

Databases: Second Semester Project

La Salle AIR 2020-2021

List of members (name and email): Lluís Gumbau (lluis.gumbau@students.salle.url.edu), Narcís Cisquella (narcis.cisquella@students.salle.url.edu), Marc Postils (marc.postils@students.salle.url.edu) i Joan Llobet (joan.llobet@students.salle.url.edu)

Date of finalisation: 30/06/2021

Summary of tasks

Task	Subtask	Status (completed, in progress, not started, 3/5 completed...)
Requirement 1 Flights	Queries x5	4/5 completed
	Triggers x3	1/3 completed
	Event x1	completed
Requirement 2 Airports	Queries x5	4/5 completed
	Triggers x3	1/3 completed
	Event x1	Not completed
Requirement 3 Shopping	Queries x5	4/5 completed
	Triggers x3	1/3 completed
	Event x1	completed
Requirement 4 Luggage	Queries x5	3/5 completed
	Triggers x2	2/2 completed
	Event x1	Not completed
Requirement 5 Cross-module	Queries x6	1/6 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

1	INTRODUCTION (1 PAGE).....	3
2	LSAIR RELATIONAL DATABASE REQUIREMENTS.....	4
2.1	REQUIREMENT 1 FLIGHTS.....	4
2.2	AIRPORT MODULE.....	21
2.3	SHOPPING MODULE	35
2.4	LUGGAGE MODULE	53
2.5	CROSS MODULE QUERIES.....	71
3	LSAIR GRAPH DATABASE REQUIREMENTS	76
3.1	CASE STUDY 1: AIRPLANE, AIRPORT, CITY, AND COUNTRY	76
3.2	CASE STUDY 2: PILOTS, FLIGHT ATTENDEE, LANGUAGE, FLIGHT AND AIRPORT	88
4	CONCLUSIONS	101
4.1	USE OF RESOURCES	101
4.2	LESSONS LEARNT (1 PAGE)	101
4.3	FUTURE WORK AND CONCLUSIONS (1 PAGE)	102

1 Introduction (1 page)

Els principals objectius d'aquest projecte és posar en pràctica tot el que hem après en el curs des de la lectura i enteniment dels models d'una base de dades fins a tot tipus de consultes avançades. I finalment, comprovar que tot sigui correcte amb les validacions corresponents.

El projecte es basa en tot el comportament i tot el que forma un aeroport, des de les característiques i manteniment d'un avió fins a les botigues que hi ha dins d'aquest.

El projecte està dividit en 5 blocs principals, que son els següents:

Bloc Airport: És l'encarregat de guardar i vincular tota la informació referent a l'aeroport. Els avions, els manteniments de l'avió, les rutes entre aeroports, el tipus de l'avió... tota la informació del aeroport i dels avions que hi ha en el aeroport la trobem en aquest bloc.

Bloc Shopping: És l'encarregat de guardar i vincular tota la informació referent a les compres. Tota activitat que pots fer dins d'un aeroport, compres, àrees de descans, productes dins d'una botiga... Tota aquesta informació la trobem dins d'aquest bloc.

Bloc Flight: És l'encarregat de guardar i vincular tota la informació referent als vols. El pilot del vol, les auxiliars de vol, el checkin, els passatgers... tota la informació vinculada a un vol, la trobem en aquest bloc.

Bloc Luggage: És l'encarregat de guardar i vincular tota la informació referent a l'equipatge. Les maletes que hi en un vol, els objectes perduts, les reclamacions... tota la informació de l'equipatge es guarda en aquest bloc.

Bloc Shared: És l'encarregat de guardar i vincular tota la informació referent a les persones. Aquí és on tractem tota la informació de les persones, els empleats, els llenguatges que parla cada persona, la ciutat i el país, tot i que aquest dues últimes entitas també estan vinculades a taules que no tenen res a veure amb les persones, com per exemple amb Airline.

No obstant que cada bloc tingui la seva funció, estan vinculats entre ells mateixos per donar credibilitat al model, ja que no té sentit que guardem la informació de tot l'equipatge d'un vol, i tota aquesta informació no estigui vinculada al seu vol corresponent.

En primer lloc, ens vam repartir la feina de manera equitativa, cadascú va fer un requeriment, en el nostre cas en Marc es va encarregar de fer el Requeriment 1, en Lluís el Requeriment 2, en Joan el 3 i en Narcís el 4. Un cop feta aquesta primera part, ens vam repartir la feina restant, on en Marc i en Joan es van enfocar més en els Triggers i els Events, mentres que en Narcís i en Lluís es van centrar en la part del Neo4j. Tot i que hem intentant seguir aquest ordre, al final ens hem acabat ajudant mútuament i tothom ha tocat cada part del projecte, ja que si algun ens quedàvem encallats els altres intentàvem ajudar i seguir avançant.

2 LSAIR Relational Database requirements

2.1 Requirement 1 Flights

2.1.1 Requirement (Query) 1.1 Anticipating countries

2.1.1.1 Solution

```
(SELECT 'most anticipating country' AS 'Type of country', c.name AS 'country name', AVG(f.date - ft.date_of_purchase) AS 'difference in hours', AVG(ft.price) AS 'Average price'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight AS f ON ft.flightID = f.flightID
GROUP BY c.countryID
HAVING COUNT(pe.personID) > 300
ORDER BY AVG(f.date - ft.date_of_purchase) DESC
LIMIT 1)
UNION
(SELECT 'least anticipating country' AS 'Type of country', c.name AS 'country name', AVG(f.date - ft.date_of_purchase) AS 'difference in hours', AVG(ft.price) AS 'Average price'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight AS f ON ft.flightID = f.flightID
GROUP BY c.countryID
HAVING COUNT(pe.personID) > 300
ORDER BY AVG(f.date - ft.date_of_purchase) ASC
LIMIT 1);
```

Type of country	country name	difference in hours	Average price
most anticipating country	Bolivia	1638.2201	375.9254
least anticipating country	Guam	680.7079	330.9010

2.1.1.2 Explanation

En aquesta query fem servir un UNION per mostrar els dos resultats, un pel most anticipating country i l'altre pel least anticipating country. Per ambdós resultats, fem el SELECT de les dades que se'ns demana, els JOINS necessaris, fem el GROUP BY de countryID ja que després al HAVING fem una subquery en que imposem que als països han de haver-hi més de 300 persones. Finalment, ordenem el resultat per la antelació mitjana en que la gent del país ha comprat el ticket (de major a menor o de menor a major depenent de que ens interressi) i posem el LIMIT 1 als dos resultats ja que sols en interessa mostrar el most i el least.

2.1.1.3 Query validation

```
SELECT DISTINCT c.name, COUNT(pe.personID) AS 'número de persones al país'
FROM person AS pe JOIN country AS c ON pe.countryID = c.countryID
WHERE c.name = 'Bolivia';
```

country name	número de persones al país
Bolivia	348

En aquesta query de validació comprovem que el país que ens surt que és el most anticipating country (Bolivia), efectivament té més de 300 persones.

```
SELECT DISTINCT c.name AS 'country name', COUNT(pe.personID) AS 'número de personas al país'
FROM person AS pe JOIN country AS c ON pe.countryID = c.countryID
WHERE c.name = 'Guam';
```

country name	número de personas al país
Guam	313

En aquesta query de validació comprovem que el país que ens surt que és el least anticipating country (Guam), efectivament té més de 300 persones.

També hem fet les següents queries de validació:

```
SELECT c.name AS 'country name', AVG(f.date - ft.date_of_purchase) AS 'difference in hours',
AVG(ft.price) AS 'Average price'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight AS f ON ft.flightID = f.flightID
GROUP BY c.countryID
HAVING (SELECT COUNT(pe2.personID)
        FROM person AS pe2 JOIN country AS c2 ON pe2.countryID = c2.countryID
        WHERE c.countryID = c2.countryID) > 300
ORDER BY AVG(f.date - ft.date_of_purchase) DESC;
```

country name	difference in hours	Average price
Bolivia	1638.2201	375.9254
Botswana	1634.5132	355.1842
Yemen	1572.0833	316.3841
Kiribati	1526.4808	376.8413
Zambia	1509.5913	414.1435
Greenland	1507.6533	284.7538
Cameroon	1473.8578	297.9174
Mauritania	1469.9810	383.0048
Tanzania	1459.4770	333.9247
Venezuela	1451.3981	445.7184
South Korea	1446.7750	368.1875
Portugal	1446.4327	373.2837
Central Afric...	1442.4826	379.0448
Greece	1439.1914	328.7368
Bahamas	1435.7531	370.7037
Marshall Isla...	1411.4592	347.4678
Israel	1407.4979	388.2638
Djibouti	1404.6127	338.9827
United Arab ...	1403.8935	359.4068
Colombia	1305.1753	271.5418

Aquí comprovem que efectivament el país que compra els bitllets amb més antelació és Bolívia, ordenem de més a menys diferència d'hores.

```

SELECT c.name AS 'country name', AVG(f.date - ft.date_of_purchase) AS 'difference in hours',
AVG(ft.price) AS 'Average price'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN country AS c ON pe.countryID = c.countryID
JOIN flight AS f ON ft.flightID = f.flightID
GROUP BY c.countryID
HAVING (SELECT COUNT(pe2.personID)
FROM person AS pe2 JOIN country AS c2 ON pe2.countryID = c2.countryID
WHERE c.countryID = c2.countryID) > 300
ORDER BY AVG(f.date - ft.date_of_purchase) ASC;

```

country name	difference in hours	Average price
Guam	680.7079	330.9010
Peru	738.3709	450.5587
Saint Pierre and Miquelon	803.4932	376.0090
Wake Island	810.3618	425.2846
French Guiana	836.0821	440.2995
Haiti	846.4154	393.5179
Cote d'Ivoire	846.5348	371.8261
South Sudan	856.1554	409.8008
Qatar	880.3153	389.8829
Saint Helena	889.1410	291.1752
Uruguay	904.0614	369.0877
Bulgaria	906.6043	407.7362
Nicaragua	912.4643	378.7277
Laos	922.3527	294.6027
Oman	926.2451	355.3083
Barbados	926.6189	344.2336
Norfolk Island	929.5571	349.8219
Bosnia and Herzegovina	930.8166	373.4541

Aquí comprovem que efectivament el país que compra els bitllets amb menys antelació és Guam, ordenem de menys a més diferència d'hores.

Les comprovacions de la mitjana de la diferència en hores i la mitjana del preu no és necessari fer-la ja que la funció AVG ja et calcula la mitjana.

2.1.2 Requirement (Query) 1.2 Passengers and turbulences

2.1.2.1 Solution

```

SELECT DISTINCT pe.personID AS 'person id', pe.name, pe.surname, pe.born_date AS 'born date'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN flight AS f ON ft.flightID = f.flightID
WHERE (SELECT f2.date
      FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID = pa2.passengerID
      JOIN flighttickets AS ft2 ON pa2.passengerID = ft2.passengerID
      JOIN flight AS f2 ON ft2.flightID = f2.flightID
      WHERE pe.personID = pe2.personID
      ORDER BY f2.date DESC
      LIMIT 1) = (SELECT f3.date
                FROM flight AS f3 JOIN flighttickets AS ft3 ON ft3.flightID = f3.flightID
                JOIN passenger AS pa3 ON pa3.passengerID = ft3.passengerID
                JOIN person AS pe3 ON pe3.personID = pa3.passengerID
                JOIN status AS s3 ON f3.statusID = s3.statusID
                WHERE s3.status = "Strong Turbulences" AND pe.personID =
pe3.personID
                GROUP BY pe3.personID
                HAVING COUNT(s3.statusID) = 1
                ORDER BY f3.date DESC
                LIMIT 1)
ORDER BY pe.personID ASC;

```

person id	name	surname	born date
607	Shelby	Martinson	1937-12-19
828	Shannon	Janos	1956-08-29
1008	Renault	Grahlmans	1989-09-21
1267	Wenona	Gasker	1971-01-06
1379	Rikki	Balthasar	1988-01-14
1381	Frazer	Powis	1932-03-27
1517	Chadwick	Harlock	1931-06-15
1607	Lay	Houtby	1938-05-14
1639	Dode	Pyrton	1968-08-04
1718	Brandon	Gowanson	1965-10-18
1746	Molli	Jacobbe	1960-03-09
1774	Celestyn	Jarrette	1947-07-31
1809	Jamil	Shirtliff	1972-01-25
1863	Adolph	Willan	1941-04-03
1878	Millicent	Zuann	2010-09-06
1957	Giffie	Lamprey	1978-06-08
1990	Bobette	Bamfield	1983-01-10
2003	Penrod	Perllman	1926-09-18

2.1.2.2 Explanation

Fem un SELECT DISTINCT per diferenciar els passatgers i que no surtin repetits i fem els JOINS amb les taules necessàries. Al WHERE fem dues SUBQUERIES, en la primera mostrem la data del últim vol del passatger i en la segona fem que sigui la mateixa data en que el status sigui “Strong Turbulences” ja que ens diuen que l’últim vol que han agafat és de “Strong Turbulences”. La condició del HAVING COUNT = 1 és molt important ja que fa que el passatger només hagi agafat un vol amb “Strong Turbulences” i que aquest sigui l’últim vol. El ORDER BY de la última línia no és necessari.

2.1.2.3 Query validation

```
SELECT pe.personID AS 'person id', pe.name, pe.surname, pe.born_date AS 'born date', s.status,
f.date
```

```
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
```

```
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
```

```
JOIN flight AS f ON ft.flightID = f.flightID
```

```
JOIN status AS s ON f.statusID = s.statusID
```

```
WHERE (SELECT f2.date
```

```
FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID = pa2.passengerID
```

```
JOIN flighttickets AS ft2 ON pa2.passengerID = ft2.passengerID
```

```
JOIN flight AS f2 ON ft2.flightID = f2.flightID
```

```
WHERE pe.personID = pe2.personID
```

```
ORDER BY f2.date DESC
```

```
LIMIT 1) = (SELECT f3.date
```

```
FROM flight AS f3 JOIN flighttickets AS ft3 ON ft3.flightID = f3.flightID
```

```
JOIN passenger AS pa3 ON pa3.passengerID = ft3.passengerID
```

```
JOIN person AS pe3 ON pe3.personID = pa3.passengerID
```

```
JOIN status AS s3 ON f3.statusID = s3.statusID
```

```
WHERE s3.status = "Strong Turbulences" AND pe.personID =
```

```
pe3.personID
```

```
GROUP BY pe3.personID
```



```

HAVING COUNT(s3.statusID) = 1
ORDER BY f3.date DESC
LIMIT 1)
ORDER BY pe.personID ASC, f.date DESC;

```

person id	name	surname	born date	status	date
607	Shelby	Martinson	1937-12-19	Strong turbulences	2010-11-25
607	Shelby	Martinson	1937-12-19	Perfect	1987-01-08
828	Shannon	Janos	1956-08-29	Strong turbulences	1981-06-12
1008	Renault	Grahlmans	1989-09-21	Strong turbulences	1992-10-26
1008	Renault	Grahlmans	1989-09-21	5 hour delayed	1991-04-14
1267	Wenona	Gasker	1971-01-06	Strong turbulences	2019-01-01
1267	Wenona	Gasker	1971-01-06	Perfect	2013-05-04
1379	Rikki	Balthasar	1988-01-14	Strong turbulences	2014-01-07
1379	Rikki	Balthasar	1988-01-14	Little turbulences	1999-05-28
1379	Rikki	Balthasar	1988-01-14	1 day delayed	1995-07-29
1381	Frazer	Powis	1932-03-27	Strong turbulences	1981-06-12
1381	Frazer	Powis	1932-03-27	Perfect	1965-05-07
1517	Chadwick	Harlock	1931-06-15	Strong turbulences	1971-01-09
1607	Lay	Houtby	1938-05-14	Strong turbulences	2005-05-05
1639	Dode	Pyrton	1968-08-04	Strong turbulences	2014-11-30
1718	Brandon	Gowanson	1965-10-18	Strong turbulences	2016-03-06
1746	Molli	Jacobbe	1960-03-09	Strong turbulences	2016-01-28
1774	Celestyn	Jarrette	1947-07-31	Strong turbulences	2000-09-03
1809	Jamil	Shirtliff	1972-01-25	Strong turbulences	1986-12-20

La query és la mateixa que la principal però treien el distinct per a que surtin tots els vols dels passatger i afegint-li el select el estatut del vol i la data del vol. Hem ordenat descendentment pel personID i per la data del vol per a que els diferents vols dels passatgers ens surtin junts i per a que la data del vol estigui ordenada i sigui més fàcil de comprovar els resultats. Com es pot veure a la captura, l'últim vol que han fet els passatgers és de "Strong Turbulences", està ordenat per la data de més recent a més antiga. Fent scroll down després de compilar de query es pot veure com tots els casos compleixen amb el requeriment de l'enunciat.

```

SELECT pe.personID AS 'person id', pe.name, pe.surname, pe.born_date AS 'born date', s.status,
f.date
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN flight AS f ON ft.flightID = f.flightID
JOIN status AS s ON f.statusID = s.statusID
WHERE (SELECT f2.date
FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID = pa2.passengerID
JOIN flighttickets AS ft2 ON pa2.passengerID = ft2.passengerID
JOIN flight AS f2 ON ft2.flightID = f2.flightID
WHERE pe.personID = pe2.personID
ORDER BY f2.date DESC
LIMIT 1) = (SELECT f3.date
FROM flight AS f3 JOIN flighttickets AS ft3 ON ft3.flightID = f3.flightID
JOIN passenger AS pa3 ON pa3.passengerID = ft3.passengerID
JOIN person AS pe3 ON pe3.personID = pa3.passengerID
JOIN status AS s3 ON f3.statusID = s3.statusID

```

```

WHERE s3.status = "Strong Turbulences" AND pe.personID =
pe3.personID
GROUP BY pe3.personID
HAVING COUNT(s3.statusID) = 2
ORDER BY f3.date DESC
LIMIT 1)
ORDER BY pe.personID ASC, f.date DESC;

```

person id	name	surname	born date	status	date
11604	Heddi	Gredden	1964-05-05	Strong turbulences	2020-11-15
11604	Heddi	Gredden	1964-05-05	Perfect	2016-07-07
11604	Heddi	Gredden	1964-05-05	Perfect	2005-05-08
11604	Heddi	Gredden	1964-05-05	Strong turbulences	2005-04-02
11604	Heddi	Gredden	1964-05-05	Little turbulences	2003-08-05
13091	Jeramie	Hanbidge	1961-09-20	Strong turbulences	2014-02-22
13091	Jeramie	Hanbidge	1961-09-20	Perfect	2008-09-29
13091	Jeramie	Hanbidge	1961-09-20	Strong turbulences	2007-12-03
14762	Garnet	Viles	1955-05-03	Strong turbulences	2019-01-01
14762	Garnet	Viles	1955-05-03	Strong turbulences	2012-01-26
14762	Garnet	Viles	1955-05-03	3 hour delayed	2007-03-28
14762	Garnet	Viles	1955-05-03	Perfect	2003-04-27
14762	Garnet	Viles	1955-05-03	Perfect	2001-05-17
14762	Garnet	Viles	1955-05-03	Perfect	1986-09-11
14916	Berry	Blyth	1920-01-22	Strong turbulences	2014-10-14
14916	Berry	Blyth	1920-01-22	Strong turbulences	2011-12-19
14916	Berry	Blyth	1920-01-22	Perfect	2006-11-14
15034	Gordy	Cunningham	1972-12-06	Strong turbulences	2016-08-18
15034	Gordy	Cunningham	1972-12-06	Strong turbulences	2010-07-05
15034	Gordy	Cunningham	1972-12-06	Perfect	1997-10-20

Una altra manera de comprovar de que el resultat és correcte, és canviant el HAVING COUNT de la segona subquery i igualar-lo a 2 i d'aquesta manera forçar que el resultat surti incorrecte, d'aquesta manera sortiran passatgers que el seu últim vol ha estat de "Strong Turbulences" i, a més a més, anteriorment han tingut un altre vol amb "Strong Turbulences" prèviament. Això demostra que el HAVING COUNT(s3.statusID) = 1 de la query principal funciona correctament.

2.1.3 Requirement (Query) 1.3 Pilots and co-pilots

2.1.3.1 Solution

```

SELECT p.flying_license AS 'flying license', COUNT(f.pilotID) AS '# times she/he was pilot',
p.grade
FROM pilot AS p JOIN flight AS f ON p.pilotID = f.pilotID
WHERE p.grade - (SELECT AVG(grade)
FROM pilot) > 2
GROUP BY f.pilotID
HAVING COUNT(f.pilotID) < (SELECT COUNT(p2.copilotID)
FROM pilot AS p2
WHERE f.pilotID = p2.copilotID)
ORDER BY COUNT(f.pilotID) DESC;

```

flying license	# times she/he was pilot	grade
90-984-9951	2	9.46
05-564-1878	2	9.6
82-324-0620	2	9.56
15-741-5329	2	9.68
73-636-9959	1	9.56
92-674-3016	1	9.5
82-958-9279	1	9.62
08-440-1486	1	9.4
54-988-7758	1	9.52
65-983-7024	1	9.42
25-504-6465	1	9.76
22-707-0728	1	9.54
87-994-3423	1	9.64
60-173-1960	1	9.98

2.1.3.2 Explanation

Primer fem el SELECT de les dades que ens demanen, el JOIN de flight amb pilot i fem una SUBQUERY al where per imposar que el grade del pilot ha de ser superior per 2 punts a la mitjana de tots els pilots. Agrupem pel id del pilot i al HAVING posem la condició de que el pilot ha estat més vegades copilot que no pas pilot, això ho fem amb una SUBQUERY i igulem el id del pilot amb el del copilot perquè volem mirar la mateixa persona. Finalment ordenem descendentment pel número de vegades que ha estat pilot.

2.1.3.3 Query validation

```

SELECT p.flying_license AS 'flying license', COUNT(f.pilotID) AS '# times she/he was pilot',
p.grade, p.pilotID
FROM pilot AS p JOIN flight AS f ON p.pilotID = f.pilotID
WHERE p.grade - (SELECT AVG(grade)
                  FROM pilot) > 2
GROUP BY f.pilotID
HAVING COUNT(f.pilotID) < (SELECT COUNT(p2.copilotID)
                           FROM pilot AS p2
                           WHERE f.pilotID = p2.copilotID)
ORDER BY COUNT(f.pilotID) DESC;

```

flying license	# times she/he was pilot	grade	pilotID
90-984-9951	2	9.46	136388
82-324-0620	2	9.56	135813
15-741-5329	2	9.68	133771
05-564-1878	2	9.6	135499
25-504-6465	1	9.76	131695
54-988-7758	1	9.52	134864
22-707-0728	1	9.54	134751
92-674-3016	1	9.5	133922
60-173-1960	1	9.98	136024
65-983-7024	1	9.42	134927
87-994-3423	1	9.64	136339
08-440-1486	1	9.4	131719
73-636-9959	1	9.56	134363
82-958-9279	1	9.62	132047

Abans de res, copiem la mateixa query tal qual però afegint-li una columna pel pilotID, ja que per les següents querys ens interessa saber el ids dels diferents pilots que compleixen les condicions especificades a l'enunciat.

```
SELECT AVG(p.grade)
FROM pilot AS p;
```

AVG(p.grade)
7.399678428592708

Fem aquesta query per calcular el valor numèric exacte de la mitja dels pilots, que veiem que dona 7.399. Per tant, segons ens diu l'enunciat els pilots que surten a la query principal han de tenir un grade superior a 9.399 ($7.399 + 2 = 9.399$). I, efectivament, si anem mirant pilot a pilot a la primera captura, veiem que tots tenen un grade superior a 9.399.

```
SELECT f.pilotID, COUNT(f.pilotID) AS 'vegades que ha estat pilot'
FROM flight AS f
WHERE f.pilotID = 136388;
```

pilotID	vegades que ha estat pilot
136388	2

En aquesta query, a la condició del where hem de posar els ids dels pilots que compleixen les condicions especificades a l'enunciat, es a dir els ids dels pilots que surten a la segona captura. En aquest cas, posem l'exemple amb el primer pilot que surt a la captura, que de id té 136388. Veiem que ha exercit dues vegades de pilot. A continuació veurem quantes vegades ha exercit de copilot.

```
SELECT p.pilotID, COUNT(p.copilotID) AS 'vegades que ha estat copilot'
FROM pilot AS p
WHERE p.copilotID = 136388;
```

pilotID	vegades que ha estat copilot
131678	3

Com podem veure, aquest pilot ha fet 3 vegades de copilot, per tant és correcte ja que ha exercit més vegades de copilot que no de pilot.

Aquestes dues ultimes querys les hauríem de repetir per cada pilotID que surten a la segona captura, i veuríem que tots compleixen les condicions requerides, nosaltres mateixos ens hem encarregat de comprovar-ho un a un.

2.1.4 Requirement (Query) 1.4 Very old passengers cannot communicate

2.1.4.1 Solution

```
SELECT DISTINCT pe.name AS 'passenger name', pe.surname, pe.born_date AS 'born date'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN languageperson AS lp ON pe.personID = lp.personID
```

```

JOIN language AS l ON lp.languageID = l.languageID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
WHERE NOT EXISTS (SELECT *
  FROM language AS l2 JOIN languageperson AS lp2 ON l2.languageID = lp2.languageID
  JOIN person AS pe2 ON lp2.personID = pe2.personID
  JOIN employee AS e ON pe2.personID = e.employeeID
  JOIN flight_attendant AS fa ON e.employeeID = fa.flightattendantID
  JOIN flight_flightattendant AS ffa ON fa.flightattendantID = ffa.flightAttendantID
  WHERE l.languageID = l2.languageID AND ft.flightID = ffa.flightID)
AND TIMESTAMPDIFF(YEAR, pe.born_date, NOW()) >= 100
ORDER BY pe.born_date DESC;

```

passenger name	surname	born date
Beatriz	England	1921-06-11
Cherish	Olivazzi	1921-06-11
Ashly	Zanitti	1921-06-10
Alyda	France	1921-06-07
Flo	Whawell	1921-06-06
Cirillo	Saladin	1921-06-05
Rosabel	Sadler	1921-06-05
Dredi	Minmagh	1921-06-04
Baxy	Filipovic	1921-06-03
Elwin	Canto	1921-06-03
Blondy	Seeler	1921-06-03
Tobin	Scowcraft	1921-06-02
Martica	Paddefield	1921-06-01
Nina	Britnell	1921-05-31
Koenraad	Jaffray	1921-05-31
Rosetta	Meaden	1921-05-30
Keenan	Cullum	1921-05-30
Jacky	Meijer	1921-05-28
Cobbie	Budik	1921-05-28
Gordon	Boorman	1921-05-27

2.1.4.2 Explanation

Abans de res fem el SELECT DISTINCT de la informació del passatgers que se'ns demana, el distinct es perquè ens surti 1 fila per passatger. Fem els JOINS de passatger amb les taules que ens interessen i al WHERE fem una SUBQUERY usant JOINS relacionats amb els flightattendants i usem el NOT EXISTS ja que no ens interessen mostrar passatgers que parlin el mateix idioma que els flightattendants, per tant a la condició del where de la subquery iguaem els languageID que parlen passatgers i flightattendants d'aquell mateix vol (iguaem el ID del vol del passatger amb el del flightattendant). Finalment a la query principal li afegim la condició de que els passatgers hagin nascut fa més de 100 anys. Fem servir la funció TIMESTAMPDIFF que ens calcula la diferència en anys entre la data actual i la data de naixement del passatger i retorna el resultat amb un número enter. En el moment de la captura la data actual era el 2021-06-12 i com hem ordenat la query per la data de naixement descendent, ens estalviem una query de validació per validar que tots els passatgers van néixer fa més de 100 anys. Finalment, afegir que al compilar la query ens surten 441 passatgers, hi ha tants ja que en molts casos hi ha vols en que no disposen de flightattendants però hem afegit igualment aquest passatgers, ja que tècnicament si no hi ha flightattendants no parlen cap idioma i per tant parlaran idiomes diferents segur.

2.1.4.3 Query validation

```
SELECT pe.name AS 'passenger name', pe.surname, pe.born_date AS 'born date', l.name,
ft.flightID
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN languageperson AS lp ON pe.personID = lp.personID
JOIN language AS l ON lp.languageID = l.languageID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
WHERE NOT EXISTS (SELECT * -- posem * perquè vaig llegir per internet que al NOT EXISTS
posavem * al SELECT
FROM language AS l2 JOIN languageperson AS lp2 ON l2.languageID = lp2.languageID
JOIN person AS pe2 ON lp2.personID = pe2.personID
JOIN employee AS e ON pe2.personID = e.employeeID
JOIN flight_attendant AS fa ON e.employeeID = fa.flightattendantID
JOIN flight_flightattendant AS ffa ON fa.flightattendantID = ffa.flightAttendantID
WHERE l.languageID = l2.languageID AND ft.flightID = ffa.flightID)
AND TIMESTAMPDIFF(YEAR, pe.born_date, NOW()) >= 100
ORDER BY ft.flightID DESC;
```

passenger name	surname	born date	name	flightID
Robinet	Styan	1921-05-26	English	20965
Rudiger	D'Almeida	1920-03-05	French	20714
Hewett	Wodham	1920-11-05	Spanish	20704
Christyna	Gairdner	1921-01-04	Mandarin	20446
Cherish	Olivazzi	1921-06-11	English	20440
Cherish	Olivazzi	1921-06-11	French	20440
Cherish	Olivazzi	1921-06-11	Spanish	20440
Raimund	Jowitt	1921-02-23	Mandarin	19982
Raimund	Jowitt	1921-02-23	Mandarin	19970
Oliviero	Abrashkin	1920-03-27	English	19964
Yehudi	Carrabot	1920-03-24	Albanian	19961
Starr	Farfull	1920-09-25	Chao-ch...	19954
Craig	Killimister	1921-01-28	Maltese	19953
Kalindi	Tilzey	1920-02-17	Indonesian	19952
Amalie	Ryan	1920-08-22	German	19946
Elliott	Girk	1920-06-08	Greek	19938
Cherri	Doeg	1920-03-03	Hungarian	19938
Cherri	Doeg	1920-03-03	Ukrainian	19938
Tildie	Kemmer	1920-05-28	German	19927

Primer copiem la mateixa query però traient el DISTINCT i afegint-li el nom del idioma i el flightID perquè volem mostrar tots els idiomes que parlen els passatgers juntament amb el seu flightID per comprovar mitjançant la següent query si algun d'aquests idiomes és parlat pels flightattendants d'aquell mateix vol. Ordenem pel flightID per saber tots els idiomes que són parlats pels passatgers per cada vol en concret.

```
SELECT fa.flightattendantID, l.name
FROM flight_attendant AS fa JOIN employee AS e ON fa.flightattendantID = e.employeeID
JOIN person AS pe ON e.employeeID = pe.personID
JOIN languageperson AS lp ON pe.personID = lp.personID
JOIN language AS l ON lp.languageID = l.languageID
JOIN flight_flightattendant AS ffa ON fa.flightattendantID = ffa.flightAttendantID
```


WHERE ffa.flightID = 19938;

flightID	flightattendantID	idioma/es parlats
19938	140337	English
19938	140337	Spanish
19938	147346	French
19938	147346	Mandarin

En aquesta query volem comprovar que per cada flightID que surt a la query anterior, en cap d'ells els passatgers parlin el mateix o mateixos idiomes que els flightattendants. Aleshores si anem a la query anterior i agafem per exemple el penúltim flightID que es el 19938, veiem que els passatgers d'aquest vol parlen Greek, Hungarian i Ukranian. I si per aquest flightID mirem els idiomes que parlen els flightattendants surten English, Spanish, French i Mandarin. Per tant, és correcte ja que no coincideix cap idioma. Per fer la validació completa hem de agafar cada flightID que surt a la primera query de validació i mirar que els idiomes que surtin que parlen els passatgers no coincideixin amb els idiomes que parlen els flightattendants.

2.1.5 Requirement (Query) 1.5 Window passengers

2.1.5.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.1.5.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.1.5.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.1.6 Requirement (Trigger) 1.6 Invalid tickets

2.1.6.1 Solution

TODO: Write the solution of the trigger including the creation of any table

2.1.6.2 Explanation

TODO: To explain how the trigger works using 5-10 sentences.

2.1.6.3 Trigger validation

TODO: Write the tests you have done to test that the trigger works: insert... update... delete... You can include results of any query.

2.1.7 Requirement (Trigger) 1.7 Credit card criminals

2.1.7.1 Solution

```
DROP TABLE IF EXISTS CrimeSuspect;
CREATE TABLE CrimeSuspect(
    passengerID BIGINT(20),
    name VARCHAR(255),
```

```

        surname VARCHAR(255),
        passport VARCHAR(11),
        phone VARCHAR(20),
        PRIMARY KEY (passengerID)
    );

DELIMITER $$
DROP TRIGGER IF EXISTS card_criminals $$
CREATE TRIGGER card_criminals
    AFTER INSERT
    ON passenger
    FOR EACH ROW
BEGIN

    IF (SELECT COUNT(*)
        FROM passenger AS pa
        WHERE pa.creditCard = NEW.creditCard) > 1 THEN

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

    END IF;

END $$

DELIMITER ;

```

2.1.7.2 *Explanation*

Primer de tot, creem la taula CrimeSuspect amb els atributs que se'ns especifica a l'enunciat. Seguidament, creem el TRIGGER i posem que sigui AFTER INSERT ON passenger ja que a l'enunciat ens diu que quan fem insert a passenger i la tarjeta de crèdit ja existeixi dins de la taula passenger, doncs inserir la informació del possible impostor a CrimeSuspect. Dins del trigger posem el condicional de que si a la taula passenger ja existeix almenys un passatger on la seva targeta de crèdit sigui la mateixa que la del passatger que acabem d'inserir, doncs inserim a la taula CrimeSuspect el ID d'aquest passatger, el seu nom, cognom, passaport i telèfon mòbil.

2.1.7.3 *Trigger validation*

Abans que res, mostrem la taula passenger ordenada descendentment pel ID del passatger.

```

SELECT * FROM passenger
ORDER BY passengerID DESC;

```


passengerID	creditCard
98112	676724230706410144
98111	3543138936988290
98110	3580162461718405
98109	3560317575037571
98108	6376503437891431
98107	3574490493777877
98106	4508149907395044
98105	4026073742696734
98104	3557207029561190
98103	3573139056129791
98102	374288011914099
98101	337941499683463

Comprovem que la taula CrimeSuspect està buida:

```
SELECT * FROM crimesuspect
```

passengerID	name	surname	passport	phone
NULL	NULL	NULL	NULL	NULL

Inserim noves persones (a la taula person) per tal de fer les comprovacions a posteriori:

```
INSERT INTO person(personID, name, surname, countryID, passport, email, phone_number,
born_date, sex)
VALUES(98113, "Lluis", "Gumbau", 71, "821-29-2356", "lluis.gumbau@students.salle.url.edu",
"+34 673 956 294", "1999-05-26", "M");
```

```
INSERT INTO person(personID, name, surname, countryID, passport, email, phone_number,
born_date, sex)
VALUES(98114, "Narcís", "Cisquella", 71, "751-34-7321",
"narcis.cisquella@students.salle.url.edu", "+34 623 562 211", "1999-02-09", "M");
```

```
INSERT INTO person(personID, name, surname, countryID, passport, email, phone_number,
born_date, sex)
VALUES(98115, "Joan", "Llobet", 71, "341-35-5855", "joan.llobet@students.salle.url.edu", "+34
622 344 788", "1999-06-13", "M");
```

```
INSERT INTO person(personID, name, surname, countryID, passport, email, phone_number,
born_date, sex)
VALUES(98116, "Marc", "Postils", 71, "441-86-6577", "marc.postils@students.salle.url.edu", "+34
673 040 646", "1999-03-27", "M");
```

Així es veu la taula person:

```
SELECT * FROM person
WHERE personID BETWEEN 98100 AND 98200
ORDER BY personID DESC;
```

personID	name	surname	countryID	passport	email	phone_number	born_date	sex
98116	Marc	Postils	71	441-86-6577	marc.postils@students.salle.url.edu	+34 673 040 646	1999-03-27	M
98115	Joan	Llobet	71	341-35-5855	joan.llobet@students.salle.url.edu	+34 622 344 788	1999-06-13	M
98114	Narcís	Cisquella	71	751-34-7321	narcis.cisquella@students.salle.url.edu	+34 623 562 211	1999-02-09	M
98113	Lluís	Gumbau	71	821-29-2356	lluis.gumbau@students.salle.url.edu	+34 673 956 294	1999-05-26	M
98112	Florrie	Piperley	210	215-03-3936	florrie.piperley@youtube.com	+62 175 343 6035	1956-01-16	F
98111	Patrizius	McCaughen	4	398-42-2405	patrizius.mccaughen@dagondesign.com	+505 563 122 2901	2008-04-08	M
98110	Jodi	Reynault	192	801-28-5330	jodi.reynault@slideshare.net	+63 435 309 0506	1978-12-21	F
98109	Mayer	Phillipson	58	110-84-4018	mayer.phillipson@unicef.org	+93 885 841 3192	2000-05-16	M
98108	Corty	Holehouse	15	237-12-6408	corty.holehouse@livejournal.com	+504 636 301 8139	1971-07-17	M
98107	Haleigh	Goscomb	151	858-80-3582	haleigh.goscomb@pen.io	+381 652 961 1160	1951-02-17	F
98106	Murry	MacPeake	71	713-92-7102	murphy.macpeake@japanpost.jp	+86 524 735 8902	1994-02-11	M
98105	Randy	Van den V...	23	321-20-4100	randy.van den velden@spotify.com	+54 443 775 1862	1992-11-10	F
98104	Paul	Lartice	151	216-04-5478	paul.lartice@census.gov	+994 690 869 7813	1950-08-16	M
98103	D'arcy	Elsy	192	861-60-0611	d'arcy.elsy@linkedin.com	+389 198 583 9165	1982-02-28	M

Comencem amb els INSERTS a la taula passanger:

```
INSERT INTO passenger(passengerID, creditCard)
VALUES (98113, 3543138936988290);
```

Per al passatger amb ID = 98113 (el que hem ensenyat que hem inserit) li hem posat una targeta de crèdit que ja existia a la taula passanger, és la mateixa targeta de crèdit del passatger amb ID = 98111, com es pot veure a la captura feta anteriorment de la taula passanger. Per tant, com un altre passatger ja ha utilitzat aquesta targeta de crèdit, aquest passatger que acabem d'introduir ha d'anar directament a la taula CrimeSuspect.

```
SELECT * FROM crimesuspect;
```

passengerID	name	surname	passport	phone
98113	Lluís	Gumbau	821-29-2356	+34 673 956 294
NULL	NULL	NULL	NULL	NULL

Veiem que la seva informació coincideix perfectament amb el passatger que hem inserit amb ID = 98113 i també veiem com s'ha afegit correctament a la taula CrimeSuspect.

Fem un insert a la taula passatger de la persona amb ID = 98114 i li posem una targeta de crèdit nova, és a dir, que no està registrada a la taula passanger per cap passatger.

```
INSERT INTO passenger(passengerID, creditCard)
VALUES (98114, 8543138933982290);
```

D'aquesta manera, aquest passatger no s'hauria d'introduir a la taula CrimeSuspect. Comprovem-ho:

```
SELECT * FROM crimesuspect;
```

passengerID	name	surname	passport	phone
98113	Lluís	Gumbau	821-29-2356	+34 673 956 294
NULL	NULL	NULL	NULL	NULL

Fem un insert a la taula passanger de la persona amb ID = 98115 i li posem una targeta de crèdit que ja existeix dins de la taula passanger, de fet és la mateixa targeta de crèdit que la del passatger anterior que tenia un ID = 98114.

```
INSERT INTO passenger(passengerID, creditCard)
VALUES (98115, 8543138933982290);
```

Aleshores, aquest nou passatger s'hauria d'introduir a la taula CrimeSuspect.

```
SELECT * FROM crimesuspect;
```

passengerID	name	surname	passport	phone
98113	Lluís	Gumbau	821-29-2356	+34 673 956 294
98115	Joan	Llobet	341-35-5855	+34 622 344 788
NULL	NULL	NULL	NULL	NULL

S'ha introduït correctament i les dades del passatger coincideixen.

Per últim fem un insert d'un altre passatger que utilitza la mateixa targeta de crèdit que els dos anteriors i, per tant, també s'ha d'introduir a la taula CrimeSuspect.

```
INSERT INTO passenger(passengerID, creditCard)
VALUES (98116, 8543138933982290);
```

```
SELECT * FROM crimesuspect;
```

passengerID	name	surname	passport	phone
98113	Lluís	Gumbau	821-29-2356	+34 673 956 294
98115	Joan	Llobet	341-35-5855	+34 622 344 788
98116	Marc	Postils	441-86-6577	+34 673 040 646
NULL	NULL	NULL	NULL	NULL

S'ha introduït correctament i les dades del passatger coincideixen.

2.1.8 Requirement (Trigger) 1.8 Cancelled flights

2.1.8.1 Solution

TODO: Write the solution of the trigger including the creation of any table

2.1.8.2 Explanation

TODO: To explain how the trigger works using 5-10 sentences.

2.1.8.3 Trigger validation

TODO: Write the tests you have done to test that the trigger works: insert... update... delete... You can include results of any query.

2.1.9 Requirement (Event) 1.9 Flight statistics

2.1.9.1 Solution

```
DROP TABLE IF EXISTS DailyFlights;
CREATE TABLE DailyFlights(
    date DATE,
    number_of_flights INTEGER,
    PRIMARY KEY (date)
);
```

```

DROP TABLE IF EXISTS MonthlyFlights;
CREATE TABLE MonthlyFlights(
    average_of_flights FLOAT,
    month DATE,
    PRIMARY KEY (month)
);

DELIMITER $$
DROP PROCEDURE IF EXISTS flight_check $$
CREATE PROCEDURE flight_check()
BEGIN
    INSERT INTO DailyFlights(date, number_of_flights)
    SELECT CURRENT_DATE(), COUNT(*)
    FROM flight
    WHERE date = CURRENT_DATE();

    INSERT INTO MonthlyFlights(average_of_flights, month)
    SELECT AVG(number_of_flights),
        STR_TO_DATE(CONCAT(YEAR(CURRENT_DATE()), '-',
        MONTH(CURRENT_DATE()), '-', '1'), '%Y-%m-%d')
    FROM DailyFlights
    WHERE MONTH(CURRENT_DATE()) = MONTH(date) AND
    YEAR(CURRENT_DATE()) = YEAR(date)
    GROUP BY month(date)
    ON DUPLICATE KEY UPDATE average_of_flights = (SELECT
AVG(number_of_flights)
FROM DailyFlights
WHERE
month(CURRENT_DATE()) = month(date) AND year(CURRENT_DATE()) = year(date)
GROUP BY
month(date));
END $$

DELIMITER ;
DROP EVENT IF EXISTS flight_statistics;
CREATE EVENT flight_statistics
ON SCHEDULE EVERY 24 HOUR
STARTS '2021-01-01 23:59:59'
ON COMPLETION PRESERVE
DO CALL
    flight_check();

```

2.1.9.2 Explanation

En primer lloc creem les dues taules que ens demanen, DailyFlights i MonthlyFlights. Un cop creades, creem un STORED PROCEDURE, que serà l'encarregat d'insertar la informació corresponent a DailyFlights i a MonthlyFlights. A la taula DailyFlights insertarem la data actual, que és la data on cada dia insertarem nous vols (fent INSERTS a la taula flight) i el número de vols que hi ha hagut aquell dia, d'aquesta manera fem un COUNT del número de files que hi ha a la taula flight just aquell dia. A la taula MonthlyFlights, a mesura que es van afegint vols diàriament a DailyFlights, doncs inserim la mitja del número de vols de cada mes que hi ha a la taula DailyFlights. Per tant, a

MonthlyFlights fem el AVG del número de vols de cada mes junt amb la data del mes, per exemple pel mes de juny d'aquest any li hem posat '2021-06-1', posem el dia 1 com podríem haver posat el dia XX (referint-nos a que engloba al mes de juny). Posem la línia de ON DUPLICATE KEY UPDATE average_of_flights ja que com cada dia insertem nous vols, doncs la mitja de vols va variant i s'ha d'anar actualitzant. Finalment, creem l'EVENT per a que s'actualitzi diàriament i cridem al stored procedure.

2.1.9.3 Trigger validation

```
INSERT INTO flight(flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)
VALUES (21001, 136372, 2245, 1324, DATE(NOW()), 'I5', 2424, '00:23:00', 4);
INSERT INTO flight(flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)
VALUES (21002, 136372, 2245, 1324, DATE(NOW()), 'I5', 2424, '00:23:00', 4);
INSERT INTO flight(flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)
VALUES (21003, 136372, 2245, 1324, DATE(NOW()), 'I5', 2424, '00:23:00', 4);
INSERT INTO flight(flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)
VALUES (21004, 136372, 2245, 1324, DATE(NOW()), 'I5', 2424, '00:23:00', 4);
INSERT INTO flight(flightID, pilotID, planeID, routeID, date, gate, fuel, departure_hour, statusID)
VALUES (21005, 136372, 2245, 1324, DATE(NOW()), 'I5', 2424, '00:23:00', 4);
```

Fem els INSERTS de diferents vols, el primer el vam fer el dia 17-06-2021, el segon el 19-06-2021, i els tres últims el dia 21-06-2021.

Aleshores, a la taula DailyFlights surt la data en que vam fer el insert i el número de vols fets aquell dia, per tant, amb els inserts que hem posat anteriorment s'omple d'aquesta manera la taula DailyFlights:

date	number_of_flights
2021-06-17	1
2021-06-19	1
2021-06-21	3
NULL	NULL

I a la taula MonthlyFlights mostrarà la mitja de vols d'aquell mes i el mes en qüestió. Per tant, en aquest cas que en el mes de juny hi ha hagut 5 vols en 3 dies diferents, doncs la mitja de vols es calcula dividint número de vols/dies, per tant, $5/3 = 1,6$ periòdic. Com pel dia 18-06-2021 i el 20-06-2021 no vam executar manualment l'event no surt que el number_of_flights és 0, però si un dia no inserim cap vol a flight i executem l'event doncs si que surt el number_of_flights = 0 per aquell dia

average_of_flights	month
1.66667	2021-06-01
NULL	NULL

2.2 Airport module

2.2.1 Requirement (Query) 2.1 Airlines and petrol capacity

2.2.1.1 Solution

```
SELECT al.name AS 'airline name', COUNT(r.routeID) AS '# routes'
FROM airline AS al JOIN routeairline AS ral ON ral.airlineID = al.airlineID
JOIN route AS r ON r.routeID = ral.routeID
JOIN airport AS ap ON ap.airportID = r.departure_airportID
JOIN planetype AS pt ON pt.planetypeID = ral.planetypeID
```

```

JOIN city AS c ON ap.cityID = c.cityID
JOIN country AS cy ON cy.countryID = c.countryID
JOIN airport AS a2 ON r.destination_airportID = a2.airportID
JOIN city AS c2 ON a2.cityID = c2.cityID
JOIN country AS cy2 ON c2.countryID = cy2.countryID
WHERE r.minimum_petrol > pt.petrol_capacity
AND cy.countryID <> cy2.countryID
GROUP BY al.airlineID
ORDER BY COUNT(r.routeID) DESC;

```

airline name	# routes
Delta Air Lines	99
United Airlines	85
Air France	83
Lufthansa	76
Lufthansa Cargo	76
US Airways	70
American Airlines	67
Etihad Airways	55
Qatar Airways	53
Alitalia	49
TAP Portugal	48
Air Berlin	45
Singapore Airlines	42
Singapore Airlin...	42
Air Canada	42
Qantas	42
China Eastern A...	41
Aeroflot Russia...	38
China Southern ...	36

2.2.1.2 Explanation

Fem el SELECT de les dades que ens demanen i fem els JOINS de les taules necessàries, inicialment vam relacionar les taules planetype amb plane i plane amb airline i després amb les rutes però llegint l'enunciat ens vam donar compte que plantype l'havíem de relacionar amb routeairline ja que ens diuen que hem de mirar la capacitat del aircraft assignada a la ruta. Al WHERE, posem la condició de que el petroli mínim necessari (minimum_petrol) per fer la ruta sigui superior a la capacitat de petroli (petrol_capacity) del aircraft assignat per aquesta ruta i també fem que el país origen de la ruta sigui diferent al país destí d'aquella ruta. Agrupem pel ID de la aerolínea i ordenem pel número de rutes de major a menor de cada aerolínea que compleixen les condicions.

2.2.1.3 Query validation

```

SELECT al.name AS 'airline name', COUNT(r.routeID) AS '# routes', al.airlineID
FROM airline AS al JOIN routeairline AS ral ON ral.airlineID = al.airlineID
JOIN route AS r ON r.routeID = ral.routeID
JOIN airport AS ap ON ap.airportID = r.departure_airportID
JOIN planetype AS pt ON pt.planetypeID = ral.planetypeID
JOIN city AS c ON ap.cityID = c.cityID
JOIN country AS cy ON cy.countryID = c.countryID
JOIN airport AS a2 ON r.destination_airportID = a2.airportID
JOIN city AS c2 ON a2.cityID = c2.cityID

```

```

JOIN country AS cy2 ON c2.countryID = cy2.countryID
WHERE r.minimum_petrol > pt.petrol_capacity
AND cy.countryID <> cy2.countryID
GROUP BY al.airlineID
ORDER BY COUNT(r.routeID) DESC;

```

airline name	# routes	airlineID
Delta Air Lines	99	2009
United Airlines	85	5209
Air France	83	137
Lufthansa	76	3320
Lufthansa Cargo	76	3321
US Airways	70	5265
American Airlines	67	24
Etihad Airways	55	2222
Qatar Airways	53	4091
Alitalia	49	596
TAP Portugal	48	4869
Air Berlin	45	214
Air Canada	42	330
Singapore Airlines	42	4435
Singapore Airline	42	4464
Qantas	42	4089
China Eastern Airlines	41	1758
Aeroflot Russia	38	130
China Southern Airlines	36	1767

Copiem la query principal però afegint-li al select al airlineID per comprovar amb la següent query que compleix les condicions.

```

SELECT al.name AS 'airline name', al.airlineID, r.routeID, r.minimum_petrol, pt.petrol_capacity,
(r.minimum_petrol - pt.petrol_capacity), cy.name AS 'país aeroport origen', cy2.name AS 'país
aeroport destí'
FROM airline AS al JOIN routeairline AS ral ON ral.airlineID = al.airlineID
JOIN route AS r ON r.routeID = ral.routeID
JOIN airport AS ap ON ap.airportID = r.departure_airportID
JOIN planetype AS pt ON pt.planetypeID = ral.planetypeID
JOIN city AS c ON ap.cityID = c.cityID
JOIN country AS cy ON cy.countryID = c.countryID
JOIN airport AS a2 ON r.destination_airportID = a2.airportID
JOIN city AS c2 ON a2.cityID = c2.cityID
JOIN country AS cy2 ON c2.countryID = cy2.countryID
WHERE r.minimum_petrol > pt.petrol_capacity
AND cy.countryID <> cy2.countryID
AND al.airlineID = 2009
ORDER BY (r.minimum_petrol - pt.petrol_capacity) ASC;

```


airline name	airlineID	routeID	minimum_petrol	petrol_capacity	(r.minimum_petrol - pt.petrol_capacity)	país aeroport origen	país aeroport destí
Delta Air Lines	2009	24080	23148	23112	36	Australia	Nepal
Delta Air Lines	2009	5451	7385	6723	662	Kuwait	Saudi Arabia
Delta Air Lines	2009	12781	23912	23112	800	United States	Spain
Delta Air Lines	2009	11820	24721	23490	1231	United States	Mexico
Delta Air Lines	2009	13176	27349	26100	1249	Australia	Mexico
Delta Air Lines	2009	5057	27776	26100	1676	Mexico	Australia
Delta Air Lines	2009	11049	134267	131544	2723	United States	Japan
Delta Air Lines	2009	21732	25920	23112	2808	Nepal	Australia
Delta Air Lines	2009	5054	29111	26100	3011	Mexico	United States
Delta Air Lines	2009	23589	29338	26100	3238	United States	Jamaica
Delta Air Lines	2009	14610	10051	6723	3328	Canada	Netherlands
Delta Air Lines	2009	24453	26983	23112	3871	Australia	Netherlands
Delta Air Lines	2009	12847	45731	41769	3962	United States	Italy
Delta Air Lines	2009	858	10780	6723	4057	Burkina Faso	Niger
Delta Air Lines	2009	12123	15449	11340	4109	United States	Australia
Delta Air Lines	2009	15664	10959	6723	4236	Netherlands	Canada
Delta Air Lines	2009	2902	11010	6723	4287	Rwanda	Uganda
Delta Air Lines	2009	5365	12298	6723	5575	Bahrain	United Arab Emira...
Delta Air Lines	2009	12164	29460	23490	5970	United States	Mexico
Delta Air Lines	2009	11538	55061	40455	6006	United States	Italy

77 12:17:16 SELECT airline AS 'airline name', 99 rows returned

0.032 sec / 0.000 sec

Aquesta query és completa ja que validem que compleix totes les condicions. Hem agafat de referencia el airlineID = 2009, que és de la aerolínia Delta Air Airlines, que és el primer resultat de la query principal. Aleshores, hem seleccionat totes les rutes d'aquesta aerolínia que compleixen les condicions especificades i, efectivament, les compleix. Primer de tot, si ens fixem en el número de rows o files que ens han sortit (2^a captura) posa que hi ha 99 rows, que són el número de rutes que hi ha i que coincideixen amb el '# routes' de la primera aerolínia. En segon lloc, si ens fixem en la columna de (r.minimum_petrol - pt.petrol_capacity), el primer número que surt és el 36 i és el número més petits de tots ja que hem ordenat per aquest número de menor a major. Això significa que tots els números d'aquesta columna són positius i, per tant, compleixen la condició de que (mínimum_petrol - petrol_capacity > 0). Per últim, totes les rutes també compleixen que el país de l'aeroport origen i destí siguin diferents, fent scroll down podem veure que en cap ruta coincideixen.

Hauríem d'anar mirant per cada aerolínia que ens surt a la query i posar-lo a la query de validació i comprovaríem que tots els resultats coincideixen.

2.2.2 Requirement (Query) 2.2 Mechanic grades

2.2.2.1 Solution

```

SELECT CONCAT(CAST(FLOOR(m.grade) AS CHAR(2)), '-', CAST(FLOOR(m.grade)+1 AS
CHAR(2))) AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
where (SELECT COUNT(pm2.pieceID)
FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON pm2.maintenanceID =
m2.maintenanceID
WHERE pm.maintenanceID= pm2.maintenanceID
GROUP BY m2.maintenanceID
LIMIT 1) < 10
GROUP BY FLOOR(m.grade);

```


grade range	duration average
0-1	47.0667
1-2	44.2867
2-3	44.6220
3-4	37.9719
4-5	48.8378
5-6	40.6696
6-7	46.4852
7-8	40.4459
8-9	41.3014
9-10	41.1959

2.2.2.2 Explanation

Comencem seleccionant els rangs de les puntuacions, això ho fem amb el CONCAT, el qual agrupa les strings que introduïm. El FLOOR trunca el resultat, és a dir arrodoneix cap abaix a l'enter més proper. Només tenim en compte aquells manteniments que s'han reemplaçat menys de 10 peces. Finalment mostrem el rang de la puntuació i la mitja de totes les puntuacions dins del rang corresponent.

2.2.2.3 Query validation

A continuació us adjuntem una query on adjuntem totes les possibilitats en una sola query, així podem comprovar que el concat funciona a la perfecció, i si executem query a query, és a dir la query que compleix cada rang, podem comprovar que els resultats coincideixin. Així que comprovem dos dels aspectes més rellevants de la query principal.

```

SELECT '0-1' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 0 AND 1
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '1-2' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 1 AND 2
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '2-3' AS 'grade range', AVG(mt.duration) AS 'duration average'

```

```

FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 2 AND 3
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '3-4' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 3 AND 4
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '4-5' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 4 AND 5
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '5-6' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 5 AND 6
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '6-7' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID

```

```

WHERE m.grade BETWEEN 6 AND 7
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '7-8' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 7 AND 8
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '8-9' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 8 AND 9
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1) -- peces utilitzades en el manteniment (mt.maintenanceID)
UNION
SELECT '9-10' AS 'grade range', AVG(mt.duration) AS 'duration average'
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID
JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 9 AND 10
AND 10 > (SELECT COUNT(pm2.pieceID)
          FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON
pm2.maintenanceID = m2.maintenanceID
          WHERE pm.maintenanceID= pm2.maintenanceID
          GROUP BY m2.maintenanceID
          LIMIT 1);

```

A més a més, amb aquesta query comprovem que cada manteniment té menys de 10 peces canviades. Si traiem la condició del where de la subquery de les 10 peces, podem observar com apareixen més manteniments amb més de 10 peces.

```

SELECT '0-1' AS 'grade range', AVG(mt.duration) AS 'duration average', COUNT(pm.pieceID)
FROM mechanic AS m JOIN maintenance AS mt ON mt.mechanicID = m.mechanicID
JOIN piecemaintenance AS pm ON pm.maintenanceID = mt.maintenanceID

```

```

JOIN piece AS p ON p.pieceID = pm.pieceID
WHERE m.grade BETWEEN 0 AND 1
AND 10 > (SELECT COUNT(pm2.pieceID)
         FROM maintenance AS m2 JOIN piecemaintenance AS pm2 ON pm2.maintenanceID =
m2.maintenanceID
         WHERE pm.maintenanceID= pm2.maintenanceID
         GROUP BY m2.maintenanceID
         LIMIT 1)
GROUP BY mt.maintenanceID;

```

2.2.3 Requirement (Query) 2.3 Airports and mean distance routes

2.2.3.1 Solution

```

SELECT a.airportID AS 'airport id', cy.countryID AS 'country id', AVG(r.distance) AS 'average
distance'
FROM airport AS a JOIN route AS r ON r.departure_airportID = a.airportID
JOIN city AS c ON c.cityID = a.cityID
JOIN country AS cy ON cy.countryID = c.countryID
GROUP BY a.airportID
HAVING AVG(r.distance) > (SELECT AVG(r2.distance)
FROM airport AS a2 JOIN route AS r2 ON r2.departure_airportID = a2.airportID
JOIN city AS c2 ON c2.cityID = a2.cityID
JOIN country AS cy2 ON cy2.countryID = c2.countryID
WHERE cy.countryID = cy2.countryID);

```

	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

2.2.3.2 Explanation

Primer fem un SELECT dels atributs que se'ns indica a l'enunciat, seleccionem i agrupem les taules (JOINS) que necessitem i afegim les condicions per filtrar els resultats. La SUBQUERY és l'encarregada de complir la condició de que la distància mitja de rutes que surten d'un aeroport sigui superior a la distància mitja de rutes que surten dels aeroports que són del mateix país que el seleccionat anteriorment, per això a la subquery afegim la condició de que comparteixin país.

2.2.3.3 Query validation

```

SELECT a.airportID, cy.countryID, AVG(distance)
FROM airport AS a JOIN route AS r ON r.departure_airportID = a.airportID

```

```

JOIN city AS c ON c.cityID = a.cityID
JOIN country AS cy ON cy.countryID = c.countryID
WHERE a.airportID = 5;

```

airportID	countryID	AVG(distance)
5	1	1422.4063

En aquesta query comprovem que efectivament la distancia mitja de les routes que surten del primer aeroport que ens surt a la query principal (airportID = 5) és de 14422,40.

```

SELECT cy2.countryID, AVG(r2.distance)
FROM airport AS a2 JOIN route AS r2 ON r2.departure_airportID = a2.airportID
JOIN city AS c2 ON c2.cityID = a2.cityID
JOIN country AS cy2 ON cy2.countryID = c2.countryID
WHERE cy2.countryID = 1;

```

countryID	AVG(r2.distance)
1	1057.4850

Aquí comprovem que agafant el countryID del aeroport que ens ha sortit abans (airportID = 5) i calculant la distancia mitja de les rutes que surten del país d'aquest aeroport, veiem que efectivament la distancia és inferior ($1422.40 > 1057.48$).

Seleccionem l'airportID i el countryID que ens interessa (els que surten a la captura), i anem comprovant que els resultats siguin els esperats.

2.2.4 Requirement (Query) 2.4 Airlines and routes

2.2.4.1 Solution

```

SELECT a.name AS "airline name", a.airlineID AS "airline id", cy.name AS "country name",
MAX(r.time) AS "longest route duration"
FROM airline AS a JOIN routeairline AS ra ON a.airlineID = ra.airlineID
JOIN route AS r ON ra.routeID = r.routeID
JOIN country AS cy ON cy.countryID = a.countryID
WHERE a.active = 'Y'
AND cy.countryID NOT IN (SELECT DISTINCT cy2.countryID
                        FROM route AS r2 JOIN airport AS a2 ON r2.destination_airportID
                        = a2.airportID
                        JOIN city AS c2 ON c2.cityID = a2.cityID
                        JOIN country AS cy2 ON cy2.countryID = c2.countryID
                        JOIN routeairline AS ra2 ON ra2.routeID = r2.routeID
                        WHERE a.airlineID = ra2.airlineID)
AND cy.countryID NOT IN (SELECT DISTINCT cy3.countryID
                        FROM route AS r3 JOIN airport AS a3 ON r3.departure_airportID =
a3.airportID
                        JOIN city AS c3 ON c3.cityID = a3.cityID
                        JOIN country AS cy3 ON cy3.countryID = c3.countryID
                        JOIN routeairline AS ra3 ON ra3.routeID = r3.routeID
                        WHERE a.airlineID = ra3.airlineID)
GROUP BY a.airlineID

```

ORDER BY MAX(r.time) DESC;

airline name	airline id	country name	longest route duration
Qatar Airways	4091	Qatar	14:29:00
Korean Air	3163	Republic of Korea	13:17:00
Asiana Airlines	28	Republic of Korea	12:46:00
Singapore Airlines	4435	Singapore	12:33:00
Singapore Airlines Cargo	4464	Singapore	12:33:00
Compagnie Africaine d'Aviation	11806	Congo (Kinshasa)	10:07:00
Astrakhan Airlines	462	Russia	10:03:00
LAN Argentina	3201	Argentina	09:01:00
Indigo	2855	United States	08:50:00
Transavia France	8745	France	08:44:00
Sat Airlines	4808	Kazakhstan	08:33:00
Royal Brunei Airlines	4255	Brunei	08:27:00
Air Namibia	153	Namibia	07:52:00
City Connexion Airlines	1790	Burundi	07:40:00
Sky Express	4374	Greece	07:40:00
Scoot	17891	Singapore	07:32:00
AirAsia X	2417	Malaysia	07:32:00
Yemenia	5496	Yemen	07:31:00
Ietstar Asia Airways	3021	Singapore	07:11:00

2.2.4.2 Explanation

Fem el SELECT de les dades que ens demanen, i els JOINS necessaris i posem la condició al WHERE de que les aerolínies siguin actives (a.active = 'Y'). Fem dues SUBQUERIES, la primera per imposar que el país de la aerolínia sigui diferent al país del aeroport destí de totes les rutes que tingui la aerolínia. La segona per imposar que el país de la aerolínia sigui diferent al país del aeroport origen de totes les rutes que tingui la aerolínia. Hem utilitzat el NOT IN en ambdós casos ja que no volem que el país de la aerolínia estigui dins del llistat dels països dels aeroports destí i orígens d'aquella aerolínia. Agrupem pel ID de l'aerolínia i ordenem els resultats de major a menor duració de la ruta més llarga de cada aerolínia.

2.2.4.3 Query validation

SELECT al.name AS 'airline name', al.airlineID, al.active AS "activa?", r.routeID, r.time AS 'duració', cy3.name AS 'pais aerolinia', cy.name AS 'país aeroport origen', cy2.name AS 'país aeroport destí'

FROM airline AS al JOIN routeairline AS ral ON ral.airlineID = al.airlineID

JOIN route AS r ON r.routeID = ral.routeID

JOIN airport AS ap ON ap.airportID = r.departure_airportID

JOIN city AS c ON ap.cityID = c.cityID

JOIN country AS cy ON cy.countryID = c.countryID

JOIN airport AS ap2 ON ap2.airportID = r.destination_airportID

JOIN city AS c2 ON ap2.cityID = c2.cityID

JOIN country AS cy2 ON cy2.countryID = c2.countryID

JOIN country AS cy3 ON al.countryID = cy3.countryID

WHERE al.airlineID = 4091

GROUP BY r.routeID DESC

ORDER BY r.time DESC;

airline name	airlineID	activa?	routeID	duració	país aerolínia	país aeroport origen	país aeroport destí
Qatar Airways	4091	Y	31335	14:29:00	Qatar	Spain	Philippines
Qatar Airways	4091	Y	9275	11:45:00	Qatar	Philippines	United States
Qatar Airways	4091	Y	31292	11:31:00	Qatar	Japan	Philippines
Qatar Airways	4091	Y	13000	11:18:00	Qatar	United States	Philippines
Qatar Airways	4091	Y	32465	11:02:00	Qatar	Philippines	Canada
Qatar Airways	4091	Y	32453	10:35:00	Qatar	Philippines	Spain
Qatar Airways	4091	Y	9273	09:55:00	Qatar	Philippines	United States
Qatar Airways	4091	Y	22624	09:25:00	Qatar	Philippines	Japan
Qatar Airways	4091	Y	12795	08:54:00	Qatar	United States	Philippines
Qatar Airways	4091	Y	32422	08:27:00	Qatar	Philippines	Brazil
Qatar Airways	4091	Y	32410	07:54:00	Qatar	Philippines	Turkey
Qatar Airways	4091	Y	32803	07:18:00	Qatar	United States	Philippines
Qatar Airways	4091	Y	32431	07:13:00	Qatar	Philippines	South Africa
Qatar Airways	4091	Y	32405	07:11:00	Qatar	Philippines	Indonesia
Qatar Airways	4091	Y	9276	07:07:00	Qatar	Philippines	United States
Qatar Airways	4091	Y	9272	06:58:00	Qatar	Philippines	Yemen
Qatar Airways	4091	Y	3987	06:45:00	Qatar	Italy	United States
Qatar Airways	4091	Y	32402	06:38:00	Qatar	Philippines	United States
Qatar Airways	4091	Y	20075	06:35:00	Qatar	Japan	Philippines
Qatar Airways	4091	Y	32322	06:33:00	Qatar	South Africa	Philippines

Aquesta query mostra, d'una aerolínia en concret, en aquest cas la aerolínia amb ID = 4091, que es la que té la ruta més llarga i, per tant, es el primer resultat de la query principal com es pot veure, mostra el país origen de la aerolínia, si és activa o no, i també totes les rutes d'aquella aerolínia junt amb el país origen i destí de cada ruta. Està ordenada per la duració de les rutes de major a menor. Si mirem el resultat de la query (la captura), podem comprovar que la aerolínia és activa (columna de "activa?" és igual a "Y", que significa yes), que la ruta més llarga dura 14 hores i 29 minuts (ja que esta ordenat per la duració de major a menor) i coincideix amb el resultat de la query principal i finalment també comprovem que el país de la aerolínia es diferent del país del aeroport destí i origen. Fent scroll down podem comprovar que en cap cas el país origen o destí es de Qatar.

Una altra forma que hem fet per comprovar lo dels països més ràpidament sense haver de fer scroll down i anar mirant els resultats és afegint la condició de que el país de la aerolínia sigui igual o al país origen o al país destí de totes les rutes d'aquella aerolínia. D'aquesta manera forcem l'error i si executem la query ens surt que no hi ha cap ruta ja que significa que ho hem fet correctament. Aquesta query és la següent:

```

SELECT al.name AS 'airline name', al.airlineID, al.active AS "activa?", r.routeID, r.time AS 'duració', cy3.name AS 'país aerolínia', cy.name AS 'país aeroport origen', cy2.name AS 'país aeroport destí'
FROM airline AS al JOIN routeairline AS ral ON ral.airlineID = al.airlineID
JOIN route AS r ON r.routeID = ral.routeID
JOIN airport AS ap ON ap.airportID = r.departure_airportID
JOIN city AS c ON ap.cityID = c.cityID
JOIN country AS cy ON cy.countryID = c.countryID
JOIN airport AS ap2 ON ap2.airportID = r.destination_airportID
JOIN city AS c2 ON ap2.cityID = c2.cityID
JOIN country AS cy2 ON cy2.countryID = c2.countryID
JOIN country AS cy3 ON al.countryID = cy3.countryID
WHERE al.airlineID = 4091
AND (cy3.countryID = cy.countryID OR cy3.countryID = cy2.countryID)

```

GROUP BY r.routeID DESC
ORDER BY r.time DESC;

airline name	airlineID	activa?	routeID	duració	país aerolínia	país aeroport origen	país aeroport destí
-----------------	-----------	---------	---------	---------	-------------------	-------------------------	------------------------

Ho hem validat pel cas de la aerolínia amb ID = 4091, aleshores, si voleu validar-ho per les altres aerolínies que surten a la query principal, tan sols heu de canviar el airportID de les queries de validació pel que vulgueu mirar. I veureu que tots els resultats són correctes.

2.2.5 Requirement (Query) 2.5 Pieces replaced

2.2.5.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.2.5.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.2.5.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.2.6 Requirement (Trigger) 2.6 Routes cancelled

2.2.6.1 Solution

TODO: Write the solution of the trigger including the creation of any table

2.2.6.2 Explanation

TODO: To explain how the trigger works using 5-10 sentences.

2.2.6.3 Trigger validation

TODO: Write the tests you have done to test that the trigger works: insert... update... delete... You can include results of any query.

2.2.7 Requirement (Trigger) 2.7 Mechanics firings

2.2.7.1 Solution

TODO: Write the solution of the trigger including the creation of any table

2.2.7.2 Explanation

TODO: To explain how the trigger works using 5-10 sentences.

2.2.7.3 Trigger validation

TODO: Write the tests you have done to test that the trigger works: insert... update... delete... You can include results of any query.

2.2.8 Requirement (Trigger) 2.8 Petrol updates

2.2.8.1 Solution

```
DROP TABLE IF EXISTS EnvironmentalReductions;
CREATE TABLE EnvironmentalReductions(
```



```

        route VARCHAR(255),
        difference FLOAT,
        date DATE,
        PRIMARY KEY (route, date)
    );

DELIMITER $$
DROP TRIGGER IF EXISTS petrol_updates $$
CREATE TRIGGER petrol_updates
    AFTER UPDATE
    ON route
    FOR EACH ROW
BEGIN
    INSERT INTO EnvironmentalReductions(route, difference, date)
    SELECT CONCAT(departure_airportID, " ", destination_airportID), NEW.minimum_petroli
- OLD.minimum_petroli, CURRENT_DATE()
    FROM route
    WHERE routeID = NEW.routeID;
END $$

DELIMITER ;

```

2.2.8.2 *Explanation*

Primer de tot, creem la taula EnvironmentalReductions amb tots els atributs que se'ns requereix a l'enunciat. Seguidament, creem el TRIGGER i posem que sigui AFTER UPDATE ja que al enunciat ens diuen que volem tenir un historial de les vegades que s'actualitza la quantitat de petroli per realitzar una ruta, per tant, després de fer el update de la quantitat de petroli, afegirem aquest canvis a la taula EnvironmentalReduction. Posem ON route ja que la taula route es on hi ha la quantitat de petroli, els destination_airportID i departure_airportID, que són els atributs que ens interessin per a la taula EnvironmentalReduction. Dins del trigger farem el insert a la taula EnvironmentalReductions del departure_airportID i del destination_airportID d'aquella ruta, fent servir la funció CONCAT per unir els dos IDs ja que ens diuen que ha de ser un varchar amb els dos IDs junts. També hem d'insertar la diferencia del valor de la quantitat de petroli que posarem al UPDATE (NEW.minimum_petroli) i el valor de la quantitat de petroli que hi ha a la taula route abans de fer el update (OLD.minimum_petroli) i finalment la data en que es fa el update, que és la CURRENT_DATE(), que és la data actual. A la condició del where fem que sigui la mateixa ruta ja que volem modificar una ruta en concret, concretament la ruta que posem al update.

2.2.8.3 *Trigger validation*

```

SELECT *
FROM route
ORDER BY routeID DESC;

```

Abans de res, fem el select de la taula route, ordenat pel routeID descendentment.

routeID	destination_airportID	departure_airportID	distance	minimum_petrol	time
37016	4056	6980	714	6000	00:52:03
37015	5457	6944	910	3061	00:27:09
37014	5753	6631	345	8492	00:46:33
37013	5641	6631	538	1947	00:41:53
37012	5924	6631	149	9522	00:43:57
37011	5765	6631	935	16748	01:25:37
37010	5560	6507	875	6616	00:38:04

A continuació, creem la taula EnvironmentalReduction. Inicialment està buida.

```
SELECT *
FROM environmentalreductions
```

route	difference	date
NULL	NULL	NULL

Ara executem el trigger i fem els següents updates:

```
UPDATE route SET minimum_petrol = 5000
WHERE routeID = 37016;
```

```
UPDATE route SET minimum_petrol = 4050
WHERE routeID = 37015;
```

```
UPDATE route SET minimum_petrol = 10734
WHERE routeID = 37014;
```

Aleshores, fent el select de la taula route veiem que s'ha actualitzat amb els valors que li hem assignat.

```
SELECT *
FROM route
ORDER BY routeID DESC;
```

routeID	destination_airportID	departure_airportID	distance	minimum_petrol	time
37016	4056	6980	714	5000	00:52:03
37015	5457	6944	910	4050	00:27:09
37014	5753	6631	345	10734	00:46:33
37013	5641	6631	538	1947	00:41:53
37012	5924	6631	149	9522	00:43:57
37011	5765	6631	935	16748	01:25:37
37010	5560	6507	875	6616	00:38:04

I si fem el select de la taula EnvironmentalReduction, veiem que enregistra les actualitzacions que hem realitzat.

route	difference	date
6631 5753	2242	2021-06-21
6944 5457	989	2021-06-21
6980 4056	-1000	2021-06-21
NULL	NULL	NULL

La primera fila pertany a la routeID = 37014, que té el departure_airportID = 6631 (origen), destination_airportID = 5753 (destí), la diferència de la quantitat de petroli entre el nou valor que li hem assignat i el valor existent que hi havia a la taula route ($10734 - 8492 = 2242$). També guardem la data en que en el seu moment era la CURRENT_DATE(), la data en que vam fer el update (2021-06-21).

La segona fila pertany a la routeID = 37015, que té el departure_airportID = 6944 (origen), destination_airportID = 5457 (destí), la diferència de la quantitat de petroli entre el nou valor que li hem assignat i el valor existent que hi havia a la taula route ($4050 - 3061 = 989$). També guardem la data en que en el seu moment era la CURRENT_DATE(), la data en que vam fer el update (2021-06-21).

La tercera fila pertany a la routeID = 37016, que té el departure_airportID = 6980 (origen), destination_airportID = 4056 (destí), la diferència de la quantitat de petroli entre el nou valor que li hem assignat i el valor existent que hi havia a la taula route ($5000 - 6000 = -1000$). També guardem la data en que en el seu moment era la CURRENT_DATE(), la data en que vam fer el update (2021-06-21).

2.2.9 Requirement (Event) 2.9 Yearly maintenance costs

2.2.9.1 Solution

TODO: Write the solution of the event including the creation of any table

2.2.9.2 Explanation

TODO: To explain how the event works using 5-10 sentences.

2.2.9.3 Trigger validation

TODO: Write the tests you have done to test that the event works: insert... update... delete... You can include results of any query.

2.3 Shopping module

2.3.1 Requirement (Query) 3.1 Local commerce

2.3.1.1 Solution

```
SELECT DISTINCT c.name AS 'company name', cy.name AS 'country name'
FROM company AS c JOIN product AS p ON c.companyID = p.companyID
JOIN food AS f ON f.foodID = p.productID
JOIN country AS cy ON c.countryID = cy.countryID
WHERE ((SELECT COUNT(f2.foodID)
        FROM company AS c2 JOIN product AS p2 ON c2.companyID = p2.companyID
        JOIN food AS f2 ON f2.foodID = p2.productID
        WHERE c.companyID = c2.companyID
```

```

AND c.countryID = f2.countryID) / (SELECT COUNT(f3.foodID)
FROM company AS c3 JOIN product AS p3 ON
c3.companyID = p3.companyID
JOIN food AS f3 ON f3.foodID = p3.productID
WHERE c.countryID = f2.countryID)) >= 1/40;

```

company name	country name
Browsetype	Canada
Centidel	Mexico

2.3.1.2 Explanation

Fem el SELECT de les dades que ens demanen i posem el DISTINCT per diferenciar els resultats. Fem els JOINS de les taules necessàries i fem 2 SUBQUERYs. La primera mostra el número de productes alimentaris en que el seu país de procedència és el mateix que el país de la seva empresa. La segona mostra la quantitat total de productes alimentaris que té aquella empresa. I, segons ens diu l'enunciat, hem de mostrar les empreses en que tenen mínim 1 de 40 productes alimentaris, per tant fem la operació de que el número de productes alimentaris (1a subquery) / el número total de productes alimentaris d'aquella empresa (2a subquery) ha de ser superior o igual a 1/40.

2.3.1.3 Query validation

```

SELECT DISTINCT c.name AS 'company name', cy.name AS 'country name', c.companyID,
cy.countryID AS 'país de la empresa'
FROM company AS c JOIN product AS p ON c.companyID = p.companyID
JOIN food AS f ON f.foodID = p.productID
JOIN country AS cy ON c.countryID = cy.countryID
WHERE ((SELECT COUNT(f2.foodID)
FROM company AS c2 JOIN product AS p2 ON c2.companyID = p2.companyID
JOIN food AS f2 ON f2.foodID = p2.productID
WHERE c2.companyID = c2.companyID
AND c2.countryID = f2.countryID) / (SELECT COUNT(f3.foodID)
FROM company AS c3 JOIN product AS p3 ON
c3.companyID = p3.companyID
JOIN food AS f3 ON f3.foodID = p3.productID
WHERE c2.countryID = c3.countryID)) >= 1/40;

```

company name	country name	companyID	país de la empresa
Centidel	Mexico	91	108
Browsetype	Canada	79	4

Primer de tot, copiem la mateixa query que la principal però afegint-li el companyID i countryID de l'empresa ja que per a les següents query ens interessarà.

```

SELECT f.foodID, f.countryID AS 'país procedent del producte alimentari', c.countryID AS 'país de la empresa', c.companyID
FROM company AS c JOIN product AS p ON c.companyID = p.companyID
JOIN food AS f ON f.foodID = p.productID
JOIN country AS cy ON c.countryID = cy.countryID
WHERE c.countryID = 91
AND f.countryID = 108;

```

foodID	país procedent del producte alimentari	país de la empresa	companyID
3208	108	108	91

En aquesta query mostrem/calculem quants productes alimentaris de la primera empresa que ens surt (companyID = 91) hi ha en que el país de procedència dels productes alimentari sigui el mateix que el país de l'empresa. Veiem que sols hi ha 1. Al select podríem haver posat SELECT COUNT(f.foodID) i directament et diu quants productes alimentaris hi ha amb aquestes condicions, però nosaltres volíem mostrar-los.

```
SELECT f.foodID, f.countryID AS 'país procedent del producte alimentari', c.countryID AS 'país de la empresa', c.companyID
FROM company AS c JOIN product AS p ON c.companyID = p.companyID
JOIN food AS f ON f.foodID = p.productID
JOIN country AS cy ON c.countryID = cy.countryID
WHERE c.companyID = 91;
```

foodID	país procedent del producte alimentari	país de la empresa	companyID
229	243	108	91
766	231	108	91
1314	104	108	91
1844	82	108	91
2421	250	108	91
3208	108	108	91
3628	55	108	91
4054	113	108	91
4342	24	108	91
4547	170	108	91
4877	135	108	91
5597	95	108	91
5669	41	108	91
5829	192	108	91
5933	164	108	91
6778	192	108	91
7347	142	108	91
7357	59	108	91

mult 157

263 15:16:26 SELECT f.foodID, f.countryID, c.countryID, c.companyID FROM company AS c JOIN product AS p ON c.companyID = p.companyID JOIN food AS f ON f.foodID = p.productID JOIN country AS cy ON c.countryID = cy.countryID WHERE c.companyID = 91; 38 row(s) returned

0.000 sec. / 0.000 sec.

En aquesta query mostrem/calculem el número total de productes alimentaris que hi ha a la empresa amb companyID = 91. Veiem que retorna 38 files, per tant hi ha 38. Al select podríem haver posat simplement SELECT COUNT(f.foodID) però volíem mostrar tots els productes i comprovar que efectivament, entre tots ells, un és el que té foodID = 3208.

Per tant, si fem la operació $1/38$, veiem que és major a $1/40$ i per tant compleix la condició.

Hauríem de repetir aquest procés per l'altra empresa que surt a la query principal, que té companyID = 79. Tan sols hauríem de canviar el companyID i countryID a les queries de validació i també veuríem com compleix la condició.

2.3.2 Requirement (Query) 3.2 Best trade relations

2.3.2.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.3.2.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.3.2.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.3.3 Requirement (Query) 3.3 The best restaurant

2.3.3.1 Solution

```
SELECT c.name AS 'company name', ROUND(MAX(r.score), 2) AS 'score'
FROM company AS c JOIN waitingarea AS wa ON c.companyID = wa.companyID
JOIN restaurant AS r ON wa.waitingAreaID = r.restaurantID
WHERE c.companyID = (SELECT c2.companyID
                     FROM company AS c2 JOIN waitingarea AS wa2 ON c2.companyID =
wa2.companyID
                     GROUP BY c2.companyID
                     ORDER BY COUNT(wa2.waitingAreaID) DESC
                     LIMIT 1);
```

	company name	score
▶	Zazio	9.47

2.3.3.2 Explanation

Seleccionem els atributs que se'ns requereix a l'enunciat, i, en l'score, fem servir la funció MAX per seleccionar l'score més alt dels restaurants d'aquella empresa i el ROUND per arrodonir a dos decimals. Fem els JOINS convenients per l'obtenció de les dades que necessitem. Hem emprat una SUBQUERY que s'encarrega de comprovar que sigui la company amb més waiting areas, ho hem fet ordenant descendentment pel nombre de waiting areas i posant el límit a 1 per a que surti la company amb més waitings areas.

2.3.3.3 Query validation

```
SELECT c.name AS 'company name', COUNT(wa.waitingAreaID) AS 'nombre de waiting areas'
FROM company AS c JOIN waitingarea AS wa ON c.companyID = wa.companyID
GROUP BY c.companyID
ORDER BY COUNT(wa.waitingAreaID) DESC;
```

company name	nombre de waiting areas
Zazio	25
Quire	21
Skaboo	20
Aimbo	20
Quinu	19
Wordpedia	18
Babbleopia	18
Shufflester	18
Mymm	17
Twimm	17
Eimbee	17
Realcube	17
Pixoboo	16
Miboo	16
Liveth the	16

En aquesta query mirem quina es la company que té més waiting areas. Veiem que la empresa es diu Zazio.

```
SELECT c.name, r.score
FROM company AS c JOIN waitingarea AS wa ON c.companyID = wa.companyID
JOIN restaurant AS r ON wa.waitingAreaID = r.restaurantID
WHERE c.name = 'Zazio'
ORDER BY r.score DESC;
```

company name	score
Zazio	9.47
Zazio	9.26
Zazio	8.77
Zazio	8.7
Zazio	8
Zazio	7.97
Zazio	7.96
Zazio	7.79
Zazio	7.32
Zazio	6.93
Zazio	6.64
Zazio	6.4

D'aquesta empresa anomenada Zazio, mostrem la puntuació (score) de tots els seus restaurants i ho ordenem de més alta a més baixa. Veiem que el restaurant amb major puntuació 9.47 i, per tant, el resultat coincideix amb el resultat de la query principal.

2.3.4 Requirement (Query) 3.4 Stores and restaurants

2.3.4.1 Solution

```
SELECT c.name AS 'company name', c.company_value AS 'company value'
FROM company AS c
WHERE c.companyID IN (SELECT c.companyID
                      FROM (SELECT s.storeID, (COUNT(*) / (SELECT
COUNT(*)
```

```

FROM store AS s2 JOIN productstore AS ps2 ON s2.storeID = ps2.storeID

WHERE s2.storeID = s.storeID

GROUP BY s2.storeID)) AS same_product_ratio
FROM store AS s JOIN productstore
AS ps ON s.storeID = ps.storeID
JOIN product AS p ON p.productID =
ps.productID
JOIN company AS productcompany
ON productcompany.companyID = p.companyID
GROUP BY p.companyID, s.storeID)
AS same_product_ratio_x_store
JOIN waitingarea AS w ON
same_product_ratio_x_store.storeID = w.waitingareaID
JOIN company AS c ON w.companyID =
c.companyID
WHERE
same_product_ratio_x_store.same_product_ratio > 0.2)
AND c.companyID IN (SELECT c.companyID
FROM restaurant AS r JOIN waitingarea AS w ON
r.restaurantID = w.waitingareaID
JOIN company AS c ON c.companyID = w.companyID
GROUP BY c.companyID, r.type
HAVING COUNT(r.type) > 1);

```

company name	company value
Devshare	32945576
Centidel	21096882
Linklinks	30410268
Voomm	20521030
Fatz	8653178
Oyondu	13059034
Buzzdog	22842172

2.3.4.2 Explanation

Fem el SELECT de les dades que ens demanen i fem els JOINS necessaris. Aleshores, comprovem, emprant subqueries, les condicions que requereix l'enunciat. En la condició del WHERE c.companyID IN i tot lo que ve a continuació fins el següent AND, comprovem que una company té botigues que venen, com a mínim, el 20% del total dels productes de la seva company. La segona part de la query, a partir del AND c.companyID IN i lo que ve a continuació, comprovem que té restaurants de diferents tipus. Emprem 2 IN ja que ens interessa aquelles companys que compleixen les condicions explicades anteriorment.

2.3.4.3 Query validation

Fiquem la mateixa query que la principal però afegint-li la companyID per a posteriori fer validacions.

```
SELECT c.name AS 'company name', c.company_value AS 'company value', c.companyID
```



```

FROM company AS c
WHERE c.companyID IN (SELECT c.companyID
                      FROM (SELECT s.storeID, (COUNT(*) / (SELECT
COUNT(*)
FROM store AS s2 JOIN productstore AS ps2 ON s2.storeID = ps2.storeID
WHERE s2.storeID = s.storeID
GROUP BY s2.storeID)) AS same_product_ratio
FROM store AS s JOIN productstore
JOIN product AS p ON p.productID =
JOIN company AS productcompany
GROUP BY p.companyID, s.storeID)
AS same_product_ratio_x_store
same_product_ratio_x_store.storeID = w.waitingareaID
JOIN company AS c ON w.companyID =
c.companyID
WHERE
same_product_ratio_x_store.same_product_ratio > 0.2)
AND c.companyID IN (SELECT c.companyID
FROM restaurant AS r JOIN waitingarea AS w ON
r.restaurantID = w.waitingareaID
JOIN company AS c ON c.companyID = w.companyID
GROUP BY c.companyID, r.type
HAVING COUNT(r.type) > 1);

```

company name	company value	companyID
Devshare	32945576	151
Centidel	21096882	154
Linklinks	30410268	157
Voomm	20521030	160
Fatz	8653178	164
Oyodu	13059034	166
Buzzdog	22842172	169
NULL	NULL	NULL

Fem una query per comprovar que la primera companya que surt a la captura de la query principal (companyID = 151), té alguna botiga que ven més del 20% dels productes totals.

```

SELECT c.companyID, s.storeID, COUNT(p.productID) AS 'número de productes'
FROM store AS s JOIN productstore AS ps ON s.storeID = ps.storeID
JOIN product AS p ON p.productID = ps.productID
JOIN company AS c ON p.companyID = c.companyID
WHERE c.companyID = 151

```

GROUP BY c.companyID, s.storeID;

companyID	storeID	número de productes
151	643	96
151	1458	1

En aquestes cas, veiem que la storeID = 643 ven més del 20%, de fet, ven el $(96/97) * 100 = 98,96\%$.

En aquesta query mirem que la company que surt amb ID = 151 tingui restaurants diferents:

```
SELECT c.companyID, r.restaurantID, r.type
FROM company AS c JOIN waitingarea AS w ON c.companyID = w.companyID
JOIN restaurant AS r ON w.waitingareaID = r.restaurantID
WHERE c.companyID = 151;
```

companyID	restaurantID	type
151	1688	Fast Food
151	1713	Mexican
151	1886	Mexican

Efectivament, té restaurants de diferents tipus.

Per últim, si volem validar tots els resultats de la query principal, hauriem de canviar el companyID pel que vulguem mirar a les validacions.

2.3.5 Requirement (Query) 3.5 Waiting areas without shop keepers

2.3.5.1 Solution

```
SELECT c.name AS 'company name', wa.opening_hour AS 'opening hour', wa.close_hour AS
'close hour', wa.airportID AS 'airport ID', wa.waitingareaID AS 'waiting area ID'
FROM company AS c JOIN waitingarea AS wa ON wa.companyID = c.companyID
WHERE (HOUR(wa.close_hour - wa.opening_hour)) > (SELECT
SUM((HOUR(sk2.weekly_hours)/7))
FROM company AS c2 JOIN
waitingarea AS wa2 ON wa2.companyID = c2.companyID
JOIN shopkeeper AS sk2 ON
sk2.waitingareaID = wa2.waitingareaID
WHERE wa.waitingareaID =
wa2.waitingareaID)
AND wa.close_hour < '24:00:00' AND
wa.close_hour > '06:00:00';
```

2.3.5.2 Explanation

Seleccionem els atributs que se'ns requereix a l'enunciat, fem els JOINS necessaris per l'obtenció de dades i posem les condicions pertinents. La subquery, en aquest cas, comprova que les hores que manté obert durant el dia la waiting area sigui superior a la suma de les hores laborals dels

shopkeepers d'aquella waiting area en concret, ja que ens ho requereix l'enunciat. La funció HOUR retorna únicament la hora, no els minuts. A més a més, també comprovem que la botiga tanqui abans de mitjanit, que són les dues últimes condicions de la query. Considerem que una waiting area tanca abans de la mitjanit si tanca abans de les 12 de la nit i després de les 6 de la matinada.

2.3.5.3 Query validation

```
SELECT c.name AS 'company name', wa.airportID, wa.waitingareaID, HOUR(wa.close_hour -
wa.opening_hour) AS 'hores que obra al dia', SUM(HOUR(sk.weekly_hours)/7) 'hores totals diaries
dels shopkeepers'
```

```
FROM company AS c JOIN waitingarea AS wa ON wa.companyID = c.companyID
```

```
JOIN shopkeeper AS sk ON sk.waitingareaID = wa.waitingareaID
```

```
AND wa.close_hour < "24:00:00" AND wa.close_hour > "06:00:00";
```

```
GROUP BY wa.waitingareaID
```

```
ORDER BY wa.waitingareaID ASC;
```

company name	airportID	waitingareaID	hores que obra al dia	hores totals diaries dels shopkeepers
Miboo	1187	1	7	7.1429
Meedoo	5757	2	13	10.4286
Kazio	575	3	13	10.2857
Twitterbeat	1187	4	8	17.0000
Twitterbeat	337	5	11	22.1429
Babbleopia	5560	6	10	22.5714
Feedfish	2101	7	9	8.2857
Realcube	2916	8	16	20.2857
Twinder	5560	9	14	6.7143
Blogspan	1187	10	15	5.4286
Divavu	1347	11	10	19.1429
Realcube	5757	12	14	20.4286
Shufflester	5449	13	11	14.8571
Twitterbeat	2101	14	8	48.4286
Photobug	5486	15	10	0.0000
Photospace	3213	16	13	38.1429
Skajo	5449	17	11	35.7143
Roomm	575	18	11	21.5714
Skaboo	5449	19	15	23.4286

Aquí mostrem totes les waiting areas que tanquen abans de les 12 de la nit i després de les 6 de la matinada. D'aquestes waiting areas, ens hem de fixar en les que la columna 'hores que obra al dia' sigui major a 'hores totals diàries dels shopkeepers' ja que ens ho requereix l'enunciat. Per tant, si anem mirant veiem que hem de tenir en compte les waiting areas amb ID 2, 3, 7, 9, 10, 15... Per tant, coincideixen perfectament amb els resultats de la query principal.

2.3.6 Requirement (Trigger) 3.6 Waiting areas shutdown

2.3.6.1 Solution

TODO: Write the solution of the trigger including the creation of any table

2.3.6.2 Explanation

TODO: To explain how the trigger works using 5-10 sentences.

2.3.6.3 Trigger validation

TODO: Write the tests you have done to test that the trigger works: insert... update... delete... You can include results of any query.

2.3.7 Requirement (Trigger) 3.7 Updated products

2.3.7.1 Solution

```
DROP TABLE IF EXISTS PriceUpdates;
CREATE TABLE PriceUpdates(
    productID bigint(20),
    product_name VARCHAR(255),
    companyID bigint(20),
    previous_price VARCHAR(255),
    later_price VARCHAR(255),
    date_of_change DATETIME,
    comentari TEXT,
    PRIMARY KEY (productID, date_of_change)
);
DELIMITER $$
DROP TRIGGER IF EXISTS updated_products $$
CREATE TRIGGER updated_products
    AFTER UPDATE
    ON product
    FOR EACH ROW
BEGIN
    IF NEW.price <> OLD.price THEN
        SET @Comment = " ";

        IF (SELECT COUNT(*)
            FROM PriceUpdates
            WHERE productID = OLD.productID
            AND date_of_change <> CURRENT_TIMESTAMP()) > 0 THEN

            IF (SELECT later_price
                FROM PriceUpdates
                WHERE productID = OLD.productID
                ORDER BY date_of_change DESC
                LIMIT 1) > NEW.price THEN

                SET @Comment = "This product has been changing over time, it is
possible that it is a strategy of the company.";

            END IF;

        END IF;

        INSERT INTO PriceUpdates(productID, product_name, companyID,
previous_price, later_price, date_of_change, comentari)
            VALUES (OLD.productID, OLD.name, OLD.companyID, OLD.price, NEW.price,
NOW(), @Comment);

    END IF;
```

END \$\$

DELIMITER ;

2.3.7.2 Explanation

Primer de tot, creem la taula PriceUpdates amb tots els atributs que se'ns requereix a l'enunciat, però afegint-li el productID per després al trigger fer unes comprovacions (els becaris de l'assignatura ens van dir que no hi havia problema en fer-ho). Seguidament, creem el TRIGGER i posem que sigui AFTER UPDATE ON product ja que al enunciat ens diuen clarament que cada vegada que fem un update del preu d'un producte, hem d'inserir certa informació a la taula PriceUpdates. Dins del trigger, posem la condició de que si el preu que hem actualitzat del producte és diferent al que hi havia anteriorment, doncs posem la condició de que si a la taula PriceUpdates ja hi ha un producte amb el mateix ID que ha canviat de preu i amb una data de modificació diferent a la data de la nova actualització, doncs posem la condició de que si l'últim preu en que s'ha actualitzat aquest producte és superior al preu que hem fet update, doncs hem de fer el insert a PriceUpdates i posar el comentari "This product has been changing over time, it is possible that it is a strategy of the company.". Si no compleix els dos últims condicionals no posarem cap comentari i significarà que o no hi ha cap producte amb el mateix productID del que hem fet update a PriceUpdates o que el preu del que hem fet update no s'ha decrementat respecte l'última actualització a PriceUpdates. Finalment, fem el insert dels atributs corresponents a PriceUpdates amb el comentari corresponent.

2.3.7.3 Trigger validation

Abans que res, dir que l'atribut 'date_of_change' de la taula PriceUpdates l'hem posat de tipus DATETIME ja que sigui més pràctic a l'hora de fer les validacions, ja que si haguéssim posat el tipus DATE, doncs tan sols podríem afegir un producte amb el mateix ID per dia per a que surti el comentari de "This product has been changing over time, it is possible that it is a strategy of the company", ja que a l'enunciat ens diu que per a que surti aquest comentari un dels requisits és que la data sigui diferent. I si haguéssim posat que fos de tipus DATE, dins del trigger, a la condició de AND date_of_change <> CURRENT_TIMESTAMP() hauria de ser AND date_of_change <> CURRENT_DATE().

Primer mostrem la taula product, ordenat descendentment pel productId.

```
SELECT * FROM product
ORDER BY productId DESC;
```

productId	companyID	name	weight	price
17000	154	Asoka	19.54	19.52
16999	110	Span	3.45	5.48
16998	116	Keylex	16.63	2.23
16997	27	Wrapsafe	17.27	1669
16996	125	Daltfresh	3.1	21.45
16995	56	Toughjoyfax	15.96	245
16994	31	Flexidy	3.85	1063
16993	110	Alpha	16.52	1.41
16992	154	It	1.81	24.61
16991	60	Quo Lux	15.6	117

Comprovem que la taula PriceUpdates està buida.

SELECT * FROM priceupdates;

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fem uns inserts a la taula product per no modificar els productes que ja havien a la base dades.

```
INSERT INTO product (productID, companyID, name, weight, price)
VALUES (17001, 110, "Standoff", 17.5, 53.9);
```

```
INSERT INTO product (productID, companyID, name, weight, price)
VALUES (17002, 116, "Raid", 34.1, 77.4);
```

productID	companyID	name	weight	price
17002	116	Raid	34.1	77.4
17001	110	Standoff	17.5	53.9
17000	154	Asoka	19.54	19.52
16999	110	Span	3.45	5.48
16998	116	Keylex	16.63	2.23
16997	27	Wrapsafe	17.27	1669
16996	125	Daltfresh	3.1	21.45
16995	56	Toughjoyfax	15.96	245
16994	31	Flexidy	3.85	1063
16993	110	Alpha	16.52	1.41

Ara ja podem fer els UPDATE del preu d'algun producte. Comensem fent-lo pel productID = 17001.

```
UPDATE product SET price = 40 WHERE productID = 17001;
```

Mostrem la taula PriceUpdates:

SELECT * FROM priceupdates;

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
17001	Standoff	110	53.9	40	2021-06-29 14:57:29	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

S'afegeix correctament el producte amb el seu ID = 17001, el seu nom que és Standoff, el seu companyID que és 110, el preu que costava anteriorment que és 53.9 (mirar-ho a la captura de la taula producte), el preu que costa actualment que és el preu que hem posat al update que és 40, la data en que s'ha fet el canvi i el comentari buit ja que és el primer producte que afegim a PriceUpdates.

Fem un altre UPDATE pel mateix productID però ara incrementant el preu. El comentari hauria de sortir buit ja que no s'ha decrementat el preu.

```
UPDATE product SET price = 45 WHERE productID = 17001;
```

Mostrem la taula PriceUpdates:

SELECT * FROM priceupdates;

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
17001	Standoff	110	53.9	40	2021-06-29 14:57:29	
17001	Standoff	110	40	45	2021-06-29 15:01:30	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

El previous_price és el preu que havia abans que era 40 i el preu actual és 45. Com no hem decrementat el preu el comentari surt buit.

Fem un altre UPDATE pel mateix productID però ara decrementant el preu respecte l'última actualització.

UPDATE product SET price = 43 WHERE productID = 17001;

Mostrem la taula PriceUpdates:

SELECT * FROM priceupdates;

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
17001	Standoff	110	53.9	40	2021-06-29 14:57:29	
17001	Standoff	110	40	45	2021-06-29 15:01:30	
17001	Standoff	110	45	43	2021-06-29 15:03:47	This product has been changing over time, it is possible that it is a strategy of the company.
NULL	NULL	NULL	NULL	NULL	NULL	NULL

El preu anterior era de 45, l'actual de 43, per tant, com s'ha decrementat el preu respecte l'última actualització, s'ha canviat en diferents dates (és la mateixa, tan sols canvia la hora pero com hem dit abans hem posat la hora per a que la data fos diferent i pugues sortir el comentari sense haver de fer els updates en diez diferents) i el producte ja surt anteriorment a la taula, doncs mostra aquest comentari "This product has been changing over time, it is possible that it is a strategy of the company".

Si ara fem un update i li incrementem el valor respecte l'últim preu, doncs el comentari hauria de tornar a sortir buit.

UPDATE product SET price = 47 WHERE productID = 17001;

Mostrem la taula PriceUpdates:

SELECT * FROM priceupdates;

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
17001	Standoff	110	53.9	40	2021-06-29 14:57:29	
17001	Standoff	110	40	45	2021-06-29 15:01:30	
17001	Standoff	110	45	43	2021-06-29 15:03:47	This product has been changing over time, it is possible that it is a strategy of the company.
17001	Standoff	110	43	47	2021-06-29 15:09:04	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ara, per exemple, fem update del producte amb id = 17002 i veurem que les seves dades són correctes i el comentari surt buit ja que sera el primer cop que aquest producte surti a la taula.

UPDATE product SET price = 80 WHERE productID = 17002;

Mostrem la taula PriceUpdates:

```
SELECT * FROM priceupdates;
```

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
17001	Standoff	110	53.9	40	2021-06-29 14:57:29	
17001	Standoff	110	40	45	2021-06-29 15:01:30	
17001	Standoff	110	45	43	2021-06-29 15:03:47	This product has been changing over time, it is possible that it is a strategy of the company.
17001	Standoff	110	43	47	2021-06-29 15:09:04	
17002	Raid	116	77.4	80	2021-06-29 15:11:55	
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Per últim li fem un update a aquest producte per a que surti el comentari llarg. Per tant, li hem de decrementar el preu:

```
UPDATE product SET price = 70 WHERE productID = 17002;
```

Mostrem la taula PriceUpdates:

```
SELECT * FROM priceupdates;
```

productID	product_name	companyID	previous_price	later_price	date_of_change	comentari
17001	Standoff	110	53.9	40	2021-06-29 14:57:29	
17001	Standoff	110	40	45	2021-06-29 15:01:30	
17001	Standoff	110	45	43	2021-06-29 15:03:47	This product has been changing over time, it is possible that it is a strategy of the company.
17001	Standoff	110	43	47	2021-06-29 15:09:04	
17002	Raid	116	77.4	80	2021-06-29 15:11:55	
17002	Raid	116	80	70	2021-06-29 15:13:18	This product has been changing over time, it is possible that it is a strategy of the company.
NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.3.8 Requirement (Trigger) 3.8 Product values per m2

2.3.8.1 Solution

TODO: Write the solution of the trigger including the creation of any table

2.3.8.2 Explanation

TODO: To explain how the trigger works using 5-10 sentences.

2.3.8.3 Trigger validation

TODO: Write the tests you have done to test that the trigger works: insert... update... delete... You can include results of any query.

2.3.9 Requirement (Event) 3.9 Expired food

2.3.9.1 Solution

```
DROP TABLE IF EXISTS ExpiredProducts;  
CREATE TABLE ExpiredProducts(  
    productID BIGINT(20),  
    expiry_date DATE,  
    dia DATE,  
    PRIMARY KEY (productID)  
);
```

```
DELIMITER $$  
DROP PROCEDURE IF EXISTS food_check $$  
CREATE PROCEDURE food_check()  
BEGIN
```



```

INSERT INTO ExpiredProducts(productID, expiry_date, dia)
SELECT foodID, expiration_date, CURRENT_DATE()
FROM food JOIN product AS p ON foodID = p.productID
JOIN productstore AS ps ON p.productID = ps.productID
WHERE expiration_date < CURRENT_DATE()
GROUP BY foodID

```

```

ON DUPLICATE KEY UPDATE productID = productID;
END $$

```

```

DELIMITER ;
DROP EVENT IF EXISTS expired_food;
CREATE EVENT expired_food
ON SCHEDULE EVERY 24 HOUR
STARTS '2021-01-01 23:59:59'
ON COMPLETION PRESERVE
DO CALL
    food_check();

```

2.3.9.2 *Explanation*

En primer lloc, creem la taula ExpiredProducts amb els atributs que ens demanen. A continuació, creem el STORED PROCEDURE, que serà l'encarregat d'insertar la informació correspondent a ExpiredProducts sempre que un producte alimentari estigui en venda i estigui caducat. En aquesta taula, insertarem el ID del producte alimentari, la data de caducitat d'aquest producte i la data en que s'ha afegit a la taula ExpiredProducts (la data actual). Dins del cos del stored procedure, per saber si un producte alimentari està en venda, fem el JOIN amb productstore. I posem la condició de que si la data de caducitat és inferior a la data actual doncs significa que està caducat i l'hem d'insertar a ExpiredProducts. Finalment, creem l'EVENT per a que s'actualitzi diàriament i cridem al stored procedure anomenat food_check().

2.3.9.3 *Trigger validation*

Primer de tot fem uns inserts a product, productstore i finalment a food.

```

INSERT INTO product(productID, companyID, name, weight, price)
VALUES (17003, 116, "Pringles", 20.5, 2);

```

```

INSERT INTO productstore(storeID, productID)
VALUES (601, 17003);

```

```

INSERT INTO food(foodID, expiration_date, countryID)
VALUES (17003, "2021-01-04", 125);

```

Com es pot veure, hem insertat un producte alimentari que està caducat, ja que actualment estem a 2021-06-30 i la data que hem posat es 2021-01-04. Hem posat que el producte estigui en venda ja que l'hem afegit a productstore.

Mostrem el contingut que hi ha a ExpiredProducts:

```

SELECT *
FROM expiredproducts

```

ORDER BY dia DESC;

productID	expiry_date	dia
737	2021-06-29	2021-06-30
2626	2021-06-29	2021-06-30
2982	2021-06-29	2021-06-30
3405	2021-06-29	2021-06-30
5264	2021-06-29	2021-06-30
6169	2021-06-29	2021-06-30
6895	2021-06-29	2021-06-30
6935	2021-06-29	2021-06-30
7154	2021-06-29	2021-06-30
7347	2021-06-29	2021-06-30
7713	2021-06-29	2021-06-30
10195	2021-06-29	2021-06-30
12297	2021-06-29	2021-06-30
13553	2021-06-29	2021-06-30
13930	2021-06-29	2021-06-30
14668	2021-06-29	2021-06-30
16256	2021-06-29	2021-06-30
17003	2021-01-04	2021-06-30

Com es pot veure, el producte alimentari que hem introduït (ID = 17003) apareix a la taula amb la data actual amb la que s'ha inserit a ExpiredProducts, és a dir, la data en que hem executat l'event. A part del producte alimentari amb ID = 17003, veiem que també s'han inserit els productes ja caducats i que estan en venda que es troben a la taula food.

Afegir també que el primer dia que vam executar l'event va ser el dia 2021-06-19 i aleshores aquest dia es van afegir tots els productes alimentaris que estaven caducats i a la venda que tenien una data de caducitat inferior a 2021-06-19. Us ho mostrem a continuació:

```
SELECT *  
FROM expiredproducts  
ORDER BY dia DESC;
```

productID	expiry_date	dia
30	2021-06-18	2021-06-19
852	2021-06-18	2021-06-19
901	2021-06-18	2021-06-19
1133	2021-06-18	2021-06-19
1556	2021-06-18	2021-06-19
1902	2021-06-18	2021-06-19
1914	2021-06-18	2021-06-19
3631	2021-06-18	2021-06-19
3707	2021-06-18	2021-06-19
4131	2021-06-18	2021-06-19
4479	2021-06-18	2021-06-19
4889	2021-06-18	2021-06-19
5081	2021-06-18	2021-06-19
5152	2021-06-18	2021-06-19
6440	2021-06-18	2021-06-19
6508	2021-06-18	2021-06-19
6847	2021-06-18	2021-06-19
7242	2021-06-18	2021-06-19
7734	2021-06-18	2021-06-19
8019	2021-06-18	2021-06-19

A part d'aquests hi ha un munt més que no es veuen a la captura.

El segon dia que vam executar l'event va ser el 2021-06-21, aleshores es van afegir el productes alimentaris caducats entre el 2021-06-19 i el 2021-06-21 que hi havia a la taula food.

productID	expiry_date	dia
302	2021-06-20	2021-06-21
1066	2021-06-20	2021-06-21
2672	2021-06-20	2021-06-21
4169	2021-06-20	2021-06-21
4248	2021-06-20	2021-06-21
5807	2021-06-20	2021-06-21
5891	2021-06-20	2021-06-21
6475	2021-06-20	2021-06-21
6673	2021-06-20	2021-06-21
7936	2021-06-20	2021-06-21
8284	2021-06-20	2021-06-21
9227	2021-06-20	2021-06-21
9243	2021-06-20	2021-06-21
11570	2021-06-20	2021-06-21
12097	2021-06-20	2021-06-21
12154	2021-06-20	2021-06-21
12521	2021-06-20	2021-06-21
13419	2021-06-20	2021-06-21
13974	2021-06-20	2021-06-21
14531	2021-06-20	2021-06-21

El tercer dia que vam executar l'event va ser el 2021-06-24, aleshores es van afegir el productes alimentaris caducats entre el 2021-06-21 i el 2021-06-23 que hi havia a la taula food.

productID	expiry_date	dia
181	2021-06-23	2021-06-24
229	2021-06-23	2021-06-24
1873	2021-06-23	2021-06-24
3650	2021-06-23	2021-06-24
4693	2021-06-23	2021-06-24
5035	2021-06-23	2021-06-24
5791	2021-06-23	2021-06-24
5810	2021-06-23	2021-06-24
7344	2021-06-23	2021-06-24
7929	2021-06-23	2021-06-24
8134	2021-06-23	2021-06-24
9277	2021-06-23	2021-06-24
10189	2021-06-23	2021-06-24
10258	2021-06-23	2021-06-24
10535	2021-06-23	2021-06-24
10833	2021-06-23	2021-06-24
10988	2021-06-23	2021-06-24
12700	2021-06-23	2021-06-24
14503	2021-06-23	2021-06-24
14690	2021-06-23	2021-06-24

I així uns dies més fins executar-lo per última vegada el dia 2021-06-30 (primera captura).

A continuació, fem un insert d'un producte alimentari que està en venda i que no està caducat. Per tant, no s'hauria d'inserir a ExpiredProducts.

```
INSERT INTO product(productID, companyID, name, weight, price)
VALUES (17004, 27, "Lays", 15.2, 3);
```

```
INSERT INTO productstore(storeID, productID)
VALUES (8, 17004);
```

```
INSERT INTO food(foodID, expiration_date, countryID)
VALUES (17004, "2022-03-11", 81);
```

Comprovem que no s'ha afegit a ExpiredProducts:

```
SELECT *
FROM expiredproducts
ORDER BY dia DESC;
```

productID	expiry_date	dia
737	2021-06-29	2021-06-30
2626	2021-06-29	2021-06-30
2982	2021-06-29	2021-06-30
3405	2021-06-29	2021-06-30
5264	2021-06-29	2021-06-30
6169	2021-06-29	2021-06-30
6895	2021-06-29	2021-06-30
6935	2021-06-29	2021-06-30
7154	2021-06-29	2021-06-30
7347	2021-06-29	2021-06-30
7713	2021-06-29	2021-06-30
10195	2021-06-29	2021-06-30
12297	2021-06-29	2021-06-30
13553	2021-06-29	2021-06-30
13930	2021-06-29	2021-06-30
14668	2021-06-29	2021-06-30
16256	2021-06-29	2021-06-30
17003	2021-01-04	2021-06-30
490	2021-06-28	2021-06-29
1146	2021-06-28	2021-06-29

Efectivament, no s'ha afegit ja que no està caducat.

2.4 *Luggage module*

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

2.4.1.1 *Solution*

```
SELECT DISTINCT p.personID AS 'person id' , p.name, p.surname, e.salary
FROM person as p JOIN employee AS e ON p.personID = e.employeeID
JOIN luggagehandler as l ON e.employeeID = l.luggagehandlerID
JOIN flightluggagehandler AS fl ON l.luggagehandlerID = fl.luggagehandlerID
GROUP BY fl.flightID
HAVING COUNT(fl.luggagehandlerID) = 1 AND e.salary < (SELECT AVG(e2.salary)
FROM person as p2 JOIN employee AS e2 ON p2.personID = e2.employeeID
JOIN luggagehandler as l2 ON e2.employeeID = l2.luggagehandlerID)
ORDER BY p.personID ASC;
```

person id	name	surname	salary
147967	Price	Hazelgreave	28756
147969	Beilul	Gisbey	48758
147971	Collie	Scase	25692
147977	Orlando	Menguy	63940
147979	Violet	Richie	70718
147980	Althea	Jeannot	58008
147986	Karylin	Fritche	57087
147995	Waylon	Koepe	10017
148003	Derrick	Newitt	26532
148006	Regine	Pargetter	19879
148009	Jorge	Sapsed	62052
148010	Ronni	Guihen	35744
148012	Lilas	McNabb	63459
148014	Ragnar	Colleford	37310
148017	Standf...	Nowick	55614
148022	Rozelle	Heminsley	38830
148023	Anya	McPhelimy	27828
148024	Legra	Garbett	70498
148028	Martie	Quinet	55978

2.4.1.2 Explanation

Primer fem el SELECT DISITINCT per diferenciar les persones i que no surtin repetides, fem els JOINS de les taules necessàries, agrupem pel flightID ja que hem de posar la condició al HAVING de que sols pot haver-hi un luggage handler per vol i que el seu salari sigui inferior a la mitja del salari dels lugaggehandlers, això últim ho hem feta amb una SUBQUERY i posant la condició. El order by no és necessari, és més per estètica.

2.4.1.3 Query validation

```
SELECT p.personID AS 'person id' , p.name, p.surname, e.salary, fl.flightID
FROM person as p JOIN employee AS e ON p.personID = e.employeeID
JOIN luggagehandler as l ON e.employeeID = l.luggagehandlerID
JOIN flightluggagehandler AS fl ON l.luggagehandlerID = fl.luggagehandlerID
GROUP BY fl.flightID
HAVING COUNT(fl.luggagehandlerID) = 1 AND e.salary < (SELECT AVG(e2.salary)
FROM person as p2 JOIN employee AS e2 ON p2.personID = e2.employeeID
JOIN luggagehandler as l2 ON e2.employeeID = l2.luggagehandlerID)
ORDER BY fl.flightID ASC;
```

person id	name	surname	salary	flightID
151396	Orbadiah	Darrach	31859	5
150907	Domenic	Cordero	17468	24
149341	Birgit	Piercy	49857	33
150187	Shea	Goode	10310	40
148799	Sam	Burwood	66243	65
149152	Bryana	Langsdon	33039	69
153341	Lindsay	Koppes	79885	85
148357	Stefa	Di Claudio	46408	107
149782	Sansone	Glazer	73678	115
148981	Bardlay	De Antoni	70702	125
153447	Carolyn	Baudone	78474	132
153283	Clay	Filchakov	50700	145
151560	Sayres	Browell	50871	147
152340	Belicia	Teissier	20230	150
151748	Papagena	De Rye B...	64976	151
150751	Pierson	Lockwood	79550	153
148906	Delbert	Cuddehay	75842	181
152824	Letty	Frankton	43432	188
152466	Corny	Savill	68456	200
152322	Shana	Usher	55488	215

Aquesta query és la mateixa que la principal però afegint-li el flightID i ordenant pel flightID de menor a major. El flightID ens interessarà per la següent query.

```
SELECT flightID
FROM flightluggagehandler
GROUP BY flightID
HAVING COUNT(luggageHandlerID) = 1
```

flightID
5
6
10
16
17
22
24
26
33
40
42
43
64
65
68
69
73
84
85
93

Aquesta query mostra la llista dels flightID en que sols hi ha un luggage handler, és a dir, són els vols en que només hi treballava un luggage handler. Aquests son els vols que ens interessin ja que és una condició que diu l'enunciat, que volem vols on només hagi carregat les maletes un luggage handler. Per tant, si mirem la captura de la primera query de validació on surt també el flightID, doncs aquets valors han de estar dins de la llista de valors que acabem de mostrar. I, efectivament, si anem fent scroll down i mirant, tots ho estan. Ara falta comprovar la condició del salari.

```
SELECT AVG(e.salary) AS 'mitja del salari dels luggagehandlers'
FROM person as p JOIN employee AS e ON p.personID = e.employeeID
JOIN luggagehandler as l ON e.employeeID = l.luggagehandlerID;
```

mitja del salari dels luggagehandlers
80295.8341

Aquesta query ens mostra la mitja del salari dels luggagehandlers, que és de 80295,8431. Per tant, segons ens diu l'enunciat, de la llista dels luggage handlers que surten a la llista principal, doncs el seu salari ha de ser inferior a 80295,8341.

```
SELECT DISTINCT p.personID AS 'person id' , p.name, p.surname, e.salary
FROM person as p JOIN employee AS e ON p.personID = e.employeeID
JOIN luggagehandler as l ON e.employeeID = l.luggagehandlerID
JOIN flightluggagehandler AS fl ON l.luggagehandlerID = fl.luggagehandlerID
GROUP BY fl.flightID
HAVING COUNT(fl.luggagehandlerID) = 1 AND e.salary <= (SELECT AVG(e.salary)
FROM person as p JOIN employee AS e ON p.personID = e.employeeID
JOIN luggagehandler as l ON e.employeeID = l.luggagehandlerID)
ORDER BY e.salary DESC;
```

person id	name	surname	salary
149940	Laurice	Meir	80149
152305	Tuckie	Etienne	80120
148955	Marisa	Gontier	80116
153124	Sayer	Fincham	80045
152140	Aldric	Boult	80040
149366	Johnath	Hauck	80030
150914	Annamarie	Tawn	79964
153341	Lindsay	Koppes	79885
152086	Hall	Pherps	79734
148095	Natasha	Mougeot	79720
151613	Lita	Kerwick	79718
151307	Harvey	Jaine	79687
152687	Quinn	Patterfield	79621
150096	Alethea	Dive	79568
150751	Pierson	Lockwood	79550
151112	Selle	Izhakov	79532
153079	Markos	Woolland	79490
150939	Gabriela	Guillain	79427
148367	Shen	Woolacott	79423

Posem la mateixa query que la principal però ordenant pel salari de major a menor, i podem veure que el luggage handler que té el salari més alt és de 80149 i, efectivament, es menor a la mitja del salari dels luggage handlers ($80295.8341 > 80149$).

2.4.2 Requirement (Query) 4.2 Luggage details

2.4.2.1 Solution

```
SELECT pe.name AS 'passenger name', pe.email, cy.name AS 'country', l.color, l.brand, l.weight,
NULL AS 'volume', cl.extra_cost AS 'extra cost', so.fragile AS 'fragility'
FROM person AS pe JOIN luggage AS l ON pe.personID = l.passengerID
JOIN country AS cy ON pe.countryID = cy.countryID
JOIN checkedluggage AS cl ON cl.checkedluggageID = l.luggageID
JOIN specialobjects AS so ON so.specialobjectID = cl.checkedluggageID
WHERE LEFT(pe.name, 4) = LEFT(cy.name, 4)
AND (SELECT COUNT(l2.luggageID)
```

```
FROM luggage AS l2
WHERE l2.passengerID = pe.personID) > 1
AND l.luggageID = (SELECT l2.luggageID
FROM luggage AS l2
WHERE l2.passengerID = pe.personID
ORDER BY l2.weight ASC
LIMIT 1)
```

```
UNION
SELECT pe.name, pe.email, cy.name, NULL, NULL, NULL, NULL, NULL, NULL
FROM person AS pe JOIN country AS cy ON pe.countryID = cy.countryID
WHERE LEFT(pe.name, 4) = LEFT(cy.name, 4)
AND (SELECT COUNT(l2.luggageID)
```

```
FROM luggage AS l2
WHERE l2.passengerID = pe.personID) = 0
```

```
UNION
SELECT pe.name, pe.email, cy.name, l.color, l.brand, l.weight, (hl.size_x*hl.size_y*hl.size_z) AS
'volume', NULL, NULL
FROM person AS pe JOIN luggage AS l ON pe.personID = l.passengerID
JOIN country AS cy ON pe.countryID = cy.countryID
JOIN handluggage AS hl ON hl.handluggageID = l.luggageID
WHERE LEFT(pe.name, 4) = LEFT(cy.name, 4)
AND (SELECT COUNT(l2.luggageID)
```

```
FROM luggage AS l2
WHERE l2.passengerID = pe.personID) > 1
AND l.luggageID = (SELECT l2.luggageID
FROM luggage AS l2
WHERE l2.passengerID = pe.personID
ORDER BY l2.weight ASC
LIMIT 1);
```

passenger name	email	country	color	brand	weight	volume	extra cost	fragility
Germaine	germaine.elderkin@smugmug.com	Germany	NULL	NULL	NULL	NULL	NULL	NULL
Camella	camella.arnaldy@ucla.edu	Cameroon	NULL	NULL	NULL	NULL	NULL	NULL
Maurice	maurice.thomasson@cdc.gov	Mauritania	NULL	NULL	NULL	NULL	NULL	NULL
Maurits	maurits.jury@blogtalkradio.com	Mauritania	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
Franz	franz.tithecote@si.edu	France	NULL	NULL	NULL	NULL	NULL	NULL
Francine	francine.natrass@amazon.de	France	NULL	NULL	NULL	NULL	NULL	NULL
Francesca	francesca.farlowe@theatlantic.com	France	NULL	NULL	NULL	NULL	NULL	NULL
Frandlyn	frandlyn.castellucci@bloglovin.com	France	NULL	NULL	NULL	NULL	NULL	NULL
Franzen	franz.en.capner@jimdo.com	France	NULL	NULL	NULL	NULL	NULL	NULL
Austina	austina.sanzio@wikispaces.com	Austria	NULL	NULL	NULL	NULL	NULL	NULL
Saudra	saudra.guild@google.com	Saudi Ara...	NULL	NULL	NULL	NULL	NULL	NULL
Saudra	saudra.hammel@mozilla.com	Saudi Ara...	NULL	NULL	NULL	NULL	NULL	NULL
Phillis	phillis.berredoth@ovh.net	Philippines	NULL	NULL	NULL	NULL	NULL	NULL
Britt	britt.purse@twitpic.com	British Vir...	NULL	NULL	NULL	NULL	NULL	NULL
Easter	easter.roistone@xing.com	East Timor	NULL	NULL	NULL	NULL	NULL	NULL
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
Norton	norton.bowling@surveymonkey.com	North Korea	NULL	NULL	NULL	NULL	NULL	NULL
Norton	norton.east@feedburner.com	North Korea	NULL	NULL	NULL	NULL	NULL	NULL
Northrup	northrup.woodburne@gnu.org	North Korea	NULL	NULL	NULL	NULL	NULL	NULL
Montgomery	montgomery.greatreax@opensour...	Montserrat	NULL	NULL	NULL	NULL	NULL	NULL
Austin	austin.tumbelty@myspace.com	Austria	Black	Sams...	1.4	19200	NULL	NULL
Franklin	franklin.bocke@w3.org	France	Red	East...	1.6	11832	NULL	NULL
Fransisco	fransisco.pickerin@sakura.ne.jp	France	Red	Amer...	4.6	66960	NULL	NULL

2.4.2.2 Explanation

Fem el SELECT dels atributs que ens demanen i els JOINS necessaris. Fem servir 2 UNION, per una banda, pel checkedluggage no conte l'atribut volum i per això el posem a NULL, però sí conte el extra_cost i la fragilitat. En canvi, una de les altres parts del union es pel handluggages, que sí conte el volum per no el extra_cost ni la fragilitat, per això els posem a NULL. Per l'última part del union és per ficar la query en que el passatger no té cap luggage, les altres dos querys que he comentat abans eren pels passatgers que tenen més d'un luggage.

2.4.2.3 Query validation

Com es pot veure a la captura de la query principal, les 4 primeres paraules del nom del passatger coincideix amb les 4 primeres paraules del país. A les columns en que el color, brand, weight, volume, extracost i fragility estan a NULL, significa que el passatger no té cap luggage. Si alguns d'aquests valors que acabem de comentar no estan a NULL, significa que el passatger té més d'un luggage.

2.4.3 Requirement (Query) 4.3 Lost objects

2.4.3.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.4.3.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.4.3.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.4.4 Requirement (Query) 4.4 Special objects

2.4.4.1 Solution

```
SELECT '0' AS 'hazardous level', ROUND(AVG(c.extra_cost), 2) AS 'average extra cost'
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 0
UNION
SELECT '1' AS 'hazardous level', ROUND(AVG(c.extra_cost), 2)
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 1
UNION
SELECT '2' AS 'hazardous level', ROUND(AVG(c.extra_cost), 2)
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 2
UNION
SELECT '3' AS 'hazardous level', ROUND(AVG(c.extra_cost), 2)
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 3;
```

hazardous level	average extra cost
0	59.56
1	84.81
2	109.67
3	134.25

2.4.4.2 Explanation

Bàsicament fem 3 UNIONS ja que hi ha 4 hazardous levels diferents. En el nivell 0 fem la mitja del cost extra dels objectes especials que compleixen que no són ni fràgils ni corrosius ni inflamables, en el nivell 1 igual però han de complir que siguin o fràgils o corrosius o inflamables, en el nivell 2 igual però han de complir dues de les tres característiques i, finalment, pel nivell 3 han de complir que siguin fràgils, corrosius i inflamables. Un objecte especial és fràgil/corrosiu/fràgil si és 1 o no ho és si és 0.

2.4.4.3 Query validation

```
SELECT '0' AS 'hazardous level', c.extra_cost, s.fragile, s.corrosive, s.flammable
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 0;
```

hazardous level	extra_cost	fragile	corrosive	flammable
0	45.6	0	0	0
0	36	0	0	0
0	99	0	0	0
0	58.7	0	0	0
0	29.5	0	0	0
0	75.7	0	0	0
0	21.4	0	0	0
0	30.8	0	0	0
0	57.9	0	0	0
0	40.3	0	0	0
0	92.5	0	0	0
0	29	0	0	0
0	22.3	0	0	0
0	66.2	0	0	0
0	60.4	0	0	0
0	27.6	0	0	0
0	84.1	0	0	0
0	98.8	0	0	0
0	77.4	0	0	0
0	85.0	0	0	0

Si posem la primera part de la query però mostrant tots els resultats, fent scroll down podem comprovar que tots compleixien la condició de que l'objecte especial no és ni fràgil ni corrosiu ni inflamable (tots estan a 0). La mitja del cost extra també és correcta ja que amb la funció AVG te la calcula directament.

```
SELECT '1' AS 'hazardous level', c.extra_cost, s.fragile, s.corrosive, s.flammable
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 1;
```

hazardous level	extra_cost	fragile	corrosive	flammable
1	107	0	1	0
1	71	0	0	1
1	124	0	1	0
1	74	0	1	0
1	139	0	0	1
1	75	0	0	1
1	75	0	1	0
1	63.5	1	0	0
1	62	0	0	1
1	40	0	0	1
1	99	0	1	0
1	41	1	0	0
1	36	1	0	0
1	36.7	1	0	0
1	55	0	1	0
1	89.9	1	0	0
1	106	0	0	1
1	21.5	1	0	0
1	36	0	1	0
1	127	0	1	0

Si posem la segona part de la query però mostrant tots els resultats, fent scroll down podem comprovar que tots compleixien la condició de que l'objecte especial o és fràgil o corrosiu o inflamable (un dels tres està a 1). La mitja del cost extra també és correcta ja que amb la funció AVG te la calcula directament.

```
SELECT '2' AS 'hazardous level', c.extra_cost, s.fragile, s.corrosive, s.flammable
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 2;
```

hazardous level	extra_cost	fragile	corrosive	flammable
2	142	1	0	1
2	90	1	1	0
2	68	1	0	1
2	153	1	1	0
2	182	1	0	1
2	83	0	1	1
2	53	1	0	1
2	76	1	1	0
2	138	0	1	1
2	101	1	1	0
2	134	0	1	1
2	86	1	1	0
2	108	1	1	0
2	79	1	1	0
2	41	1	1	0
2	142	0	1	1
2	163	0	1	1
2	138	0	1	1
2	114	1	0	1
2	40	1	1	0

Si posem la tercera part de la query però mostrant tots els resultats, fent scroll down podem comprovar que tots compleixien la condició de que l'objecte especial té 2 dels 3 atributs a 1. La mitja del cost extra també és correcta ja que amb la funció AVG te la calcula directament.

```
SELECT '3' AS 'hazardous level', c.extra_cost, s.fragile, s.corrosive, s.flammable
FROM checkedluggage as c JOIN specialobjects as s ON s.specialobjectID = c.checkedluggageID
WHERE (s.fragile + s.corrosive + s.flammable) = 3;
```

hazardous level	extra_cost	fragile	corrosive	flammable
3	203	1	1	1
3	72	1	1	1
3	184	1	1	1
3	117	1	1	1
3	95	1	1	1
3	142	1	1	1
3	58	1	1	1
3	107	1	1	1
3	154	1	1	1
3	192	1	1	1
3	143	1	1	1
3	150	1	1	1
3	56	1	1	1
3	185	1	1	1
3	91	1	1	1
3	138	1	1	1
3	108	1	1	1
3	187	1	1	1
3	210	1	1	1
3	108	1	1	1

Si posem la quarta part de la query però mostrant tots els resultats, fent scroll down podem comprovar que tots compleixen la condició de que l'objecte especial és fràgil, corrosiu i inflamable (tots estan a 1). La mitja del cost extra també és correcta ja que amb la funció AVG te la calcula directament.

2.4.5 Requirement (Query) 4.5 Claims

2.4.5.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.4.5.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.4.5.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.4.6 Requirement (Trigger) 4.6 Refunded tickets

2.4.6.1 Solution

```
DROP TABLE IF EXISTS RefundsAlterations;
CREATE TABLE RefundsAlterations(
    personID BIGINT(20),
    ticketID BIGINT(20),
    comment TEXT
);
```

```

DELIMITER $$
DROP TRIGGER IF EXISTS refundedTickets $$
CREATE TRIGGER refundedTickets
    AFTER INSERT
    ON refund
    FOR EACH ROW
BEGIN
    IF (SELECT COUNT(*)
        FROM refund AS r
        WHERE r.flightTicketID = NEW.flightTicketID AND r.accepted = 1) > 1 THEN

        INSERT INTO RefundsAlterations(personID, ticketID, comment)
        SELECT c.passengerID, NEW.flightTicketID, "Refund of a ticket already processed
correctly"
        FROM claims AS c JOIN refund AS r ON c.claimID = r.refundID
        WHERE r.flightTicketID = NEW.flightTicketID AND r.refundID = NEW.refundID;
    END IF;

    IF (SELECT COUNT(flightTicketID)
        FROM refund AS r
        WHERE flightTicketID = NEW.flightTicketID AND r.accepted = 0) >= 3 THEN

        INSERT INTO RefundsAlterations(personID, ticketID, comment)
        SELECT c.passengerID, NEW.flightTicketID, "Excessive Attempts"
        FROM claims AS c JOIN refund AS r ON c.claimID = r.refundID
        WHERE r.flightTicketID = NEW.flightTicketID AND r.refundID = NEW.refundID;
    END IF;

END $$

```

2.4.6.2 *Explanation*

Primer de tot, creem la taula RefundsAlterations amb els atributs que se'ns especifica a l'enunciat. Seguidament, creem el TRIGGER i posem que sigui AFTER INSERT ja que a l'enunciat ens diu que per cada refund que es fa, volem inserir informació a la taula RefundsAlterations si és compleixen una sèrie de condicions que ara especificarem. Posem ON refund ja que refund es la taula on hi ha el flightTicketID, si ha estat acceptat o no i, apart, a l'enunciat ens ho deixa molt clar. Dins del trigger posem el condicional de que si a la taula refund hi ha 2 o més refunds acceptades (accepted = 1) pel mateix ticketID, doncs inserim a RefundsAlterations el ID de la persona que ha fet els refunds, el ID del ticket i el comentari de "Refund of a ticket already processed correctly". Per altra banda, posem un altre condicional que es compleix si a la taula refund hi ha mínim 3 refunds rebutjats (accepted = 0) amb el mateix ticketID. Si es compleix, doncs inserirem l'ID de la persona que ha fet les refunds, el ID del ticket i el comentari de "Excessive Attempts".

2.4.6.3 *Trigger validation*

```

SELECT *
FROM refund
ORDER BY refundID DESC;

SELECT *
FROM claims

```

ORDER BY claimID DESC;

refundID	flightTicketID	argument	accepted	amount
9999	22505	Other	1	697
9998	33520	Flight Delayed	0	53
9997	59982	Other	0	38
9993	34833	Other	0	545
9992	57746	Flight Delayed	0	543
9991	47854	Flight Delayed	1	108
9990	22733	Other	1	588
9989	5501	Other	1	133
9987	45012	Other	1	909
9983	36152	Other	1	85

claimID	passengerID	date
10000	73488	2016-09-29
9999	20751	2018-05-09
9998	44253	2009-11-21
9997	15297	2019-08-23
9996	44992	2007-05-20
9995	29178	2002-08-08
9994	40809	1988-07-12
9993	39253	2002-01-11
9992	17712	2009-02-26
9991	94728	2011-05-14

Abans de res, mostrem tota la informació de les taules refund i claims ordenades pel seu ID de major a menor.

També comprovem que la taula RefundsAlterations està buida:

```
SELECT *  
FROM refundsalterations;
```

personID	ticketID	comment
----------	----------	---------

Primer de tot, fem el primer cas que és que si hi ha més d'una refund acceptada amb el mateix flightTicketID i feta pel mateix passatger, doncs hem d'inserir a la taula RefundsAlterations el personID, ticketID i el comentari de "Refund of a ticket already processed correctly".

Per fer-ho, hem de fer 2 inserts de dos (o més) refunds que tinguin el mateix flightTicketID i que les refunds estiguin acceptades. Tal que així:

```
INSERT INTO refund(refundID, flightTicketID, argument, accepted, amount)  
VALUES (10000, 74970, 'Flight Delayed', 1, 750);
```

Ara, per fer a poder fer el segon INSERT a refund, necessitem un claim amb claimID = 100001. Fem un INSERT a claim.

```
INSERT INTO claims(claimID, passengerID, date)  
VALUES (10001, 73488, CURRENT_DATE());
```

Com pel refundID = 10000, el claimID era el 10000, que té de passengerID = 73488, doncs a aquets nou claim li posem aquest passengerID i qualsevol data, la data actual per exemple. I ja podem fer l'insert del segon refund:

```
INSERT INTO refund(refundID, flightTicketID, argument, accepted, amount)  
VALUES (10001, 74970, 'Flight Delayed', 1, 750);
```

Un cop fets quests inserts, a la taula RefundsAlterations s'hauria d'haver afegit una nova fila amb les següents dades: personID = 73488, ticketID = 74970 i comment = "Refund of a ticket already processed correctly". Comprovem-ho:

```
SELECT *
```


FROM refundsalterations;

personID	ticketID	comment
73488	74970	Refund of a ticket already processed correctly

Aquesta primera condició funciona correctament, provem amb la segona. La segona ens diu que si a si a la taula refund hi ha mínim 3 refunds rebutjats amb el mateix ticketID, doncs hauríem d'inserir a la taula RefundsAlterations el personID, ticketID i el comentari de "Excessive Attempts".

Per fer-ho, hem de fer 3 inserts (o més) a la taula refund en que aquestes refunds siguin rebutjades (accepted = 0), que tinguin el mateix flightTicketID i siguin del mateix passatger. En aquest cas, com a la taula refund hi ha diversos passatgers que tenen refunds rebutjades, doncs agafarem el seu ID i així només caldrà fer 2 inserts. Per exemple, hem agafat el passatger del refundID = 9998, en que de flightTicketID té el 33520 i pel seu claimID = 9998, té de passengerID = 44253. Comencem:

```
INSERT INTO claims(claimID, passengerID, date)
VALUES (10002, 44253, '2018-01-20');
```

```
INSERT INTO claims(claimID, passengerID, date)
VALUES (10003, 44253, '2018-02-23');
```

Al igual que abans, abans necessitem fer inserts a claim pel mateix ID dels inserts de refund que farem.

Posem els claimIDs 10002 i 10003, passengerID el que acabem de dir, el 44253 i la data qualsevol. Ara si procedim a fer els inserts a refund.

```
INSERT INTO refund(refundID, flightTicketID, argument, accepted, amount)
VALUES (10002, 33520, 'Other', 0, 53);
```

```
INSERT INTO refund(refundID, flightTicketID, argument, accepted, amount)
VALUES (10003, 33520, 'Other', 0, 53);
```

El refundID 10002 i 10003, el flightTicketID el que acabem de dir, que és el 33520 i posem el accepted = 0.

Un cop fets aquests inserts, a la taula RefundsAlterations s'hauria d'haver afegit una nova fila amb les següents dades: personID = 44253, ticketID = 33520 i comment = "Excessive Attempts". Comprovem-ho:

personID	ticketID	comment
73488	74970	Refund of a ticket already processed correctly
44253	33520	Excessive Attempts

Com podem observar, funciona correctament.

2.4.7 Requirement (Trigger) 4.7 Lost object days

2.4.7.1 Solution

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

```

CREATE TABLE LostObjectsDays(
    lostObjectID BIGINT(20),
    num_days INTEGER,
    avg_num_days FLOAT,
    PRIMARY KEY (lostObjectID)
);

DELIMITER $$
DROP TRIGGER IF EXISTS lost_object_days $$
CREATE TRIGGER lost_object_days
    AFTER UPDATE
    ON lostobject
    FOR EACH ROW
BEGIN
    IF OLD.founded = 0 AND NEW.founded = 1 THEN
        INSERT INTO LostObjectsDays(lostObjectID, num_days, avg_num_days)
        SELECT NEW.lostObjectID, DATEDIFF(CURRENT_DATE(), c.date), (SELECT
        COALESCE(AVG(num_days), 0)
                                FROM LostObjectsDays AS lod JOIN lostobject AS lo ON
lod.lostObjectID = lo.lostObjectID
                                WHERE lo.description = NEW.description)
        FROM claims AS c
        WHERE c.claimID = NEW.lostObjectID;

    END IF;

END $$

DELIMITER ;

```

2.4.7.2 *Explanation*

Primer de tot, creem la taula LostObjectDays amb els atributs que se'ns especifica a l'enunciat. Seguidament, creem el TRIGGER i posem que sigui AFTER UPDATE ja que a l'enunciat ens diu que quan un objecte passa de no trobat a trobat l'hem d'inserir a la nova taula, això significa que fem de fer un update del lostobject i que passi de founded = 0 a founded = 1. Posem ON lostobject ja que és la taula que conté l'atribut founded i és on farem l'update. Dins del trigger posem el condicional de que si un objecte perdut passa de no trobat a trobat, doncs fer l'insert a la taula LostObjectDays del ID del objecte perdut (ja trobat), el número de dies que ha estat perdut i la mitjana de dies perduts que han estat aquell tipus d'objecte. Per saber quants dies ha estat perdut, doncs restem la data actual, que és quan s'ha trobat l'objecte, amb la data en que es va posar la reclamació (per això posem el FROM claim). Per fer-ho, hem usat la funció DATEDIFF, que retorna el número de dies entre les dues dates. Per calcular la mitja de dies que ha estat perdut un tipus d'objecte, primer posem la condició de que tinguin la mateixa descripció (tal i com ens diuen a l'enunciat) i després usant la funció COALESCE per a que, per exemple, pel primer lostobject que li fem update doncs com la mitja de dies perduts serà 0 però sortirà null ja que encara no hem fet l'insert a LostObjectDays, doncs fem servir aquesta funció per a que enlloc de que surti null, surti un 0.

2.4.7.3 *Trigger validation*

```
SELECT * FROM lostobject
```

ORDER BY lostObjectID DESC;

lostObjectID	luggageID	description	color	founded
9994	NULL	User-centric interactive system engine	Blue	1
9986	NULL	Centralized responsive frame	White	1
9981	NULL		White	0
9979	83198	Balanced bottom-line help-desk	Black	0
9975	NULL	Digitized logistical architecture	Pink	1
9973	NULL	Organized value-added moderator	Black	1
9972	NULL	Proactive client-driven alliance	Blue	1
9967	NULL		Yellow	1
9959	NULL	Upgradable fresh-thinking installation	Blue	1
9958	NULL		Blue	1
9954	NULL	Sharable well-modulated architecture	Yellow	1
9941	NULL		Yellow	1
9936	11260	Balanced methodical model	Blue	1
9933	NULL	Multi-channelled client-driven help-desk	Black	1

Primer de tot, mostrem la informació de la taula lostobject ordenada pel seu lostobjectID descendent, ja que en els següents passos farem diversos inserts.

SELECT * FROM lostobjectsdays;

lostObjectID	num_days	avg_num_days
NULL	NULL	NULL

Aquí comprovem que la taula LostObjectDays està buida.

INSERT INTO lostobject(lostObjectID, luggageID, description, color, founded)
VALUES (9995, NULL, 'Synchronization architected', 'Blue', 0);

INSERT INTO lostobject(lostObjectID, luggageID, description, color, founded)
VALUES (9996, NULL, 'Synchronization architected', 'Red', 0);

INSERT INTO lostobject(lostObjectID, luggageID, description, color, founded)
VALUES (9997, NULL, 'Synchronization architected', 'Green', 0);

lostObjectID	luggageID	description	color	founded
9997	NULL	Synchronization architected	Green	0
9996	NULL	Synchronization architected	Red	0
9995	NULL	Synchronization architected	Blue	0
9994	NULL	User-centric interactive system engine	Blue	1
9986	NULL	Centralized responsive frame	White	1
9981	NULL		White	0
9979	83198	Balanced bottom-line help-desk	Black	0
9975	NULL	Digitized logistical architecture	Pink	1
9973	NULL	Organized value-added moderator	Black	1
9972	NULL	Proactive client-driven alliance	Blue	1
9967	NULL		Yellow	1
9959	NULL	Upgradable fresh-thinking installation	Blue	1
9958	NULL		Blue	1
9954	NULL	Sharable well-modulated architecture	Yellow	1

Fem diferents inserts a la taula lostobjects, amb la mateixa descripció (aquesta descripció no estava prèviament a aquesta taula, és inventat) i founded = 0.

```
UPDATE lostobject SET founded = 1 WHERE lostObjectID = 9995;
```

Fem UPDATE del primer lost object que hem inserit (lostObjectID = 9995) i que passi de founded = 0 a founded = 1.

```
SELECT * FROM lostobjectsdays;
```

lostObjectID	num_days	avg_num_days
9995	6898	0
NULL	NULL	NULL

Veiem que s'ha inserit correctament a la taula LostObjectDays. El lostObjectID coincideix, per saber si el nombre de dies perduts és correcte, hem de mostrar aquest lostObjectID a la taula claim, ja que allà surt la data en que es va posar la reclamació.

```
SELECT * FROM claims  
ORDER BY claimID DESC;
```

claimID	passengerID	date
10003	44253	2018-02-23
10002	44253	2018-01-20
10001	73488	2021-06-25
10000	73488	2016-09-29
9999	20751	2018-05-09
9998	44253	2009-11-21
9997	15297	2019-08-23
9996	44992	2007-05-20
9995	29178	2002-08-08
9994	40809	1988-07-12
9993	39253	2002-01-11
9992	17712	2009-02-26
9991	61738	2011-05-11

Aleshores, si restem la CURRENT_DATE(), que avui som 2021-06-27 amb la data de la reclamació (del claimID = 9995), que és 2002-08-08, doncs la funció DATEDIFF ens retorna els dies que han passat.

```
SELECT DATEDIFF(CURRENT_DATE(), '2002-08-08') AS 'diferència en dies';
```

diferència en dies
6898

Comprovem que el resultat és correcte.

La mitja de número de dies perduts és 0 ja que és el primer cop que fem update, i aquest no el comptem.

Ara fem un nou UPDATE pel segon lostObjectID que hem inserit (lostObjectID = 9996) i que passi de founded = 0 a founded = 1.

```
UPDATE lostobject SET founded = 1 WHERE lostObjectID = 9996;
```

```
SELECT * FROM lostobjectsdays;
```

lostObjectID	num_days	avg_num_days
9995	6898	0
9996	5152	6898
NULL	NULL	NULL

Veiem que el segon objecte perdut s'ha inserit correctament a LostObjectDays.

Mirant a captura de la taula claim que hem mostrat anteriorment, veiem que pel claimID = 9996, la data de reclamació és al 2007-05-20 i la CURRENT_DATE() és el 2021-06-27. Comprovem que el número de dies perdut és correcte:

```
SELECT DATEDIFF(CURRENT_DATE(), '2007-05-20') AS 'diferència en dies';
```

diferència en dies
5152

La mitja de número de dies perduts d'aquell objecte és 6898 ja que és la mitja de dies en que han estan perdut els objectes amb descripció 'Synchronization architected'. Com només hi ha hagut un prèviament i aquest ha estat 6898 dies perduts, doncs aquesta és la mitja.

Fem el UPDATE del tercer lostobject que hem introduït (lostObjectID = 9997) i que passi de founded = 0 a founded = 1:

```
UPDATE lostobject SET founded = 1 WHERE lostObjectID = 9997;
```

Mirem si s'ha afegit a LostObjectDays:

```
SELECT * FROM lostobjectsdays;
```

lostObjectID	num_days	avg_num_days
9995	6898	0
9996	5152	6898
9997	674	6025
NULL	NULL	NULL

Mirant a captura de la taula claim que hem mostrat anteriorment, veiem que pel claimID = 9997, la data de reclamació és al 2019-08-23 i la CURRENT_DATE() és el 2021-06-27. Comprovem que el número de dies perdut és correcte:

```
SELECT DATEDIFF(CURRENT_DATE(), '2019-08-23') AS 'diferència en dies';
```

diferència en dies
674

Ara, hem de comprovar que la mitja de dies perduts és correcte. Com prèviament ja hi havien 2 objectes perduts que han passat de founded = 0 a founded = 1 i són del mateix tipus que acabem d'inserir, doncs la mitja es calcula $(6898 + 5152) / 2 = 6025$. Efectivament, concorda amb el resultat.

Per acabar, fem INSERT d'un objecte perdut que tingui una descripció diferents als tres anteriors per comprovar que la mitja de dies sigui 0.

```
INSERT INTO lostobject(lostObjectID, luggageID, description, color, founded)
VALUES (9998, NULL, 'Optimization powered', 'Red', 0);
```

Posem una descripció random que no està a la base de dades i posem el founded a 0.

Fem el update correspondent:

```
UPDATE lostobject SET founded = 1 WHERE lostObjectID = 9998;
```

Mirem si s'ha afegit a LostObjectDays:

```
SELECT * FROM lostobjectsdays;
```

lostObjectID	num_days	avg_num_days
9995	6898	0
9996	5152	6898
9997	674	6025
9998	4236	0
NULL	NULL	NULL

Mirant a captura de la taula claim que hem mostrat anteriorment, veiem que pel claimID = 9997, la data de reclamació és al 2009-11-21 i la CURRENT_DATE() és el 2021-06-27. Comprovem que el número de dies perdut és correcte:

```
SELECT DATEDIFF(CURRENT_DATE(), '2009-11-21') AS 'diferència en dies';
```

diferència en dies
4236

Finalment, la mitja de dies perduts és correcte i es 0 ja que prèviament no hi hagut cap objecte perdut d'aquest tipus (mateixa descripció) que hagi passat de no trobat a trobat.

2.4.8 Requirement (Event) 4.8 Transported statistics

2.4.8.1 Solution

TODO: Write the solution of the event including the creation of any table

2.4.8.2 Explanation

TODO: To explain how the event works using 5-10 sentences.

2.4.8.3 Trigger validation

TODO: Write the tests you have done to test that the event works: insert... update... delete... You can include results of any query.

2.5 Cross module queries

2.5.1 Requirement (Query) 5.1 Overbooked airlines

2.5.1.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

2.5.1.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.5.1.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

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

2.5.2.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.5.2.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.5.2.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.5.3 Requirement (Query) 5.3 Flight attendant vacancy

2.5.3.1 Solution

```
SELECT pe.name AS 'passenger name', pe.surname, pe.phone_number AS 'Phone number',
COUNT(lp.languageID) AS '# languages spoken'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN languageperson AS lp ON lp.personID = pe.personID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN flight AS f ON f.flightID = ft.flightID
JOIN route AS r ON f.flightID = r.routeID
JOIN airport AS ar ON r.departure_airportID = ar.airportID
JOIN city AS cy ON cy.cityID = ar.cityID
WHERE (SELECT COUNT(lp2.languageID)
      FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID = pa2.passengerID
      JOIN languageperson AS lp2 ON lp2.personID = pe2.personID
      WHERE pe2.personID = pe.personID
      GROUP BY pe2.personID) >= 2
AND (SELECT l4.languageID
     FROM language AS l4
     WHERE l4.name = "Chavacano") IN (SELECT lp3.languageID
```

```

FROM person AS pe3 JOIN passenger AS pa3 ON
pe3.personID = pa3.passengerID
JOIN languageperson AS lp3 ON lp3.personID =
pe3.personID
WHERE pe3.personID = pe.personID)
AND ABS(cy.timezone - (SELECT cy2.timezone
FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID =
pa2.passengerID
JOIN flighttickets AS ft2 ON pa2.passengerID = ft2.passengerID
JOIN flight AS f2 ON f2.flightID = ft2.flightID
JOIN route AS r2 ON f2.flightID = r2.routeID
JOIN airport AS ar2 ON r2.destination_airportID = ar2.airportID
JOIN city AS cy2 ON cy2.cityID = ar2.cityID
WHERE pe2.personID = pe.personID
AND r.routeID = r2.routeID
GROUP BY cy2.cityID)) >= 3
GROUP BY pe.personID;

```

passenger name	surname	Phone number	# languages spoken
Jeffry	Alloway	+86 442 483 6691	2
Bryanty	Shory	+965 133 396 9574	2
Emelen	Tuxill	+54 425 822 9376	2
Bowie	Franca	+86 148 886 7529	2
Web	Pischel	+86 343 530 9882	2
Rhetta	Santoro	+63 482 848 0819	3
Durand	Bearman	+7 469 130 5151	2
Clim	Mucklo	+420 639 614 8184	6
Leeanne	Oxnam	+386 535 807 0526	2
Andras	Silverstone	+66 259 417 4314	2
Alister	Lansdale	+502 879 970 1517	2
Sallyanne	Dictus	+55 253 922 2204	2
Carina	Dubois	+62 979 814 8567	2
Boycey	Younghus...	+299 501 388 0686	2
Weston	Karlicek	+57 659 610 1463	2
Petronille	Savin	+62 450 706 1430	2

2.5.3.2 Explanation

Fem el SELECT de les dades que ens demanen, i els JOINS. En la primera SUBQUERY comprovem que el passatger parla mínim 2 idiomes. A la segona i tercera SUBQUERY, fem que l'idioma "Chavacano" estigui dins de la llista dels idiomes (fem servir el IN) que parla el passatger, és a dir, que dels 2 o més idiomes que parla el passatger, un d'ells es el "Chavacano". I finalment, posem la condició de que la diferencia horària entre les ciutats origen i destí d'alguna de les rutes dels vols que ha fet el passatger sigui major o igual a 3 hores, tal com se'ns especifica a l'enunciat. Fem el valor absolut de la resta entre les dues hores ja que la timezone pot ser negativa.

2.5.3.3 Query validation

Primer posem la mateixa query que la principal però afegint-li el ID del passatger per a posteriori fer les comprovacions:

```

SELECT pe.name AS 'passenger name', pe.surname, pe.phone_number AS 'Phone number',
COUNT(lp.languageID) AS '# languages spoken', pe.personID

```



```

FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN languageperson AS lp ON lp.personID = pe.personID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN flight AS f ON f.flightID = ft.flightID
JOIN route AS r ON f.flightID = r.routeID
JOIN airport AS ar ON r.departure_airportID = ar.airportID
JOIN city AS cy ON cy.cityID = ar.cityID
WHERE (SELECT COUNT(lp2.languageID)
      FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID = pa2.passengerID
      JOIN languageperson AS lp2 ON lp2.personID = pe2.personID
      WHERE pe2.personID = pe.personID
      GROUP BY pe2.personID) >= 2
AND (SELECT l4.languageID
      FROM language AS l4
      WHERE l4.name = "Chavacano") IN (SELECT lp3.languageID
      FROM person AS pe3 JOIN passenger AS pa3 ON
pe3.personID = pa3.passengerID
      JOIN languageperson AS lp3 ON lp3.personID =
pe3.personID
      WHERE pe3.personID = pe.personID)
AND ABS(cy.timezone - (SELECT cy2.timezone
      FROM person AS pe2 JOIN passenger AS pa2 ON pe2.personID =
pa2.passengerID
      JOIN flighttickets AS ft2 ON pa2.passengerID = ft2.passengerID
      JOIN flight AS f2 ON f2.flightID = ft2.flightID
      JOIN route AS r2 ON f2.flightID = r2.routeID
      JOIN airport AS ar2 ON r2.destination_airportID = ar2.airportID
      JOIN city AS cy2 ON cy2.cityID = ar2.cityID
      WHERE pe2.personID = pe.personID
      AND r.routeID = r2.routeID
      GROUP BY cy2.cityID)) >= 3
GROUP BY pe.personID;

```

passenger name	surname	Phone number	# languages spoken	personID
Randee	Ivashkin	+86 261 146 2524	2	2002
Heinrick	Courteney	+86 453 889 1385	3	14452
Jerrine	Fishleigh	+86 541 620 6313	6	14563
Jeffry	Alloway	+86 442 483 6691	2	15518
Bryanty	Shory	+965 133 396 9574	2	16389
Emelen	Tuxill	+54 425 822 9376	2	25346
Bowie	Franca	+86 148 886 7529	2	29019
Web	Pischel	+86 343 530 9882	2	33272
Rhetta	Santoro	+63 482 848 0819	3	34997
Durand	Bearman	+7 469 130 5151	2	38554
Clim	Mucklo	+420 639 614 8184	6	41585
Leeanne	Oxnam	+386 535 807 0526	2	48965
Andras	Silverstone	+66 259 417 4314	2	51187
Alister	Lansdale	+502 879 970 1517	2	53417
Sallyanne	Dictus	+55 253 922 2204	2	56789
Carina	Dubois	+62 979 814 8567	2	57701
Boycey	Younghus...	+299 501 388 0686	2	76081
Weston	Karlicek	+57 659 610 1463	2	80790
Petronille	Savin	+62 450 706 1430	2	91210

Ara, comprovem pel primer passatger que ens surt, que té un ID = 2002, que parla mínim 2 idiomes i que un d'ells és el "Chavacano".

```
SELECT lp.personID, l.languageID, l.name AS 'nom del idioma'
FROM languageperson AS lp JOIN language AS l ON lp.languageID = l.languageID
WHERE personID = 2002;
```

personID	languageID	nom del idioma
2002	16	Chavacano
2002	21	English

Finalment, comprovem que de tots els vols que ha agafat el passatger amb ID = 2002, hi hagi algun en que la diferència horària sigui mínim de 3 hores.

```
SELECT pe.personID, f.flightID, cy.timezone, cy2.timezone, ABS((cy.timezone) - (cy2.timezone))
AS 'diferència horària'
FROM person AS pe JOIN passenger AS pa ON pe.personID = pa.passengerID
JOIN flighttickets AS ft ON pa.passengerID = ft.passengerID
JOIN flight AS f ON f.flightID = ft.flightID
JOIN route AS r ON f.flightID = r.routeID
JOIN airport AS ar2 ON r.destination_airportID = ar2.airportID
JOIN airport AS ar ON r.departure_airportID = ar.airportID
JOIN city AS cy2 ON cy2.cityID = ar2.cityID
JOIN city AS cy ON cy.cityID = ar.cityID
WHERE pe.personID = 2002;
```

personID	flightID	timezone	timezone	diferència horària
2002	5197	-6	-5	1
2002	14350	-9	-4	5

Efectivament, el vol amb ID = 14350, la diferència horària entres les ciutats origen i destí de la ruta és major a 3.

Per comprovar que tots els resultats de la query principal són correctes, hauríem de canviar el ID de la persona pel que vulguem mirar.

2.5.4 Requirement (Query) 5.4 Smuggling airports

2.5.4.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.5.4.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.5.4.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.5.5 Requirement (Query) 5.5 Reliable plane types

2.5.5.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.5.5.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.5.5.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

2.5.6 Requirement (Query) 5.6 Employees languages

2.5.6.1 Solution

TODO: Write the query in text following this example:

```
SELECT name
FROM table
WHERE name LIKE "%a"
ORDER BY name
```

TODO: Include a picture of the result.

2.5.6.2 Explanation

TODO: To explain the query using 4-5 sentences.

2.5.6.3 Query validation

TODO: Explain why the query works, you can use other queries to support the explanation.

3 LSAIR Graph database requirements

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

Per aquest Case Study, hem hagut de muntar les relacions entre les següents entitats: Airport, Plane, Country i City. Per fer-ho, hem de tenir en compte que només hem de seleccionar els avions que s'han retirat en els tres últims anys i que pertanyin a aerolínies situades en un país que el nom comenci per la lletra 'S'. Un cop seleccionada la informació anterior, dins d'aquesta selecció, hem de mostrar els avions que hagin completat una ruta. Finalment, havíem de seleccionar els avions que tenien com a aeroport destí o origen algun dels mostrats en el Dataset 2.

3.1.1 Data migration

En primer lloc, vam fer les queries per obtenir les dades que ens demanaven en l'enunciat, un cop fetes i validades, és a dir comprovar que la informació fos correcte, vam inserir la informació de les queries en unes taules creades específicament per guardar i organitzar-ho tot.

```
DROP TABLE IF EXISTS Dataset1 CASCADE;
CREATE TABLE Dataset1(
  planeID BIGINT(20),
  retirement_year bigint(20),
  type_plane VARCHAR(255),
  name VARCHAR(255),
  maintenance_count INTEGER,
  piece_count INTEGER,
  cost_sum BIGINT(20),

  PRIMARY KEY (planeID)
);
INSERT INTO Dataset1(planeID, retirement_year, type_plane, name, maintenance_count,
piece_count, cost_sum)
SELECT p.planeID, p.retirement_year, pt.type_name, a.name, COUNT(m.maintenanceID),
COUNT(pm.maintenanceID), SUM(pc.cost)
FROM plane AS p JOIN planetype AS pt ON pt.planetypeID = p.planetypeId
JOIN airline AS a ON p.airlineID = a.airlineID
JOIN maintenance AS m ON m.planeID = p.planeID
JOIN piecemaintenance AS pm ON m.maintenanceID = pm.maintenanceID
JOIN piece AS pc ON pc.pieceID = pm.pieceID
JOIN country AS cy ON a.countryID = cy.countryID
WHERE cy.name LIKE 'S%'
AND (EXTRACT(YEAR FROM current_date()) - p.retirement_year) < 3
GROUP BY planeID
UNION
SELECT p.planeID, p.retirement_year, pt.type_name, a.name, 0, 0, 0
FROM plane AS p JOIN planetype AS pt ON pt.planetypeID = p.planetypeId
```

```

JOIN airline AS a ON p.airlineID = a.airlineID
JOIN country AS cy ON a.countryID = cy.countryID
WHERE cy.name LIKE 'S%' AND NOT EXISTS(SELECT * FROM maintenance AS m WHERE
m.planeID = p.planeID)
AND (EXTRACT(YEAR FROM current_date()) - p.retirement_year) < 3
GROUP BY planeID;

```

-- Dataset 12

```

DROP TABLE IF EXISTS Dataset12 CASCADE;
CREATE TABLE Dataset12(
    planeID BIGINT(20),
    departure BIGINT(20),
    destination BIGINT(20),
    PRIMARY KEY(planeID,departure,destination),
    FOREIGN KEY (planeID) REFERENCES Dataset1(planeID)
);
INSERT INTO Dataset12(planeID,departure, destination)
SELECT DISTINCT f.planeID,r.departure_airportID, r.destination_airportID
FROM flight as f JOIN route AS r ON r.routeID=f.routeID
WHERE f.planeID IN (select planeID from dataset1);

```

-- Dataset2

```

DROP TABLE IF EXISTS Dataset2 CASCADE;
CREATE TABLE Dataset2(
    airportID BIGINT(20),
    airport_name VARCHAR(255),
    airport_altitude INT(11),
    cityID BIGINT(20),
    city_name VARCHAR(255),
    city_timezone INT(11),
    country_name VARCHAR(255),

    PRIMARY KEY(airportID)
);

INSERT INTO Dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone,
country_name)
SELECT DISTINCT ap.airportID, ap.name, ap.altitude, c.cityID, c.name, c.timezone, cy.name
FROM Dataset12 AS d12 JOIN airport AS ap ON ap.airportID = d12.destination
JOIN city AS c ON ap.cityID = c.cityID
JOIN country AS cy ON c.countryID = cy.countryID
UNION
SELECT DISTINCT ap.airportID, ap.name, ap.altitude, c.cityID, c.name, c.timezone, cy.name
FROM Dataset12 AS d12 JOIN airport AS ap ON ap.airportID = d12.departure
JOIN city AS c ON ap.cityID = c.cityID
JOIN country AS cy ON c.countryID = cy.countryID;

```

```

-- INSERT per fer les querys
-- insert all routes lagos south korea query 5

```

```

INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone,
country_name) VALUES(2272,'Gimpo International Airport',59,2175,'Seoul',9,'South Korea');
INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone,
country_name) VALUES(2265,'Jeju International Airport',118,2168,'Cheju',9,'South Korea');
INSERT INTO Dataset12(planeID, departure, destination) VALUES(1259,2272,2957);
INSERT INTO Dataset12(planeID, departure, destination) VALUES(1259,2265,2957);

```

-- QUERY4

```

INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone,
country_name) VALUES(5195,'Sembawang Air Base',86,2997,'Sembawang',8,'Singapore');
INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone,
country_name) VALUES(1430,'Limnos Airport',14,1373,'Limnos',2,'Greece');

```

```

INSERT INTO Dataset1(planeID, retirement_year, type_plane, name, maintenance_count,
piece_count, cost_sum) VALUES(1,2020,'Spain AM','NGM',5,5,2195297);
INSERT INTO Dataset1(planeID, retirement_year, type_plane, name, maintenance_count,
piece_count, cost_sum) VALUES(2,2020,'Spain AM2','NGM',6,6,2995297);

```

```

INSERT INTO Dataset12(planeID, departure, destination) VALUES(1344,5195,1430);
INSERT INTO Dataset12(planeID, departure, destination) VALUES(1,5195,1473);
INSERT INTO Dataset12(planeID, departure, destination) VALUES(2,1473,1430);

```

Un cop creades i emplenades totes les taules, vam exportar la informació obtinguda a format csv, la exportació la vam fer des de el propi MySQL. Quan ja teníem tots els csvs creats, els vam penjar a les fulles de càlcul de google, on vam posar accés públic perquè tothom pogués accedir-hi. Un cop tot penjat al núvol, vam copiar l'enllaç de cada csv i vam modificar el link per poder importar tota la informació en la nostra base de dades del Neo4j. Finalment, amb els enllaços ja creats, vam importar tota la informació amb les següents comandes:

-- Dataset1

```

LOAD CSV WITH HEADERS FROM 'https://docs.google.com/spreadsheets/d/1RwUs_cg-
FIfKNncIQfWJ-a32LNR2YiWOL-R7DwaW6P4/gviz/tq?tqx=out:csv&sheet=0' AS d1
CREATE (p:Plane{planeID: ToInteger(d1.planeID), retirement_year:
ToInteger(d1.retirement_year),plane_type_name:d1.type_plane, airline: d1.name, maintenances:
ToInteger(d1.maintenance_count), pieces: ToInteger(d1.piece_count), total_cost:
ToInteger(d1.cost_sum)});

```

-- Dataset2

```

LOAD CSV WITH HEADERS FROM 'https://docs.google.com/spreadsheets/d/1Q4CG--
lyg29N4q9MUW3KJgi94OUJ7-QRLhcPQyXAdXw/gviz/tq?tqx=out:csv&sheet=0' AS d2
CREATE(a:Airport {airportID: ToInteger(d2.airportID), airport_name: d2.airport_name, altitude:
ToInteger(d2.airport_altitude)})
MERGE(c:City{ cityID: ToInteger(d2.cityID), city_name: d2.city_name, time_zone:
ToInteger(d2.city_timezone)})
MERGE(co:Country{ country_name: d2.country_name})
CREATE(a)-[:Located_in]->(c)
CREATE(c)-[:City_from]->(co)

```

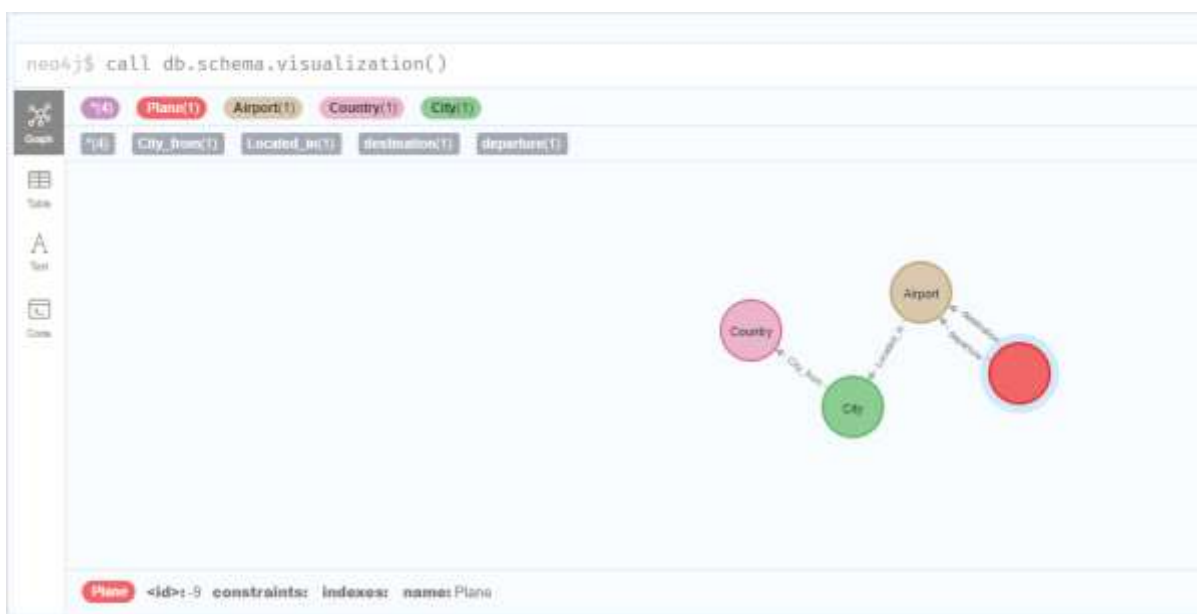
--Dataset12

```

LOAD          CSV          WITH          HEADERS          FROM
'https://docs.google.com/spreadsheets/d/1yAfXPbNu_hoZc05ZxNS0ijnpi-
edzbJ75xZGDhX4ItA/gviz/tq?tqx=out:csv&sheet=0'AS d12 WITH d12
MATCH(a:Airport)
MATCH(a2:Airport)
MATCH(p:Plane)
WHERE p.planeID = ToInteger(d12.planeID)
AND a.airportID = ToInteger(d12.departure)
AND a2.airportID= ToInteger(d12.destination)
CREATE
(p)-[:departure]->(a),
(p)-[:destination]->(a2)

```

3.1.2 Neo4J Data model



Com podem veure en la captura la nostra estructura principal per aquest “Case Study 1” es centra en la creació dels nodes següents: Country, City, Airport i Plane, els quals estan vinculats per les seves relacions, en el cas de Country i City és relacionen per “City_from”, ja que la ciutat és troba dins del país, Aiport-City es relacionen per “Located_in”, ja que un aeroport es troba dins en una ciutat, i finalment Aiport-Plane, té dues relacions, ja que com se’ns indica a l’enunciat, l’aeroport pot ser el de destí(destination_airport) o el d’origen(departure_airport), així els podem diferenciar.

3.1.3 Query 1

3.1.3.1 Solution

```

MATCH (p:Plane)
WHERE NOT (p)-[:departure]->(:Airport) AND NOT (p)-[:destination]->(:Airport) AND
p.pieces<8
RETURN p

```

```
neo4j$ MATCH (p:Plane) WHERE NOT (p)-[:departure]->(:Airport) AND NOT (p)-[:destination]->(:Airport) AND p.pieces<8 RETURN p
```

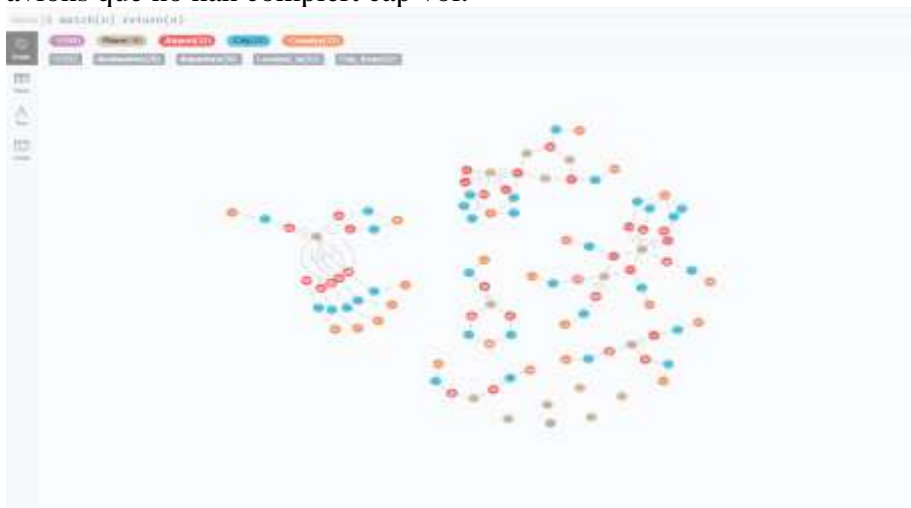
"p"
{ "pieces": 5, "total_cost": 2095207, "planeID": 651, "plane_type_name": "Canadair Regional Jet 900", "retirement_year": 2020, "airline": "Iberia Airlines", "maintenances": 15 }
{ "pieces": 0, "total_cost": 0, "planeID": 1113, "plane_type_name": "Airbus A340-300", "airline": "Scandinavian Airlines System", "retirement_year": 2021, "maintenances": 0 }
{ "pieces": 0, "total_cost": 0, "planeID": 1758, "plane_type_name": "Airbus A319", "retirement_year": 2021, "airline": "SilkAir", "maintenances": 0 }
{ "pieces": 0, "total_cost": 0, "planeID": 3404, "plane_type_name": "De Havilland Canada DHC-2-400 Dash 8Q", "retirement_year": 2021, "airline": "Mova Airline", "maintenances": 0 }
{ "pieces": 0, "total_cost": 0, "planeID": 3782, "plane_type_name": "Boeing 767", "retirement_year": 2020, "airline": "Iberia Airlines", "maintenances": 0 }
{ "pieces": 7, "total_cost": 3831793, "planeID": 9949, "plane_type_name": "Airbus A319", "retirement_year": 2020, "airline": "Vueling Airlines", "maintenances": 7 }

3.1.3.2 Explanation

En primer lloc seleccionem el planes amb la condició que no hagin realitzat cap vol, és a dir que no hagin ni sortit ni arribat a cap aeroport, com podem veure en el where. A més a més també hem de seleccionar només els avions que se li hagin reemplaçat menys de 8 peces. Finalment, retornem els avions que compleixen totes les condicions anteriors.

3.1.3.3 Query validation

En primer lloc amb la següent comanda: MATCH (n) RETURN (n), podem comprovar que hi ha 6 avions que no han complert cap vol.



Aquí mirem que realment tingui menys de 8 peces reemplaçades.

```
MATCH (p:Plane)
WHERE NOT (p)-[:departure]->(:Airport) AND NOT (p)-[:destination]->(:Airport) AND
p.pieces<8
RETURN p, p.pieces
```



```
neo4j$ MATCH (p:Plane) WHERE NOT (p)-[:departure]->(a:Airport) AND NOT (p)-[:destination]->(a:Airport) AND p.pieces<8 RETURN p, p.pieces
```

"p"	"p.pieces"
{ "pieces":5, "total_cost":2198297, "planeID":651, "plane_type_name":"Canal Regional Jet 300", "airline":"Iberia Airlines", "retirement_year":2020, "maintenance":5 }	
{ "pieces":0, "total_cost":0, "planeID":1113, "plane_type_name":"Airbus A3 40-300", "airline":"Scandinavian Airlines System", "retirement_year":2001, "maintenance":0 }	
{ "pieces":0, "total_cost":0, "planeID":1799, "plane_type_name":"Airbus A3 30", "airline":"El Al", "retirement_year":2021, "maintenance":0 }	
{ "pieces":0, "total_cost":0, "planeID":3406, "plane_type_name":"De Havilland Canada DHC-6-400 Dash 8Q", "airline":"Norse Airlines", "retirement_year":2021, "maintenance":0 }	
{ "pieces":0, "total_cost":0, "planeID":1792, "plane_type_name":"Boeing 76 0", "airline":"Iberia Airlines", "retirement_year":2020, "maintenance":0 }	
{ "pieces":7, "total_cost":3881795, "planeID":4949, "plane_type_name":"Airbus A318", "airline":"Wotling Airlines", "retirement_year":2020, "maintenance":7 }	

3.1.4 Query 2

3.1.4.1 Solution

```
MATCH (p:Plane)-[:departure]->(a:Airport)
WITH p, collect(distinct a) as dif_airport, count(distinct a) as count_airport
MATCH (p:Plane)-[:destination]->(a2:Airport)
WHERE NOT a2 IN dif_airport
WITH p as plane, count_airport + count(distinct a2) as total_airports
ORDER BY total_airports DESC
RETURN plane, total_airports
```

```
neo4j$ MATCH (p:Plane)-[:departure]->(a:Airport) WITH p, collect(distinct a) as dif_airport, count(distinct a) as count_airport MATCH (p:Plane)-[:destination]->(a2:Airport) WHERE NOT a2 IN dif_airport WITH p as plane, count_airport + count(distinct a2) as total_airports ORDER BY total_airports DESC RETURN plane, total_airports
```

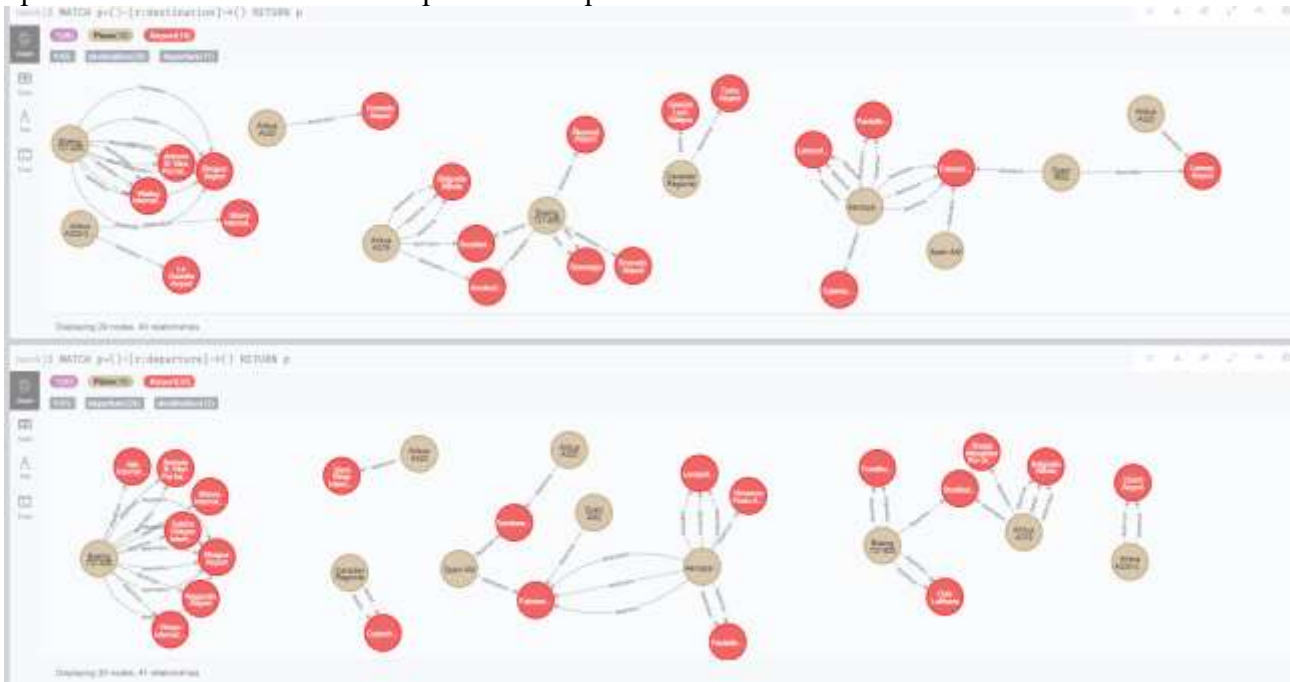
"plane"	"total_airports"
{ "pieces":4, "total_cost":3990328, "planeID":1244, "plane_type_name":"Boeing 737-800", "airline":"Fly Air", "retirement_year":2018, "maintenance":0 }	
{ "pieces":0, "total_cost":0, "planeID":1113, "plane_type_name":"Boeing 76 0", "airline":"Scandinavian Airlines System", "retirement_year":2001, "maintenance":0 }	
{ "pieces":14, "total_cost":4594912, "planeID":1244, "plane_type_name":"Boeing 737-800", "airline":"Fly Air", "retirement_year":2018, "maintenance":14 }	
{ "pieces":0, "total_cost":0, "planeID":479, "plane_type_name":"Airbus A318", "airline":"Fly Air", "retirement_year":2021, "maintenance":0 }	
{ "pieces":10, "total_cost":600536, "planeID":610, "plane_type_name":"Canal Regional Jet 300", "airline":"Scandinavian Airlines System", "retirement_year":2020, "maintenance":10 }	
{ "pieces":0, "total_cost":0, "planeID":1792, "plane_type_name":"Airbus A318", "airline":"Iberia Airlines", "retirement_year":2020, "maintenance":0 }	
{ "pieces":0, "total_cost":2198297, "planeID":651, "plane_type_name":"Canal Regional Jet 300", "airline":"Iberia Airlines", "retirement_year":2020, "maintenance":0 }	
{ "pieces":0, "total_cost":1299507, "planeID":1113, "plane_type_name":"Boeing 76 0", "airline":"Scandinavian Airlines System", "retirement_year":2001, "maintenance":0 }	
{ "pieces":14, "total_cost":4594912, "planeID":1244, "plane_type_name":"Boeing 737-800", "airline":"Fly Air", "retirement_year":2018, "maintenance":14 }	
{ "pieces":0, "total_cost":0, "planeID":1799, "plane_type_name":"Airbus A330-300", "airline":"El Al", "retirement_year":2021, "maintenance":0 }	
{ "pieces":7, "total_cost":3881795, "planeID":4949, "plane_type_name":"Airbus A318", "airline":"Wotling Airlines", "retirement_year":2020, "maintenance":7 }	
{ "pieces":0, "total_cost":0, "planeID":3406, "plane_type_name":"De Havilland Canada DHC-6-400 Dash 8Q", "airline":"Norse Airlines", "retirement_year":2021, "maintenance":0 }	

3.1.4.2 Explanation

En primer lloc seleccionem tant l'avió que surt (departure) d'un aeroport i les arribades(destination) que te el mateix avió a un altre aeroport, així tenim en compte tots els aeroports on cada avió ha estat. Com se'n indica en l'enunciat, ordenem la llista descendentment depenent del nombre d'aeroports on ha estat cada avió.

3.1.4.3 Query validation

A continuació mostrem les arribades i sortides de cada avió, així podem tots els aeroports que ha estat i podem contar el número d'aeroports en els que ha estat.



Per obtenir les imatges anterior posem: `MATCH p=()-[r:destination]->() RETURN p`, per la primera foto, i `MATCH p=()-[r:departure]->() RETURN p`, per la segona foto.

Aquí, en canvi, mostrem el nom de tots el aeroports sense repetir, d'on ha estat cada avió i ho mostrem en forma de taula, d'aquesta manera podem comprovar el nom de cada node, i podem validar que és igual a la imatge anterior.

```
MATCH (p:Plane)-[:departure]->(a:Airport)
WITH p, collect(distinct a.airport_name) as dif_airport, count(distinct a) as count_airport
MATCH (p:Plane)-[:destination]->(a2:Airport)
WHERE NOT a2.airport_name IN dif_airport
WITH p as plane, count_airport + count(distinct a2) as
total_airports,dif_airport+collect(a2.airport_name) as collect_airports
ORDER BY total_airports DESC
RETURN plane,total_airports,collect_airports
```

Query 3: MATCH (p:Plane)-[:departure]->(a:Airport)-[:Located_in]->()[:City_from]->(c:Country) WHERE a.altitude > 100 WITH p, collect(distinct c) as dep_countries, count(distinct c) as count_airport MATCH (p2:Plane)-[:destination]->(a2:Airport)-[:Located_in]->()[:City_from]->(c:Country) WHERE p2.planeID = p.planeID AND a2.altitude > 100 AND NOT c IN dep_countries WITH p as plane, count_countries + count(distinct c) as total_countries RETURN plane, total_countries

plane	total_countries	count_airport
{ "plane": 1, "total_cost": 1000000, "planeID": 1001, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Haguel Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 2, "total_cost": 1000000, "planeID": 1002, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 3, "total_cost": 1000000, "planeID": 1003, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 4, "total_cost": 1000000, "planeID": 1004, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 5, "total_cost": 1000000, "planeID": 1005, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 6, "total_cost": 1000000, "planeID": 1006, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 7, "total_cost": 1000000, "planeID": 1007, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 8, "total_cost": 1000000, "planeID": 1008, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 9, "total_cost": 1000000, "planeID": 1009, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]
{ "plane": 10, "total_cost": 1000000, "planeID": 1010, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1	["Antonio M. G. S. Per. International Airport", "Hague Airport", "Hague International Airport", "Hague Airport International Airport", "Hague Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport", "Hague International Airport"]

3.1.5 Query 3

3.1.5.1 Solution

```

MATCH (p:Plane)-[:departure]->(a:Airport)-[:Located_in]->()[:City_from]->(c:Country)
WHERE a.altitude > 100
WITH p, collect(distinct c) as dep_countries, count(distinct c) as count_countries
MATCH (p2:Plane)-[:destination]->(a2:Airport)-[:Located_in]->()[:City_from]->(c:Country)
WHERE p2.planeID = p.planeID AND a2.altitude > 100 AND NOT c IN dep_countries
WITH p as plane, count_countries + count(distinct c) as total_countries
RETURN plane, total_countries

```

Query 3: MATCH (p:Plane)-[:departure]->(a:Airport)-[:Located_in]->()[:City_from]->(c:Country) WHERE a.altitude > 100 WITH p, collect(distinct c) as dep_countries, count(distinct c) as count_countries MATCH (p2:Plane)-[:destination]->(a2:Airport)-[:Located_in]->()[:City_from]->(c:Country) WHERE p2.planeID = p.planeID AND a2.altitude > 100 AND NOT c IN dep_countries WITH p as plane, count_countries + count(distinct c) as total_countries RETURN plane, total_countries

plane	total_countries
{ "plane": 1, "total_cost": 1000000, "planeID": 1001, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 2, "total_cost": 1000000, "planeID": 1002, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 3, "total_cost": 1000000, "planeID": 1003, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 4, "total_cost": 1000000, "planeID": 1004, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 5, "total_cost": 1000000, "planeID": 1005, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 6, "total_cost": 1000000, "planeID": 1006, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 7, "total_cost": 1000000, "planeID": 1007, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 8, "total_cost": 1000000, "planeID": 1008, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 9, "total_cost": 1000000, "planeID": 1009, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1
{ "plane": 10, "total_cost": 1000000, "planeID": 1010, "plane_type_name": "Boeing 737-800", "airline": "Ryan Air", "retirement_year": 2019, "maintenance": 0 }	1

3.1.5.2 Explanation

En primer lloc seleccionem els avions que hagin sortit o aterrat a un aeroport, el qual està situat en una ciutat. També, hem de mirar que l'aeroport està a una altitud superior a 100. Finalment mostrem els avions i el total de països diferents on ha estat.

3.1.5.3 Query validation

En aquesta primera query, mostrem els països on ha estat, així podem validar que el count està fent la seva funció i que els països són diferents.

```

MATCH (p:Plane)-[:departure]->(a:Airport)-[:Located_in]->()[:City_from]->(c:Country)
WHERE a.altitude > 100
WITH p, collect(distinct c) as dep_countries, count(distinct c) as count_countries
MATCH (p2:Plane)-[:destination]->(a2:Airport)-[:Located_in]->()[:City_from]->(c:Country)
WHERE p2.planeID = p.planeID AND a2.altitude > 100 AND NOT c IN dep_countries

```

```
new $ MATCH (p:Plane)-[:departures]->(a:Airport)-[:located_in]->(c1:City)->(c2:Country) WHERE a.altitude > 100 WITH p, collect(distinct c) as dep_countries, com
```

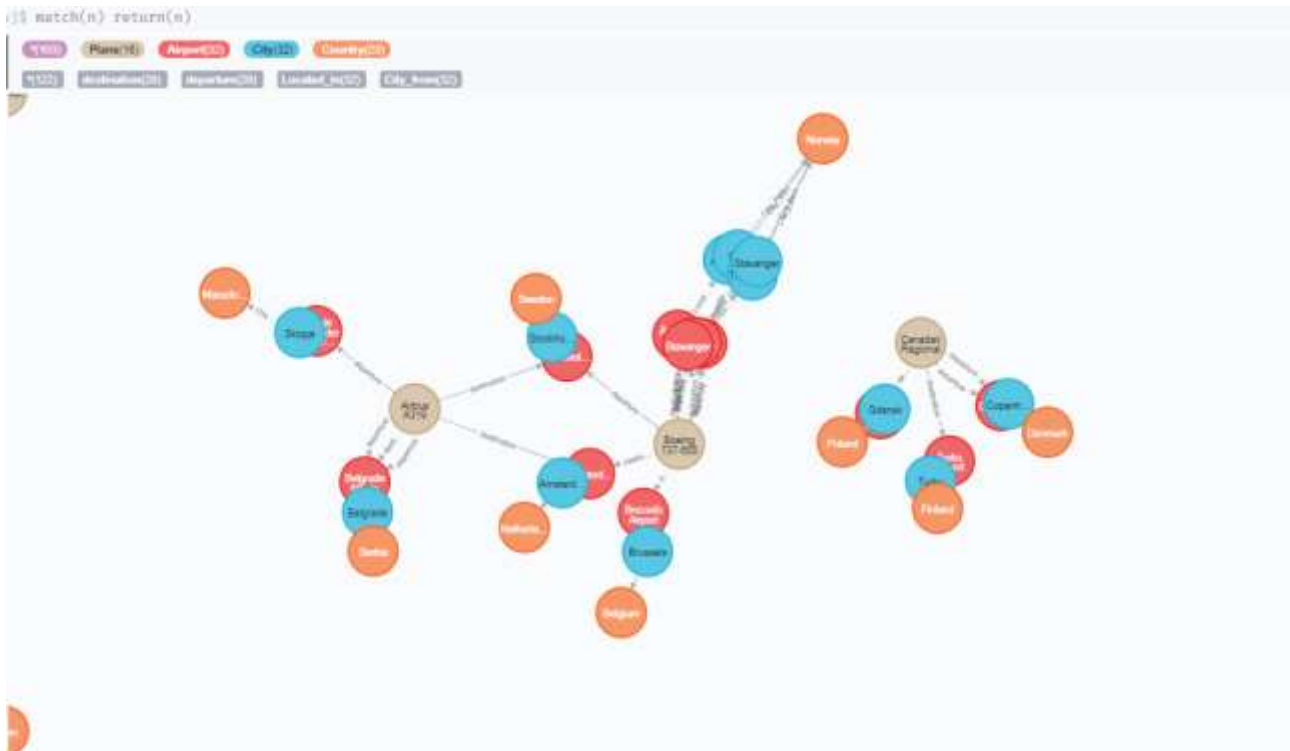
```

MATCH (p:Plane)-[:departure]->(a:Airport)-[:Located_in]->()-[:City_from]->(c:Country)
WITH p, collect(distinct c) as dep_countries, count(distinct c) as count_countries
MATCH (p2:Plane)-[:destination]->(a2:Airport)-[:Located_in]->()-[:City_from]->(c:Country)
WHERE p2.planeID =p.planeID AND NOT c IN dep_countries
WITH p as plane, count_countries + count(distinct c ) as total_countries,collect(distinct c)+
dep_countries as total_c
RETURN plane,total_countries,total_c

```

"plane"	"total_passenger"	"total_cargo"
("plane":12,"total_pass":8888336,"planeID":104,"plane_type_name":"Bombardier CRJ-900","airline":"Canadian Airlines System","retirement_year":2000,"maintenance":12)	22	("country_name":"Finland"), ("country_name":"Finland"), ("country_name":"Denmark")
("plane":15,"total_pass":6,"planeID":1057,"plane_type_name":"Bombardier CRJ-900","airline":"Canadian Airlines System","retirement_year":2010,"maintenance":1)	1	("country_name":"Belgium"), ("country_name":"Belgium"), ("country_name":"Belgium"), ("country_name":"Belgium")
("plane":14,"total_pass":2443107,"planeID":104,"plane_type_name":"Boeing 747-400","retirement_year":2017,"airline":"KLM","maintenance":8)	1	("country_name":"Sweden"), ("country_name":"Italy")
("plane":16,"total_pass":6,"planeID":1061,"plane_type_name":"Airbus A320neo","airline":"KLM International Air Lines","retirement_year":2020,"maintenance":0)	1	("country_name":"United States"), ("country_name":"Bulgaria")
("plane":10,"total_pass":18,"planeID":107,"plane_type_name":"Airbus A319neo","airline":"KLM","retirement_year":2011,"maintenance":0)	1	("country_name":"Netherlands"), ("country_name":"Sweden"), ("country_name":"Netherlands")
("plane":18,"total_pass":118110,"planeID":1176,"plane_type_name":"Boeing 737-800","airline":"KLM","retirement_year":2017,"maintenance":8)	7	("country_name":"Italy"), ("country_name":"France"), ("country_name":"Netherlands"), ("country_name":"Netherlands"), ("country_name":"Netherlands"), ("country_name":"Netherlands"), ("country_name":"Netherlands")
("plane":15,"total_pass":18,"planeID":107,"plane_type_name":"Airbus A320neo","airline":"KLM International Air Lines","retirement_year":2017,"maintenance":0)	1	("country_name":"Indonesia"), ("country_name":"Netherlands")
("plane":16,"total_pass":118110,"planeID":1176,"plane_type_name":"Boeing 737-800","airline":"KLM","retirement_year":2017,"maintenance":8)	2	("country_name":"Italy"), ("country_name":"Singapore")
("plane":14,"total_pass":118110,"planeID":1176,"plane_type_name":"Boeing 737-800","airline":"KLM","retirement_year":2017,"maintenance":8)	1	("country_name":"Sweden"), ("country_name":"Singapore")

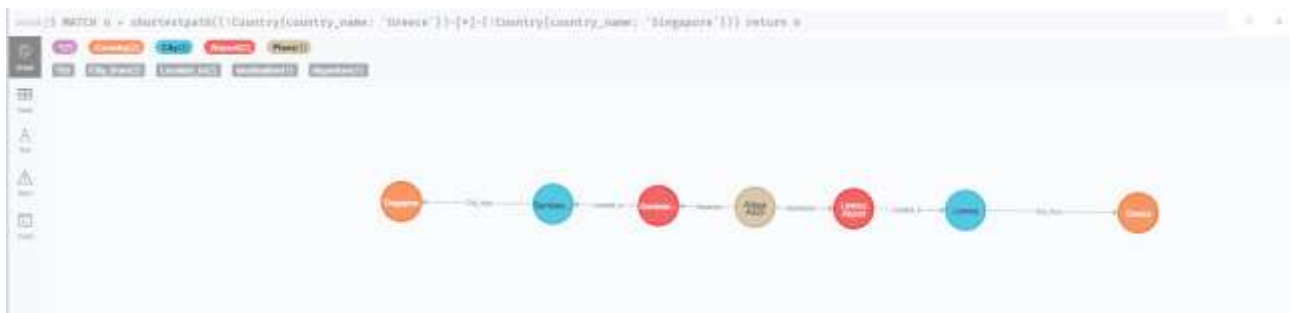
MATCH (n) RETURN(n)
Aquí us posem un exemple de la compilació.



3.1.6 Query 4

3.1.6.1 Solution

MATCH n = shortestpath((:Country{country_name: 'Greece'})-[*]-(:Country{country_name: 'Singapore'})) RETURN n



3.1.6.2 Explanation

En primer lloc, seleccionem qualsevol relació que hi ha entre els països requerits, que en aquest cas son Grècia i Singapur. A continuació, seleccionem la ruta més curta que separa els dos països, això ho fem mitjançant `shortestpath`, el qual calcula el camí més curt entre els dos nodes. Finalment, mostrem aquesta ruta, que en el nostre cas li hem assignat `n`.

3.1.6.3 Query validation

Inicialment, en la nostra base de dades, no teníem ni Grècia ni Singapur, i per poder fer la query vam haver de inserir-ho manualment amb les comandes a continuació:


```

INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name,
city_timezone, country_name) VALUES(5195,'Sembawang Air
Base',86,2997,'Sembawang',8,'Singapore');
INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name,
city_timezone, country_name) VALUES(1430,'Limnos Airport',14,1373,'Limnos',2,'Greece');
INSERT INTO Dataset1(planeID, retirement_year, type_plane, name, maintenance_count,
piece_count, cost_sum) VALUES(1,2020,'Spain AM','NGM',5,5,2195297);
INSERT INTO Dataset1(planeID, retirement_year, type_plane, name, maintenance_count,
piece_count, cost_sum) VALUES(2,2020,'Spain AM2','NGM',6,6,2995297);

```

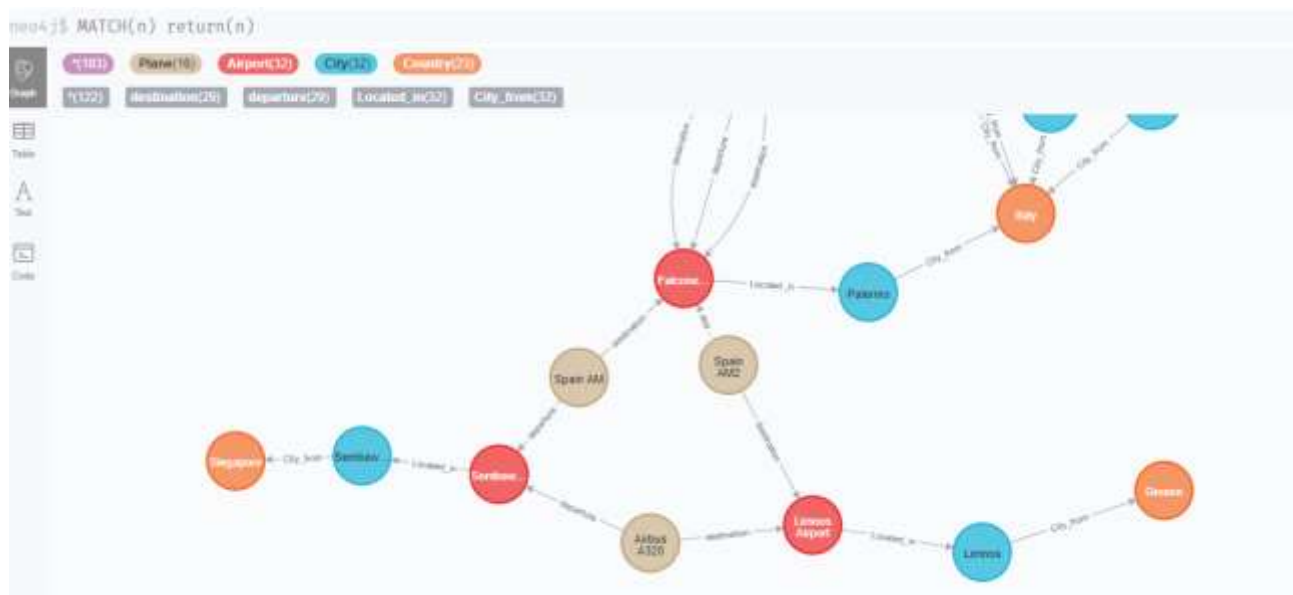
```

INSERT INTO Dataset12(planeID, departure, destination) VALUES(1344,5195,1430);
INSERT INTO Dataset12(planeID, departure, destination) VALUES(1,5195,1473);
INSERT INTO Dataset12(planeID, departure, destination) VALUES(2,1473,1430);

```

Gràcies a això, vam poder realitzar la query.

A més a més, vam haver de crear les relacions entre els aeroports i els avions, per poder-ho fer, vam haver d'inserir informació a Dataset 1 com a Dataset 12.

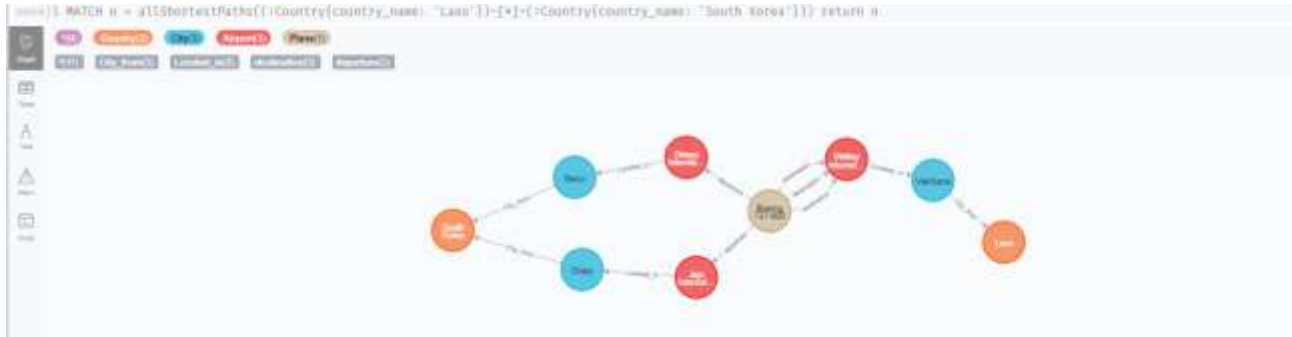


Amb la comanda MATCH (n) RETURN (n), podem veure que hi ha dues rutes entre Singapur i Grècia i validem que la més curta és la que estem mostrant en la query principal.

3.1.7 Query 5

3.1.7.1 Solution

```
MATCH n = allShortestPaths((:Country{country_name: 'Laos'})-[*]-(:Country{country_name: 'South Korea'})) RETURN n
```



3.1.7.2 Explanation

En primer lloc, seleccionem totes les relacions que hi ha entre els països requerits, que en aquest cas son Laos i Corea del Sud. A continuació, seleccionem totes les rutes que separen els dos països, això ho fem mitjançant `allShortestPaths`, el qual mostra els diferents camins que hi ha entre els dos nodes. Finalment, mostrem aquestes rutes, que en el nostre cas li hem assignat `n`.

3.1.7.3 Query validation

Com en el cas anterior, en la nostra base de dades, no teníem ni Laos ni Corea del Sud, i per poder fer la query vam haver de inserir-ho manualment amb les comandes a continuació:

```
INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone, country_name) VALUES(2272,'Gimpo International Airport',59,2175,'Seoul',9,'South Korea');
INSERT INTO dataset2(airportID, airport_name, airport_altitude, cityID, city_name, city_timezone, country_name) VALUES(2265,'Jeju International Airport',118,2168,'Cheju',9,'South Korea');
INSERT INTO Dataset12(planeID, departure, destination) VALUES(1259,2272,2957);
INSERT INTO Dataset12(planeID, departure, destination) VALUES(1259,2265,2957);
```

Gràcies a això, vam poder realitzar la query.

A més a més, vam haver de crear les relacions entre els aeroports i els avions, per poder-ho fer, vam haver d'inserir informació a Dataset 1 com a Dataset 12.

[illegible]

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

Per aquest Case Study, en canvi, hem hagut de relacionar les següents entitats: Airport, Flight, FlightAttendant, Pilot i Language, on havíem de relacionar aquestes 5 entitats depenent de les llengües que parlen i dels aeroports on han estat. Per començar, hem hagut de seleccionar els pilots actius, és a dir, que no estan retirats, que parlin més de 3 llengües diferents i que tinguin un sou superior a 100.000. A més a més, a partir de la informació obtinguda anteriorment, les/els auxiliars de vol que han estat en un vol amb els pilots seleccionats en el dataset 1, és a dir, com ja hem dit, la informació anterior. Finalment, hem hagut de connectar els dos datasets, mostrant així els vols en els que els pilots i les/els auxiliars de vol han coincidit.

3.2.1 Data migration

En primer lloc, vam fer les queries per obtenir les dades que ens demanaven en l'enunciat, un cop fetes i validades, és a dir comprovar que la informació fos correcta, vam inserir la informació de les queries en unes taules creades específicament per guardar i organitzar-ho tot.

```
DROP TABLE IF EXISTS Dataset2_1 CASCADE;
CREATE TABLE Dataset2_1(
```



```

        pilotID BIGINT(20),
        name VARCHAR(255),
        surname VARCHAR(255),
        email VARCHAR(255),
        sex CHAR(1),
        salary INT(11),
        years_working INT(11),
        languages VARCHAR(255),
        PRIMARY KEY(pilotID)
    );

```

```

INSERT INTO Dataset2_1 (pilotID, name, surname, email, sex, salary, years_working, languages)
SELECT pi.pilotID, p.name, p.surname, p.email, p.sex, e.salary, e.years_working,
GROUP_CONCAT(l.name SEPARATOR ', ') AS 'Languages she/he speaks'
FROM pilot AS pi JOIN person AS p ON pi.pilotID = p.personID
JOIN employee AS e ON e.employeeID = p.personID
JOIN languageperson AS lp ON lp.personID = p.personID
JOIN language AS l ON lp.languageID = l.languageID
WHERE (SELECT COUNT(lp2.languageID)
      FROM languageperson AS lp2
      WHERE lp2.personID = p.personID
      GROUP BY lp2.personID)>3 AND e.salary > 100000
GROUP BY pi.pilotID
ORDER BY pi.pilotID ASC;

```

-- Dataset 2

```

DROP TABLE IF EXISTS Dataset2_2;
CREATE TABLE Dataset2_2(
    flightattendantID BIGINT(20),
    name VARCHAR(255),
    surname VARCHAR(255),
    sex CHAR(1),
    salary INT(11),
    years_working INT(11),
    languages VARCHAR(255),
    PRIMARY KEY (flightattendantID )
);

```

```

INSERT INTO Dataset2_2 (flightattendantID, name, surname, sex, salary, years_working,
languages)
SELECT fa.flightattendantID, p.name, p.surname, p.sex, e.salary, e.years_working,
GROUP_CONCAT(l.name SEPARATOR ', ') AS 'Languages she/he speaks'
FROM flight_attendant AS fa JOIN person AS p ON fa.flightattendantID = p.personID
JOIN employee AS e ON e.employeeID = p.personID
JOIN languageperson AS lp ON lp.personID = p.personID
JOIN language AS l ON lp.languageID = l.languageID
JOIN flight_flightattendant as ffa ON fa.flightattendantID = ffa.flightattendantID
JOIN flight as f ON f.flightID = ffa.flightID
WHERE f.pilotID IN (SELECT pilotID FROM Dataset2_1)

```

```
GROUP BY fa.flightattendantID;
```

```
DROP TABLE IF EXISTS Dataset2_12;  
CREATE TABLE Dataset2_12(  
    pilotID BIGINT(20),  
    flightattendantID BIGINT(20),  
    flightID BIGINT(20),  
    date DATE,  
    destination_airportID BIGINT(20),  
    departure_airportID BIGINT(20),  
    PRIMARY KEY (flightattendantID, pilotID, flightID),  
    FOREIGN KEY (pilotID) REFERENCES Dataset2_1(pilotID),  
    FOREIGN KEY (flightattendantID) REFERENCES Dataset2_2(flightattendantID)  
);
```

```
INSERT INTO Dataset2_12 (pilotID, flightattendantID, flightID, date, destination_airportID,  
departure_airportID)  
SELECT d1.pilotID, d2.flightattendantID, f.flightID, f.date, r.destination_airportID,  
r.departure_airportID  
FROM Dataset2_1 AS d1 JOIN flight as f ON d1.pilotID=f.pilotID  
JOIN flight_flightattendant as ffa ON ffa.flightID=f.flightID  
JOIN Dataset2_2 as d2 ON d2.flightattendantID = ffa.flightattendantID  
JOIN route as r ON r.routeID = f.routeID;
```

Un cop creades i emplenades totes les taules, vam exportar la informació obtinguda a format csv, la exportació la vam fer des de el propi MySQL. Quan ja teníem tots els csvs creats, els vam penjar a les fulles de càlcul de google, on vam posar accés públic perquè tothom pogués accedir-hi. Un cop tot penjat al núvol, vam copiar l'enllaç de cada csv i vam modificar el link per poder importar tota la informació en la nostra base de dades del Neo4j. Finalment, amb els enllaços ja creats, vam importar tota la informació amb les següents comandes:

```
LOAD CSV WITH HEADERS FROM  
'https://docs.google.com/spreadsheets/d/13QonPrcKzPn0YjHpxcX-  
xP2ZNDboezsOUBAk0PeV6qU/gviz/tq?tx=out:csv&sheet=0' as d1  
WITH SPLIT(d1.languages, ", ") AS lang  
FOREACH (l IN lang | MERGE (:Language {language_name: l}));
```

```
LOAD CSV WITH HEADERS FROM  
'https://docs.google.com/spreadsheets/d/13QonPrcKzPn0YjHpxcX-  
xP2ZNDboezsOUBAk0PeV6qU/gviz/tq?tx=out:csv&sheet=0'  
AS d1  
CREATE (p:Pilot{pilotID: ToInteger(d1.pilotID), pilot_name: d1.name, surname:  
d1.surname, email: d1.email,  
sex: d1.sex, salary: ToInteger(d1.salary), years_working:  
ToInteger(d1.years_working)});
```

```

LOAD CSV WITH HEADERS FROM
'https://docs.google.com/spreadsheets/d/13QonPrcKzPn0YjHpXcX-
xP2ZNDboezsOUBAk0PeV6qU/gviz/tq?tqx=out:csv&sheet=0'
AS d1
MATCH (l:Language),(p:Pilot{pilotID: ToInteger(d1.pilotID)}) WHERE d1.languages
CONTAINS l.language_name
CREATE (l)-[:Speak]-(p)

```

--Dataset2_2

```

LOAD CSV WITH HEADERS FROM
'https://docs.google.com/spreadsheets/d/1rROZSrXF_F7FjCkWS_ycLu-
dUeCsuFFCOQrFsVc7J54/gviz/tq?tqx=out:csv&sheet=0' as d2
WITH SPLIT(d2.languages, ", ") AS lang
FOREACH (l IN lang | MERGE (:Language {language_name: l}));

```

```

LOAD CSV WITH HEADERS FROM
'https://docs.google.com/spreadsheets/d/1rROZSrXF_F7FjCkWS_ycLu-
dUeCsuFFCOQrFsVc7J54/gviz/tq?tqx=out:csv&sheet=0'
AS d2
CREATE(ft:FlightAttendant {flightattendantID: ToInteger(d2.flightattendantID),
FlightAttendant_name: d2.name, surname: (d2.surname),sex:d2.sex, salary:
ToInteger(d2.salary), years_working: ToInteger(d2.years_working),
languages:d2.languages});

```

```

LOAD CSV WITH HEADERS FROM
'https://docs.google.com/spreadsheets/d/1rROZSrXF_F7FjCkWS_ycLu-
dUeCsuFFCOQrFsVc7J54/gviz/tq?tqx=out:csv&sheet=0'
AS d2
MATCH (l:Language),(ft:FlightAttendant {flightattendantID:
ToInteger(d2.flightattendantID)}) WHERE d2.languages CONTAINS l.language_name
CREATE (l)-[:Speak]-(ft);

```

--Dataset2_12

```

LOAD CSV WITH HEADERS FROM 'https://docs.google.com/spreadsheets/d/1WBJbm-
YJ1hgUSqZi1KY4ipMwYlIZR1-PUoWIg485Rmc/gviz/tq?tqx=out:csv&sheet=0' as d12
MERGE (f:Flight{flight_name:ToInteger(d12.flightID), date: date(d12.date)})
MERGE (a_destination:Airport{airport_name:ToInteger(d12.destination_airportID)})
MERGE (a_departure:Airport{airport:ToInteger(d12.departure_airportID)})
CREATE (f)-[:Departs]->(a_departure),
(f)-[:Arrive]->(a_destination)

```

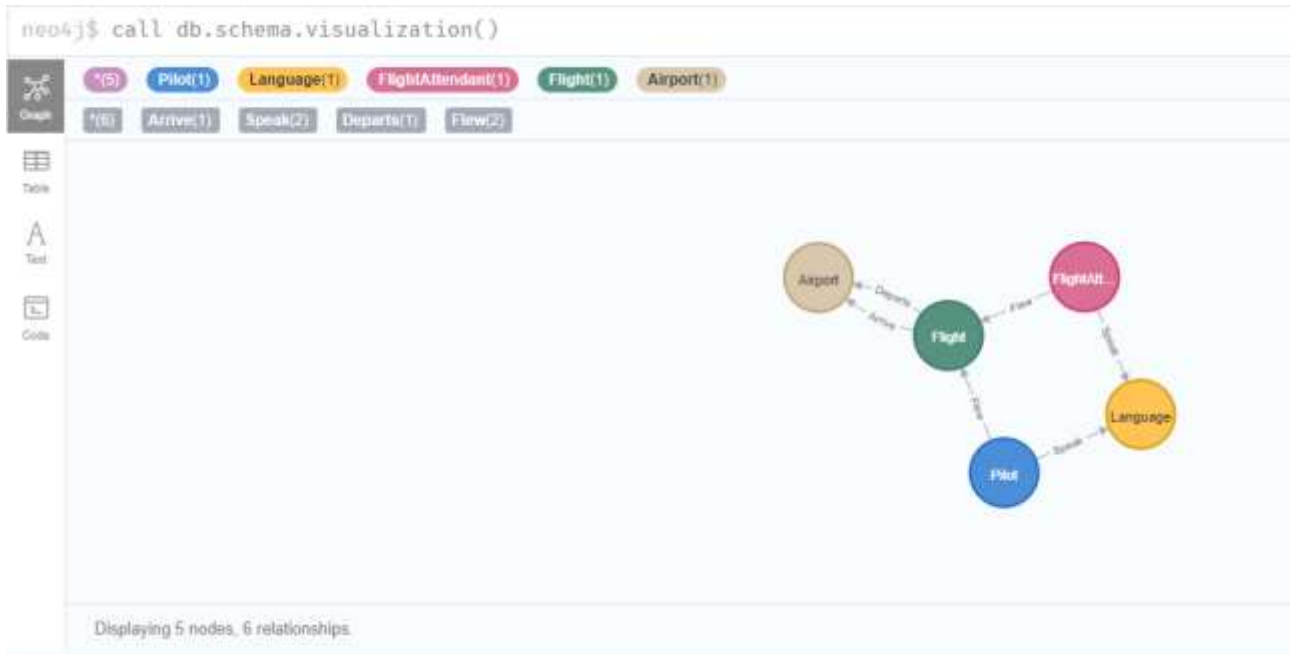
```

LOAD CSV WITH HEADERS FROM 'https://docs.google.com/spreadsheets/d/1WBJbm-
YJ1hgUSqZi1KY4ipMwYlIZR1-PUoWIg485Rmc/gviz/tq?tqx=out:csv&sheet=0' as d12
MATCH(f:Flight)
MATCH(ft:FlightAttendant)
MATCH(p:Pilot)
WHERE p.pilotID = ToInteger(d12.pilotID)
AND ft.flightattendantID = ToInteger(d12.flightattendantID)
AND f.flight_name= ToInteger(d12.flightID)
CREATE

```

(p)-[:Flew]->(f),
 (ft)-[:Flew]->(f)

3.2.2 Neo4J Data model



Com podem veure l'estructura principal per al "Case Study 2" es centra en la creació d'aquests 5 nodes: Pilot, Flight Attendant, Language, Flight i Airport. Tots tenen i estan vinculats per unes relacions que són les següents: Airport-Flight, té dues relacions, ja que el flight pot ser d'arribada(Arrive) o de sortida(Departs), Flight-FlightAttendant, ja que en cada vol hi ha auxiliars de vol, en aquest cas li hem dit "Flew", a la relació entre aquests dos nodes, FlightAttendant-Language, cada auxiliar de vol parla un o més idiomes, per tant la relació entre aquests dos nodes rep el nom de "Speak", la relació entre Language-Pilot, rep el nom de "Speak" pel mateix motiu, un pilot també pot parlar un o més idiomes, finalment, Pilot-Flight, ja que en cada vol hi ha d'haver un pilot, en aquest cas la relació rep el nom de "Flew".

3.2.3 Query 1

3.2.3.1 Solution

```
MATCH(fa1:FlightAttendant)-[:Flew]->(:Flight)<-[:Flew]-(fa2:FlightAttendant)
MERGE (fa1)-[c:Colleague]-(fa2)
RETURN (fa1)-[c]->(fa2)
```



3.2.3.2 Explanation

En primer lloc seleccionem les auxiliars de vol que han coincidit en el mateix vol i creen una nova relació entre les dues que rep el nom de “Colleague”. Finalment, mostrem totes les Flight Attendants que han coincidit en el mateix vol amb vinculades amb la nova relació Colleagues.

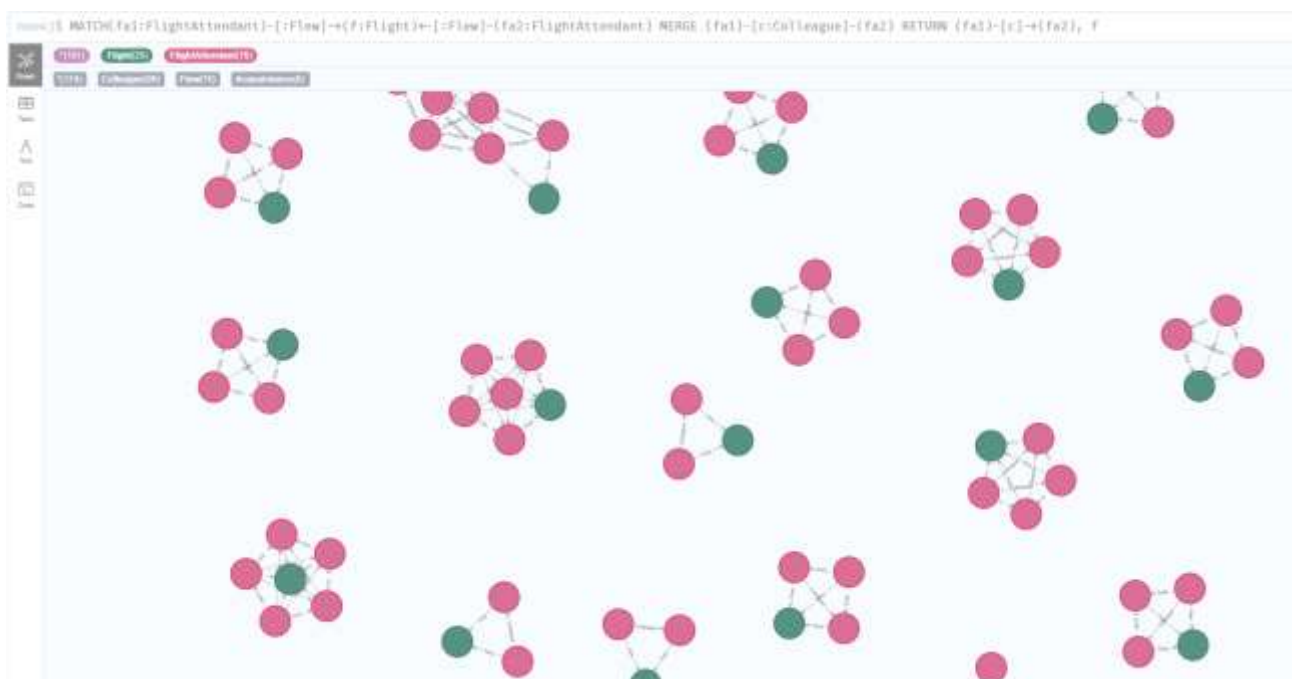
3.2.3.3 Query validation

A continuació, comprovem que realment les FlightAttendant que mostrem van coincidir en el mateix vol i que la relació “Colleague” s’ha creat correctament, ho hem fet amb la següent comanda:

```

MATCH (fa1:FlightAttendant)-[:Flew]->(f:Flight)-[:Flew]-(fa2:FlightAttendant)
MERGE (fa1)-[:Colleague]-(fa2)
RETURN (fa1)-[:c]->(fa2), f

```



3.2.4 Query 2

3.2.4.1 Solution

```
CALL {
    MATCH(fa:FlightAttendant)-[:Flew]->(:Flight)-[:Departs]->(:Airport)<-[:Arrive]-
(f2:Flight)<-[:Flew]-(fa2:FlightAttendant)
    WHERE (fa)-[:Speak]->(:Language)<-[:Speak]-(fa2)
    AND NOT (fa)-[:Colleague]-(fa2)
    RETURN fa, fa2

    UNION ALL

    MATCH(fa:FlightAttendant)-[:Flew]->(:Flight)-[:Departs]->(:Airport)<-[:Departs]-
(f2:Flight)<-[:Flew]-(fa2:FlightAttendant)
    WHERE (fa)-[:Speak]->(:Language)<-[:Speak]-(fa2)
    AND NOT (fa)-[:Colleague]-(fa2)
    RETURN fa, fa2

    UNION ALL

    MATCH(fa:FlightAttendant)-[:Flew]->(:Flight)-[:Arrive]->(:Airport)<-[:Arrive]-
(f2:Flight)<-[:Flew]-(fa2:FlightAttendant)
    WHERE (fa)-[:Speak]->(:Language)<-[:Speak]-(fa2)
    AND NOT (fa)-[:Colleague]-(fa2)
    RETURN fa, fa2
}
MERGE (fa)-[aq:Acquaintance]->(fa2)
RETURN (fa)-[aq]->(fa2)
```



3.2.4.2 Explanation

En primer lloc hem de buscar les flight attendants que han coincidit en un mateix aeroport, ja sigui que les dues sortint del mateix, arribin al mateix o que una arribi i l'altre surti del mateix, per això fem els unions amb els tres casos. A més a més, amb el where mirem que les dues flight attendants parlín el mateix i que no es coneixen. Finalment enllacem aquestes auxiliars amb una relació que les categoritza com a conegudes, ja que no son amigues, però han coincidit. El call, s'encarrega d'executar tot el que hi ha dins dels claudàtors.

3.2.4.3 Query validation

Per fer aquesta validació, mostrem les flights attendants i l'aeroport on han coincidit, així podem comprovar que realment han estat en al mateix aeroport que no es coneixien d'abans. Hem obtingut aquesta informació amb la comanda següent:

```
CALL {
    MATCH(fa:FlightAttendant)-[:Flew]->(:Flight)-[:Departs]->(a:Airport)<-[:Arrive]-
(f2:Flight)<-[:Flew]-(fa2:FlightAttendant)
    WHERE (fa)-[:Speak]->(:Language)<-[:Speak]-(fa2)
    AND NOT (fa)-[:Colleague]-(fa2)
    RETURN fa, fa2, a

    UNION ALL

    MATCH(fa:FlightAttendant)-[:Flew]->(:Flight)-[:Departs]->(a:Airport)<-[:Departs]-
(f2:Flight)<-[:Flew]-(fa2:FlightAttendant)
    WHERE (fa)-[:Speak]->(:Language)<-[:Speak]-(fa2)
    AND NOT (fa)-[:Colleague]-(fa2)
    RETURN fa, fa2, a

    UNION ALL

    MATCH(fa:FlightAttendant)-[:Flew]->(:Flight)-[:Arrive]->(a:Airport)<-[:Arrive]-
(f2:Flight)<-[:Flew]-(fa2:FlightAttendant)
    WHERE (fa)-[:Speak]->(:Language)<-[:Speak]-(fa2)
    AND NOT (fa)-[:Colleague]-(fa2)
    RETURN fa, fa2, a
}
MERGE (fa)-[aq:Acquaintance]->(fa2)
RETURN (fa)-[aq]->(fa2), a
```

Per acabar, també podem fer zoom a la solució obtinguda a la query principal o a la de validació i veiem que realment hi ha auxiliars de vol que no es coneixien d'abans i que s'ha establert la nova relació entre elles “Acquaintance”

3.2.5 Query 3

3.2.5.1 Solution

```
MATCH(pi:Pilot)-[:Flew]->(:Flight)-[:Flew]-(fa:FlightAttendant)
WHERE (pi)-[:Speak]->(:Language)-[:Speak]-(fa)
AND ABS(pi.years_working - fa.years_working) < 10
MERGE (pi)-[af:Affair]-(fa)
RETURN (pi)-[af]-(fa)
```



3.2.5.2 Explanation

En primer lloc seleccionem els pilots i les auxiliars de vol que han viatjat junts en un mateix vol. A més a més, també, amb el where, ens encarreguem que tant el pilot com les flights attendants parlin el mateix idioma i que la diferencia d'anys treballats entre els dos sigui superior a 10 anys. Finalment, enllacem el nostre resultat amb una relació de “Affair”, com se'ns indica en l'enunciat. Mostrem les relacions i nodes entre els pilots i les auxiliars de vol, afegint la relació creada d’Affair. Com podem observar, hi ha pilots que han tingut aventures amb dues col·legues, que és el que ens demanen en la query 5.

3.2.5.3 Query validation

A continuació mostrem els anys que ha treballat cada persona i la diferencia d'anys treballats entre les flight attendants i pilots que han coincidit en un vol, parlen el mateix idioma i que han tingut una aventura, ho fem amb la següent comanda:

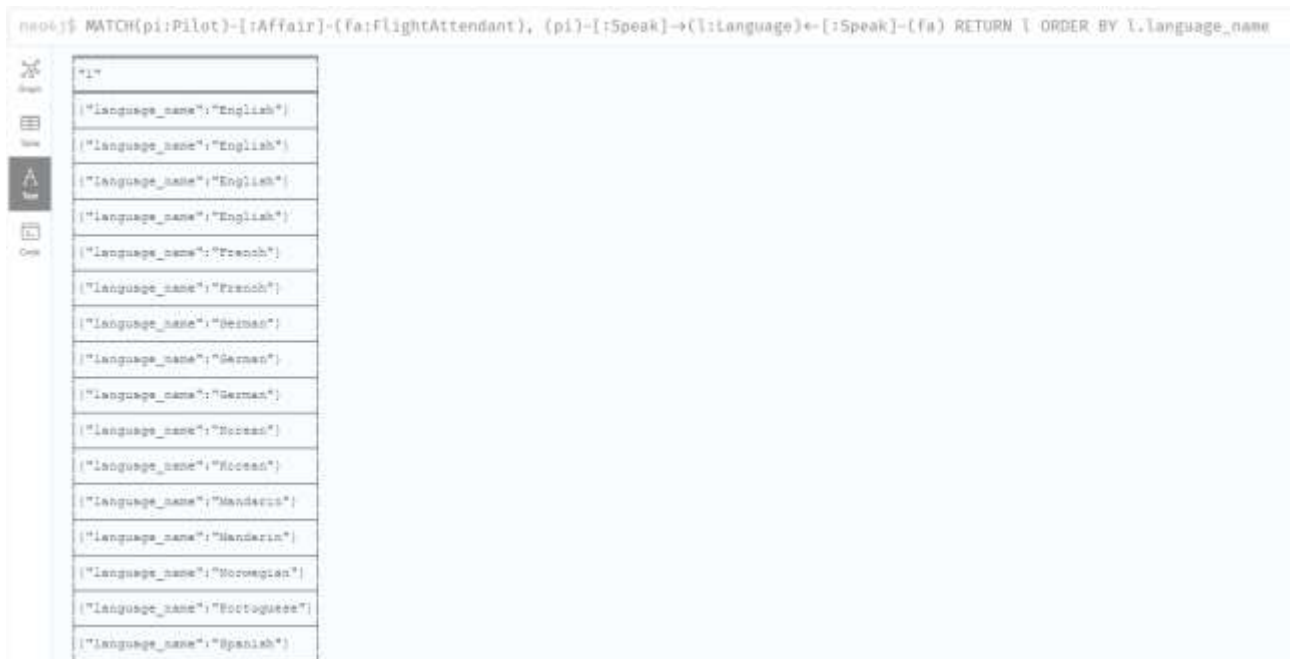
```
MATCH(pi:Pilot)-[:Flew]->(:Flight)-[:Flew]-(fa:FlightAttendant)
WHERE (pi)-[:Speak]->(:Language)-[:Speak]-(fa)
AND ABS(pi.years_working - fa.years_working) < 10
MERGE (pi)-[af:Affair]-(fa)
RETURN (pi)-[af]-(fa), pi.years_working, fa.years_working , ABS(pi.years_working -
fa.years_working)
```



```

MATCH(pi:Pilot)-[:Affair]-(fa:FlightAttendant),
      (pi)-[:Speak]->(l:Language)-[:Speak]-(fa)
RETURN l
ORDER BY l.language_name

```



neo4j\$ MATCH(pi:Pilot)-[:Affair]-(fa:FlightAttendant), (pi)-[:Speak]->(l:Language)-[:Speak]-(fa) RETURN l ORDER BY l.language_name

1
("language_name": "English")
("language_name": "English")
("language_name": "English")
("language_name": "English")
("language_name": "French")
("language_name": "French")
("language_name": "German")
("language_name": "German")
("language_name": "Italian")
("language_name": "Italian")
("language_name": "Mandarin")
("language_name": "Mandarin")
("language_name": "Norwegian")
("language_name": "Portuguese")
("language_name": "Spanish")

Com podem veure, el número de vegades que es repeteix el language, coincideix amb el count de la query principal, per tant podem dir que la query funciona correctament.

3.2.7 Query 5

3.2.7.1 Solution

```

MATCH(fa:FlightAttendant)-[:Affair]-(pi:Pilot)-[:Affair]-(fa2:FlightAttendant)
WHERE (fa)-[:Colleague]-(fa2)
RETURN pi

```



neo4j\$ MATCH(fa:FlightAttendant)-[:Affair]-(pi:Pilot)-[:Affair]-(fa2:FlightAttendant) WHERE (fa)-[:Colleague]-(fa2) RETURN pi

Graph visualization showing three blue nodes connected by lines, representing the relationship between a pilot and two flight attendants who are colleagues.

Record details:

```

#0  <id>:25 email: starleigh.chicott@nibpages.com pilotId: 134555 pilot_name: Starleigh salary: 125168 sex: M surname: Chicott years_working: 0

```

3.2.7.2 Explanation

En primer lloc seleccionem el pilots que han tingut dues relacions amb flights attendants diferents. A continuació, en el where, posem com a condició que les dues flight attendants amb les que el pilot ha

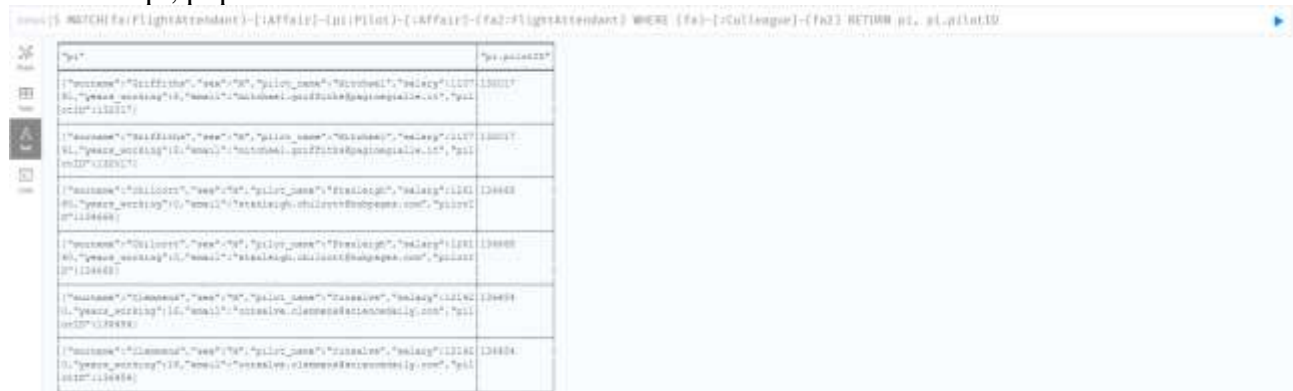
tingut relació, és coneixen. Finalment, mostrem per pantalla la informació sobre aquests pilots, com se'ns demana en l'enunciat.

3.2.7.3 Query validation

A continuació mostrem la query amb el ID corresponent a cada pilot, així podrem comprovar que els pilots que surten son els que realment han tingut una aventura amb dues auxiliars de vol que son amigues entre elles.

Primer, amb questa comanda, mostrem els IDs del pilots involucrats:

```
MATCH(fa:FlightAttendant)-[:Affair]-(pi:Pilot)-[:Affair]-(fa2:FlightAttendant)
WHERE (fa)-[:Colleague]-(fa2)
RETURN pi, pi.pilotID
```



pi	pi.pilotID
{ "surname": "Stiffitha", "sex": "M", "pilot_name": "Stiffitha", "salary": 1127, "years_working": 18, "email": "stiffitha@hugoboss.com", "pilotID": 112117 }	112117
{ "surname": "Stiffitha", "sex": "M", "pilot_name": "Stiffitha", "salary": 1127, "years_working": 18, "email": "stiffitha@hugoboss.com", "pilotID": 112117 }	112117
{ "surname": "Stiffitha", "sex": "M", "pilot_name": "Stiffitha", "salary": 1121, "years_working": 18, "email": "stiffitha@hugoboss.com", "pilotID": 112117 }	112117
{ "surname": "Stiffitha", "sex": "M", "pilot_name": "Stiffitha", "salary": 1121, "years_working": 18, "email": "stiffitha@hugoboss.com", "pilotID": 112117 }	112117
{ "surname": "Stiffitha", "sex": "M", "pilot_name": "Stiffitha", "salary": 1121, "years_working": 18, "email": "stiffitha@hugoboss.com", "pilotID": 112117 }	112117
{ "surname": "Stiffitha", "sex": "M", "pilot_name": "Stiffitha", "salary": 1121, "years_working": 18, "email": "stiffitha@hugoboss.com", "pilotID": 112117 }	112117

I ara, mostrem el gràfic amb les aventures i si passem el cursor pels nodes dels pilots, podem veure que els ID son correctes.



3.2.8 Query 6

3.2.8.1 Solution

```
MATCH (pi:Pilot)-[:Affair]-(fa:FlightAttendant)
WITH COUNT(*) AS num_affaires, pi AS pi
WHERE num_affaires > 1
RETURN (pi)-[:Affair]-(fa), (pi)-[:Colleague]-(fa), (pi)-[:Acquaintance]-(fa)
```



3.2.8.2 Explanation

En primer lloc seleccionem els pilots que han tingut alguna relació amb un altre persona. També fem un comptador de les relacions que ha tingut i ens encarreguem de mostrar els que han tingut més d'una relació. Finalment mostrem els pilots quines d'aquestes relacions han tingut: affair, colleague, and acquaintance.

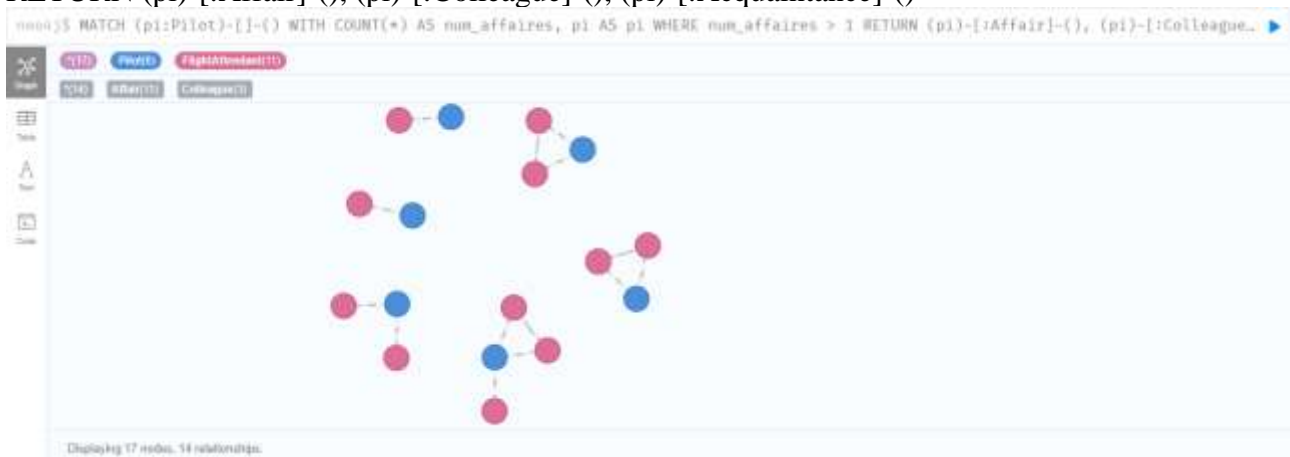
3.2.8.3 Query validation

A continuació mostrem totes les relacions que hi ha entre pilots, sense la necessitat de que sigui una aventura amorosa, així podem comprovar que realment mostrem els pilots amb més d'1 affair i que els pilots que mostrem tinguin una aventura amb algú. Ho fem amb la següent query:

`MATCH (pi:Pilot)-[]-()`

`WITH COUNT(*) AS num_affaires, pi AS pi`

`RETURN (pi)-[:Affair]-(), (pi)-[:Colleague]-(), (pi)-[:Acquaintance]-()`



4 Conclusions

4.1 Use of resources

Stage	Joan	Narcís	Marc	Lluís	Total
SQL queries	10	13	15	14	52
Triggers	7	5	9	2	23
Events	2	2	3	2	9
Neo4J population	2	10	2	8	22
Neo4J queries	2	4	1	5	12
Documentation	5	2	10	5	22
Total:	28	36	40	36	140

La taula és una aproximació de les hores que hem empleat, ho hem intentat representar de la manera més adient, tot i així no creiem que sigui del tot correcte, ja que realment no hem calculat el temps que hi hem dedicat.

SQL queries: La realització de totes les queries de la primer part. Cada membre del grup es va enfocar en un requeriment, és per això que les hores son semblants entre els diferents membres.

Triggers: Com hem dit abans, en Joan i en Marc, al principi es van centrar més en aquest punt, és per això la diferencia d'hores, tot i així en Narcís i en Lluís també hi van col·laborar.

Events: Com hem dit abans, en Joan i en Marc, al principi es van centrar més en aquest punt, és per això la diferencia d'hores, tot i així en Narcís i en Lluís també hi van col·laborar.

Neo4J population: Com hem dit abans, en Narcís i en Lluís, al principi es van centrar més en aquest punt, és per això la diferencia d'hores, tot i així en Narcís i en Marc també hi van col·laborar.

Neo4J queries: Com hem dit abans, en Narcís i en Lluís, al principi es van centrar més en aquest punt, és per això la diferencia d'hores, tot i així en Joan i en Marc també hi van col·laborar.

Documentation: Havíem d'inserir molta informació, tot i que la quantitat d'hores sembli exagerada, realment hi hem dedicat moltes hores en fer tota la memòria, validacions, explicacions....

4.2 Lessons learnt (1 page)

Fent aquesta pràctica hem posat a prova i hem acabat de polir tot el que hem anat veient en el curs. Començant per l'inici del primer semestre, fent aquesta pràctica hem practicat la lectura de models, en aquest cas d'un model relacional. Així mateix, també hem hagut de realitzar queries complexes, posant a prova el contingut de queries après en el primer semestre combinat amb tot lo après relacionat amb queries en el segon semestre, com podria ser l'ús de subqueries, triggers, events o stored procedures. A més a més, al haver de validar totes les queries que hem creat també hem après com fer-ho, i que ens aporta per la millora i optimització de cada query. Finalment, hem hagut de fer servir el que hem après de Neo4j. En resum, creiem que com ja hem dit, aquesta pràctica ens ha servit per fer un repàs a fons de tot el que hem après durant aquest any cursant l'assignatura Base de Dades.

Per l'altre banda, al ser un treball en grup de quatre persones, cal afegir que també hem après a treballar en equip, organitzar-nos i comunicar-nos entre els membres de l'equip ha estat una part essencial per a la realització d'aquesta pràctica. Aquest últim punt ens servirà de molt per al nostre futur laboral com a futurs enginyers, ja que estem segurs que en algun punt haurem de formar equip amb altres enginyers per tirar el projecte endavant, que és exactament el que hem fet durant el desenvolupament d'aquesta última pràctica.

4.3 Future work and conclusions (1 page)

Com ja hem dit abans, creiem que aquest últim projecte ens ha anat bé per fer un repàs pràctic de tot el que hem vist durant el curs. Creiem que el contingut de la pràctica és el convenient i ha estat encertat, no obstant creiem que ens heu donat poc temps per fer-la, ja que al final és la nota d'aquest projecte la que “decideix” si aprovem l'assignatura o no.

També creiem que hi ha apartats de la pràctica que sumen molt o són molt extensos per lo poc que els hem treballat a classe, com per exemple tot l'apartat de neo4j, sota el nostre parer un tema que hem treballat a dos laboratoris no hauria de tenir tanta rellevància en la nota final del projecte, ja que no només hem hagut de fer el que hem treballat a classe, a més hem hagut de crear uns csvs amb els requeriments que se'ns indicava, exportar-los i importar-los al neo4j, coses que no hem vista a classe i essencials per fer tot l'apartat de Neo4j... Està bé que ens deixeu un apartat on haguem de fer recerca i d'alguna manera “buscar-nos la vida”, però creiem que aquest apartat no hauria de tenir tan de pes.

Com sempre agrair als becaris i al professorat, ja que sempre que ho hem necessitat ens heu donat l'ajut que necessitàvem.