliminem informació de la suposada taula RouteTrigger que en realitat és Route, però per raons prèviament establertes, fem una replica d'aquesta taula, amb tal de que el trigger s'activi a l'espera d'afegir informació a RoutesCancelled)

DELETE FROM RouteTrigger WHERE routeID >= 1 AND routeID <= 5;

(Comprovem que s'han eliminat les files especificades a la taula Route/Trigger)

SELECT \* FROM routeTrigger;

6	3	2	356	8123	01:04:56
7	4	2	112	7479	00:49:24
8	6	2	292	5831	00:41:09
9	1	3	124	9313	00:41:31
10	4	3	545	2158	00:59:14
11	2	3	451	4551	00:41:39
12	3425	3	523	8369	00:42:29
13	5	3	260	10679	00:51:32
14	3428	3	533	9082	00:44:17
15	3427	3	885	9298	00:56:47
16	6	3	924	12429	01:11:39
17	3415	4	965	13318	01:37:51
18	1	4	155	6612	00:50:48
19	3	4	481	11364	00:53:30
20	3418	4	937	13740	01:02:00

(Comprovem que s'ha afegit la informació a la taula RoutesCancelled)

SELECT \* FROM RoutesCancelled;

	destination_name	origin_name	num_airlines	date_route_deletion
▶	Goroka	Mount Hagen	0	2021-05-26
	Goroka	Nadzab	0	2021-05-26
	Goroka	Madang	0	2021-05-26
	Goroka	Port Moresby	1	2021-05-26
	Madang	Goroka	0	2021-05-26

(I ja ho tindriem verificada el trigger d'aquest apartat/Requirement.

Òbviament, he posat d'exemple només 5 files. Però si posem 300 per exemple, sortirien resultats amb un num\_airlines de valors més alts)

## 2.2.7 Requirement (Trigger) 2.7 Mechanics firings

### 2.2.7.1 Solution

```
DROP TABLE IF EXISTS MechanicsFirings;
```

```
CREATE TABLE MechanicsFirings(
```

```
    mechanic_id INT,  
    name VARCHAR(50),  
    surname VARCHAR(50),  
    birth_date DATE,  
    firing_reason TEXT
```

```
);
```

```
-- • Retirement: If the date when the person is deleted from the table is 65 years old or  
-- older.
```

```
DELIMITER $$
```

```
DROP TRIGGER IF EXISTS dismissal $$
```

```
CREATE TRIGGER dismissal
```

```
    BEFORE DELETE ON Mechanic FOR EACH ROW
```

```
BEGIN
```

```
-- • Retirement: If the date when the person is deleted from the table is 65 years old or  
-- older.
```

```
IF (SELECT (YEAR(NOW()) - YEAR(person.born_date)) >= 65 FROM person WHERE  
person.personID = OLD.mechanicID) THEN
```

```
    INSERT INTO MechanicsFirings(mechanic_id, name, surname, birth_date, firing_reason)  
        SELECT p.personID, p.name, p.surname, p.born_date, "Retirement"  
        FROM Person AS p  
        JOIN Mechanic AS mech ON mech.mechanicID = p.personID  
        JOIN Country AS c ON p.countryID = c.countryID  
        JOIN City AS city ON c.countryID = city.cityID  
        JOIN Airport AS airp ON city.cityID = airp.cityID  
        WHERE mech.mechanicID = old.mechanicID  
        GROUP BY mech.mechanicID;
```

```
-- • Not completing the evaluation period: When the sum of repairs that the mechanic has  
-- done does not add up to more than 10 hours.
```

```
ELSEIF (SELECT SUM(m.duration) <= 10 FROM maintenance AS m WHERE m.mechanicid =  
OLD.mechanicID) THEN
```

```
    INSERT INTO MechanicsFirings(mechanic_id, name, surname, birth_date, firing_reason)  
        SELECT p.personID, p.name, p.surname, p.born_date, "Not completing the evaluation period"  
        FROM Person AS p  
        JOIN Mechanic AS mech ON mech.mechanicID = p.personID  
        JOIN Maintenance AS m ON mech.mechanicID = m.mechanicID  
        WHERE mech.mechanicID = old.mechanicID  
        GROUP BY mech.mechanicID;
```

```
-- • Firing without reason: When it does not belong to any of the previous types.
```

```
ELSE
```

```
    INSERT INTO MechanicsFirings(mechanic_id, name, surname, birth_date, firing_reason)  
        SELECT p.personID, p.name, p.surname, p.born_date, "Firing without reason"  
        FROM Person AS p  
        JOIN Mechanic AS mech ON mech.mechanicID = p.personID
```

```

JOIN Maintenance AS m ON mech.mechanicID = m.mechanicID
    WHERE mech.mechanicID = old.mechanicID
    GROUP BY mech.mechanicID;
END IF;

DELETE maintenance, piecemaintenance FROM maintenance JOIN piecemaintenance
    WHERE maintenance.mechanicID IN (SELECT mech.mechanicID FROM Mechanic AS mech
    WHERE mech.mechanicID = old.mechanicID)
        AND piecemaintenance.maintenanceID = maintenance.maintenanceID;
END $$

DELIMITER ;

```

#### 2.2.7.2 *Explanation*

La idea principal d'aquest trigger és que hi han tres raons per les quals es pot dur a terme un acomiadament als mecànics de l'aeroport: ja sigui perquè tenen més (o són) de 65 anys, que la suma d'hores de les reparacions portades a terme no doni més de 10 hores de realització, o cap de les dues raons prèviament donades. Aquesta part en concret s'ha portat a terme amb una estructura IF-ELSEIF-ELSE a dins del trigger, i segons quina circumstància ocorregués, hauríem de guardar varies dades a dins d'una taula nova creada per nosaltres anomenada MechanicFirings amb un text com a paràmetre dient la raó de perquè s'ha portat a terme l'acomiadament.

Apart d'això, sigui quina sigui la raó de l'acomiadament, acte seguit s'ha d'eliminar el manteniment i les peces reemplaçades en aquell manteniment trobades a dins de la taula PieceMaintenance a través d'un DELETE normal i corrent amb les seves relacions adients fetes amb JOINs i la condició de WHERE de que coincideixi el mecànic que s'acomienda.

#### 2.2.7.3 *Trigger validation*

Respecte la validació del trigger, al igual que la query anterior, es porta a terme de la següent forma: (Primer de tot creem tant la taula de MechanicsFirings com taules auxiliars de Person, Piecemaintenance i Mechanic, anomenats respectivament PersonTrigger, PiecemaintenanceTrigger i MechanicTrigger on li emmagatzemem la informació de les taules prèviament mencionades amb tal de no tocar les dades oficials de l'exportació del projecte)

En aquest cas, posarem com a exemple de validació al **mechanic** amb **id = 103** i els seus respectius **manteniments** amb **ids 74, 539 i 980**:

```

SELECT * FROM MechanicTrigger AS mech
RIGHT JOIN Maintenance AS m ON m.mechanicID = mech.mechanicID
JOIN PiecemaintenanceTrigger AS mt ON mt.maintenanceID = m.maintenanceID
JOIN Piece AS p ON p.pieceID = mt.pieceID
WHERE mech.mechanicID = 103
GROUP BY mt.maintenanceID
ORDER BY mech.mechanicID, mt.maintenanceID;

```

	mechanicID	grade	maintenanceID	duration	planeID	mechanicID	date	maintenanceID	pieceID	pieceID	name	cost
▶	103	8.54	74	31	1333	103	1981-10-01	74	1	1	Aileron	610271
	103	8.54	539	30	3438	103	1992-08-27	539	3	3	Cockpit	232972
	103	8.54	980	42	1544	103	1989-02-22	980	2	2	Antenna	900145

(Comprovem que MechanicTrigger conté totes les dades copiades de Mechanic abans de fer el trigger i la eliminació de dades)

SELECT \* FROM MechanicTrigger;

	mechanicID	grade
	100	6.67
	101	7.81
	102	5.04
	103	8.54
	104	5.39
	105	6.36
	106	8.97
	107	8.65
	108	5.34

(Fem el mateix procediment tant per PiecemaintenanceTrigger com MaintenanceTrigger)

SELECT \* FROM PiecemaintenanceTrigger;

	maintenanceID	pieceID
	63	1
	65	1
	70	1
	71	1
	72	1
	73	1
	74	1
	75	1
	76	1

SELECT \* FROM MaintenanceTrigger  
ORDER BY mechanicID;

	maintenanceID	duration	planeID	mechanicID	date
	838	13	3516	101	2004-03-19
	1159	91	4151	101	2005-12-23
	74	31	1333	103	1981-10-01
	539	30	3438	103	1992-08-27
	980	42	1544	103	1989-02-22

(Executem el trigger establert)

(Esborrem la informació la taula MechanicTrigger, en aquest cas la id del mechanic = 103)

DELETE FROM mechanicTrigger WHERE mechanicID = 103;  
SELECT \* FROM MechanicTrigger; (com es pot veure, s'esborra el mechanicID = 103)

	mechanicID	grade
	100	6.67
	101	7.81
	102	5.04
	104	5.39
	105	6.36
	106	8.97
	107	8.65
	108	5.34

(Visualitzem el resultat final de la taula de MechanicFirings per comprovar que s'ha afegit a la taula els mecànics acomiadats)

```
SELECT * FROM MechanicFirings
GROUP BY mechanic_id;
```

	mechanic_id	name	surname	birth_date	firing_reason
▶	103	Muriel	Winslow	1946-09-10	Retirement

(I per últim, comprovem que tant els manteniments d'aquest mechanic com els piecemaintenances pertanyents a aquest, s'esborren de les taules PiecemaintenanceTrigger i MaintenanceTrigger)

	maintenanceID	pieceID
	71	1
	72	1
	73	1
	75	1
	76	1

(no mechanicID = 74)

	maintenanceID	duration	planeID	mechanicID	date
	73	28	4388	98	2008-06-29
	75	12	3833	92	1982-09-08
	76	13	629	194	2018-01-11
	77	38	1654	26	1997-10-04
	78	64	3165	204	2013-09-14

(no mechanicID = 74)

	maintenanceID	duration	planeID	mechanicID	date
	536	39	4020	100	1996-08-20
	537	56	578	31	1973-08-23
	538	57	2154	119	1987-10-28
	540	43	4107	229	1982-02-06
	541	67	4624	282	2020-12-01

(no mechanicID = 539)

	maintenanceID	duration	planeID	mechanicID	date
	978	64	1576	137	1984-04-20
	979	74	4266	137	2001-08-28
	981	15	4705	13	1996-01-11
	982	78	413	245	1985-05-26
	983	4	260	148	2014-01-24

(no mechanicID = 980)

I ja estaríem amb la verificació d'aquest trigger. Afegim informació tant a la taula de MechanicFirings com la eliminació dels manteniments i piecemaintenances dels mecànics esborrats.

Ja de pas, adjunto el script utilitzat per a la verificació d'aquesta trigger, incloent la creació de taules auxiliars com SELECTs i queries de comprovació de resultats:

```
DROP TABLE IF EXISTS MechanicsFirings;
```

```
CREATE TABLE MechanicsFirings(
```

```
    mechanic_id INT,  
    name VARCHAR(50),  
    surname VARCHAR(50),  
    birth_date DATE,  
    firing_reason TEXT
```

```
);
```

```
CREATE TABLE PiecemaintenanceTrigger
```

```
SELECT * FROM Piecemaintenance;
```

```
DELETE FROM piecemaintenancetrigger;
```

```
INSERT INTO Piecemaintenancetrigger
```

```
SELECT * FROM Piecemaintenance;
```

```
DROP TABLE IF EXISTS MechanicTrigger;
```

```
CREATE TABLE MechanicTrigger
```

```
SELECT * FROM Mechanic;
```

```
INSERT INTO MechanicTrigger
```

```
SELECT * FROM Mechanic;
```

```
CREATE TABLE MaintenanceTrigger
```

```
SELECT * FROM Maintenance;
```

```
DELETE FROM MaintenanceTrigger;
```

```
INSERT INTO MaintenanceTrigger
```

```
SELECT * FROM Maintenance;
```

```
-- • Retirement: If the date when the person is deleted from the table is 65 years old or
```

```
-- older.
```

```
DELIMITER $$
```

```
DROP TRIGGER IF EXISTS dismissal $$
```

```
CREATE TRIGGER dismissal
```

```
    BEFORE DELETE ON MechanicTrigger FOR EACH ROW
```

```
BEGIN
```

```
-- • Retirement: If the date when the person is deleted from the table is 65 years old or
```

```

-- older.
IF (SELECT (YEAR(NOW()) - YEAR(person.born_date)) >= 65 FROM person WHERE
person.personID = OLD.mechanicID) THEN
    INSERT INTO MechanicsFirings(mechanic_id, name, surname, birth_date, firing_reason)
    SELECT p.personID, p.name, p.surname, p.born_date, "Retirement"
    FROM Person AS p
    JOIN Mechanic AS mech ON mech.mechanicID = p.personID
    JOIN Country AS c ON p.countryID = c.countryID
    JOIN City AS city ON c.countryID = city.cityID
    JOIN Airport AS airp ON city.cityID = airp.cityID
    WHERE mech.mechanicID = old.mechanicID
    GROUP BY mech.mechanicID;

-- • Not completing the evaluation period: When the sum of repairs that the mechanic has
-- done does not add up to more than 10 hours.
ELSEIF (SELECT SUM(m.duration) <= 10 FROM maintenance AS m WHERE m.mechanicid =
OLD.mechanicID) THEN
    INSERT INTO MechanicsFirings(mechanic_id, name, surname, birth_date, firing_reason)
    SELECT p.personID, p.name, p.surname, p.born_date, "Not completing the evaluation period"
    FROM Person AS p
    JOIN Mechanic AS mech ON mech.mechanicID = p.personID
    JOIN Maintenance AS m ON mech.mechanicID = m.mechanicID
    WHERE mech.mechanicID = old.mechanicID
    GROUP BY mech.mechanicID;

-- • Firing without reason: When it does not belong to any of the previous types.
ELSE
    INSERT INTO MechanicsFirings(mechanic_id, name, surname, birth_date, firing_reason)
    SELECT p.personID, p.name, p.surname, p.born_date, "Firing without reason"
    FROM Person AS p
    JOIN Mechanic AS mech ON mech.mechanicID = p.personID
    JOIN Maintenance AS m ON mech.mechanicID = m.mechanicID
    WHERE mech.mechanicID = old.mechanicID
    GROUP BY mech.mechanicID;
END IF;

DELETE maintenancettrigger, piecemaintenancettrigger FROM maintenancettrigger JOIN
piecemaintenancettrigger
WHERE maintenancettrigger.mechanicID IN (SELECT mech.mechanicID FROM
MechanicTrigger AS mech WHERE mech.mechanicID = old.mechanicID)
    AND piecemaintenancettrigger.maintenanceID = maintenancettrigger.maintenanceID;
END $$

DELIMITER ;

SELECT * FROM MechanicTrigger;

select * from piecemaintenance;

SELECT * FROM mechanic;

SELECT * FROM mechanicsFirings

```

```

GROUP BY mechanic_id;

SELECT * FROM PiecemaintenanceTrigger;
SELECT * FROM MaintenanceTrigger
ORDER BY mechanicID;

DELETE FROM mechanicTrigger WHERE mechanicID = 103;

SELECT * FROM MechanicTrigger AS mech
RIGHT JOIN Maintenance AS m ON m.mechanicID = mech.mechanicID
JOIN PiecemaintenanceTrigger AS mt ON mt.maintenanceID = m.maintenanceID
JOIN Piece AS p ON p.pieceID = mt.pieceID
WHERE mech.mechanicID = 103
GROUP BY mt.maintenanceID
ORDER BY mech.mechanicID, mt.maintenanceID;

```

Tot i així, i ja com última anotació, la resposta oficial de l'apartat de la solució, i torno a reiterar, que en aquest cas seria substituir les taules auxiliars de versió Trigger per les normals corresponents per a poder eliminar la informació de Piecemaintenance i Maintenance.

### 2.2.8 Requirement (Trigger) 2.8 Petrol updates

#### 2.2.8.1 Solution

```
DROP TABLE IF EXISTS EnvironmentalReductions;
CREATE TABLE EnvironmentalReductions(
    route VARCHAR(100),
    difference INT,
    up_date DATE
);

DELIMITER $$

DROP TRIGGER IF EXISTS history_petrol $$

CREATE TRIGGER history_petrol
    BEFORE UPDATE ON Route FOR EACH ROW
BEGIN
    INSERT INTO EnvironmentalReductions(route, difference, up_date)
        SELECT CONCAT('DEPARTURE:', c.name, '(', country.name, ')', '-->', 'DESTINATION:',
        calt.name, '(', countryalt.name, ')') AS route_dest_dep, ABS(NEW.minimum_petrol -
        old.minimum_petrol) AS diff, NOW() FROM Route AS routealt
        JOIN Airport AS airp ON old.departure_airportID = airp.airportID
        JOIN Airport AS airpalt ON old.destination_airportID = airpalt.airportID
        JOIN City AS c ON c.cityID = airp.cityID
        JOIN City AS calt ON calt.cityID = airpalt.cityID
        JOIN Country AS country ON c.countryID = country.countryID
        JOIN Country AS countryalt ON calt.countryID = countryalt.countryID
        WHERE routealt.routeID = old.routeID;
END $$

DELIMITER ;
```

#### 2.2.8.2 Explanation

Per a aquest trigger, havíem de crear no només la taula que se'ns demanava de EnvironmentalReductions, sinó també especificar al SELECT les relacions adients dels JOINs amb la resta de taules que s'han de relacionar, a més de tenir en compte de fer un UPDATE BEFORE per a captar la informació tant nova com l'anterior, a més d'especificar al SELECT la concatenació de dades a especificar sobre la ruta d'origen i de destí.

Més enllà d'això, aquest trigger era senzill en comparació als dos anteriors.

#### 2.2.8.3 Trigger validation

Al igual que a les dues queries anteriors, he creat en aquesta ocasió també taules auxiliars amb el “cognom” de ‘Trigger’ afegit al nom de la taula per a diferenciar-les de les originals, degut a raons prèviament explicades relacionades amb el meu portàtil. En aquest cas només necessitava crear una taula auxiliar de Route anomenada ‘RouteTrigger1’ (per a diferenciar-lo del RouteTrigger de la query 2.6) a on li afegia la informació de la taula ‘Route’ original.

La query utilitzada per a actualitzar les dades i veure els resultats, va ser la següent:

```
UPDATE RouteTrigger1
SET minimum_petrol = 30000
WHERE routeid >= 1 AND routeid <= 10;
```

A on volem assignar-li un nou valor mínim de petroli de 30000 a les rutes de id = 1 fins a 10, per a poder veure la variació de resultats.

El procés de comprovació va estar el següent:

(Comprovem que la taula de EnvironmentalReductions s'ha creat)

DROP TABLE IF EXISTS EnvironmentalReductions;

CREATE TABLE EnvironmentalReductions(

    route VARCHAR(100),

    difference INT,

    up\_date DATE

);

SELECT \* FROM EnvironmentalReductions;

	route	difference	up_date

(Comprovem que la taula RouteTrigger1 conté la informació original de RouteTrigger)

SELECT \* FROM RouteTrigger1;

	routeID	destination_airportID	departure_airportID	distance	minimum_petrol	time
▶	1	3	1	795	6876	00:47:51
	2	4	1	688	5684	00:37:53
	3	2	1	957	4099	00:51:52
	4	5	1	818	7834	01:02:40
	5	1	2	222	8110	00:54:47
	6	3	2	356	8123	01:04:56
	7	4	2	112	7479	00:49:24
	8	6	2	292	5831	00:41:09
	9	1	3	124	9313	00:41:31
	10	4	3	545	2158	00:59:14

(Executem el trigger establert amb l'UPDATE mostrar prèviament)

(Comprovem si la informació ha canviat a la taula RouteTrigger1, que en la resposta oficial en aquest cas seria Route normal)

SELECT \* FROM RouteTrigger1; (On comprovem que les taules 1 al 10, s'han imposat el nou valor establert de petroli mínim)

	routeID	destination_airportID	departure_airportID	distance	minimum_petrol	time
▶	1	3	1	795	30000	00:47:51
	2	4	1	688	30000	00:37:53
	3	2	1	957	30000	00:51:52
	4	5	1	818	30000	01:02:40
	5	1	2	222	30000	00:54:47
	6	3	2	356	30000	01:04:56
	7	4	2	112	30000	00:49:24
	8	6	2	292	30000	00:41:09
	9	1	3	124	30000	00:41:31
	10	4	3	545	30000	00:59:14
	11	2	3	451	4551	00:41:39
	12	3425	3	523	8369	00:42:29
	13	5	3	260	10679	00:51:32

(Acte seguit comprovem la taula de EnvironmentalReductions per a veure si s'ha afegit la informació correctament, i efectivament es pot comprovar que sí per la diferència de valor entre el valor nou introduït i l'antic)

SELECT \* FROM EnvironmentalReductions;

route	difference	up_date
▶ DEPARTURE:Goroka(Papua New Guinea) --> DESTINATION:Mount Hagen(Papua New Guinea)	23124	2021-05-29
DEPARTURE:Goroka(Papua New Guinea) --> DESTINATION:Nadzab(Papua New Guinea)	24316	2021-05-29
DEPARTURE:Goroka(Papua New Guinea) --> DESTINATION:Madang(Papua New Guinea)	25901	2021-05-29
DEPARTURE:Goroka(Papua New Guinea) --> DESTINATION:Port Moresby(Papua New Guinea)	22166	2021-05-29
DEPARTURE:Madang(Papua New Guinea) --> DESTINATION:Goroka(Papua New Guinea)	21890	2021-05-29
DEPARTURE:Madang(Papua New Guinea) --> DESTINATION:Mount Hagen(Papua New Guinea)	21877	2021-05-29
DEPARTURE:Madang(Papua New Guinea) --> DESTINATION:Nadzab(Papua New Guinea)	22521	2021-05-29
DEPARTURE:Madang(Papua New Guinea) --> DESTINATION:Wewak(Papua New Guinea)	24169	2021-05-29
DEPARTURE:Mount Hagen(Papua New Guinea) --> DESTINATION:Goroka(Papua New Guinea)	20687	2021-05-29
DEPARTURE:Mount Hagen(Papua New Guinea) --> DESTINATION:Nadzab(Papua New Guinea)	27842	2021-05-29

(A més de que una altra forma de comprobar si funciona, és comparant la ruta d'origen de EnvironmentalReductions amb la de departure\_airportID de RouteTrigger1) (VERD)

	routeID	destination_airportID	departure_airportID	distance	minimum_petrol	time
▶	1	3	1	795	30000	00:47:51
	2	4	1	688	30000	00:37:53
	3	2	1	957	30000	00:51:52
	4	5	1	818	30000	01:02:40
	5	1	2	222	30000	00:54:47
	6	3	2	356	30000	01:04:56
	7	4	2	112	30000	00:49:24
	8	6	2	292	30000	00:41:09
	9	1	3	124	30000	00:41:31
	10	4	3	545	30000	00:59:14

(Que com es poden observar, són les mateixes, donant així més veritat del correcte funcionament)

### 2.2.9 Requirement (Event) 2.9 Yearly maintenance costs

#### 2.2.9.1 Solution

```
DROP TABLE IF EXISTS MaintenanceCost;
CREATE TABLE MaintenanceCost(
year int,
planeName VARCHAR(255),
price INT
);

delimiter $$

DROP EVENT IF EXISTS YearlyMaintenanceCosts $$

CREATE EVENT YearlyMaintenanceCosts
    ON SCHEDULE
        EVERY 1 YEAR
    DO
        BEGIN

            INSERT INTO MaintenanceCost
            SELECT YEAR(CURDATE()), planetype.type_name , sum(cost) FROM
                piece INNER JOIN piecemaintenance ON piece.pieceID = piecemaintenance.pieceid
                INNER JOIN maintenance ON piecemaintenance.maintenanceID =
                    maintenance.maintenanceID
                INNER JOIN plane ON plane.planeID = maintenance.planeID
                INNER JOIN planetype ON planetype.planetypename = plane.planetypename
                WHERE YEAR(maintenance.date) = YEAR(CURDATE())
                GROUP BY planetype.planetypename;

        END $$

DELIMITER ;
```

#### 2.2.9.2 Explanation

Per a aquesta ocasió, a l'hora d'implementar aquest trigger se'ns demana tenir en compte que cada any s'introdueixi varis valors de paràmetres a dins d'una taula anomenada YearlyMaintenanceCosts, el qual per a portar-ho a terme, posem el temps d'activació ON SCHEDULE EVERY 1 YEAR i fem els JOINs adients a l'hora de relacionar les taules del problema. D'aquesta forma garantim que cada any la informació es vagi introduint a la taula de forma eficient.

#### 2.2.9.3 Trigger validation

Els passos a seguir a l'hora de comprovar aquest event ha estat el següent:

(Creem la taula de MaintenanceCost i mirem que està creada sense errors)

```
DROP TABLE IF EXISTS MaintenanceCost;
CREATE TABLE MaintenanceCost(
year int,
planeName VARCHAR(255),
price INT
```

```
);
SELECT * FROM MaintenanceCost;
```

	year	planeName	price

(Al event indiquem que l'SCHEDULE el faci cada 1 MINUTE, per a no estar esperant un any)

```
ON SCHEDULE
EVERY 1 MINUTE
DO
```

(Al igual que als triggers previs per les raons prèviament explicitades en referència al meu portàtil, creem una taula auxiliar anomenada MaintenanceEvent on li farem un update del seu apartat date per a tenir dades del 2021 que no teníem abans per predeterminat amb la taula original Maintenance)

```
CREATE TABLE MaintenanceEvent
SELECT * FROM Maintenance;
DELETE FROM MaintenanceEvent;
INSERT INTO MaintenanceEvent
SELECT * FROM Maintenance;
```

```
SELECT * FROM Maintenanceevent
WHERE YEAR(date) = 2021;
```

	maintenanceID	duration	planeID	mechanicID	date

(Fem un UPDATE de tots aquells events que han succeït abans del 1970, a 2021-02-02)

```
UPDATE MaintenanceEvent
SET date = '2021-02-02'
WHERE YEAR(date) < 1970;
```

```
SELECT * FROM Maintenanceevent
WHERE YEAR(date) = 2021;
```

	maintenanceID	duration	planeID	mechanicID	date
▶	3	16	321	125	2021-02-02
	16	53	2778	334	2021-02-02
	30	25	3134	334	2021-02-02
	54	62	4781	75	2021-02-02
	71	13	3148	248	2021-02-02
	103	33	237	31	2021-02-02
	142	60	2016	31	2021-02-02
	266	44	608	229	2021-02-02
	293	78	2856	229	2021-02-02
	368	7	1251	334	2021-02-02
	376	74	3335	229	2021-02-02
	378	20	4052	248	2021-02-02
	415	17	4917	339	2021-02-02
	515	76	2541	334	2021-02-02
	586	52	2152	364	2021-02-02
	693	79	3329	217	2021-02-02
	737	65	4635	214	2021-02-02
	787	20	2019	143	2021-02-02
	843	30	1858	217	2021-02-02
	845	16	3634	248	2021-02-02
	880	79	3089	334	2021-02-02
	939	61	4710	334	2021-02-02
	948	71	1915	217	2021-02-02
	1042	15	2083	359	2021-02-02

(Apart d'això, creem una taula auxiliar de pieceevent on a la maintenanceID = 71 li assignem la seva pieceID = 1 un cost = 0)

CREATE TABLE PieceEvent

SELECT \* FROM Piece;

DELETE FROM PieceEvent;

INSERT INTO PieceEvent

SELECT \* FROM Piece;

SELECT \* FROM pieceevent

WHERE pieceID = 1;

	pieceID	name	cost
▶	1	Aileron	610271

SELECT \* FROM piecemaintenance

WHERE pieceID = 1;

	maintenanceID	pieceID
	71	1
	72	1
	73	1
	74	1
	75	1
	76	1
	81	1

```
UPDATE pieceevent SET cost = 0 WHERE pieceID = 1;
SELECT * FROM pieceevent
WHERE pieceID = 1;
```

	pieceID	name	cost
▶	1	Aileron	0

(Fem un SET GLOBAL event\_scheduler = ON per a activar l'event)

	Id	User	Host	db	Command	Time	State	Info
▶	8	root	localhost:56877	Isair	Sleep	81		NULL
	9	root	localhost:56878	Isair	Query	0	starting	SHOW processlist
	10	event_scheduler	localhost	NULL	Daemon	28	Waiting for next activation	NULL

(Deixem passar 1 minut de temps, i ens anirà introduint a la taula de MaintenanceCosts la informació demanada, però en aquest cas com és d'1 minut de temps, aquesta informació es repeteix cada 1 minut. Però si és de forma anual, llavors la informació no es repeteix i farà la seva tasca com hauria de ser)

```
SELECT * FROM MaintenanceCost;
```

	year	planeName	price
▶	2021	Aerospatiale/Alenia ATR 42-300	2506822
	2021	Aerospatiale/Alenia ATR 72	980785
	2021	Airbus A319	2278339
	2021	Airbus A320	25243217
	2021	Airbus A330-300	2950403
	2021	Avro RJ100	5926315
	2021	Boeing 737	2834232
	2021	Boeing 737-300	7254792
	2021	Boeing 737-800	13302284
	2021	Boeing 757	2059762
	2021	Boeing 767	3462846
	2021	Boeing 777-300	1402925
	2021	Embraer 175	4073649
	2021	Saab SF340A/B	4539826

I fins aquí, aquest event queda validat. I abaix deixo el script utilitzat per a la validació d'aquesta última query:

```
UPDATE MaintenanceEvent
SET date = '2021-02-02'
WHERE YEAR(date) < 1970;

CREATE TABLE MaintenanceEvent
SELECT * FROM Maintenance;
DELETE FROM MaintenanceEvent;
INSERT INTO MaintenanceEvent
SELECT * FROM Maintenance;

CREATE TABLE PieceEvent
SELECT * FROM Piece;
DELETE FROM PieceEvent;
INSERT INTO PieceEvent
SELECT * FROM Piece;

SELECT date FROM maintenance;
SELECT * FROM Maintenanceevent
WHERE YEAR(date) = 2021;

SELECT * FROM piecemaintenance
WHERE pieceID = 1;

SELECT * FROM pieceevent
WHERE pieceID = 1;

-- SELECT * FROM piece;
UPDATE pieceevent SET cost = 0 WHERE pieceID = 1;

DROP TABLE IF EXISTS MaintenanceCost;
CREATE TABLE MaintenanceCost(
year int,
planeName VARCHAR(255),
price INT
);

DELIMITER $$
DROP EVENT IF EXISTS YearlyMaintenanceCosts $$
CREATE EVENT YearlyMaintenanceCosts
ON SCHEDULE
EVERY 1 MINUTE
DO
BEGIN

    INSERT INTO MaintenanceCost
    SELECT YEAR(CURDATE()), planetype.type_name , sum(cost) FROM
        pieceevent INNER JOIN piecemaintenance ON pieceevent.pieceID =
        piecemaintenance.pieceid
```

```
    INNER JOIN maintenanceevent ON piecemaintenance.maintenanceID =
maintenanceevent.maintenanceID
```

```
    INNER JOIN plane ON plane.planeID = maintenanceevent.planeID
```

```
    INNER JOIN planetype ON planetype.planetypeID = plane.planetypeID
```

```
    WHERE YEAR(maintenanceevent.date) = YEAR(CURDATE())
```

```
    GROUP BY planetype.planetypeID;
```

```
END $$
```

```
DELIMITER ;
```

```
SELECT * FROM MaintenanceCost;
```

```
SHOW processlist;
```

```
SET GLOBAL event_scheduler = OFF;
```

## 2.3 Shopping module

### 2.3.1 Requirement (Query) 3.1 Local commerce

#### 2.3.1.1 Solution

```
select distinct c.name, co.name
from product as p
join company as c on p.companyID = c.companyID
join food as f on f.foodID = p.productID
join country as co on c.countryID = co.countryID
where c.countryid = f.countryid
and c.companyid in (select co.companyID
                     from company as co
                     join product as pr on co.companyID = pr.companyID
                     join food as fo on pr.productID = fo.foodID
                     group by co.companyID
                     having count(fo.foodID) >= 40 )  
;
```

name	name
Babblestorm	Japan
Cogibox	Mexico
Dabtype	Australia
Demizz	Switzerland
Devshare	Japan
Dynabox	Saudi Arabia
Dynazzy	China
Katz	Netherlands
Kaymbo	Norway
Latz	Kenya
Linktype	Norway
Meezzy	Spain
Mynte	United Kingdom
Quamba	Kenya
Quimba	Spain
Rhybox	United States
Rhycero	Netherlands
Riffwire	Canada
Roombo	Mexico

\*Estem visualitzant 19 files de les 28 totals que ens retorna la query.

#### 2.3.1.2 Explanation

En aquesta query se'ns demana de torna les companyies que tenen almenys 1 de 40 productes que sigui local. És a dir, la companyia ha de tenir com a mínim un producte local i mínim 40 productes de menjar en total. Per tant en la meva query primer verifico en un subquery que hi hagi un mínim de 40 productes de menjar. Un cop sé quines companyies compleixen aquest requeriment afegeixo la subquery en el “where” per tal que les companyies que compleixen la subquery també compleixin l’altra verificació en el “where” que es que el país de on prové el menjar sigui el mateix que el país de la companyia. Per tant les companyies que compleixin aquestes dues condicions seran les companyies que retornarem.

### 2.3.1.3 Query validation

Per verificar aquesta query he utilitzat dues queries diferents. En la primera query verifico que les companyies tinguin com a mínim 40 productes del tipus menjar:

```
select co.name, co.companyID, count(fo.foodID)
from company as co
join product as pr on co.companyID = pr.companyID
join food as fo on pr.productID = fo.foodID
group by co.companyID
having count(fo.foodID) >= 40 ;
```

name	companyID	count(fo.foodID)
Rhyzio	100	40
Mudo	80	40
Plajo	86	41
Myworks	90	44
Snaptags	99	44
Edgeblab	88	51
Blogtags	81	51
Linklinks	84	53
Ozu	134	70
Minyx	131	77
Oyoyo	101	82
Linktype	167	84
Oyondu	166	85
Thought...	117	85
Fatz	164	86

Podem veure que al ordenar el resultat segons el numero de productes de menjar en ordre ascendent, el mínim és de 40.

\*En aquesta query estem visualitzant una part de les 77 files resultants.

Ara em queda verificar si aquestes companies tenen com a mínim 1 producte de menjar local. La query que hem permet verificar això és la següent:

```
select c.name, co.name, count(f.foodID) as local_food
from product as p
join company as c on p.companyID = c.companyID
join food as f on f.foodID = p.productID
join country as co on c.countryID = co.countryID
where c.countryid = f.countryid
and c.companyid in (select co.companyID
                    from company as co
                    join product as pr on co.companyID = pr.companyID
                    join food as fo on pr.productID = fo.foodID
                    group by co.companyID
                    having count(fo.foodID) >= 40 )
group by c.companyID;
```

name		name	local_food
Kaymbo		Norway	2
Katz		Netherlands	2
Rhybox		United States	1
Latz		Kenya	1
Mynte		United Kingdom	1
Skimia		Morocco	1
Yozio		India	1
Babblestorm		Japan	1
Zooxo		United Kingdom	1
Dabtype		Australia	1
Skimia		Switzerland	1
Linktype		Norway	1
Dynazzy		China	1
Riffwire		Canada	1
Skimia		Saudi Arabia	1

\*Aquesta query ens retorna 28 files.

Segons el resultat d'aquesta query podem veure que totes les companyies tenen com a mínim un producte local.

### 2.3.2 Requirement (Query) 3.2 Best trade relations

#### 2.3.2.1 Solution

```

select ct_vr.name, ct_re.name, count(*) as number_of_vip_rest

from vip_room as vr
join waitingarea as wa_vr on wa_vr.waitingAreaID = vr.vipID
join company as co_vr on wa_vr.companyID = co_vr.companyID
join country as ct_vr on ct_vr.countryID = co_vr.countryID

join waitingarea as wa_re
join restaurant as re on wa_re.waitingAreaID = re.restaurantID
join company as co_re on wa_re.companyID = co_re.companyID
join country as ct_re on ct_re.countryID = co_re.countryID

where ct_vr.countryID <> ct_re.countryID
and vr.restaurantID = re.restaurantID
and ct_vr.countryID in (select co.countryID
from country as co
join forbiddenproducts as fp on co.countryID = fp.countryID
join product as pr on pr.productID = fp.productID
group by co.countryID
having count(pr.productID) <= 80 )
and ct_re.countryID in (select co.countryID
from country as co
join forbiddenproducts as fp on co.countryID = fp.countryID
join product as pr on pr.productID = fp.productID
group by co.countryID
having count(pr.productID) <= 80 )

group by ct_re.countryID, ct_vr.countryID
;

```

\*En aquesta query apareixen en total 23 files.

### 2.3.2.2 Explanation

En aquesta query havíem de mostrar els països en parells. Cada país havia de tenir menys de 80 productes prohibits. Això es el que verifiquem dins de la subquery. A la resta estem aparellant els països que s'ocupen de la vip-room amb el país del restaurant de la vip-room. Per fer això hem ajuntat dos taules de paisos, una per al pais de la vip-room i l'altre per el país del restaurant. També havíem de crear dos companies i dos waitingAreas per això. Verifiquem que els dos països tinguin una vip-room o un restaurant relacionats i si es el cas doncs els mostrem.

### 2.3.2.3 Query validation

Per validar aquesta query primer he realitzat la mateixa query que la solució però traient el group by final i mostrant el id de les waitingAreas:

```
select distinct ct_vr.name, wa_vr.waitingAreaID, wa_re.waitingAreaID, ct_re.name-- , count(*) as number_of_vip_rest

from vip_room as vr
join waitingarea as wa_vr on wa_vr.waitingAreaID = vr.vipID
join company as co_vr on wa_vr.companyID = co_vr.companyID
join country as ct_vr on ct_vr.countryID = co_vr.countryID

join waitingarea as wa_re
join restaurant as re on wa_re.waitingAreaID = re.restaurantID
join company as co_re on wa_re.companyID = co_re.companyID
join country as ct_re on ct_re.countryID = co_re.countryID

where ct_vr.countryID <> ct_re.countryID
and vr.restaurantID = re.restaurantID
and ct_vr.countryID in (select co.countryID
                        from country as co
                        join forbiddenproducts as fp on co.countryID = fp.countryID
                        join product as pr on pr.productID = fp.productID
                        group by co.countryID
                        having count(pr.productID) <= 80 )
and ct_re.countryID in (select co.countryID
                        from country as co
                        join forbiddenproducts as fp on co.countryID = fp.countryID
                        join product as pr on pr.productID = fp.productID
                        group by co.countryID
                        having count(pr.productID) <= 80 )
;
```

name	waitingAreaID	waitingAreaID	name
Saudi Arabia	1569	608	China
Norway	1393	768	Argentina
Norway	1857	825	China
Argentina	1976	825	China
China	1482	909	United Kingdom
China	896	933	Argentina
United States	1073	933	Argentina
Norway	734	1020	China
Netherlands	831	1020	China
Argentina	1232	1020	China
Saudi Arabia	1679	1020	China
Norway	1892	1020	China
China	1009	1360	United States
Germany	1111	1360	United States
Norway	824	1429	Germany
China	1904	1429	Germany
Norway	1118	1438	China
India	1291	1438	China

\*En aquesta query apareixen 37 files.

Si analitzem els resultats de la solució i els de aquesta query podem veure que concorden. Per exemple si afagim la relació de “Saudi Arabia” i “China” podem veure que tenen dos vip-rooms. Podem veure en aquesta taula que Saudi Arabia té una waitingArea amb id 1569 relacionada amb una waitingArea amb id 608 pertanyen a la China, I també passa lo mateix amb les waitingArea amb id 1679 i 1020 i amb cap altre cas. Per tant els resultats concorden efectivament.

Una altre query de verificació que he utilitzat es la de mirar quants productes prohibits té cada país.

```
select co.countryID, count(pr.productID)
from country as co
join forbiddenproducts as fp on co.countryID = fp.countryID
join product as pr on pr.productID = fp.productID
group by co.countryID
having count(pr.productID);
```

countryID	count(pr.productID)
1	70
2	86
3	76
4	88
5	79
6	62
7	72
8	58
9	80
10	64
11	68
12	84
13	82
14	84
15	80
16	85
17	87
18	79

\*Aquesta query té més files de les que apareixen

Gràcies a aquesta query podem verificar si els països que apareixen haurien d'aparèixer o no. Si analitzem els resultats podem concloure que tots els països amb més de 80 productes prohibits no apareixen.

### 2.3.3 Requirement (Query) 3.3 The best restaurant

#### 2.3.3.1 Solution

```
select co.name, re.score
from company as co
join waitingarea as wa on co.companyID = wa.companyID
join restaurant as re on wa.waitingAreaID = re.restaurantID
where co.companyID = (select co2.companyID
                      from company as co2
                      join waitingarea as wa2 on co2.companyID = wa2.companyID
                      group by co2.companyID
                      order by count(wa2.waitingAreaID) desc
                      limit 1)
ORDER BY re.score DESC
limit 1;
```

name	score
Zazio	9.47

#### 2.3.3.2 Explanation

En aquesta query haig de retornar el restaurant amb la millor nota de la companyia amb més waiting areas. Al principi vaig pensar fer dos subqueries, una per escollir el millor restaurant i una segona per escollir la companyia amb més waitingAreas. Més tard em vaig adonar que ho podia fer amb una sola subquery i un order by. En la query final doncs, faig una subquery per seleccionar la companyia amb més waitingAreas. Després faig un order by la puntuació dels restaurants i ho limito a un ja que ens demanen el restaurant amb la millor puntuació.

#### 2.3.3.3 Query validation

Per a la verificació d'aquesta query he realitzat dues queries. En la primera miro el número de waitingAreas que té cada companyia.

```
select co.companyID, co.name, COUNT(wa.waitingAreaID)
from company as co
join waitingarea as wa on co.companyID = wa.companyID
group by co.companyID
order by COUNT(wa.waitingAreaID) DESC;
```

companyID	name	COUNT(wa.waitingAreaID)
251	Zazio	25
25	Quire	21
7	Skaboo	20
8	Aimbo	20
15	Quinu	19
41	Shufflester	18
50	Babbleopia	18
287	Wordpedia	18
23	Mymm	17
27	Realcube	17
48	Eimbee	17
274	Twimm	17
37	Miboo	16
39	Twitterbeat	16
46	Pixoboo	16
257	Leenti	16
266	Livetube	16
3	Meedoo	15

Podem observar que la primera que apareix es la mateixa que en el nostre resultat de la solució. Per tant “Zazio” es la companyia que té més waitingAreas.

La segona query de verificació miro la nota de cada restaurant de la primera companyia que ens retorna la query de sobre. És a dir la companyia amb id = 251.

```
select wa.companyID, wa.waitingareaID, re.score
from waitingArea as wa
join restaurant as re on wa.waitingAreaID = re.restaurantID
where wa.companyID = 251;
```

companyID	waitingareaID	score
251	355	2.16
251	460	5.84
251	478	6.4
251	563	5.14
251	576	7.97
251	600	6.64
251	679	5.05
251	943	7.32
251	998	5.38
251	1027	8
251	1047	5.44
251	1119	7.96
251	1296	1.54
251	1328	8.77
251	1339	0.02
251	1347	6.93
251	1420	9.26
251	1618	5.38
251	1834	7.79
251	1853	4.04
251	2037	2.01
251	2145	9.47
251	2206	2.07
251	2266	8.7
251	2359	3.68

Podem observar que en aquests resultats el restaurant amb millor nota es el que té nota de 9.47, el mateix que a la nostra solució.

Podem concloure que la solució es correcte.

### 2.3.4 Requirement (Query) 3.4 Stores and restaurants

#### 2.3.4.1 Solution

```

select co.name, co.company_value
from company as co
join product as po on po.companyID = co.companyID
join productstore as ps on po.productID = ps.productID

where co.companyID in (select co3.companyID
                        from company as co3
                        join waitingarea as wa on wa.companyID = co3.companyID
                        join restaurant as re on re.restaurantID = wa.waitingareaID
                        group by co3.companyID
                        having count(distinct re.type)>= 2)

group by ps.storeID, co.companyID
having 0.2 <= (select (count(po.productID)/count(po2.productID))
                  from company as co2
                  join product as po2 on po2.companyID = co2.companyID
                  where co2.companyID = co.companyID
                  group by co2.companyID)
;
```

<u>name</u>	<u>company_value</u>
Latz	35479853
Buzzdog	22842172
Rhybox	32948342
Devshare	32945576
Fatz	8653178
Skimia	40531140
Centidel	21096882
Oyondu	13059034
Kaymbo	4307913
Linklinks	30410268
Dynabox	17303884
Tagfeed	5074245

#### 2.3.4.2 Explanation

Per aquesta query havíem de mostrar les companyies que tenen botigues que venen al menys 20% dels seus productes i restaurants de almenys dos tipus.

Primer, doncs, he realitzat una subquery en la que verifico la segona condició en el where i després, en el having he fet la segona subquery per verificar la primera condició.

### 2.3.4.3 Query validation

Per a verificar aquesta query he fet dues queries.

En la primera verifico totes les companyies que tenen més de dos tipus de restaurants:

```
select co3.companyID, co3.name, co3.company_value
from company as co3
join waitingarea as wa on wa.companyID = co3.companyID
join restaurant as re on re.restaurantID = wa.waitingareaID
group by co3.companyID
having count(distinct re.type)>= 2;
```

companyID	name	company_value
150	Kaymbo	4307913
151	Devshare	32945576
152	Latz	35479853
153	Skimia	40531140
154	Centidel	21096882
156	Dynabox	17303884
157	Linklinks	30410268
159	Yodoo	23934998
160	Voomm	20521030

Després verifico les companies que verifiquem l'altre condició:

```
select co.companyID, co.name, co.company_value
from company as co
join product as po on po.companyID = co.companyID
join productstore as ps on po.productID = ps.productID
group by ps.storeID, co.companyID
having 0.2 <= (select (count(po.productID)/count(po2.productID))
                 from company as co2
                 join product as po2 on po2.companyID = co2.companyID
                 where co2.companyID = co.companyID
                 group by co2.companyID)
```

companyID	name	company_value
74	Brainsphere	21597139
63	Skinder	26521568
79	Browsertype	10506298
87	Topiczoom	26290363
46	Pixoboo	21562643
152	Latz	35479853
107	Yozio	39997093
60	Vinder	11072934
8	Aimbo	42686960

Si comparem els resultats podem veure que, per exemple “Latz” compleix els dos requeriments, per tant la query es correcta.

### 2.3.5 Requirement (Query) 3.5 Waiting areas without shop keepers

#### 2.3.5.1 Solution

```

select distinct co.name, wa.opening_hour, wa.close_hour, wa.airportID, wa.waitingareaID
from waitingarea as wa
join shopkeeper as sk on sk.waitingAreaID = wa.waitingAreaID
join company as co on wa.companyID = co.companyID
where wa.close_hour between wa.opening_hour and '23:59:59'
and time_to_sec(timediff(wa.close_hour, wa.opening_hour )*7) > (select
time_to_sec(sum(sk2.weekly_hours))

from shopkeeper as sk2
join waitingarea as wa2 on wa2.waitingareaID =
sk2.waitingareaID
where wa2.waitingareaID = wa.waitingareaID
group by wa2.waitingareaID)

;

```

name	opening_hour	close_hour	airportID	waitingareaID
Meedoo	10:00:00	23:00:00	5757	2
Kazio	07:00:00	20:00:00	575	3
Feedfish	06:00:00	15:00:00	2101	7
Twinder	08:00:00	22:00:00	5560	9
Blogspan	05:00:00	20:00:00	1187	10
Photobug	06:00:00	16:00:00	5486	15
Geba	09:00:00	20:00:00	575	24
Skajo	10:00:00	23:00:00	2101	48
Fanoodle	06:00:00	17:00:00	5560	53
Fanoodle	08:00:00	19:00:00	5757	54
Roomm	10:00:00	23:00:00	575	55
Rifffpedia	07:00:00	21:00:00	3213	80
Oodoo	08:00:00	20:00:00	3213	89
Skajo	06:00:00	23:00:00	5486	91
Tagtune	07:00:00	19:00:00	1347	96
Livetube	06:00:00	20:00:00	5449	101
Twitterlist	06:00:00	17:00:00	1347	148
Yodo	09:00:00	22:00:00	5449	151
Aimbo	09:00:00	19:00:00	5560	152
Skajo	10:00:00	23:00:00	5757	158

\*En aquesta query apareixen 345 files.

#### 2.3.5.2 Explanation

En aquesta query ens demanaven quines waitingAreas no tenen suficients shopkeepers per cobrir l'horari d'obertura. El que he fet es primer calcular el temps que esta oberta en una setmana la waitingArea (restant hora d'obertura i de tancament i multiplicant per 7) i després calcular la suma del total de hores a la setmana de cada shopkeeper d'aquesta waitingArea. Verifico si el número d'hores obertes es superior al de la suma de les hores dels shopkeepers i si és el cas vol dir que no hi han suficients shopkeepers per cobrir el horari de la waitingArea.

### 2.3.5.3 Query validation

Podem verificar aquesta query mirant el número total de hores que està oberta la waitingArea en una setmana en una query i la suma del número de hores que treballa cada shopkeeper de la waitingArea.

En aquesta query verifiquem lo primer:

```
select distinct co.name, wa.opening_hour, wa.close_hour, wa.airportID, wa.waitingareaID,
time_to_sec(timediff(wa.close_hour, wa.opening_hour )*7)
from waitingarea as wa
join shopkeeper as sk on sk.waitingAreaID = wa.waitingAreaID
join company as co on wa.companyID = co.companyID
where wa.close_hour between wa.opening_hour and '23:59:59';
```

name	opening_hour	close_hour	airportID	waitingareaID	time_to_sec(timediff(wa.close_hour, wa.opening_hour )*7)
Miboo	08:00:00	15:00:00	1187	1	176400
Meedoo	10:00:00	23:00:00	5757	2	327600
Kazio	07:00:00	20:00:00	575	3	327600
Twitterbeat	08:00:00	16:00:00	1187	4	201600
Twitterbeat	04:00:00	15:00:00	337	5	277200
Babbleopia	10:00:00	20:00:00	5560	6	252000
Feedfish	06:00:00	15:00:00	2101	7	226800
Realcube	05:00:00	21:00:00	2916	8	403200
Twinder	08:00:00	22:00:00	5560	9	352800
Blogspan	05:00:00	20:00:00	1187	10	378000
Divavu	09:00:00	19:00:00	1347	11	252000
Realcube	07:00:00	21:00:00	5757	12	352800
Shufflester	05:00:00	16:00:00	5449	13	277200
Twitterbeat	07:00:00	15:00:00	2101	14	201600
Photobug	06:00:00	16:00:00	5486	15	252000
Photospace	06:00:00	19:00:00	3213	16	327600
Skajo	04:00:00	15:00:00	5449	17	277200
Roomm	11:00:00	22:00:00	575	18	277200
Skaboo	04:00:00	19:00:00	5449	19	378000

I en aquesta segona query fem lo segon:

```
select distinct time_to_sec(sum(sk2.weekly_hours)), wa2.waitingareaID
from shopkeeper as sk2
join waitingarea as wa2 on wa2.waitingareaID = sk2.waitingareaID
-- where wa2.waitingareaID = wa.waitingareaID
group by wa2.waitingareaID;
```

time_to_sec(sum(sk2.weekly_hours))	waitingareaID
180000	1
262800	2
259200	3
428400	4
558000	5
568800	6
208800	7
511200	8
169200	9
136800	10
482400	11
514800	12
374400	13
1220400	14
0	15
961200	16
900000	17
543600	18
590400	19
716400	20

Podem veure que quan el número de hores total dels shopkeeper es inferior al número de hores totals de la waitingArea, la companyia apareix en la solució.

### **2.3.6 Requirement (Trigger) 3.6 Waiting areas shutdown**

#### **2.3.6.1 Solution**

drop table if exists EconomicReductions;

create table if not exists EconomicReductions (

```
    companyName varchar(255),
    waitingAreaName varchar(255),
    annualSavings float,
    annualExpenses float);
```

delimiter \$\$

drop trigger if exists trigger1 \$\$

create trigger trigger1 after delete on waitingarea

for each row

begin

```
declare name_waitingArea varchar(255);
declare annual_savings float;
declare annual_expenses float;
declare company_name varchar(255);
```

```
if (select OLD.waitingAreaID = vip_room.vipID from vip_room where vip_room.vipID =
OLD.waitingAreaID) then
```

```
    select 'vip_room'
```

```
    into name_waitingArea;
```

else

```
    if (select OLD.waitingAreaID = restaurant.restaurantID from restaurant where
restaurant.restaurantID = OLD.waitingAreaID) then
```

```
        select 'restaurant'
```

```
        into name_waitingArea;
```

```
    else
```

```
        if (select OLD.waitingAreaID = store.storeID from store where store.storeId
= OLD.waitingAreaID) then
```

```
            select 'store'
```

```
            into name_waitingArea;
```

```
        end if;
```

```
    end if;
```

```
end if;
```

```
    select sum(sk.weekly_hours)* 52 * 10
```

```
    into annual_savings
```

```
    from shopkeeper as sk
```

```
    where sk.waitingAreaID = OLD.waitingAreaID
```

```
    group by OLD.waitingAreaID;
```

```

select sum(sk.weekly_hours) * 52 * 10
into annual_expenses
from waitingArea as wa
join shopkeeper as sk on sk.waitingAreaID = wa.waitingAreaID
where wa.waitingAreaID <> OLD.waitingAreaID
and wa.companyID = OLD.companyID
group by wa.companyID;

```

```

select co.name
into company_name
from company as co
where co.companyID = OLD.companyID;

```

```

insert into EconomicReductions (companyName, waitingAreaName, annualSavings,
annualExpenses)
values (company_name, name_waitingArea, annual_savings, annual_expenses);

```

```
if annual_expenses = 0 then
```

```
    -- fico la relacio companyia product a null
```

```

        update product
        set companyID = null
        where companyID = OLD.companyID;
        -- elimino la companyia
        delete company
        from company
        where companyID = OLD.companyID;
    
```

```
end if;
```

```
end $$
```

```
delimiter ;
```

### 2.3.6.2 *Explanation*

En aquest trigger ens demanen d'introduir la waitingArea que estigui eliminada en una taula anomenada EconomicReductions.

En aquesta taula ens demanaven introduir el nom de la waitingArea tancada però com que en la base de dades no hi ha cap atribut de taula que es digui nom de la waitingArea he hagut de verificar per cada waitingArea si pertany a la taula de Stores, Restaurants o Vip-Room i aleshores ficar en una variable que declarem nosaltres el nom de la waitingArea. Un cop tenim el nom de la waitingArea he decidit insertar les altres dades en altres variables enlloc de fer-ho en el insert ja que trobo que es més elegant. Finalment faig el insert a la taula i si la companyia ja no té cap waitingArea, l'elimino de la base de dades.

### 2.3.6.3 Trigger validation

Per validar el meu trigger he eliminat una waitingArea:

```
-- update de la vip-room relacionada amb el restaurant
update vip_room
join restaurant on restaurant.restaurantID = vip_room.restaurantID
join waitingArea on vip_room.vipID = waitingArea.waitingAreaID
set vip_room.restaurantID = null
where waitingArea.companyID = 100;
-- delete restaurants
delete restaurant from restaurant
join waitingArea as wa on wa.waitingAreaID = restaurant.restaurantID
where wa.companyID = 100;
-- delete vip-room
delete vip_room from vip_room
join waitingArea as wa on wa.waitingAreaID = vip_room.vipID
where wa.companyID = 100;
-- delete productstore
delete productstore from productstore
join waitingArea as wa on productstore.storeID = wa.waitingAreaID
where wa.companyID = 100;
-- delete shops
delete store from store
join waitingArea as wa on wa.waitingAreaID = store.storeID
where wa.companyID = 100;
-- delete shopkeepers
delete shopkeeper from shopkeeper
join waitingArea as wa on wa.waitingAreaID = shopkeeper.waitingAreaID
where wa.companyID = 100;

-- delete waitingArea
delete waitingArea
from waitingArea
where waitingArea.companyID = 100;
```

Despres selecciono la taula en la que s'han de importar les dades:

```
select * from EconomicReductions;
```

companyName	waitingAreaName	annualSavings	annualExpenses
Meedoo	store	379600000	11133200000
Photolist	restaurant	1393600000	1918800000

Podem observar que si ho repetim per diferentes ID se'ns omple la taula amb la informació adequada

### 2.3.7 Requirement (Trigger) 3.7 Updated products

#### 2.3.7.1 Solution

```
drop table if exists PriceUpdates;
```

```
create table if not exists PriceUpdates (
```

```
    productName varchar(255),
    companyOfProduct varchar(255),
    previousPrice float,
    laterPrice float,
    dateOfChange date,
    comment varchar(255));
```

```
delimiter $$
```

```
drop trigger if exists trigger2 $$
```

```
create trigger trigger2 after update on product
for each row
begin
```

```
    declare product_name varchar(255);
    declare company_name varchar(255);
    declare previous_price float;
    declare later_price float;
    declare date_change date;
    declare comment varchar (255);
```

```
    if (EXISTS (select PriceUpdates.productName from PriceUpdates where
    PriceUpdates.productname = OLD.name and PriceUpdates.companyOfProduct = (SELECT
    comp.name
```

```
        FROM company as comp
        where
```

```
        OLD.companyID = comp.companyID)))
        then
```

```
            select "This product has been changing over time, it is possible that it is a strategy of the
            company"
```

```
            into comment;
            end if;
```

```
            select OLD.name, co.name, OLD.price, NEW.price, NOW()
            into product_name, company_name, previous_price, later_price, date_change
            from product as pr
            join company as co on OLD.companyID = co.companyID
            where old.productID = pr.productID;
```

```

insert into PriceUpdates (productName, companyOfProduct, previousPrice, laterPrice,
dateOfChange, comment)
values (product_name, company_name, previous_price, later_price, NOW(), comment);

end $$

delimiter ;

```

### 2.3.7.2 Explanation

En aquesta query ens demanaven d'introduir en la taula PriceUpdates, dades sobre un producte del que el preu ha canviat. Una d'aquestes dades és un comentari que, si el producte ja estava en la taula, hem de insertar un comentari. Per tant hem de crear un if verificant si el producte ja esta en la nostre taula i si es compleix, insertem el comentari. Després, insertem la resta de les dades en variables que he creat, perquè, com he dit abans, trobo que d'aquesta manera el codi queda més bonic i elegant.

### 2.3.7.3 Trigger validation

Per validar aquest trigger simplement he fet un update de la taula product, en el que actualitzava el preu d'algun producte qualsevol. Si ho faig varios cops, tant per productes nous com per productes que el seu preu ja ha sigut actualitzat un vegada podem veure el resultat següent:

```

update product
set price = 910.55
where productID = 41;

select * from PriceUpdates;

```

productName	companyOfProduct	previousPrice	laterPrice	dateOfChange	comment
Prodder	Skimia	1057	117	2021-05-27	NULL
Subin	Skimia	31.99	9.8	2021-05-27	NULL
Sonsing	Jaxspan	820	9	2021-05-27	NULL
Prodder	Skimia	117	9	2021-05-27	This product has been changing over time, it is ...
Prodder	Skimia	9	910.55	2021-05-27	This product has been changing over time, it is ...

Podem observar que quan el producte ja ha sigut actualitzat almenys una vegada el comentari apareix. i en cas de no existir en la taula el comentari es null.

### 2.3.8 Requirement (Trigger) 3.8 Product values per m<sup>2</sup>

#### 2.3.8.1 Solution

```
drop table if exists AverageSquareMetreValue;
```

```
create table if not exists AverageSquareMetreValue (
    storeId integer,
    valueM2 float
);
```

```
delimiter $$
```

```
drop trigger if exists trigger3 $$
```

```
create trigger trigger3 after insert on productstore
for each row
begin
```

```
    delete from AverageSquareMetreValue;
```

```
    insert into AverageSquareMetreValue (storeId, valueM2)
        select st.storeID, avg(pr.price)/st.surface
        from store as st
        join productstore as prst on prst.storeID = st.storeID
        join product as pr on prst.productID = pr.productID
        group by st.storeID;
```

```
end $$
```

```
delimiter ;
```

#### 2.3.8.2 Explanation

Per a aquesta query, per cada cop que s'insereix un producte en una tenda hem de afegir totes les tendes i el seu valor per metre quadrat. Al principi vaig pensar fer-ho mitjançant un update o un insert depenent de si ja existia o no en la taula. Més tard vaig pensar que si les borrava totes al principi de cada trigger i les inseria després seria molt més senzill. Per tant, a cada trigger, primer buidem la taula, i inserim per totes les stores el seu id i valor per metre quadrat.

#### 2.3.8.3 Trigger validation

Com a verificació simplement serveix fent un update en el que s'insereix un producte en una store:

```
insert into productstore (storeID, productID)
values (10, 21);
```

```
select * from AverageSquareMetreValue;
```

storeId	valueM2
1432	0.537286
764	0.862367
1057	0.722887
1025	13.8919
57	8.43903
758	0.332905
1038	0.0534282
737	0.582396
31	39.8269
960	2.7501
659	0.302086
692	1.67143
697	0.0751418
74	7.71351
803	0.460328
78	20.4537
37	8.62781
1322	4.21561
895	0.305842
1410	0.103277
141	11.7181

\*Aquesta query retorna 438 files

Podem veure doncs que el trigger funciona correctament.

### 2.3.9 Requirement (Event) 3.9 Expired food

#### 2.3.9.1 Solution

```
drop table if exists ExpiredProducts;
```

```
create table if not exists ExpiredProducts (
    productID integer,
    expireDate date,
    warningDay date
);
```

```
delimiter $$
```

```
DROP EVENT IF EXISTS daily_control $$
```

```
CREATE EVENT IF NOT EXISTS daily_control
ON SCHEDULE EVERY 1 DAY
```

```
DO BEGIN
```

```
    DELETE FROM ExpiredProducts;
```

```
    INSERT INTO ExpiredProducts (productID, expireDate, warningDay)
    SELECT fpr.foodID, fpr.expiration_date, NOW()
    FROM food as fpr
    join product as pr on fpr.foodID = pr.productID
    where fpr.foodID in (select f.foodID
                           from food as f
                           where f.expiration_date < NOW())
    );
```

```
END $$
```

```
DELIMITER ;
```

#### 2.3.9.2 Explanation

En aquest event hem de fer un control sobre els productes de tipus menjar. És a dir, cada dia, hem de insertar en una taula tots els productes que tingui una data de caducitat superior a la data d'avui i que encara estiguin en venta. Per tant, en el cos del event simplement faig un insert a la taula dels productes de tipus menjar que tinguin una data de caducitat inferior a la data d'avui. També em vaig fixar que al només fer inserts, els productes que ja estaven en la taula es dupliquen, per això he afegit la línia en la que buido la taula abans de inserir les dades.

#### 2.3.9.3 Trigger validation

Per validar la query simplement he canviat perque el event s'executi cada minut enllot de cada dia per així poder verificar-ho més ràpidament. El resultat és el mateix.

Si selecciono tota la informació de la taula ExpiredProducts:

productID	expireDate	warningDa
13	2021-04-05	2021-05-28
15	2021-03-29	2021-05-28
18	2021-04-18	2021-05-28
21	2021-05-12	2021-05-28
22	2021-03-24	2021-05-28
27	2021-05-06	2021-05-28
45	2021-04-20	2021-05-28
48	2021-05-11	2021-05-28
49	2021-04-14	2021-05-28
54	2021-05-23	2021-05-28
60	2021-04-27	2021-05-28
71	2021-04-10	2021-05-28
81	2021-04-08	2021-05-28
82	2021-05-18	2021-05-28
86	2021-04-03	2021-05-28
88	2021-05-17	2021-05-28
112	2021-04-12	2021-05-28
114	2021-04-20	2021-05-28
140	2021-04-01	2021-05-28
151	2021-03-29	2021-05-28
154	2021-03-31	2021-05-28

Podem observar que totes les dates son inferiors a la del warningDay, per tant tots els productes estan caducats.

## 2.4 Luggage module

### 2.4.1 Requirement (Query) 4.1 Luggage handlers' rewards

#### 2.4.1.1 Solution

```
SELECT person.personId, person.name, person.surname, salary
FROM flightluggagehandler
INNER JOIN person ON flightluggagehandler.luggageHandlerID = personID
INNER JOIN employee ON personID = employeeID
GROUP BY flightluggagehandler.flightID
HAVING count(*) = 1
AND salary < (SELECT avg(salary)
               FROM employee, luggagehandler
               WHERE employeeID = luggageHandlerID);
```

personId	name	surname	salary
151396	Orbadiah	Darrach	31859
150907	Domenic	Cordero	17468
149341	Birgit	Piercy	49857
150187	Shea	Goode	10310
148799	Sam	Burwood	66243
149152	Bryana	Langsdon	33039
153341	Lindsay	Koppes	79885
148357	Stefa	Di Claudio	46408
149782	Sansone	Glazer	73678
148981	Bardlay	De Antoni	70702
153447	Carolyn	Baudone	78474
153283	Clay	Filchakov	50700
151560	Sayres	Bowell	50871
152340	Belicia	Teissier	20230
151748	Papagena	De Rye B...	64976
150751	Pierson	Lockwood	79550
148906	Delbert	Cuddehay	75842

#### 2.4.1.2 Explanation

Primer fem joins per juntar flightluggagehandler, employee i person per obtenir les dades dels treballadors. Després, fem un group by flightID i en posem en el having que count(\*) sigui igual a 1, de manera que detectarà si un vol té un sol treballador assignat. Per acabar, afegim una subquery en el having en la que indiquem que el salari dels treballadors sigui menor a la mitja dels salaris dels treballadors de la taula luggagehandler.

#### 2.4.1.3 Query validation

<code>luggageHandlerID</code>	<code>flightID</code>	<code>avg(salary)</code>
151396	5	80295.8341
150116	6	
148686	10	
151290	16	
150593	17	
148547	22	
150907	24	
148172	26	
149341	33	
150187	40	
149516	42	
148842	43	
148122	64	
148799	65	
150715	68	

Si fem la següent query:

```
SELECT * FROM flightluggagehandler
WHERE flightID IN (SELECT flightId
                    FROM flightluggagehandler
                    INNER JOIN person ON
                    flightluggagehandler.luggageHandlerID = personID
                    INNER JOIN employee ON personID =
                    employeeID
                    GROUP BY
                    flightluggagehandler.flightID
                    HAVING count(*) = 1);
```

Podem veure que en tots els `flightID` dels treballadors que apareixen en l'anterior query, només apareixen ells. A més a més també es pot veure com cap dels treballadors de la query supera la mitjana de salari.

#### 2.4.2 Requirement (Query) 4.2 Luggage details

##### 2.4.2.1 Solution

```

(
SELECT person.name AS passenger_name, email, country.name AS country, color, brand,
weight,
(size_x * size_y * size_z) AS volume, extra_cost, fragile
FROM person LEFT JOIN luggage ON luggage.passengerID = person.personID
INNER JOIN country ON person.countryID = country.countryID
LEFT JOIN handluggage ON luggageID = handluggageID
LEFT JOIN checkedluggage ON luggageID = checkedluggageID
LEFT JOIN specialobjects ON checkedluggageID = specialobjectID
WHERE SUBSTRING(person.name, 1, 4) LIKE SUBSTRING(country.name, 1, 4)
AND luggage.luggageID IS NULL
)
UNION
(
SELECT person.name AS passenger_name, email, country.name AS country, color, brand,
weight,
(size_x * size_y * size_z) AS volume, extra_cost, fragile
FROM person LEFT JOIN luggage ON luggage.passengerID = person.personID
INNER JOIN country ON person.countryID = country.countryID
LEFT JOIN handluggage ON luggageID = handluggageID
LEFT JOIN checkedluggage ON luggageID = checkedluggageID
LEFT JOIN specialobjects ON checkedluggageID = specialobjectID
WHERE SUBSTRING(person.name, 1, 4) LIKE SUBSTRING(country.name, 1, 4)
AND luggage.luggageID = (SELECT DP.luggageID FROM luggage AS DP WHERE
DP.passengerID = luggage.passengerID ORDER BY DP.weight ASC LIMIT 1)
GROUP BY passengerID
) ORDER BY passenger_name ASC;

```

passenger_name	email	country	color	brand	weight	volume	extra_cost	fragile
Austen	austen.petley@reference.com	Australia	NULL	NULL	NULL	NULL	NULL	NULL
Austin	austin.cuddehay@biglobe.ne.jp	Australia	NULL	NULL	NULL	NULL	NULL	NULL
Austin	austin.tumbetyl@myspace.com	Austria	Black	Samsonite	1.4	19200	NULL	NULL
Austin	austin.burberry@timesonline.co.uk	Australia	Red	Samsonite	7.6	16560	NULL	NULL
Austina	austina.sanzio@wikispaces.com	Austria	NULL	NULL	NULL	NULL	NULL	NULL
Britt	britt.purse@twitpic.com	British Virgin Islands	NULL	NULL	NULL	NULL	NULL	NULL
Camella	camella.arnaldy@uda.edu	Cameroon	NULL	NULL	NULL	NULL	NULL	NULL
Chadwick	chadwick.harlock@typepad.com	Chad	Red	Victorinox	22.7	NULL	99	0
Easter	easter.roistone@xing.com	East Timor	NULL	NULL	NULL	NULL	NULL	NULL
Francesca	francesca.farlowe@theatlantic.com	France	NULL	NULL	NULL	NULL	NULL	NULL
Francine	francine.nattrass@amazon.de	France	NULL	NULL	NULL	NULL	NULL	NULL
Franklyn	franklyn.castellucci@bloglovin.com	France	NULL	NULL	NULL	NULL	NULL	NULL
Francois	francois.northern@state.gov	France	NULL	NULL	NULL	NULL	NULL	NULL
Francoise	francoise.tapley@i2i.jp	France	NULL	NULL	NULL	NULL	NULL	NULL
Franklin	franklin.bocke@w3.org	France	Red	Eastpak	1.6	11832	NULL	NULL
Fransisco	fransisco.pickerin@sakura.ne.jp	France	Red	American T...	4.6	66960	NULL	NULL
Franz	franz.tithecote@si.edu	France	NULL	NULL	NULL	NULL	NULL	NULL

##### 2.4.2.2 Explanation

Primer fem joins de person, country, handluggage, checkedluggage i specialobject per poder tenir totes les dades que necessitem. En el where, usem SUBSTRING per obtenir les primeres 4 lletres del passatger i del país per comprovar si coincideixen. Farem un union entre dues queries iguals, excepte en el where. En la primera, es busca que luggageID sigui null per trobar passatgers que no

tinguin cap equipatge, i en la segona es fa una subquery per assegurar-se que el passatger té equipatge, i que l'equipatge que seleccionem sigui el que menys pesa.

#### 2.4.2.3 *Query validation*

En els resultats de la query es pot veure que les 4 primeres lletres del nom del passatger i del país coincideixen, i si executem la següent query:

```
SELECT person.name AS passenger_name, email, country.name AS country, color, brand,
weight,
(size_x * size_y * size_z) AS volume, extra_cost, fragile
FROM person LEFT JOIN luggage ON luggage.passengerID = person.personID
INNER JOIN country ON person.countryID = country.countryID
LEFT JOIN handluggage ON luggageID = handluggageID
LEFT JOIN checkedluggage ON luggageID = checkedluggageID
LEFT JOIN specialobjects ON checkedluggageID = specialobjectID
WHERE SUBSTRING(person.name, 1, 4) LIKE SUBSTRING(country.name, 1, 4)
AND luggage.luggageID IS NOT null;
```

Austin	austin.tumbelty@myspace.com	Austria	Black	Samsonite	1.4	19200	NULL	NULL
Austin	austin.tumbelty@myspace.com	Austria	White	American Tourister	3.7	51300	NULL	NULL
Franklin	franklin.bocke@w3.org	France	Red	Eastpak	11.2	NULL	0	NULL
Austin	austin.burberry@timesonline.co.uk	Australia	Red	Samsonite	7.6	16560	NULL	NULL
Franklin	franklin.bocke@w3.org	France	Red	Eastpak	1.6	11832	NULL	NULL
Austin	austin.tumbelty@myspace.com	Austria	Red	Victorinox	5	17472	NULL	NULL
Franklin	franklin.bocke@w3.org	France	Black	American Tourister	8.2	NULL	0	NULL
Fransisco	fransisco.pickerin@sakura.ne.jp	France	White	Samsonite	14.4	NULL	0	NULL

Podem veure com alguns passatgers tenen més d'un equipatge però en la query només es mostra el menys pesat.

### 2.4.3 Requirement (Query) 4.3 Lost objects

#### 2.4.3.1 Solution

```

SELECT lostobject.color, count(*) AS lost_objects, count(distinct passengerID) AS
num_passengers, count(*)/count(distinct passengerID) AS ratio
FROM luggage INNER JOIN lostobject ON luggage.luggageID = lostobject.luggageID
GROUP BY lostobject.color
UNION
SELECT luggage.brand, count(*) AS lost_objects, count(distinct passengerID) AS
num_passengers, count(*)/count(distinct passengerID) AS ratio
FROM luggage INNER JOIN lostobject ON luggage.luggageID = lostobject.luggageID
GROUP BY luggage.brand;

```

color	lost_objects	num_passengers	ratio
Black	106	106	1.0000
Blue	45	45	1.0000
Red	147	147	1.0000
White	52	52	1.0000
American Tourister	95	95	1.0000
Eastpak	69	69	1.0000
Samsonite	99	99	1.0000
Victorinox	87	87	1.0000

#### 2.4.3.2 Explanation

Primer es fa un join de luggage amb lostobject per obtenir les dades de l'equipatge en el qual s'han perdut objectes, i es fa un union per fer un select de color i un altre de marca. En la primera es fa un group by color i en la segona un group by brand. Per obtenir el nombre d'objectes perduts es fa un count(\*), i per obtenir el nombre de persones que els han perdut, es fa un count(distinct passengerID). Aquest dos es divideixen entre ells per calcular el ratio.

#### 2.4.3.3 Query validation

Podria semblar que hi ha un error en la query, ja que el ratio sempre es 1, pero si editem les taules per fer que el mateix passatger perdi molts objectes amb:

UPDATE luggage SET passengerID = 15297 WHERE passengerID < 5000;

Podem veure que el resultat de la query canvia adientment.

color	lost_objects	num_passengers	ratio
Black	106	101	1.0495
Blue	45	45	1.0000
Red	147	142	1.0352
White	52	52	1.0000
American Tourister	95	93	1.0215
Eastpak	69	66	1.0455
Samsonite	99	96	1.0313
Victorinox	87	85	1.0235

#### 2.4.4 Requirement (Query) 4.4 Special objects

##### 2.4.4.1 Solution

```
SELECT (fragile + corrosive + flammable) AS hazardous_level, AVG(extra_cost) AS extra_cost  
FROM specialobjects INNER JOIN checkedluggage ON checkedluggageID =  
specialobjectID  
GROUP BY hazardous_level  
ORDER BY hazardous_level;
```

hazardous_level	extra_cost
0	59.55725277491978
1	84.81310421606565
2	109.67300521998509
3	134.24722838137473

##### 2.4.4.2 Explanation

Primer es fa join de specialobjects i checkedluggage per obtenir les dades dels objectes especials que hi ha en els equipatges. En el select se suma el valor de fragile, corrosive i flammable, ja que només poden ser 0 o 1, i en sumar-los es pot obtenir un "nivell de perillositat" de l'1 al 3. Per calcular la mitjana de cost extra, es fa un AVG() de l'extra cost, ja que la query té un group by per hazardous\_level, que és la suma que hem mencionat abans.

##### 2.4.4.3 Query validation

Al sumar els valors de fragile, corrosive i flammable de la taula specialobjects, ens donarà un nivell de perill basat en la quantitat d'atributs especials que té aquest objecte. Si utilitzem aquest nivell en el group by, i calculem la AVG de l'extra cost, ens donara la mitjana de preu extra segons el nivell de perillositat. Podem fer una comprovació de que la query està funcionant bé amb:

```
SELECT fragile, corrosive, flammable, AVG(extra_cost) AS extra_cost  
FROM specialobjects INNER JOIN checkedluggage ON checkedluggageID = specialobjectID  
WHERE (fragile + corrosive + flammable) = 0;
```

fragile	corrosive	flammable	extra_cost
0	0	0	59.55725277491978

#### 2.4.5 Requirement (Query) 4.5 Claims

##### 2.4.5.1 Solution

```

SELECT claims.passengerID, name, surname, (SELECT count(distinct flightID) FROM
luggage WHERE passengerID = claims.passengerID) AS n_flights, (SELECT count(distinct
c.claimID) FROM claims AS c WHERE c.passengerID = claims.passengerID) AS n_claims
FROM person INNER JOIN claims ON claims.passengerID = person.personID
INNER JOIN refund ON claims.claimID = refundID
INNER JOIN luggage ON luggage.passengerID = claims.passengerID
WHERE (SELECT count(distinct c.claimID) FROM claims AS c WHERE c.passengerID =
claims.passengerID) > (SELECT count(distinct flightID) FROM luggage WHERE
passengerID = claims.passengerID)
GROUP BY claims.passengerID
HAVING avg(accepted) = 0
ORDER BY claims.passengerID ;

```

passengerID	name	surname	n_flights	n_claims
5553	Moise	Spehr	2	3
6040	Isak	Doveston	1	2
6368	Kenn	Dillaway	1	2
6997	Simonne	Schohier	1	2
7050	Lisa	Durbin	1	2
7185	Davey	Radsdale	1	2
7189	Ruben	Bellany	1	2
7589	Raffarty	Haresign	1	3
8220	Bret	Bencher	1	2
8549	Chery	Jira	1	2
8972	Roddy	Rameau	2	3
9351	Austina	Danilenko	1	2
10044	Luigi	Finlan	2	3
10581	Merv	Kleeman	1	2
10600	Domenico	Geist	1	2

##### 2.4.5.2 Explanation

Primer fem un join entre person, claims, refund, luggage per saber les personnes que han reclamat devolucions i les que han viatjat en avió. Per veure el nombre de vols del passatger, fem en el select una subquery en la que es fa un count(distinct) de les flightId relacionades amb el passatger en la taula luggage, i per veure el nombre de reclamacions es fa el mateix amb count(distinct) de les claimID relacionades amb el passatger en la taula claims. En el where posem que el nombre de reclamacions sigui major que el de vols, i per asegurarnos de que cap devolució li hagi sigut acceptada al passatger, posem que avg() de accepted sigui igual a 0.

##### 2.4.5.3 Query validation

Si comproven el nombre de vols fet i el nombre reclamacions fetes d'algun passatger de la query utilitzant:

```

SELECT passengerID, count(distinct flightID) FROM luggage
GROUP BY passengerID;

```

passengerID	count(distinct flightID)
5553	2

```

SELECT passengerID, count(distinct claimID) FROM claims

```

GROUP BY passengerID;

passengerID	count(distinct claimID)
5553	3

Veurem que els nombres de la query són correctes, i que un és més gran que l'altre. A més a més, si fem la query:

SELECT \* FROM claims LEFT JOIN refund ON claims.claimID = refundID;

claimID	passengerID	date	refundID	flightTicketID	argument	accepted	amount
2946	5553	2008-12-06	NULL	NULL	NULL	NULL	NULL
5782	5553	2008-12-05	5782	34026	Other	0	708
8848	5553	2008-11-27	NULL	NULL	NULL	NULL	NULL

Podem veure com té tres claims distints, i cap d'ells és una devolució acceptada.

#### **2.4.6 Requirement (Trigger) 4.6 Refunded tickets**

##### **2.4.6.1 Solution**

```
DROP TABLE IF EXISTS RefundsAlterations;  
DROP TABLE IF EXISTS refund2;
```

```
CREATE TABLE IF NOT EXISTS RefundsAlterations(  
    passengerID BIGINT UNSIGNED NOT NULL DEFAULT 0,  
    flightTicketID BIGINT UNSIGNED NOT NULL DEFAULT 0,  
    comment TEXT );
```

```
CREATE TABLE IF NOT EXISTS refund2(  
    refundID      bigint unsigned,  
    flightTicketID bigint unsigned,  
    argument      text,  
    accepted      tinyint(1),  
    amount bigint unsigned  
);
```

```
delimiter //  
DROP TRIGGER IF EXISTS refunded_tickets//  
CREATE TRIGGER refunded_tickets  
BEFORE INSERT ON refund2  
FOR EACH ROW  
BEGIN  
    IF (SELECT count(*) FROM RefundsAlterations WHERE flightTicketID =  
    new.flightTicketID GROUP BY flightTicketID) >= 3 THEN  
        INSERT INTO RefundsAlterations(passengerID, flightTicketID, comment)  
        SELECT passengerID, new.flightTicketID, "Excessive Attempts"  
        FROM refund2 RIGHT JOIN claims ON claims.claimID = refund2.refundID  
        WHERE new.refundID = claims.claimID  
        LIMIT 1;  
        ELSE IF new.flightTicketID IN (SELECT flightTicketID FROM refund2 WHERE  
        accepted = 1) THEN  
            INSERT INTO RefundsAlterations(passengerID, flightTicketID, comment)  
            SELECT passengerID, new.flightTicketID, "Refund of a ticket already processed correctly"  
            FROM refund2 RIGHT JOIN claims ON claims.claimID = refund2.refundID  
            WHERE new.refundID = claims.claimID  
            LIMIT 1;  
        END IF;  
    END IF;  
END //  
delimiter ;
```

##### **2.4.6.2 Explanation**

Aquest trigger detecta quan s'ha inserit una nova fila en la taula Refund. Quan es detecta que el reemborsament que s'està intentant inserir a la taula està repetit, s'insereixen les dades d'aquest reemborsament en la taula RefundsAlterations, amb un comentari que indica que aquest

reemborsament ja havia sigut processat correctament amb anterioritat. També detecta si ja havia sigut inserit en aquesta última taula, i en el cas que s'hagi inserit tres vegades, ho guarda amb un comentari que indica que hi ha hagut un nombre excessiu d'intents de processar aquest reemborsament.

#### 2.4.6.3 Trigger validation

Per no haver de tindre en compte totes les PK i FK a l'hora d'inserir noves files a la taula Refunds, he creat una taula anomenada Refund2 exactament igual a l'original però sense FKs ni PKs. Per comprovar que el trigger funcione correctament, he executat 5 vegades això:

```
INSERT INTO refund2 (refundID, flightTicketID, argument, accepted, amount)
VALUES (252, 196, 'Flight Delayed', 1, 264);
```

I aquest és el resultat en la taula RefundsAlterations, confirmant que tot funciona correctament:

passengerID	flightTicketID	comment
85745	197	Refund of a ticket already processed correctly
85745	197	Refund of a ticket already processed correctly
85745	197	Refund of a ticket already processed correctly
85745	197	Excessive Attempts

#### 2.4.7 Requirement (Trigger) 4.7 Lost object days

##### 2.4.7.1 Solution

```
DROP TABLE IF EXISTS LostObjectsDays;
```

```
CREATE TABLE IF NOT EXISTS LostObjectsDays(
    lostObjectID BIGINT UNSIGNED,
    days_to_find INT,
    avg_days_type INT
);
```

```
delimiter //
DROP TRIGGER IF EXISTS lost_object_days//
CREATE TRIGGER lost_object_days
AFTER UPDATE ON lostobject
FOR EACH ROW
BEGIN
    IF new.founded = 1 AND new.founded <> old.founded THEN
        INSERT INTO LostObjectsDays
        SELECT new.lostObjectID, TIMESTAMPDIFF(DAY, flight.date, claims.date), (SELECT
        AVG(days_to_find)
        FROM LostObjectsDays
        INNER JOIN lostobject ON
        LostObjectsDays.lostObjectID = lostObject.lostObjectID
        WHERE LostObjectsDays.lostObjectID =
        new.lostObjectID
        GROUP BY description)
        FROM lostobject INNER JOIN claims ON claimID = lostObjectID
        INNER JOIN luggage ON luggage.luggageID = lostobject.luggageID
```

```

    INNER JOIN flight ON flight.flightID = luggage.flightID
    WHERE lostobject.lostObjectID = new.lostObjectID;
END IF;
END //
delimiter ;

```

#### 2.4.7.2 *Explanation*

Aquest trigger s'executa cada vegada que hi ha una Update en la taula lostobject. Primer, es detecta si el canvi en la taula ha sigut que un objecte ha canviat de "no trobat" a "trobat", i que si aquest és el cas, inserta la ID de l'objecte, els dies que s'ha trigat a trobar-ho, i la mitjana dels dies que es triguen a trobar els objectes del mateix tipus (és a dir, els objectes amb la mateixa descripció) en la taula LostObjectDays. Per fer la mitjana dels dies, només es té en compte els objectes que ja estan en aquesta última taula.

#### 2.4.7.3 *Trigger validation*

Per comprovar que el trigger funciona bé, he fet un update per canviar un objecte de "no trobat" a "trobat" dues vegades, per veure si ho inserta bé a la taula i si fa bé la mitjana de temps.

```

UPDATE lostobject SET founded = 0 WHERE lostObjectID = '9267';
UPDATE lostobject SET founded = 1 WHERE lostObjectID = '9267';

```

I aquest és el resultat en la taula LostObjectDays, confirmant que tot funciona correctament:

lostObjectID	days_to_find	avg_days_type
9267	18	NULL
9267	18	18

#### **2.4.8 Requirement (Event) 4.8 Transported statistics**

##### **2.4.8.1 Solution**

```
CREATE TABLE IF NOT EXISTS DailyLuggageStatistics(  
    statisticsdate DATE,  
    weight int,
```

```
    dangerobjects int,  
    acceptedclaims int  
);
```

```
CREATE TABLE IF NOT EXISTS MonthlyLuggageStatistics(  
    year int,  
    month int,  
    weight int,
```

```
    dangerobjects int,  
    acceptedclaims int  
);
```

```
CREATE TABLE IF NOT EXISTS YearlyLuggageStatistics(  
    year int,
```

```
    weight int,  
    dangerobjects int,  
    acceptedclaims int  
);
```

```
delimiter //
```

```
DROP EVENT IF EXISTS DailyLuggageStatistics//
```

```
CREATE EVENT DailyLuggageStatistics
```

```
    ON SCHEDULE
```

```
        EVERY 1 DAY
```

```
    DO
```

```
        BEGIN
```

```
            INSERT INTO DailyLuggageStatistics
```

```
                SELECT CURDATE(), (SELECT SUM(weight)
```

```
                    FROM luggage INNER JOIN flight ON
```

```
flight.flightID = luggage.flightID
```

```
                    WHERE date between date_sub(CURDATE(), INTERVAL 1 DAY) AND
```

```
CURDATE()),
```

```
                (SELECT COUNT(*)
```

```
                    FROM luggage INNER JOIN flight ON
```

```
flight.flightID = luggage.flightID
```

```
                    INNER JOIN checkedluggage ON checkedluggage.checkedluggageid =
```

```
luggage.luggageID
```

```
                    INNER JOIN specialobjects ON specialobjects.specialobjectID =
```

```
checkedluggage.checkedluggageid
```

```
                    WHERE date between date_sub(CURDATE(), INTERVAL 1 DAY) AND
```

```
CURDATE()
```

```
                    AND (corrosive OR flammable) = 1),
```

```

        (SELECT count(*)
         FROM claims INNER JOIN refund ON claimID = refundID
                           WHERE date between
date_sub(CURDATE(), INTERVAL 1 DAY) AND CURDATE()
                           AND accepted = 1);
      END//


delimiter ;

delimiter //



DROP EVENT IF EXISTS MonthlyLuggageStatistics//



CREATE EVENT MonthlyLuggageStatistics
  ON SCHEDULE
    EVERY 1 MONTH
  DO
    BEGIN

      INSERT INTO MonthlyLuggageStatistics
      SELECT YEAR(CURDATE()), MONTH(CURDATE()), (SELECT SUM(weight)
                                                   FROM luggage INNER JOIN flight ON
flight.flightID = luggage.flightID
                                                   WHERE date between DATE_SUB(CURDATE(),INTERVAL
DAYOFMONTH(CURDATE())-1 DAY) AND date_add(DATE_SUB(CURDATE(),INTERVAL
DAYOFMONTH(CURDATE())-1 DAY), INTERVAL 1 MONTH)),
             (SELECT COUNT(*)
              FROM luggage INNER JOIN flight ON
flight.flightID = luggage.flightID
              INNER JOIN checkedluggage ON checkedluggage.checkedluggageid =
luggage.luggageID
              INNER JOIN specialobjects ON specialobjects.specialobjectID =
checkedluggage.checkedluggageid
              WHERE date between DATE_SUB(CURDATE(),INTERVAL
DAYOFMONTH(CURDATE())-1 DAY) AND date_add(DATE_SUB(CURDATE(),INTERVAL
DAYOFMONTH(CURDATE())-1 DAY), INTERVAL 1 MONTH)
              AND (corrosive OR flammable) = 1),
             (SELECT count(*)
              FROM claims INNER JOIN refund ON claimID = refundID
                           WHERE date between
DATE_SUB(CURDATE(),INTERVAL DAYOFMONTH(CURDATE())-1 DAY) AND
date_add(DATE_SUB(CURDATE(),INTERVAL DAYOFMONTH(CURDATE())-1 DAY),
INTERVAL 1 MONTH)
              AND accepted = 1);
    END//


delimiter ;

delimiter //



DROP EVENT IF EXISTS YearlyLuggageStatistics//



CREATE EVENT YearlyLuggageStatistics

```

```

ON SCHEDULE
    EVERY 1 YEAR
DO
    BEGIN

        INSERT INTO YearlyLuggageStatistics
        SELECT YEAR(CURDATE()), (SELECT SUM(weight)
                                    FROM luggage INNER JOIN flight ON
flight.flightID = luggage.flightID
                                    WHERE YEAR(date) = YEAR(CURDATE())),
        (SELECT COUNT(*)
                                    FROM luggage INNER JOIN flight ON
flight.flightID = luggage.flightID
                                    INNER JOIN checkedluggage ON checkedluggage.checkedluggageid =
luggage.luggageID
                                    INNER JOIN specialobjects ON specialobjects.specialobjectID =
checkedluggage.checkedluggageid
                                    WHERE YEAR(date) = YEAR(CURDATE())
                                    AND (corrosive OR flammable) = 1),
        (SELECT count(*)
                                    FROM claims INNER JOIN refund ON claimID = refundID
                                    WHERE YEAR(date) =
YEAR(CURDATE())
                                    AND accepted = 1);
    END//
```

delimiter ;

#### 2.4.8.2 *Explanation*

Han fet falta tres Events diferents per poder guardar tota la informació. Els tres funcionen exactament igual, només que el primer guarda la informació diària, el segon guarda la informació mensual i el tercer guarda la informació anual. Cada un dels Events guarda la informació en una taula distinta. Es guarda la data de la informació que s'està guardant, i després la suma del pes de l'equipatge que s'ha portat en avions durant cert temps, el número d'equipatge que està catalogat com corrosiu o inflamable que s'ha portat en avions durant aquest temps, i per últim quantes reclamacions han sigut acceptades durant aquest temps.

#### 2.4.8.3 *Trigger validation*

Com no hi ha informació a les taules per comprovar si el Event funciona, he canviat la data d'uns quants vols i reclamacions perquè passin en una data que sí que serà detectada per aquests Events. Abans de fer-ho, he hagut de desactivar el SAFE MODE de SQL perquè ens hem permís editar la informació de les taules independentment de les FKs i PKs.

```

SET SQL_SAFE_UPDATES = 0;
UPDATE flight
SET date = CURDATE()
WHERE date < '1955-06-29';

UPDATE flight
SET date = DATE_SUB(CURDATE(), INTERVAL DAYOFMONTH(CURDATE())-1 DAY)
WHERE date < '1960-04-12';
```

```
UPDATE claims  
SET date = CURDATE()  
WHERE date < '1955-06-29';
```

```
UPDATE claims  
SET date = DATE_SUB(CURDATE(), INTERVAL DAYOFMONTH(CURDATE())-1 DAY)  
WHERE date < '1960-04-12';  
SET SQL_SAFE_UPDATES = 1;
```

I aquests són els resultats de les tres taules, demostrant que el Event funciona correctament:

statisticsdate	weight	dangerobjects	acceptedclaims	
2021-05-26	6768	35	25	
year	month	weight	dangerobjects	acceptedclaims
2021	5	13341	78	48
year	weight	dangerobjects	acceptedclaims	
2021	16733	93	62	

## 2.5 Cross module queries

### 2.5.1 Requirement (Query) 5.1 Overbooked airlines

#### 2.5.1.1 Solution

```
SELECT airline.airlineID, airline.name, (SELECT count(distinct flight.flightID)
                                         FROM flight
INNER JOIN plane ON flight.planeID = plane.planeID
                                         INNER JOIN
airline AS al ON plane.airlineID = al.airlineID
                                         INNER JOIN
flighttickets ON flight.flightID = flighttickets.flightID
                                         LEFT JOIN
checkin ON checkin.flightticketID = flighttickets.flightticketID
                                         WHERE
checkinID IS NULL AND al.airlineID = airline.airlineID) AS overbooked_flights
FROM flight
INNER JOIN plane ON flight.planeID = plane.planeID
INNER JOIN airline ON plane.airlineID = airline.airlineID
INNER JOIN flighttickets ON flight.flightID = flighttickets.flightID
LEFT JOIN checkin ON checkin.flightticketID = flighttickets.flightticketID
WHERE checkinID IS NULL
GROUP BY airline.airlineID
HAVING count(flighttickets.passengerID) >
       (SELECT count(flighttickets.passengerID)
        FROM flight INNER JOIN plane ON
flight.planeID = plane.planeID
                                         INNER JOIN airline AS al ON
plane.airlineID = al.airlineID
                                         INNER JOIN flighttickets ON
flight.flightID = flighttickets.flightID
                                         LEFT JOIN checkin ON
checkin.flightticketID = flighttickets.flightticketID
                                         WHERE al.airlineID =
airline.airlineID) * 0.1;
```

airlineID	name	overbooked_flights
3378	Malaysia Airlines	1
3830	Ozark Air Lines	1
1316	AirTran Airways	1
3871	Pakistan International Airlines	1

#### 2.5.1.2 Explanation

Per a aquesta query ens demanaven els vols que havien tingut vols amb overbooking. Considerem overbooking el fet que una persona tingui un bitllet per el vol pero no hagi fet el check in. Això ha

de passar amb al menys el 10% dels seus passagers. Per a aquesta query hem tingut algunes dificultats ja que al principi la tenim com aqui però vam aconseguir treure-li la subquery del select ja que, com se'n ha dit a classe, és millor evitar el excés de subqueries ja que fa que vagi més lent i és menys efficient. El problema és que al executar trigar més de 80 segons depenen del ordinador comparat amb la que té la subquery que triga entre 3 i 7 segons. Per tant hem decidit mantenir la query amb la subquery del select.

#### 2.5.1.3 Query validation

He utilitzat dos queries per fer la verificació. En la primera query verifiquem el número de vols amb overbooking de cada aerolínea sense importarnos si el número de passatgers es superior al 10% del total dels passatgers:

```
SELECT al.airlineID, count(distinct flight.flightID)
FROM flight INNER JOIN plane ON flight.planeID = plane.planeID
INNER JOIN airline AS al ON plane.airlineID = al.airlineID
INNER JOIN flighttickets ON flight.flightID = flighttickets.flightID
LEFT JOIN checkin ON checkin.flighthTicketID = flighttickets.flighthTicketID
WHERE checkinID IS NULL
group by al.airlineID;
```

airlineID	count(distinct flight.flightID)
1758	2
2297	2
3320	2
24	1
214	1
330	1
576	1
794	1

Podem veure que existeixen aerolínies amb 2 vols de overbooking i companies amb 1 vol amb overbooking.

L'altre verificació hem serveix per mirar si els passatgers amb overbooking es superior al 10%

```
SELECT airline.airlineID, airline.name, (SELECT count(distinct flight.flightID)
                                         FROM flight INNER
                                         JOIN plane ON flight.planeID = plane.planeID
                                         INNER JOIN airline AS
                                         al ON plane.airlineID = al.airlineID
                                         INNER JOIN
                                         flighttickets ON flight.flightID = flighttickets.flightID
                                         LEFT JOIN checkin ON
                                         checkin.flighthTicketID = flighttickets.flighthTicketID
                                         WHERE checkinID IS
                                         NULL AND al.airlineID = airline.airlineID) AS overbooked_flights,
                                         (SELECT count(flighttickets.passengerID)
```

```

flight.planeID = plane.planeID
= al.airlineID
flighttickets.flightID
= flighttickets.flightticketID
total_flight_passenger,
count(flighttickets.passengerID) as overbooked_passengers

FROM flight
INNER JOIN plane ON flight.planeID = plane.planeID
INNER JOIN airline ON plane.airlineID = airline.airlineID
INNER JOIN flighttickets ON flight.flightID = flighttickets.flightID
LEFT JOIN checkin ON checkin.flightticketID = flighttickets.flightticketID
WHERE checkinID IS NULL
GROUP BY airline.airlineID
HAVING count(flighttickets.passengerID) >
       (SELECT count(flighttickets.passengerID)
        FROM flight INNER JOIN plane ON
        flight.planeID = plane.planeID
        = al.airlineID
        flighttickets.flightID
        = flighttickets.flightticketID
        WHERE al.airlineID = airline.airlineID) * 0.1;

```

airlineID	name	overbooked_flights	total_flight_passenger	overbooked_passengers
3378	Malaysia Airlines	1	802	216
3830	Ozark Air Lines	1	548	177
1316	AirTran Airways	1	141	45
3871	Pakistan International Airlines	1	325	75

Podem veure que al fer el calcul, el numero de passagers es superior al 10%.

### 2.5.2 Requirement (Query) 5.2 Lost objects due to jet lag

#### 2.5.2.1 Solution

```

SELECT ABS(departure_city.timezone - destination_city.timezone) AS timezone_difference,
       count(distinct lostObjectID) AS lost_objects
  FROM route INNER JOIN airport AS departure ON departure.airportID =
route.departure_airportID
           INNER JOIN airport AS destination ON destination.airportID = route.destination_airportID
           INNER JOIN city AS departure_city ON departure.cityID = departure_city.cityID
           INNER JOIN city AS destination_city ON destination.cityID = destination_city.cityID
           INNER JOIN flight ON flight.routeId = route.routeID
           INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
           INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
           INNER JOIN claims ON passenger.passengerID = claims.passengerID
           INNER JOIN lostobject ON lostobject.lostobjectID = claims.claimID
           INNER JOIN luggage ON luggage.luggageID = lostobject.luggageID AND luggage.flightID =
flight.flightID
 WHERE claims.date < date_add(flight.date, INTERVAL 3 MONTH)
 GROUP BY timezone_difference
 ORDER BY timezone_difference DESC;

```

timezone_difference	lost_objects
19	1
18	1
17	3
16	5
15	6
14	3
13	1
11	6
10	3
9	7
8	5
7	8
6	11
5	5
4	16
3	16
2	28
1	99
0	126

#### 2.5.2.2 Explanation

En el SELECT de la query es calcula el valor absolut de la resta entre la zona horària de la ciutat de sortida de l'avió i la ciutat d'arribada, que també s'utilitzarà com a GROUP BY i com a ORDER BY en ordre descendent. Per a calcular el nombre d'objectes perduts, es fa un DISTINCT COUNT de les IDs dels objectes. Per obtenir totes les dades, es fa un JOIN de dos airports i dues ciutats, una per la sortida de l'avió i l'altra per l'arribada, de la tabla flight, flightTickets, passenger, claims, lostobject i luggage. En el WHERE, s'utilitza la funció date\_add per calcular un interval de temps de tres mesos després de la data del vol, i es comprova que aquest temps sigui posterior a la data en la qual es va reclamar l'objecte.

### 2.5.2.3 Query validation

Si utilitzem aquesta query per veure tots els objectes que s'han perdut quan la diferència horària és de 0 hores:

```

SELECT distinct ABS(departure_city.timezone - destination_city.timezone) AS
timezone_difference, lostObjectID AS lost_objects
FROM route INNER JOIN airport AS departure ON departure.airportID =
route.departure_airportID
INNER JOIN airport AS destination ON destination.airportID = route.destination_airportID
INNER JOIN city AS departure_city ON departure.cityID = departure_city.cityID
INNER JOIN city AS destination_city ON destination.cityID = destination_city.cityID
INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
INNER JOIN claims ON passenger.passengerID = claims.passengerID
INNER JOIN lostobject ON lostobject.lostobjectID = claims.claimID
INNER JOIN luggage ON luggage.luggageID = lostobject.luggageID AND luggage.flightID =
flight.flightID
WHERE claims.date < date_add(flight.date, INTERVAL 3 MONTH) AND
ABS(departure_city.timezone - destination_city.timezone) = 0;

```

timezone_difference	lost_objects	claim_date	flight_date
0	9397	2006-11-19	2006-09-24
0	1171	2008-07-15	2008-07-01
0	2609	2015-11-08	2015-09-09
0	1487	2011-01-10	2010-11-05
0	2047	1984-05-27	1984-04-13
0	4781	2010-03-10	2010-02-28
0	6437	2012-11-04	2012-09-18
0	5798	2019-04-23	2019-04-08
0	781	2013-06-02	2013-05-29
0	2746	1966-08-08	1966-07-16
0	420	1990-08-03	1990-07-09
0	7002	1991-11-09	1991-11-07
0	9028	1966-06-15	1966-06-02
0	9821	1998-10-16	1998-10-10
0	8035	1997-01-08	1996-11-10
0	4974	2017-01-31	2017-01-21
-----			
126 row(s) returned			

Podem veure que ens retorna el mateix nombre d'objectes que ens diu la nostra query, i que la diferència de temps entre la reclamació de l'objecte i el vol en el qual es va perdre mai supera els tres mesos.

### 2.5.3 Requirement (Query) 5.3 Flight attendant vacancy

#### 2.5.3.1 Solution

```

SELECT person.name, person.surname, person.phone_number, (SELECT count(*) FROM
languageperson AS lp WHERE lp.personID = languageperson.personID GROUP BY lp.personID)
AS languages_spoken
FROM route INNER JOIN airport AS departure ON departure.airportID =
route.departure_airportID
INNER JOIN airport AS destination ON destination.airportID = route.destination_airportID
INNER JOIN city AS departure_city ON departure.cityID = departure_city.cityID
INNER JOIN city AS destination_city ON destination.cityID = destination_city.cityID
INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
INNER JOIN plane ON flight(planeID = plane.planeID
INNER JOIN airline ON airline.airlineID = plane.airlineID
INNER JOIN person ON person.personID = passenger.passengerID
INNER JOIN languageperson ON languageperson.personID = person.personID
INNER JOIN language ON language.languageID = languageperson.languageID
WHERE ABS(departure_city.timezone - destination_city.timezone) >= 3
AND language.name LIKE 'Chavacano'
GROUP BY languageperson.personID
HAVING languages_spoken > 1;

```

name	surname	phone_number	languages_spoken
Garrard	Gohn	+86 769 463 3329	2
Tripp	Robke	+93 699 476 8778	2
Heinrick	Courteney	+86 453 889 1385	3
Jerrine	Fishleigh	+86 541 620 6313	2
Jeffry	Alloway	+86 442 483 6691	2
Margarette	Bernaert	+51 162 507 2212	3
Carly	Keyhoe	+7 722 485 3718	2
Emelen	Tuxill	+54 425 822 9376	2
Rhetta	Santoro	+63 482 848 0819	3
Andras	Silverstone	+66 259 417 4314	2
Renae	Firle	+48 261 629 1132	2
Akim	Eland	+51 320 319 9977	3
Morten	Phoenix	+387 785 989 4813	3

#### 2.5.3.2 Explanation

En aquesta query primer fem join de totes les taules que necessitem per obtenir la informació necessària. En el SELECT, per obtenir el nombre d'idiomes que parla cada usuari utilitzem una subquery. En el WHERE, calculem el valor absolut de la diferència entre la zona horària de la ciutat de sortida de l'avió i la ciutat d'arribada i mirem que sigui major que 3, i també ens assegurem de què les persones parlen l'idioma 'Chavacano'. Per últim, fem un GROUP BY per ID de cada persona, i en el HAVING ens assegurem que les persones parlen més d'un idioma.

#### 2.5.3.3 Query validation

Amb la següent query podem veure que els valors que retorna la query son correctes:

```

SELECT person.name, person.surname, person.phone_number, language.name,
ABS(departure_city.timezone - destination_city.timezone), (SELECT count(*) FROM
languageperson AS lp WHERE lp.personID = languageperson.personID GROUP BY lp.personID)
AS languages_spoken
FROM route INNER JOIN airport AS departure ON departure.airportID =
route.departure_airportID
INNER JOIN airport AS destination ON destination.airportID = route.destination_airportID
INNER JOIN city AS departure_city ON departure.cityID = departure_city.cityID
INNER JOIN city AS destination_city ON destination.cityID = destination_city.cityID
INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
INNER JOIN plane ON flight.planeID = plane.planeID
INNER JOIN airline ON airline.airlineID = plane.airlineID
INNER JOIN person ON person.personID = passenger.passengerID
INNER JOIN languageperson ON languageperson.personID = person.personID
INNER JOIN language ON language.languageID = languageperson.languageID
WHERE ABS(departure_city.timezone - destination_city.timezone) >= 3
AND language.name LIKE 'Chavacano'
GROUP BY languageperson.personID
HAVING languages_spoken > 1;

```

name	surname	phone_number	name	ABS(departure_city.timezone - destination_city.timezone)	languages_spoken
Garrard	Gohn	+86 769 463 3329	Chavacano	16	2
Tripp	Robke	+93 699 476 8778	Chavacano	19	2
Heinrick	Courteney	+86 453 889 1385	Chavacano	5	3
Jerrine	Fishleigh	+86 541 620 6313	Chavacano	16	2
Jeffry	Alloway	+86 442 483 6691	Chavacano	7	2
Margarette	Bernaert	+51 162 507 2212	Chavacano	11	3
Carly	Keyhoe	+7 722 485 3718	Chavacano	6	2
Emelen	Tuxill	+54 425 822 9376	Chavacano	3	2
Rhetta	Santoro	+63 482 848 0819	Chavacano	4	3

Podem veure que tots els usuaris parlen l'idioma 'Chavacano' i que la diferència de temps sempre és 3 o major.

### 2.5.4 Requirement (Query) 5.4 Smuggling airports

#### 2.5.4.1 Solution

```

SELECT departure.name, (SELECT count(*)
FROM route INNER JOIN airport AS dp ON dp.airportID = route.departure_airportID
INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
WHERE dp.airportID = departure.airportID
GROUP BY dp.airportID) AS total_number_passengers
FROM route INNER JOIN airport AS departure ON departure.airportID =
route.departure_airportID
INNER JOIN airport AS destination ON destination.airportID = route.destination_airportID
INNER JOIN city AS destination_city ON destination.cityID = destination_city.cityID
INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
INNER JOIN country ON country.countryID = destination_city.countryID
INNER JOIN forbiddenproducts ON forbiddenproducts.countryID = destination_city.countryID
INNER JOIN luggage ON luggage.flightID = flight.flightID
INNER JOIN handluggage ON handluggage.productID = forbiddenproducts.productID AND
handluggageID = luggage.luggageID
GROUP BY departure.airportID;

```

	<b>name</b>	<b>total_number_passengers</b>
▶	Amsterdam Airport Schiphol	428
	Charles de Gaulle International Airport	467
	Miami International Airport	361
	Fresno Yosemite International Airport	235

#### 2.5.4.2 Explanation

Per a aquesta query l'únic aspecte que hem de tenir en compte és el comptatge de passatgers que que es troben en l'aeroport en el moment que s'infiltra els productes prohibits. En aquest cas s'hauria de fer una subconsulta d'un COUNT en el SELECT de la consulta principal tenint només en compte taules com els viatges, tickets d'aquests i els propis passatgers per a obtenir la informació adient del nostre interès.

#### 2.5.4.3 Query validation

```

SELECT departure.name, COUNT(forbiddenproducts.productID), (SELECT count(*)
FROM route INNER JOIN airport AS dp ON dp.airportID = route.departure_airportID
INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
WHERE dp.airportID = departure.airportID
GROUP BY dp.airportID) AS total_number_passengers
FROM route INNER JOIN airport AS departure ON departure.airportID =
route.departure_airportID
INNER JOIN airport AS destination ON destination.airportID = route.destination_airportID
INNER JOIN city AS destination_city ON destination.cityID = destination_city.cityID

```

```

INNER JOIN flight ON flight.routeId = route.routeID
INNER JOIN flightTickets ON flighttickets.flightID = flight.flightID
INNER JOIN passenger ON flighttickets.passengerID = passenger.passengerID
INNER JOIN country ON country.countryID = destination_city.countryID
INNER JOIN forbiddenproducts ON forbiddenproducts.countryID = destination_city.countryID
INNER JOIN luggage ON luggage.flightID = flight.flightID
INNER JOIN handluggage ON handluggage.productID = forbiddenproducts.productID AND
handluggageID = luggage.luggageID
GROUP BY departure.airportID;

```

	name	COUNT(forbiddenproducts.productID)	total_number_passengers
▶	Amsterdam Airport Schiphol	34	428
	Charles de Gaulle International Airport	9	467
	Miami International Airport	11	361
	Fresno Yosemite International Airport	1	235

L'única adició feta per a la query de validació per comprovar si està bé, és la existència de productes prohibits a través de l'adició d'un paràmetre al SELECT anomenat COUNT(forbiddenproducts.productID) el qual ens retorna el nombre de products que s'han infiltrar en l'aeroport.

### 2.5.5 Requirement (Query) 5.5 Reliable plane types

#### 2.5.5.1 Solution

```

SELECT planetype.type_name
FROM planetype INNER JOIN plane ON plane.planetypeID = planetype.planetypeID
INNER JOIN flight ON flight.planeID = plane.planeID
INNER JOIN status ON status.statusID = flight.statusId
INNER JOIN route ON flight.routeID = route.routeID
INNER JOIN airline ON airline.airlineID = plane.airlineID
GROUP BY planetype.planetypeID
HAVING SUM(route.distance) > 1000000 AND count(distinct flight.flightID) > 500
AND count(distinct airline.airlineID) > 70 AND SUM(status LIKE 'Perfect') >= count(distinct
flight.flightID)*0.53
ORDER BY planetype.planetypeID;

```

	type_name
▶	Airbus A320

#### 2.5.5.2 Explanation

Per a aquesta query, els aspectes a remarcar i tenir en compte, apart de fer les relacions adients amb els JOINs, és que tant la suma de les distàncies de les rutes siguin major a 1000000 com que el nombre de viatges realitzats siguin major a 500, que hagin més de 70 diferents aerolínies i que com a mínim el status de 'Perfect' estigui present almenys als 53% dels viatges. To això es realitza amb COUNTs, DISTINCTs a dins dels COUNTs i amb SUMs i càlculs de percentatges.

#### 2.5.5.3 Query validation

```

SELECT planetype.type_name , count(distinct flight.flightID) AS flights, SUM(route.distance),
count(distinct airline.airlineID), SUM(status LIKE 'Perfect')
FROM planetype INNER JOIN plane ON plane.planetypeID = planetype.planetypeID
INNER JOIN flight ON flight.planeID = plane.planeID
INNER JOIN status ON status.statusID = flight.statusId
INNER JOIN route ON flight.routeID = route.routeID
INNER JOIN airline ON airline.airlineID = plane.airlineID
GROUP BY planetype.planetypeID
HAVING SUM(route.distance) > 1000000 AND count(distinct flight.flightID) > 500
AND count(distinct airline.airlineID) > 70 AND SUM(status LIKE 'Perfect') >= count(distinct
flight.flightID)*0.53
ORDER BY planetype.planetypeID;

```

	type_name	flights	SUM(route.distance)	count(distinct airline.airlineID)	SUM(status LIKE 'Perfect')
▶	Airbus A320	3428	4284273	137	1824

A l'hora de validar aquesta query, s'ha prioritzar mostrar al SELECT aspectes com el nombre total de viatges realitzats, la suma de les distàncies fetes entre les rutes, el nombre de viatges amb un status de 'Perfect' i el nombre de diferents aerolínies que existeixen. D'aquesta forma validem que la informació de la solució de la query té validesa i les dades són coherents.

### 2.5.6 Requirement (Query) 5.6 Employees languages

#### 2.5.6.1 Solution

```
(  
SELECT 'flight_attendant' AS employee_type, language.name, count(distinct  
languageperson.personID) AS people_who_speak  
FROM employee INNER JOIN flight_attendant ON flight_attendant.flightattendantID =  
employee.employeeId  
INNER JOIN languageperson ON languageperson.personID = employee.employeeID  
INNER JOIN language ON language.languageID = languageperson.languageID  
GROUP BY languageperson.languageID  
)  
UNION  
(  
SELECT 'other_employee', language.name, count(distinct languageperson.personID)  
FROM employee INNER JOIN languageperson ON languageperson.personID =  
employee.employeeID  
INNER JOIN language ON language.languageID = languageperson.languageID  
WHERE employee.employeeId NOT IN (SELECT flightattendantID FROM flight_attendant)  
GROUP BY languageperson.languageID  
)  
ORDER BY people_who_speak DESC;
```

employee_type	name	people_who_speak
other_employee	English	5061
flight_attendant	English	3745
flight_attendant	Spanish	2093
other_employee	German	1918
other_employee	Spanish	1876
other_employee	Mandarin	1538
flight_attendant	German	1492
flight_attendant	French	1311
other_employee	French	1247

#### 2.5.6.2 Explanation

En aquesta query havíem de mostrar el número de persones que parlen cada idioma separades en dos grups: els empleats en general i les azafates. Per tant, la query està composta de dues queries juntades amb un union. En la primera conté per les azafates i en la segona per la resta dels empleats sense comptar les azafates.

#### 2.5.6.3 Query validation

Per validar la query simplement he comptat quantes persones parlen cada idioma en general i ho comparo amb el resultat de la solució.

```
SELECT 'employee', language.name, count(distinct languageperson.personID)  
FROM employee INNER JOIN languageperson ON languageperson.personID =  
employee.employeeID
```

```

INNER JOIN language ON language.languageID = languageperson.languageID
GROUP BY languageperson.languageID
ORDER BY count(distinct languageperson.personID) DESC

```

employee	name	count(distinct languageperson.personID)
employee	English	8806
employee	Spanish	3969
employee	German	3410
employee	Mandarin	2648
employee	French	2558
employee	Russian	1810
employee	Japanese	1767
employee	Korean	1460

Podem observar que si fem la suma per cada idioma tant azafates com altres empleats , es igual al resultat d'aquesta query.

### 3 LSAIR Graph database requirements

#### 3.1 Case study 1: Airplane, airport, city, and country

Aquest case study ens servirà per veure les relacions entre els aeroports, els països, els avions basat amb les seves rutes.

##### 3.1.1 Data migration

###### 3.1.2

Gràcies a la realització de les diferents queries que se'ns demanaven en el dataset1, el dataset 2 i les seves relacions, vam poder obtenir la informació que calia integrar-se en el Neo4j.

Per obtenir aquesta informació vam exportar les diferents dades en un csv. La descàrrega en CSV va ser senzilla ja que MySQL té l'opció d'exporta-ho directament.

Un cop teníem els CSV, vam intentar llegir-los en el Neo4j, però, tot i fer una gran quantitat de intents, no ho aconseguíem. Finalment, un dels nostres companys el va aconseguir llegir al penjar-lo en el Google Drive i compartint-lo a través de la web. Al veure que ja podíem fer la lectura de les dades, vam procedir en fer la creació dels nodes i les seves relacions, però, ens vam adonar que quan feiem la inserció, se'ns creaven nodes repetits ja que en el CSV mostràvem les relacions entre cada element. Aquest fet ens va portar a modificar el SELECT de les consultes per a que ens retornessin la informació que volíem per els nodes sense repeticions.

Mitjançant els create vam anar ficant la informació a poc a poc per comprovar cada cop si es feia bé, però posteriorment vam afegir els ; per poder ficar tota la informació de cop. Ens vam trobar amb varies complicacions però un cop vam entendre el procés, va ser més senzill.

Per fer la creació dels nodes vam seguir la següent estructura:

```
LOAD amb el fitcher csv corresponent pujat al drive,  
AS CSV WITH CSV
```

##### En el cas de creació de nodes

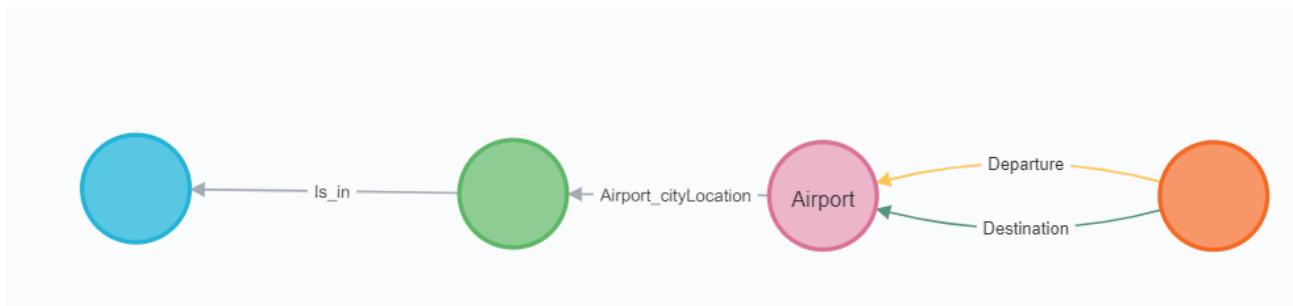
```
CREATE (del node en concret amb els seus respectius atributs)
```

##### En el cas de creació de relacions

```
MATCH (indiquem el tipus de node i li diem que busqui en una columna del csv),  
(Indiquem l'altre node i l'altra columna del CSV)
```

Finalment fem un create de les relacions de les variables dels nodes que hem fet el match just abans

### 3.1.3 Neo4J Data model



Podem observar que el nostre model de dades està compost de 4 tipus de node: Country, City, Airport i Airplane. Un avió té una destinació en un aeroport i un Departure en un altre aeroport. Aquests aeroports, cadascú té una ciutat i aquesta ciutat un País.

### 3.1.4 Query 1

#### 3.1.4.1 Solution

```
MATCH (p:Plane)
WHERE NOT (p)-->(:Airport) AND p.Different_pieces_changed < 8
RETURN p
```

```
| "p"
| 
| {"Airline_name": "Scandinavian Airlines System", "Different_pieces_chang| ed": 0, "Plane_type_name": "Airbus A340-300", "planeID": 1113, "retirement_y| ear": 2021, "Times_maintained": 0}
| 
| {"Airline_name": "Scandinavian Airlines System", "Different_pieces_chang| ed": 0, "Plane_type_name": "Boeing 737-600", "planeID": 3537, "retirement_y| ear": 2019, "Times_maintained": 0}
| 
| {"Airline_name": "Iberia Airlines", "Different_pieces_changed": 0, "Plane_| type_name": "Boeing 767", "planeID": 3782, "retirement_year": 2020, "Times_| maintained": 0}
| 
| {"Airline_name": "Vueling Airlines", "Different_pieces_changed": 7, "Plane_| _type_name": "Airbus A319", "planeID": 4949, "retirement_year": 2020, "Times_| _maintained": 7}
```

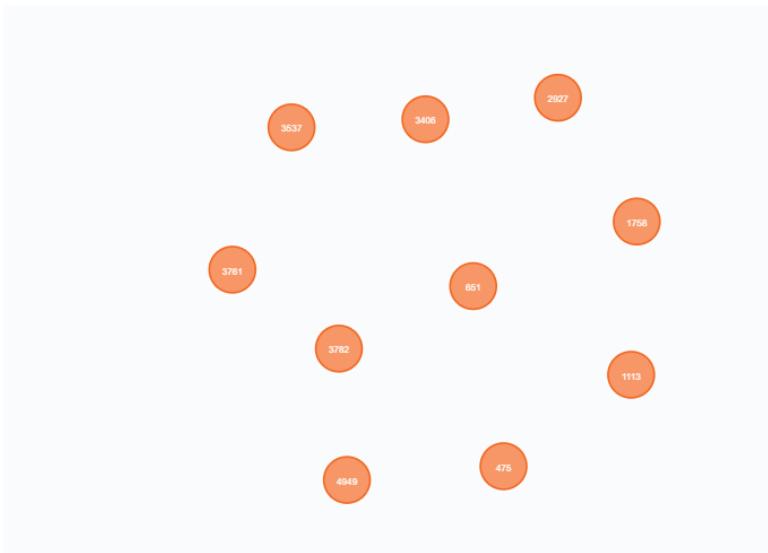
#### 3.1.4.2 Explanation

Aquesta query tracta de mostrar els avions que no han volat a cap aeroport i que tenen menys de 8 peces canviades. Per tant, primer comprovem si el avió no té cap relació amb algun aeroport i després verifiquem si té menys de 8 peces canviades.

#### 3.1.4.3 Query validation

Per validar aquesta query he realitzat dues queries, una en la que verifiquem si el número de peces canviades es inferior a 8:

```
MATCH (p:Plane)
WHERE p.Different_pieces_changed < 8
RETURN p
```



La segona verificació es si l'avió ha estat mai en un aeroport:

```
MATCH (p:Plane)
WHERE NOT (p)-->(:Airport)
RETURN p
```

```
| "p"
|
| {"Airline_name": "Scandinavian Airlines System", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A340-300", "planeID": 1113, "retirement_year": 2021, "Times_maintained": 0}
|
| {"Airline_name": "Scandinavian Airlines System", "Different_pieces_changed": 0, "Plane_type_name": "Boeing 737-600", "planeID": 3537, "retirement_year": 2019, "Times_maintained": 0}
|
| {"Airline_name": "Iberia Airlines", "Different_pieces_changed": 0, "Plane_type_name": "Boeing 767", "planeID": 3782, "retirement_year": 2020, "Times_maintained": 0}
|
| {"Airline_name": "Vueling Airlines", "Different_pieces_changed": 7, "Plane_type_name": "Airbus A319", "planeID": 4949, "retirement_year": 2020, "Times_maintained": 7}
```

Podem observar que si ajuntem aquestes dues condicions obtindrem el mateix resultat que en la solució.

### 3.1.5 Query 2

#### 3.1.5.1 Solution

```

MATCH ((c2:City)<--(ap2:Airport)-[:Destination]-(p:Plane)-[:Departure]->(ap:Airport)-->(c:City))
WHERE ap<>ap2 AND c<>c2
RETURN DISTINCT p, COUNT(DISTINCT ap) AS numAirports
ORDER BY numAirports DESC

```

"p"	"numAirports"
{"Airline_name": "Iberia Airlines", "Different_pieces_changed": 5, "Plane_type_name": "Canadair Regional Jet 900", "planeID": 651, "retirement_year": 2020, "Times_maintained": 5}	206
{"Airline_name": "Scandinavian Airlines System", "Different_pieces_changed": 11, "Plane_type_name": "Canadair Regional Jet 900", "planeID": 832, "retirement_year": 2020, "Times_maintained": 12}	120
{"Airline_name": "Swiss International Air Lines", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A330-300", "planeID": 3761, "retirement_year": 2020, "Times_maintained": 0}	104
{"Airline_name": "Jat Airways", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A319", "planeID": 475, "retirement_year": 2021, "Times_maintained": 0}	45
{"Airline_name": "SilkAir", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A319", "planeID": 1758, "retirement_year": 2021, "Times_maintained": 0}	45
{"Airline_name": "Flybaboo", "Different_pieces_changed": 12, "Plane_type_name": "Aerospatiale/Alenia ATR 72", "planeID": 3248, "retirement_year": 2021, "Times_maintained": 16}	21
{"Airline_name": "Jin Air", "Different_pieces_changed": 9, "Plane_type_name": "Boeing 737-800", "planeID": 1259, "retirement_year": 2019, "Times_maintained": 9}	15
{"Airline_name": "Jetstar Asia Airways", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A320", "planeID": 2927, "retirement_year": 2020, "Times_maintained": 0}	10
{"Airline_name": "Nova Airline", "Different_pieces_changed": 0, "Plane_type_name": "De Havilland Canada DHC-8-400 Dash 8Q", "planeID": 3406, "retirement_year": 2021, "Times_maintained": 0}	9
{"Airline_name": "Vueling Airlines", "Different_pieces_changed": 13, "Plane_type_name": "Airbus A320", "planeID": 1344, "retirement_year": 2021, "Times_maintained": 16}	3

#### 3.1.5.2 Explanation

En aquesta ocasió se'ns demana comptar el nombre d'aeroports que les diferents aerolínies han visitat. En aquest cas, s'ha de tenir en compte que els aeroports visitats siguin diferents entre sí, ja que es té en compte si són de 'Departure' o 'Destination', i això, mai poden ser el mateix aeroport del mateix país. A més de que al mostrar el llistat, els aeroports visitats siguin diferents entre ells per a evitar repeticions de països a tota costa, a ser possible.

#### 3.1.5.3 Query validation

```

MATCH ((c2:City)<--(ap2:Airport)-[:Destination]-(p:Plane)-[:Departure]->(ap:Airport)-->(c:City))
WHERE ap<>ap2 AND c<>c2
RETURN DISTINCT p, c AS numAirports
ORDER BY p.planeID DESC

```

4j\$

It is taking a long time to respond...

```
$ MATCH ((c2:City)←(:ap2:Airport)-[:Destination]-(p:Plane)-[:Departure]→(ap:Airport)→(c:City)) WHERE ap≠ap2 AND c≠c2... ⚡ ↴ ⌂ No
|lane_type_name":"Airbus A330-300","planeID":3761,"retirement_year":2020,"Times_maintained":0}
{"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
```

4j\$

It is taking a long time to respond...

```
$ MATCH ((c2:City)←(:ap2:Airport)-[:Destination]-(p:Plane)-[:Departure]→(ap:Airport)→(c:City)) WHERE ap≠ap2 AND c≠c2... ⚡ ↴ ⌂ No
|{"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
 {"City_name":"Aswan","cityID":1070,"timezone":2}
 {"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
 {"City_name":"Matei","cityID":4299,"timezone":12}
 {"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
 {"City_name":"Mae Hong Son","cityID":3657,"timezone":7}
 {"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
 {"City_name":"Lindsay","cityID":6144,"timezone":-5}
 {"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
 {"City_name":"Hurghada","cityID":1064,"timezone":2}
 {"Airline_name":"Nova Airline","Different_pieces_changed":0,"Plane_type_name":"De Havilland Canada DHC-8-400 Dash 8Q","planeID":3406,"retirement_year":2021,"Times_maintained":0}
 {"City_name":"Luxor","cityID":1066,"timezone":2}
```

Com es pot comprovar, si anem als resultats oficials i anem a una de les aerolínies amb menys nombre d'aeroports visitats, en aquest cas "DHC-8-400 Dash 8Q", amb planeID de 3406, a l'ésser 9 aeroports que ha visitat, podem comprovar llavors a quins aeroports de diferents ciutats a anat a parar, essent aquests Aswan, Matei, Mae Hong Son, Lindsay, Hurghada, Luxor, Iguatu, Jeddah i Cairo. El nombre de aeroports de les diferents ciutats visitades coincideix amb els resultats de la query de verificació.

### 3.1.6 Query 3

#### 3.1.6.1 Solution

```

MATCH (p:Plane) --> (a:Airport) -[:Airport_cityLocation]-> (:City) -[:Is_in]->(c:Country)
WHERE a.altitude > 100
RETURN p AS plane, COUNT(DISTINCT c) AS numCountries
ORDER BY COUNT(DISTINCT c) DESC

```

"plane"	"numCountries"
{"Airline_name": "Iberia Airlines", "Different_pieces_changed": 5, "Plane_type_name": "Canadair Regional Jet 900", "planeID": 651, "retirement_year": 2020, "Times_maintained": 5}	52
{"Airline_name": "Scandinavian Airlines System", "Different_pieces_changed": 11, "Plane_type_name": "Canadair Regional Jet 900", "planeID": 832, "retirement_year": 2020, "Times_maintained": 12}	37
{"Airline_name": "Swiss International Air Lines", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A330-300", "planeID": 3761, "retirement_year": 2020, "Times_maintained": 0}	36
{"Airline_name": "Jat Airways", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A319", "planeID": 475, "retirement_year": 2021, "Times_maintained": 0}	23
{"Airline_name": "SilkAir", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A319", "planeID": 1758, "retirement_year": 2021, "Times_maintained": 0}	11
{"Airline_name": "Flybaboo", "Different_pieces_changed": 12, "Plane_type_name": "Aerospatiale/Alenia ATR 72", "planeID": 3248, "retirement_year": 2021, "Times_maintained": 16}	6
{"Airline_name": "Jetstar Asia Airways", "Different_pieces_changed": 0, "Plane_type_name": "Airbus A320", "planeID": 2927, "retirement_year": 2020, "Times_maintained": 0}	5
{"Airline_name": "Jin Air", "Different_pieces_changed": 9, "Plane_type_name": "Boeing 737-800", "planeID": 1259, "retirement_year": 2019, "Times_maintained": 9}	5
{"Airline_name": "Nova Airline", "Different_pieces_changed": 0, "Plane_type_name": "De Havilland Canada DHC-8-400 Dash 8Q", "planeID": 3406, "retirement_year": 2021, "Times_maintained": 0}	4
{"Airline_name": "Vueling Airlines", "Different_pieces_changed": 13, "Plane_type_name": "Airbus A320", "planeID": 1344, "retirement_year": 2021, "Times_maintained": 16}	2

#### 3.1.6.2 Explanation

Per a aquesta query, se'ns demana realitzar una llista d'avions i el nombre de països visitats sempre i quan l'aeroport del país tingui una altitud major de 100, el qual això es fa amb una comprovació senzilla al WHERE, a més de tenir en compte d'afegir DISTINCT per a diferenciar les ciutats a comptar.

### 3.1.6.3 Query validation

```
MATCH (p:Plane) --> (a:Airport) -[:Airport_cityLocation]-> (:City) -[:Is_in]->(c:Country)
WHERE a.altitude > 100
RETURN DISTINCT a.name AS airport, a.altitude AS airport_altitude
ORDER BY airport_altitude
```

NOTA: En realitat hi han més files de les que es mostren aquí

"airport"	"airport_altitude"
"Monseñor Óscar Arnulfo Romero International Airport"	101
"Halmstad Airport"	101
"Taiwan Taoyuan International Airport"	106
"Weeze Airport"	106
"Yangon International Airport"	109
"Pangkal Pinang (Depati Amir) Airport"	109
"Sevilla Airport"	112
"Jorge Chávez International Airport"	113
"Heraklion International Nikos Kazantzakis Airport"	115
"Montreal / Pierre Elliott Trudeau International Airport"	118
"Jeju International Airport"	118

Es veu que funciona perquè no ens surt com a resultat aeròports amb un valor d'altitud de menys de 100, tal i com es pot veure a la imatge d'adalt.

### 3.1.7 Query 4

#### 3.1.7.1 Solution

```

MATCH v1 = shortestPath((p1:Country)-[:Departure|Destination|Is_in|Airport_cityLocation *..]-(p2:Country))
WHERE p1.Country_name = 'Greece' AND p2.Country_name = 'Singapore'
RETURN v1;

```



#### 3.1.7.2 Explanation

En aquesta query fem un shortestPath entre dos nodes tipus Country, en el que el primer es diu Greece i el segon Singapore. Per fer que només puguin aparèixer rutes de viatge, es limiten les relacions a departure,destination, is\_in i airport\_cityLocation. No es posa cap mena de restriccions en el nombre de relacions que hi pot haver.

#### 3.1.7.3 Query validation

Es mostra la relació de país a ciutat, de ciutat a l'aeroport i de l'aeroport a avió. A més a més, la ruta que es mostra només passa per un avió, per tant és la ruta més curta possible que es pot tomar entre dos països.

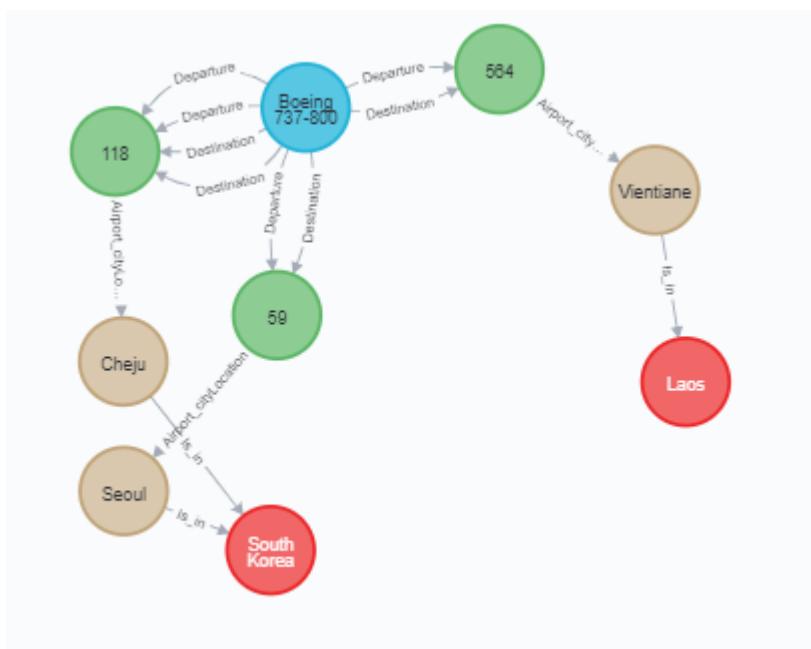
### 3.1.8 Query 5

#### 3.1.8.1 Solution

```

MATCH p=(start:Country)-[:Departure|Destination|Is_in|Airport_cityLocation *1..7]-(end:Country)
WHERE start.Country_name = 'Laos' AND end.Country_name = 'South Korea'
RETURN p

```



### *3.1.8.2 Explanation*

En questa query es mostren totes les relacions que hi ha a 7 nodes de distància (a més de 7 Neo4j deixava de respondre) que hi ha entre els països Laos i Corea del Sur. Les úniques relacions que s'han tingut en compte per això han sigut Departure, Destination, Is\_in, Airport\_city i Location, ja que son les relacions que poden representar una ruta en avió.

### *3.1.8.3 Query validation*

La query funciona bé ja que mostra relacions coherents entre els dos països, i mostra més d'una possible ruta per arribar. Si intentem buscar rutes més curtes, el programa no retorna cap resultat per què no existeixen, i si triem de trobar rutes més llargues, el programa agota el temps permitit per fer una query i no retorna cap resultat.

### **3.2 Case Study 2: Pilots, flight attendee, language, flight and airport**

En aquest case study volem veure les relacions entre els empleats d'un vol basat en el seu idioma i els llocs que han estat.

#### **3.2.1 Data migration**

En aquest case ens ha resultat més senzill fer la importació de les dades ja que ja havíem entès la metodologia gràcies al case 1.

Vam realitzar les queries dels dataset1, dataset2 , la seva relació i alguna taula més que necessitavem. Vam exportar directament varis CSV per no crear nodes repetits, els vam penjar en el Google Drive compartint-los per la web, i finalment vam fer la següent seqüència:

LOAD amb el fitxer csv corresponent pujat al drive,  
AS CSV WITH CSV

#### **En el cas de creació de nodes**

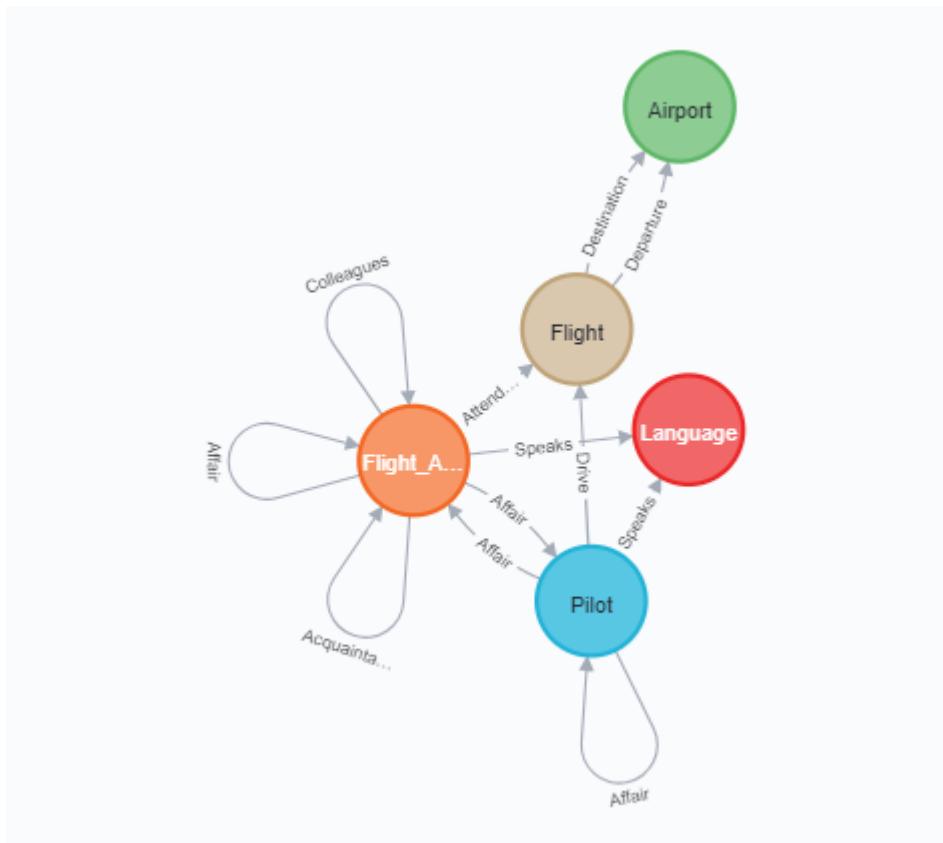
CREATE (del node en concret amb els seus respectius atributs)

#### **En el cas de creació de relacions**

MATCH (indiquem el tipus de node i li diem que busqui en una columna del csv),  
(Indiquem l'altre node i l'altre columna del CSV)

Finalment fem un create de les relacions de les variables dels nodes que hem fet el match just abans

### 3.2.2 Neo4J Data model



En aquesta base de dades, hi ha Vols (Flights), Pilots, Assistents de vol (Flight\_Attendant), Aeròports i Llenguatges. Els vols estan relacionats amb els aeròports per les relacions Destination i Departure, que indiquen en quin aeròport aterra el vol i de quin enllaire, respectivament. Cada vol també està relacionat amb assistents de vols per la relació Attends, que indica que el està treballant en aquell vol com a assistent, i amb el pilot per la relació Drive, que indica que aquell pilot està conduint el vol. Els llenguatges estan connectats amb els assistents de vol i amb els pilots per la relació Speaks, que indica que es parla aquell idioma. Els pilots i els assistents de vol estan relacionats entre si per la relació Affair, que és bidireccional i indica que hi ha hagut relacions entre aquestes dues persones. Affair també pot relacionar pilots amb altres pilots i assistents amb altres assistents. Per últim, els assistents poden tenir dos tipus més de relacions entre ells: Acquaintances, que indica que dos assistents es coneixen entre ells, i Colleagues, que indica que dos assistents treballen junts.

### 3.2.3 Query 1

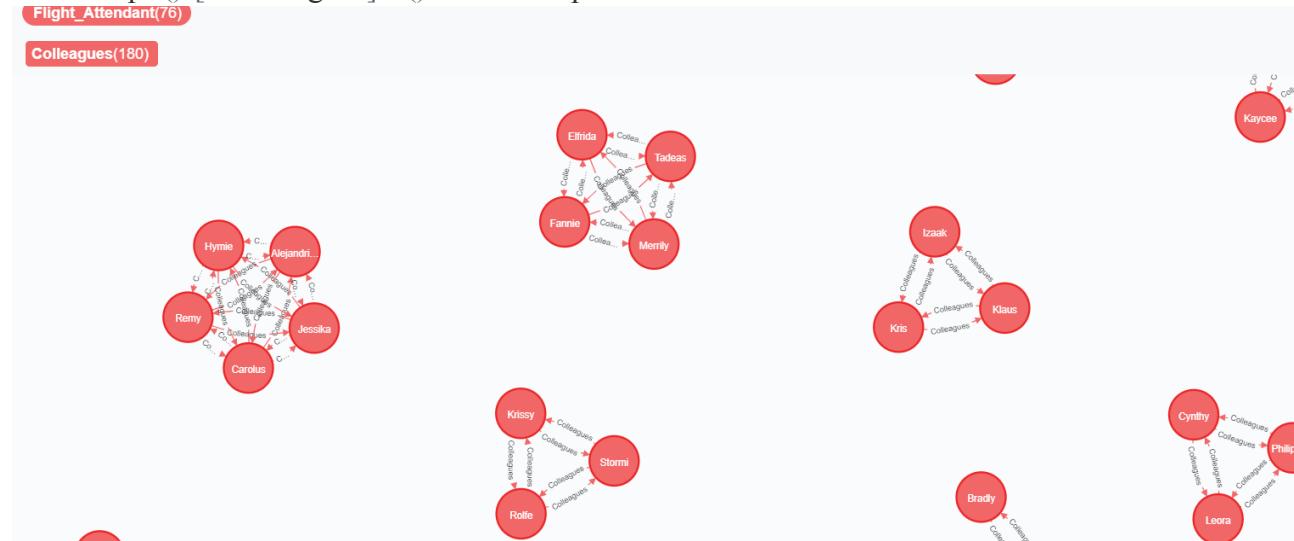
#### 3.2.3.1 Solution

```
MATCH (fa1:Flight_Attendant)-->(f: Flight) <--(fa2: Flight_Attendant)
WHERE fa1.flightAttendantID <> fa2.flightAttendantID
```

```
CREATE (fa1)-[:Colleagues]->(fa2)
```

Mostrem el resultat:

```
MATCH p=()-[r:Colleagues]->() RETURN p
```



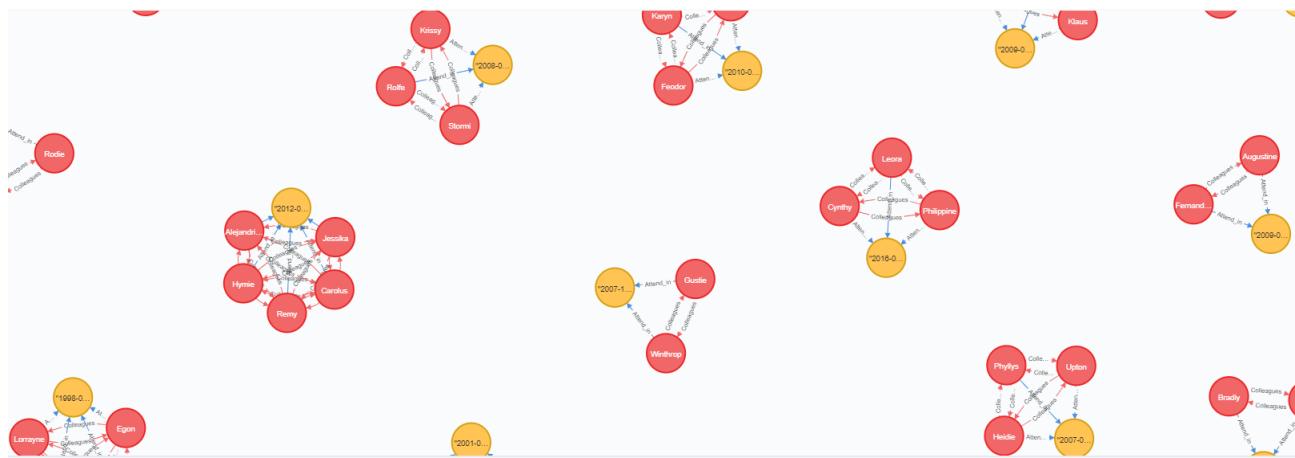
#### 3.2.3.2 Explanation

En aquesta query ens demanaven crear relacions entre els flight attendants que han coincidit en un mateix vol. Per tant, hem de fer un MATCH, en el que verifiquem si dos flight\_attendants han estat en el mateix vol. Després fem un CREATE en el que creem la relació entre els dos nodes.

#### 3.2.3.3 Query validation

Per verificar la query simplement hem de verificar si els collegues han coincidit en almenys un vol.

```
MATCH p=(f:Flight)<--()-[r:Colleagues]->()->(f) RETURN p
```



Podem observar que tots el colleagues estan agrupats amb un vol. Mirant aquest resultat es veu molt clar que tots els colleagues han coincidit en almenys un vol.

### 3.2.4 Query 2

#### 3.2.4.1 Solution

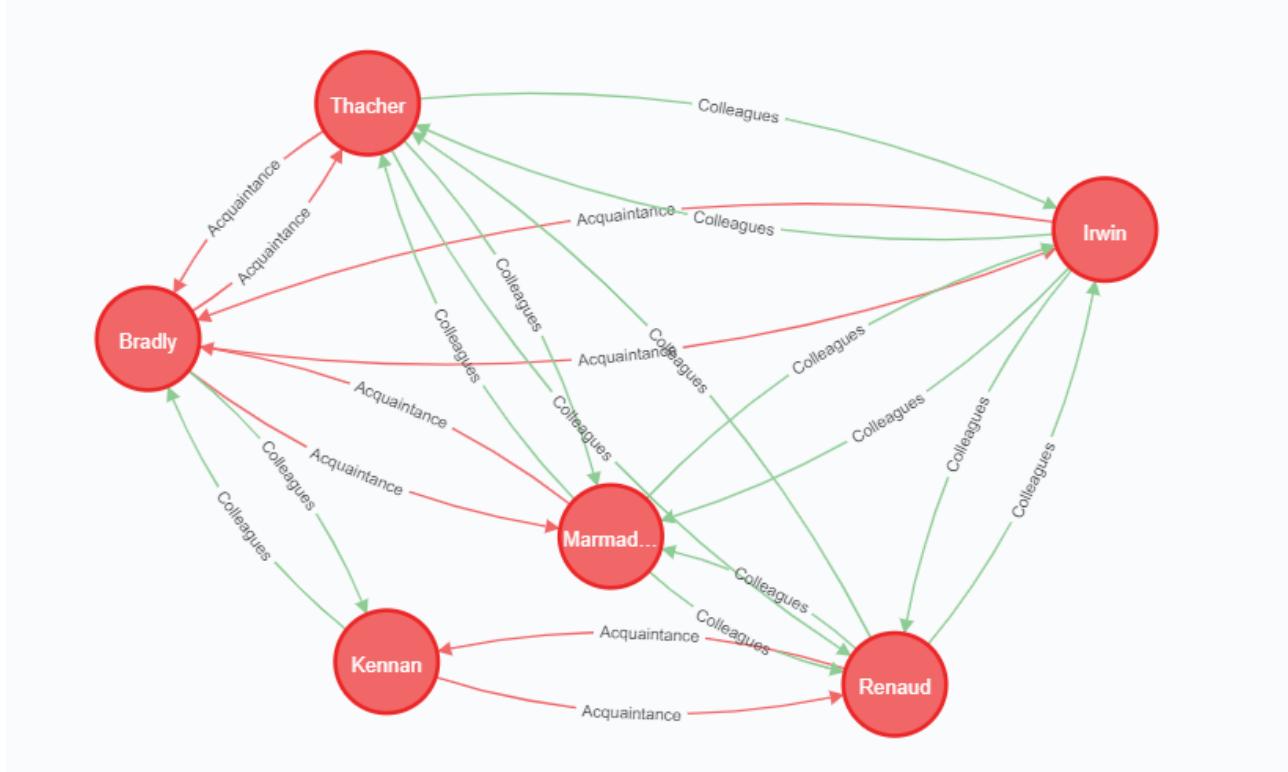
**MATCH**

(l1:Language)<--(fa1:Flight\_Attendant)-->(f1)-->(a:Airport)<--(f2)<--(fa2:Flight\_Attendant)-->(l2:Language)

**WHERE** l1.languageID = l2.languageID  
AND NOT (f1)-[:Colleagues]-(f2)

**CREATE** (fa1)-[:Acquaintance]->(fa2)

**MATCH** p=()-[r:Acquaintance]->() **RETURN** p



#### 3.2.4.2 Explanation

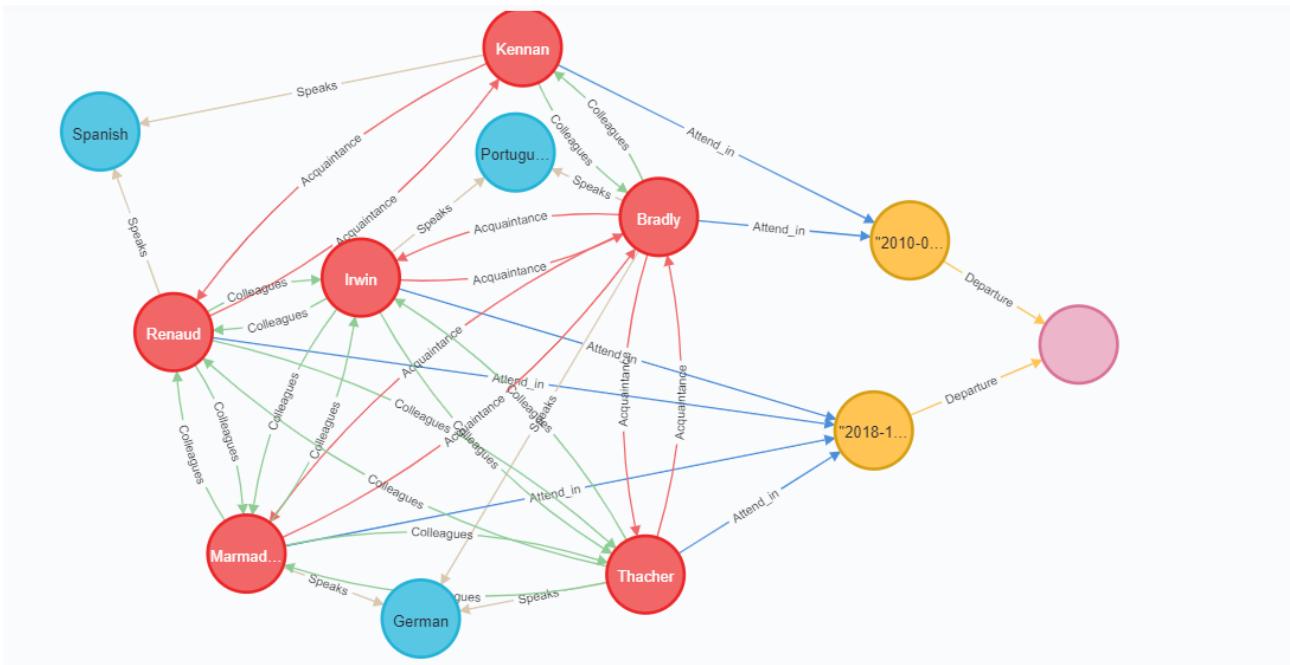
En aquesta query ens demanaven crear una relació de acquaintance quan dos flight attendants han estat al mateix aeroport i parlen el mateix idioma. A més hem de tenir en compte de que no siguin colleagues. En la query verifiquem aquestes condicions i creem la relació entre ells.

#### 3.2.4.3 Query validation

Per verificar la query simplement mostrem els aeroports que coincideixen i els idiomes que tenen en comú.

**MATCH**

p=(a:Airport)<--(f:Flight)<--(fa:Flight\_Attendant)-->(l:Language)<--(fa1:Flight\_Attendant)-->()-->(a) **RETURN** p



Podem observar que quan dues persones parlen el mateix idioma, han coincidit en un aeroport i no son colleagues, son acquaintance.

### 3.2.4 Query 3

#### 3.2.4.4 Solution

```
CREATE ()-[af:Affair]->() RETURN af;
```

**MATCH**

```
(l:Language)<-[ :Speaks ]-(p:Pilot)-[:Drive]->(f:Flight)<-[ :Attend_in ]-(fa:Flight_Attendant)-[:Speaks ]->(l2:Language)
```

```
WHERE l.languageID = l2.languageID AND abs(p.years_working - fa.years_working) < 10 AND NOT (p)-[:Affair]-(fa)
```

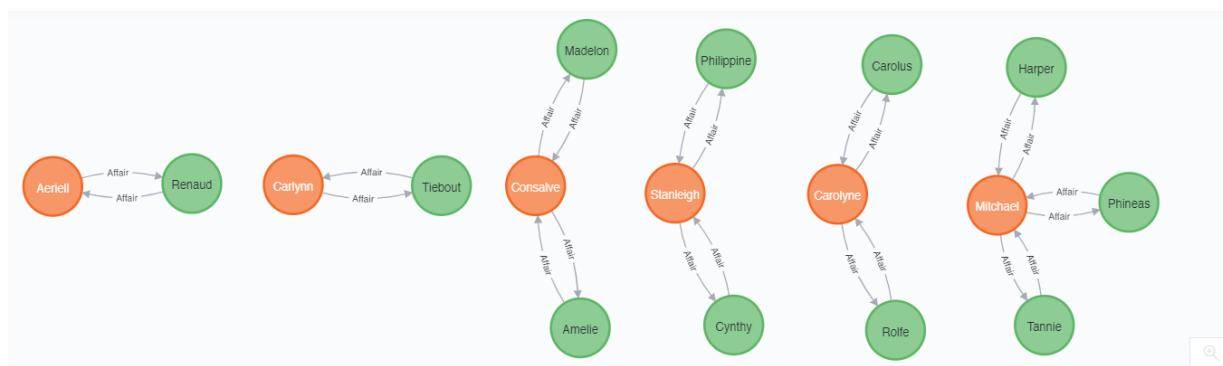
```
MERGE (p)-[:Affair]->(fa)
```

```
MERGE (fa)-[:Affair]->(p)
```

```
RETURN p, fa;
```

```
MATCH (n)-[:Affair]->() WHERE size(labels(n)) = 0 DETACH DELETE n;
```

NOTA: Acte seguit, s'hauria d'anar al menú lateral del projecte Neo4j i clicar sobre la relació Affair per a fer el display de resultats desitjats



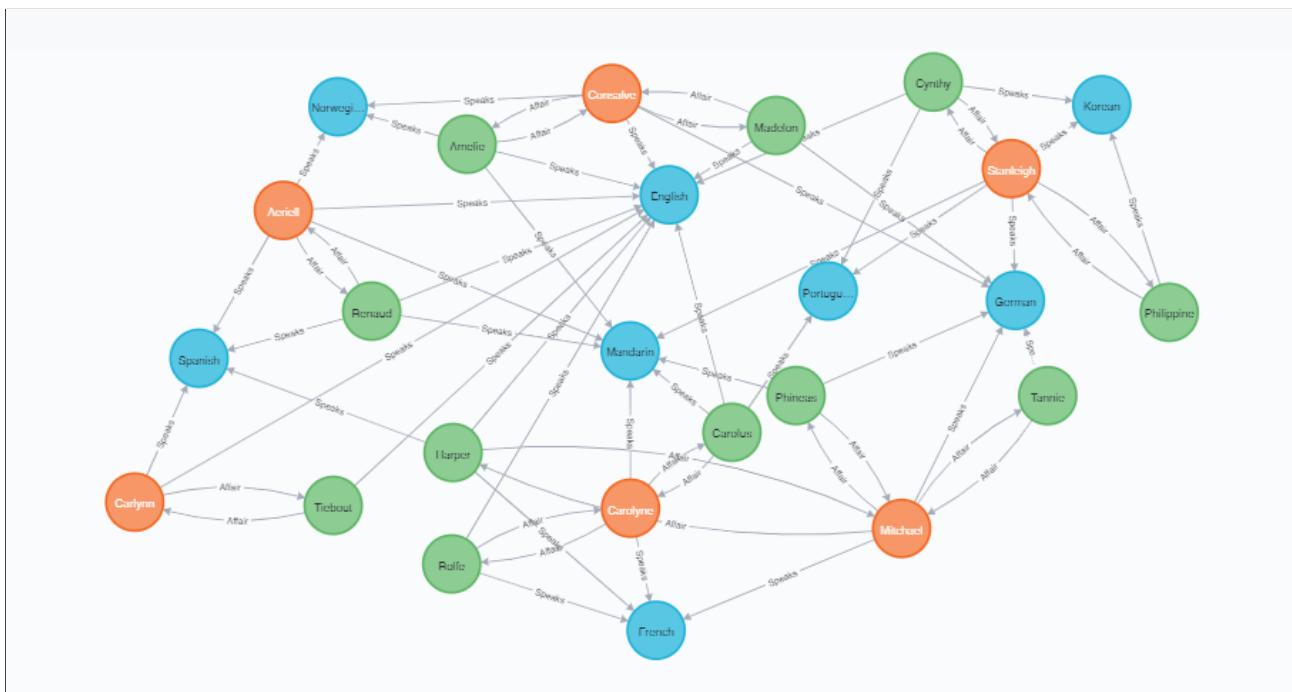
#### 3.2.4.5 Explanation

Per a aquesta ocasió, s'havia de tenir en compte varis aspectes a l'hora de realitzar aquesta query on se'ns demana afegir una relació d'Affair entre els pilots i flight attendants que tenen "aventures", a més de comptar el nombre de relacions d'Affair.

Primer de tot havíem de crear una relació d'Affair buida sense cap node connectat. Després, creàvem la query amb les especificacions donades a l'enunciat i asseguràvem abans de res de encara no conectar pilot i flight attendant com a relació d'affair. I un cop fet això, els ajuntàvem amb un MERGE la relació d'Affair d'anada i retorn per a ambdós usuaris, i ja per últim eliminàvem aquells nodes buits sense continguts buits amb relació d'Affair entre ells a través de la següent comanda:

```
MATCH (n)-[:Affair]->() WHERE size(labels(n)) = 0 DETACH DELETE n;
```

### 3.2.4.6 Query validation



### 3.2.5 Query 4

#### 3.2.5.1 Solution

**MATCH**

```
(l:Language)-[:Speaks]-(p:Pilot)-[aff:Affair]->(fa:Flight_Attendant)-[:Speaks]->(l2:Language)
RETURN DISTINCT l.name, COUNT(DISTINCT aff)
ORDER BY COUNT(distinct aff) DESC;
```

"l.name"	"COUNT(DISTINCT aff)"
"German"	14
"French"	10
"Mandarin"	10
"Hungarian"	8
"English"	8
"Malay"	6
"Arabic"	6
"Norwegian"	6
"Portuguese"	4
"Korean"	4
"Spanish"	4

"Slovak"	4
"Japanese"	2
"Hindi"	2

#### 3.2.5.2 Explanation

En aquesta ocasió se'ns demana buscar les relacions d'Affair entre el pilot i un flight attendant, sempre i quan parlin el mateix idioma. En aquest cas, l'únic aspecte a tenir en compte per a aquesta query ha estat diferenciar no només filtrar per un ORDER BY amb DISTINCT segons el nombre del llenguatge, sinó també diferenciar les relacions d'Affair entre aquests usuaris utilitzant un DISTINCT a dins del comptatge total de relacions d'Affair.

#### 3.2.5.3 Query validation

**MATCH**

```
(l:Language)-[:Speaks]-(p:Pilot)-[aff:Affair]->(fa:Flight_Attendant)-[:Speaks]->(l2:Language)
RETURN DISTINCT l.name, COUNT(DISTINCT aff), p.name, fa.name
ORDER BY COUNT(distinct aff), l.name DESC;
```

The screenshot shows the Neo4j browser interface. At the top, there's a header with icons for back, forward, search, and session management, followed by the URL <https://032b2...> and a button labeled "Iniciar sesi". Below the header, the title bar says "GERMAN" and "1/23". The main area has a progress bar at the top with the message "Server is taking a long time to respond...". Below the progress bar, the command entered is "neo4j\$ MATCH (l:Language)←[:Speaks]...". To the left of the main area, there are three vertical tabs: "Table" (selected), "Text", and "Code". The main area displays a table with four columns of data:

"Hungarian"	2	"Carolyne"	"Rolle"
"Hindi"	2	"Carlynn"	"Tiebout"
"German"	2	"Mitchael"	"Phineas"
"German"	2	"Mitchael"	"Harper"
"German"	2	"Mitchael"	"Tannie"
"German"	2	"Stanleigh"	"Cynthy"
"German"	2	"Stanleigh"	"Philippine"
"German"	2	"Consalve"	"Amelie"
"German"	2	"Consalve"	"Madelon"
"French"	2	"Mitchael"	"Phineas"
"French"	2	"Mitchael"	"Harper"
"French"	2	"Mitchael"	"Tannie"

Com es pot comprovar, en el cas de “German”, si comptem manualment, hi ha en total 14 relacions de llenguatge, les quals, de dos en dos (que fan referència tant a la relació d’X -> Y i Y -> X) demostra la relació establerta del llenguatge que utilitzen. En aquest cas, Mitchael té una relació amb Phineas i Phineas té una relació amb Mitchael on parlen en Alemany ambdós. Això explica el perquè es compta com dos, i no com un, ja que estem tenint en compte ambdues posicions d’aquests usuaris. I aquesta lògica, aplicada a la resta de llenguatges d’Affair entre els altres usuaris.

### 3.2.6 Query 5

#### 3.2.6.1 Solution

```
MATCH (fa:Flight_Attendant )<-[ :Affair ]-(p:Pilot)-[ :Affair ]->(f:Flight_Attendant)
WHERE (fa)<>(f) AND (fa)-[ :Colleagues ]->(f)
RETURN p
```



#### 3.2.6.2 Explanation

En aquesta query se'ns demana de mostrar els pilots que hagin tingut alguna “affair” amb dues o més flight attendants i que aquestes flight attendants siguin col·legues. Per tant el que busquem es un node de pilot que tingui dues relacions d'affair amb dos nodes de flight attendants diferents. Els dos nodes de flight attendant han de tenir una relació de colleagues entre ells.

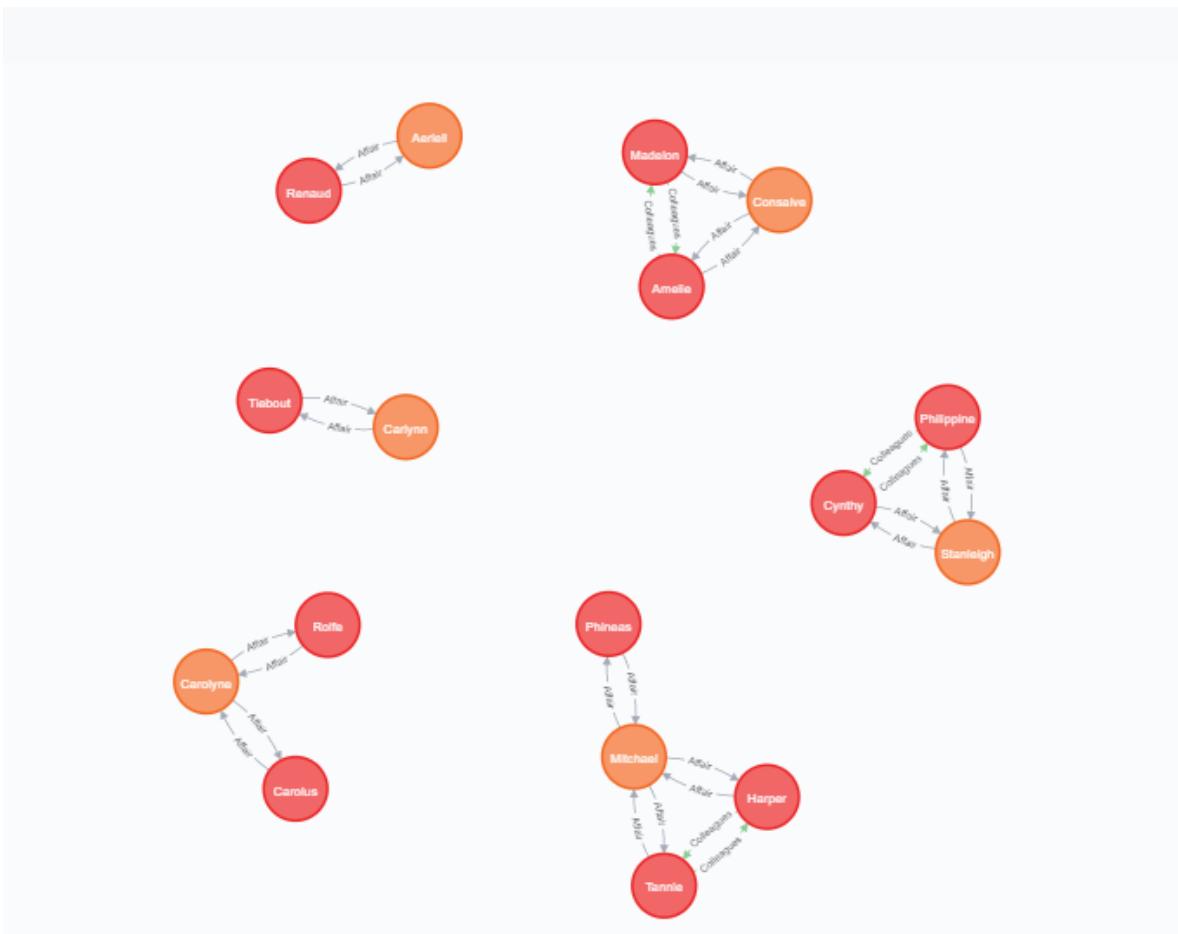
Retornem només el pilot ja que és el que s'ens demana

#### 3.2.6.3 Query validation

Per verificar aquesta query podem mirar primer les flight attendants amb qui el pilot ha tingut una affair:

```
MATCH (fa:Flight_Attendant )<-[ :Affair ]-(p:Pilot)
RETURN p, fa
```

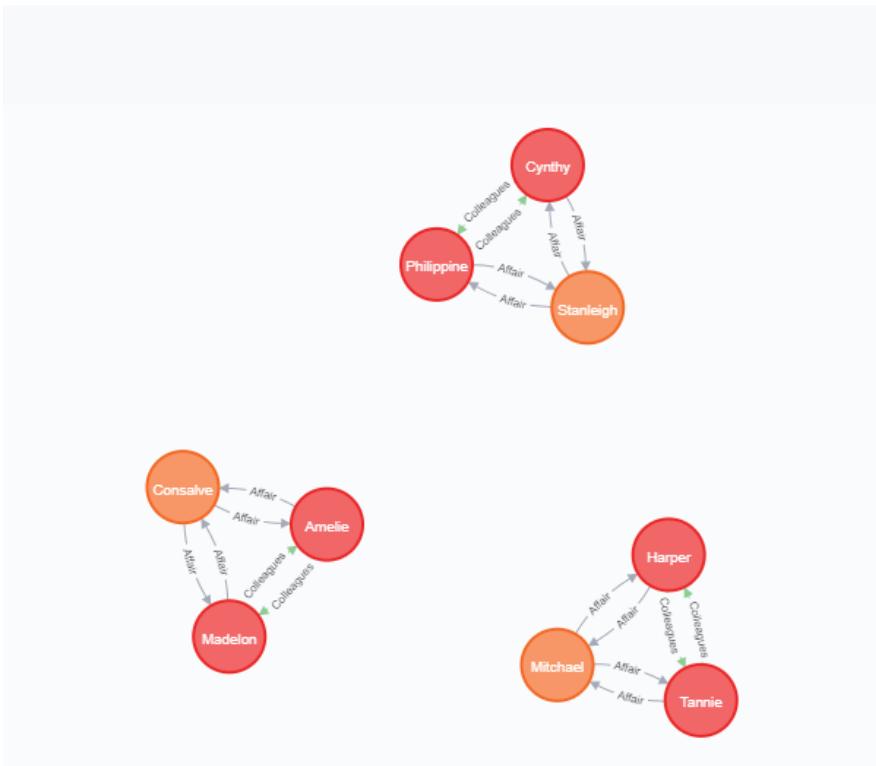
El resultat és el següent:



Podem observar que hi han pilots que han tingut una sola affair, dues affairs o fins i tot 3 affairs. Ara, els pilots que han tingut affairs amb 2 flight attendants i que a més siguin colleagues son menys:

```

MATCH (fa:Flight_Attendant)-[:Affair]-(p:Pilot)-[:Affair]->(f:Flight_Attendant)
WHERE (fa)<>(f) AND (fa)-[:Colleagues]->(f)
RETURN p,f,fa
    
```



Podem observar que aquests tres pilots son els únics que han tingut dos o més relacions “affair” amb flight attendants que son col·legues.

### 3.2.7 Query 6

#### 3.2.7.1 Solution

```
MATCH v1 = shortestpath((p1:Pilot)-[:Colleagues|Affair|Acquaintance *..]-(p2:Pilot))
MATCH (f2:Flight_Attendant)<--(p1)-->(f1:Flight_Attendant)
MATCH (f3:Flight_Attendant)<--(p1)-->(f4:Flight_Attendant)
WHERE p1<>p2
RETURN v1;
```

(no changes, no records)

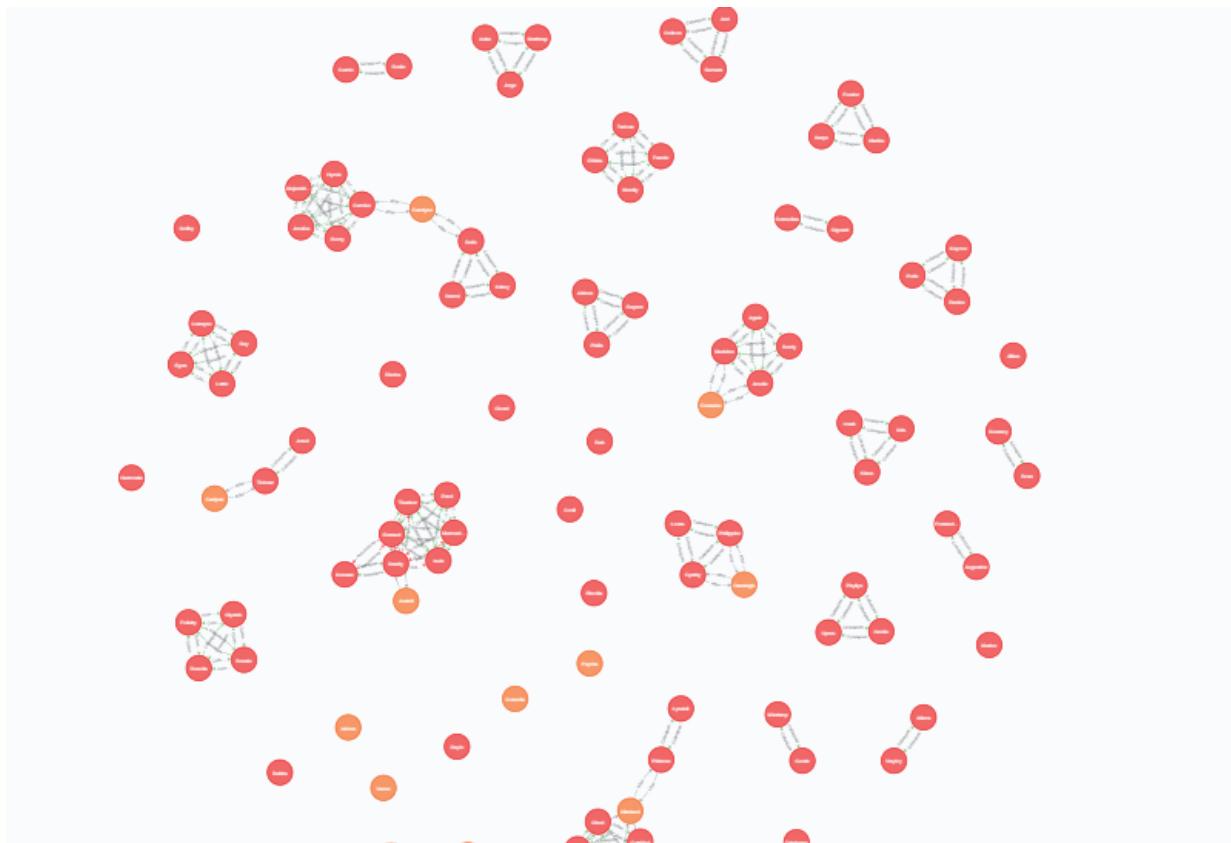
#### 3.2.7.2 Explanation

En aquesta query ens demanen de mostrar les relacions entre els pilots que tinguin més de una affaris. Aquestes relacions només poden ser Acquaintance, Colleagues o Affair. Per tant hem de realitzar un shortestpath per trobar les relacions i nodes que ens portin d'un pilot a un altre. També verifiquem que cadascun dels pilots tingui més d'una relació de affair

#### 3.2.7.3 Query validation

Mitjançant la query següent podem observar que efectivament, no hi ha cap relació entre pilots respectant les condicions del enunciat:

```
MATCH (n:Pilot)
MATCH (n2:Flight_Attendant)
RETURN n, n2
```





## 4 Conclusions

### 4.1 Use of resources

Member 4: Alex Collado

Stage	Member 1	Member 2	Member 3	Member 4	Total
SQL queries	20	25	25	25	95
Triggers	9	9	8	6	33
Events	3	2	2	3	10
Neo4J population	7	2	7	2	18
Neo4J queries	2	5	2	6	17
Documentation	9	7	7	3	25
<b>Total:</b>	<b>50</b>	<b>50</b>	<b>51</b>	<b>44</b>	<b>195</b>

### Member 4- Alex Collado Garrido

La part que més temps m'ha portat, amb diferència, han sigut les queries de SQL, ja que al fer més un mòdul, i sent el mòdul 5 només queries, hi ha sigut la part del projecte en la que més he treballat.

Els triggers i les queries de Neo4j també m'han portat una mica de temps, però no massa perquè només he hagut de fer 2 de cada. La població del Neo4j m'ha portat molt poc temps perquè és una part en la qual pràcticament no he treballat, excepte per afegir un parell de línies per fer funcionar una de les queries.

### Member 2- Wesley Lucas Mas

La part on he trigat més temps ha estat en les SQL queries ja que aquestes essent cinc de diferents dificultats cadascuna a nivell lògic, doncs té sentit que tragi bastant, conjuntament a més amb els triggers, que és el que també em va portar bastant de temps realitzar, i en algun que altre moment he necessitat ajuda dels meus companys ja no només a nivell de lògica que em donin suport, sinó a més perquè com ja vaig explicitar a la meva part del treball, el meu portàtil no va bé del tot a l'hora de fer córrer les execucions, i per tant, els meus companys van haver-hi d'ajudar-me executant les queries des dels seu workbench.

Respecte la població del Neo4j, és la part on tampoc la he treballat massa a fons. És a dir, ajudava com podia a Guillem i Laura, revisant que la info. estigués bé o per si faltaven detalls/informació a afegir, però tampoc he estat molt a fons perquè he invertit temps en la resta d'apartats. Tot i així, si necessitaven ajuda, jo els hi donava suport com pogués. I respecte les queries del Neo4j, em va portar molt de temps per la lògica de les queries del case 2 de la relació d'Affairs i de la case 1 on tenien una mica de trampa a l'hora de plantejar la lògica del Departure i Destination dels avions.

I ja per últim la documentació m'ha trigat moltíssim temps per a fer-la ja que en la part dels triggers, volia explicar amb molt de detall el com va ser el procés de validació amb tal de que tant el becari que corregís el codi com el professor que s'està llegint ara mateix aquesta memòria, veïs que la meva forma de fer les validacions tot i que era diferent a la resta dels meus companys, era degut a que estic tenint dificultats amb el meu portàtil, i com a tal, m'he assegurat moltíssim de donar molt detall per a demostrar imatge per imatge que el que he fet és correcte.

### Membre 1 - Laura Pascual López

En el meu cas, he estat bastant temps amb les meves queries. Al principi em vaig encallar una mica amb la primera query ja que no entenia del tot l'enunciat, fent que intentés fer una consulta unaica estranya que no complia amb els requisits de l'enunciat. Després de parlar amb becaris per intentar resoldre-la vaig aconseguir arribar a una idea de com fer-ho, però, un cop ho vaig acabar, el resultat no em convencia, tenia la sensació de que estava malament, pel que me la vaig estar mirant contínuament. Finalment, gràcies als becaris, em van ajudar a trobar en el meu error en el càlcul de les hores.

Respecte les altres queries, també vaig tenir alguns problemes, sobretot durant les validacions, em vaig trobar amb varis errors, fent que m'hagués de repassar gairebé totes les consultes. Una d'elles va ser la dels pilots i copilots, on vaig cometre un error bastant despistat però, em va costar adonar-me. Una altre que em va fallar va ser la dels avis que parlaven idiomes. Al principi, tal i com ho vaig fer, em donava problemes amb aquells passatgers que parlaven varis idiomes, en la consulta m'apareixien les persones que com a mínim una de les seves llengües no es trobés entre els flight Attendant (per tant era un error). Després de comentar-li a una becaria, vaig arribar a la conclusió de fer un NOT EXISTS, però un cop feta, vaig tardar molt en la seva validació, ja que els resultats em semblaven estranys i molts d'ells els vaig fer fila per fila.

Un altre motiu del perquè les queries són les que més temps he estat, ha sigut perquè en alguna revisió amb els becaris, alguna consulta que em semblava tenir bé, es podria millorar (com en el cas dels passatgers i les dobles turbulències).

Sobre els triggers, en si, no vaig tardar massa en fer-los, el principal pes del temps es va trobar en les diferents validacions d'aquests. Sobretot per la 1.6, que em vaig trobar l'error de no poder modificar la taula que s'utilitza per cridar el trigger dins el trigger. Al principi no tenia ben clar com resoldre aquest error, vaig provar de fer varies coses com fer el trigger que al fer el insert del que se'm demanava, cridés un altre trigger amb una altra taula perquè eliminés les files... però tots aquests intents no anaven. Fins que al final, em vaig plantejar escriure a la memòria que no es podia però abans, ho vaig voler confirmar amb un becat, que gràcies a ell em va donar una pista per trobar la solució.

El event en si vaig tardar poquet comparat amb els altres. Al principi em va sortir algun error però en general, entre validació i algun error, vaig tardar allò aproximadament.

En la població del NEO4J, vaig tardar una mica més perquè vaig fer les querys del case 1, i vaig participar activament en la inserció de les dades tant del case 1 com del 2.

Sobre les querys del NEO4J, al fer gran part de la inserció, em va tocar fer menys queries.

Finalment, en la memòria he tardat bastant ja que intento expressar correctament i amb detall les diferents consultes i elements que se'ns demana.

### **Member 3 - Guillem Roselló Christiaen**

Per la meva part m'han costat molt la part de les queries del MySQL ja que no vaig acabar d'entendre bé la mecànica de les subqueries i els joins. Es pot veure per les meves notes de les AC que aquesta part hem costava més. Però poc a poc li he pillat el truc i ara ja entenc bé el funcionament.

## **4.2 Lessons learnt (1 page)**

### **Alex Collado Garrido**

En aquest projecte he après a optimitzar el codi en fer querys en mySQL i escollir bé quines eines utilitzar a l'hora de fer-les. Es nota especialment la diferència si en comparo les querys que vaig fer pel projecte del primer semestre amb aquest.

També he millorat bastant en l'ús de Neo4j i Cypher, i he après com exportar les dades d'una base de dades en SQL a un altre gestor de base de dades com és Neo4j utilitzant CSVs exportats directament des de SQL.

### **Wesley Lucas Mas**

En aquest projecte extens he après no només a emprar les eines noves ensenyades a classe com els JOINs, subqueries, triggers, events, etc... sinó que a més he après sobretot a treballar en un entorn com Neo4j on l'entorn de treball respecte els sistemes gestors de bases de dades com MySQL, és diferent a la seva forma i fins i tot, complexe des del meu punt de vista personal.

Òbviament no podia faltar a més l'exportació de CSVs al Neo4j, que tot i que no he tingut un rol molt important en aquesta part, sí que diré que apart de fer revisió de dades a aquests, també he fet les meves pròpies importacions de informació a l'hora de donar suport als meus companys, ja no només per donar-lis suport, sinó també per a aprendre com fer-ho seguint la teoria de classe i del que ens documentàvem a internet.

I en general, a nivell d'assignatura, com a conclusió final del curs, combinat amb el projecte de 2on semestre de Direcció i Programació Orientada a Objectes, he après la importància que té l'àrea de les bases de dades i lo important que són a l'hora d'emmagatzemar informació i filtrar-la segons nosaltres vulguem, fent servir les eines adequades en tot moment. És molt fàcil cometre errors de tant i quan, però tot i així, amb el que he après aquest curs i a quina part de internet he de trobar les coses, sabria com guiar-me al món laboral amb aquestes eines.

### **Laura Pascual López**

Gràcies a la pràctica, he consolidat millor els coneixements que ens han donat al llarg del segon semestre. M'ha ajudat principalment que apareguessin elements que a classe vam tractar una mica per sobre per així poder-ho acabar d'entendre.

Una altre cosa que considero que he après ha estat la inserció de dades en el NEO4J ja que abans anava bastant perduda.

### **Guillem Roselló Christiaen**

Durant la realització del projecte crec que he millorat molt en SQL. Abans de la realització, i com podem veure per les meves notes de les AC setmanals de SQL hem costava molt aquest tipus de queries i no estava gens acostumat. Ara entenc molt millor el funcionament de cada element i sé com optimitzar una query. En el Neo4j, crec que no se m'ha donat gaire malament desde un bon

principi per tant no he tingut gaires dificultats per fer les queries. Sí que he après a insertar les dades al Neo4j ja que abans no sabia com fer-ho.

#### **4.3 Future work and conclusions (1 page)**

Per millorar el projecte crec que el que hauríem de fer és verificar les queries directament després d'haver-les fet i no esperar a haver acabat totes les queries per verificar-les després. Això no és una cosa que ha passat amb totes les queries però si amb algunes. Comparat al primer semestre que ens passàvem tots els fitxers per un chat de discord, ara, ens hem organitzat mitjançant un servidor del discord i tenim diversos canals destinat cadascun a una cosa en específic. Aquesta solució tot i ser millor que el que feiem en el primer semestre no és la millor. Lo millor creiem que seria utilitzar Git per així poder tenir un seguiment de les version. A més Git és una eina utilitzada per molts desenvolupadors i creiem que és necessari aprendre a utilitzar-la.

Tot i aquests problemes creiem que som un grup que ens portem molt bé entre tots i tenim un bon ambient de treball. Tots hem fet part de la nostra feina i hem ajudat als altres quan ho necessitaven. Ens hem organitzat molt bé per a aquesta pràctica i hem sapigut coordinar-nos i repartir-nos la feina correctament. No hem tingut cap conflicte intern i tothom ha sapigut fer el que se'ls demana.

Aquest projecte ha sigut molt enriquidor per a tot i ens ha permès entendre els conceptes ensenyats a classe i fer que se'ns internint.