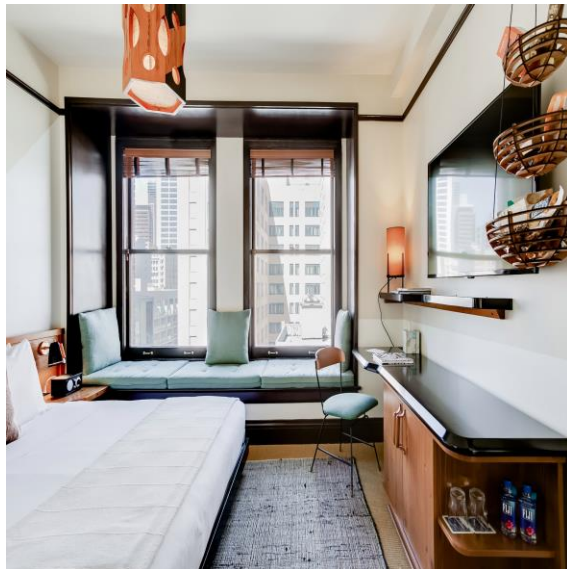


laSalle

UNIVERSITAT RAMON LLULL

Database Systems



Practical Assignment 1 - LS Eirbianbi

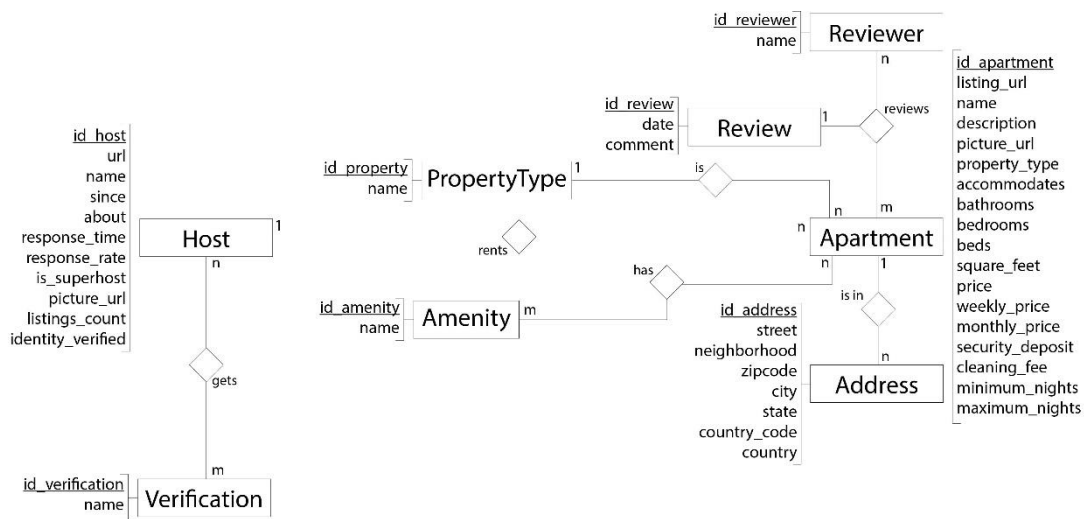
Nom: Laia Abad

Login: laia.abad

Table of contents

• Conceptual model.....	p. 2
• Relational model.....	p. 3
• Physical model.....	p. 4
• Importation verification.....	p. 13
• Queries and their verification.....	p. 13
Querie 1.....	p. 14
Querie 2.....	p. 16
Querie 3.....	p. 17
Querie 4.....	p. 18
Querie 5.....	p. 19
Querie 6.....	p. 20
Querie 7.....	p. 22
Querie 8.....	p. 24
Querie 9.....	p. 27
Querie 10.....	p. 29
• Conclusions.....	p. 30

Conceptual model



Justificació:

A la taula Host hem tret tota la informació de l'apartment que hi havia a la taula Hosts del CSV. També hem separat l'array host_verifications i hem creat una taula per a les verificacions. Com un host pot tenir varies verificacions, i diferents hosts poden tenir la mateixa verificació, es tracta d'una relació n:m.

A Apartment, hem separat amenities a la seva pròpia taula, ja que es tracta d'un array. Com un apartment pot tenir varies amenities, i diferents apartments poden tenir la mateixa amenity, es tracta d'una relació n:m.

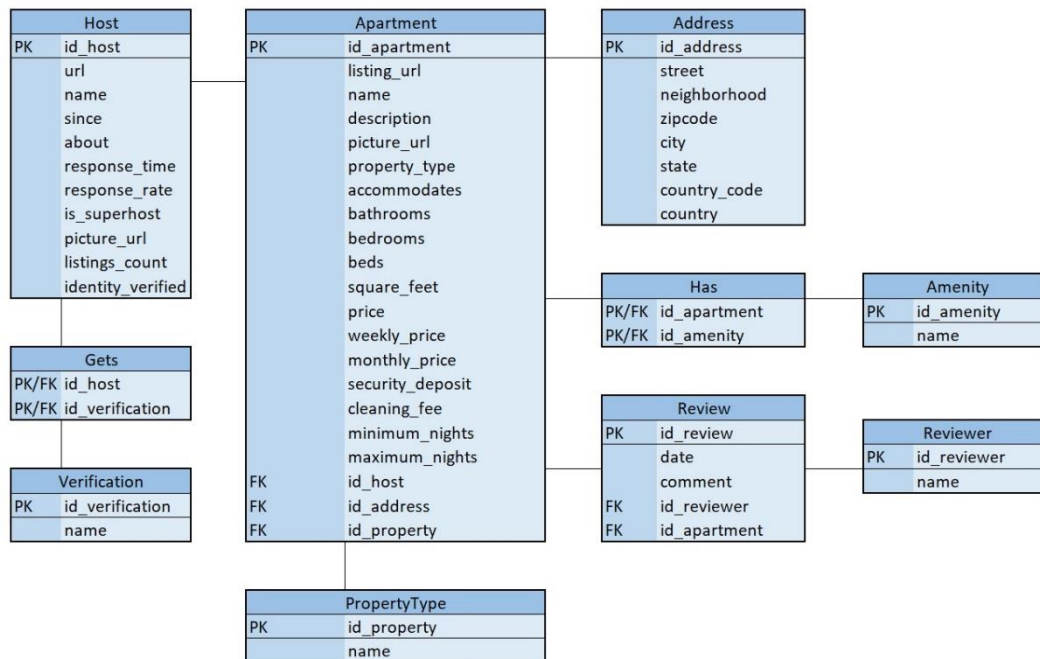
A més, hem separat l'adreça a la seva pròpia taula, ja que semblen dades atòmiques. Es tracta d'una relació 1:n, ja que 2 apartaments poden estar al mateix carrer.

La taula PropertyType ha sigut creada perquè es una dada que serveix per a classificar els apartments. Un apartment només pot ser d'un tipus, així que es una relació 1:n.

La taula Host i Apartment tenen una relació 1:n, perquè un host pot tenir més d'un apartment. Els apartments no poden ser llogats per més d'un host.

De la taula Review del CSV, hem tret dues taules, la Review i la Reviewer. La Review conté la informació sobre la review feta i la Reviewer, la informació sobre la persona que ha fet la review. Aquestes dues, formen una relació ternària amb apartment, en la que una persona pot fer una review sobre varis apartments i diferents persones poden fer una review sobre un apartment.

Relational model



Justificació:

Ja que Host i Verification tenen una relació n:m, s'ha hagut de crear una taula intermitja, que conte els PKs de les altres taules com a PK/FK.

Apartment conté les PKs de Address, PropertyType i Host com a FKs, perquè té una relació 1:n amb les totes.

En canvi, amb Amenity, s'ha hagut de crear una taula intermitja que conté els PKs de les altres taules com a PK/FK.

Per a la relació ternària, com de la relació 1:n:m, Review és la que té l'1, conté les PKs de les altres dues com a FKs

Physical model

```
DROP TABLE IF EXISTS Host CASCADE;
```

```
CREATE TABLE Host (  
    id_host SERIAL,  
    url VARCHAR(255),  
    name VARCHAR(255),  
    since DATE,  
    about TEXT,  
    response_time VARCHAR(255),  
    response_rate VARCHAR(255),  
    is_superhost BOOLEAN,  
    picture_url VARCHAR(255),  
    listings_count INT,  
    identity_verified BOOLEAN,  
    host_id INT,  
    PRIMARY KEY (id_host)
```

```
);
```

-- host_id: Als altres models no esta, ja que es una columna que eliminem. S'utilitza per a fer el producte cartesià a la taula Apartment. Conté l'id utilitzat a la taula Hosts original del CSV.

```
DROP TABLE IF EXISTS Verification CASCADE;
```

```
CREATE TABLE Verification (  
    id_verification SERIAL,  
    name VARCHAR(255),  
    PRIMARY KEY (id_verification)
```

```
);
```

```
DROP TABLE IF EXISTS Gets CASCADE;
```

```
CREATE TABLE Gets (  
    id_host INT,  
    id_verification INT,  
    PRIMARY KEY (id_host, id_verification),  
    FOREIGN KEY (id_host) REFERENCES Host(id_host),  
    FOREIGN KEY (id_verification) REFERENCES  
    Verification(id_verification)
```

```
);
```

```
DROP TABLE IF EXISTS Address CASCADE;
```

```
CREATE TABLE Address (  
    id_address SERIAL,  
    street VARCHAR(255),  
    neighborhood VARCHAR(255),  
    zipcode VARCHAR(255),  
    city VARCHAR(255),  
    state CHAR(3),  
    country_code CHAR(2),  
    country VARCHAR(255),  
    PRIMARY KEY (id_address)
```

```

);

DROP TABLE IF EXISTS PropertyType CASCADE;
CREATE TABLE PropertyType (
    id_property SERIAL,
    name VARCHAR(255),
    PRIMARY KEY (id_property)
);

DROP TABLE IF EXISTS Apartment CASCADE;
CREATE TABLE Apartment (
    id_apartment SERIAL,
    listing_url VARCHAR(255),
    name VARCHAR(255),
    description TEXT,
    picture_url VARCHAR(255),
    accommodates INT,
    bathrooms REAL,
    bedrooms INT,
    beds INT,
    square_feet INT,
    price VARCHAR(255),
    weekly_price VARCHAR(255),
    monthly_price VARCHAR(255),
    security_deposit VARCHAR(255),
    cleaning_fee VARCHAR(255),
    minimum_nights INT,
    maximum_nights INT,
    id_host INT,
    id_property INT,
    id_address INT,
    id INT,
    PRIMARY KEY (id_apartment),
    FOREIGN KEY (id_host) REFERENCES Host(id_host),
    FOREIGN KEY (id_property) REFERENCES PropertyType(id_property),
    FOREIGN KEY (id_address) REFERENCES Address(id_address)
);

-- id: Als altres models no esta, ja que es una columna que eliminem. S'utilitza per a fer el
-- producte cartesià a la taula Has. Conté l'id utilitzat a la taula Apartments original del CSV.

```

```

DROP TABLE IF EXISTS Amenity CASCADE;
CREATE TABLE Amenity (
    id_amenity SERIAL,
    name VARCHAR(255),
    PRIMARY KEY (id_amenity)
);

```

```

DROP TABLE IF EXISTS Has CASCADE;
CREATE TABLE Has (
    id_apartment INT,
    id_amenity INT,

```

```

        PRIMARY KEY (id_apartment, id_amenity),
        FOREIGN KEY (id_apartment) REFERENCES Apartment(id_apartment),
        FOREIGN KEY (id_amenity) REFERENCES Amenity(id_amenity)
    );

DROP TABLE IF EXISTS Reviewer CASCADE;
CREATE TABLE Reviewer (
    id_reviewer SERIAL,
    name VARCHAR(255),
    reviewer_id INT,
    PRIMARY KEY (id_reviewer)
);
-- reviewer_id: Als altres models no esta, ja que es una columna que eliminem. S'utilitza per a
fer el producte cartesià a la taula Reviews. Conté l'id utilitzat a la taula Review original del
CSV.

DROP TABLE IF EXISTS Review CASCADE;
CREATE TABLE Review (
    id_review SERIAL,
    id_apartment INT,
    id_reviewer INT,
    date DATE,
    comment TEXT,
    PRIMARY KEY (id_review),
    FOREIGN KEY (id_apartment) REFERENCES Apartment(id_apartment),
    FOREIGN KEY (id_reviewer) REFERENCES Reviewer(id_reviewer)
);

INSERT INTO Host(url, name, since, about, response_time, response_rate,
is_superhost, picture_url, listings_count, identity_verified, host_id)
SELECT DISTINCT host_url, host_name, host_since, host_about,
host_response_time, host_response_rate, host_is_superhost,
host_picture_url, host_listings_count, host_identity_verified, host_id
FROM Hosts;

UPDATE Host
SET response_rate = TRIM('%' FROM response_rate);

UPDATE Host
SET response_rate = NULL
WHERE response_rate LIKE 'N/A';

ALTER TABLE Host
ALTER COLUMN response_rate TYPE REAL USING response_rate::REAL;

UPDATE Host
SET response_rate = response_rate/100;
-- response_rate: Treiem el '%' i els valors que no son nombres per a poder transformar-ho en
REAL i dividir-ho entre 100, ja que un percentatge representa un valor entre 100.

DROP TABLE IF EXISTS Verifications;

```

```
CREATE TABLE Verifications (
    name VARCHAR(255),
    host_id INT
);
```

-- Verifications: es una taula auxiliar que utilitzem per passar el array de hosts_verifications a una columna amb l'id del host corresponent. Fem una taula auxiliar per dues raons: 1) per poder fer el DISTINCT i treure les verificacions repetides i 2) per a tenir els ids dels hosts corresponents i poder fer el producte cartesià, ja que si el fèiem a la taula original, al estar els ids no repetits, no es pot fer un DISTINCT.

```
INSERT INTO Verifications(name, host_id)
SELECT REGEXP_SPLIT_TO_TABLE(host_verifications, E','), host_id
FROM Hosts;
```

```
UPDATE Verifications
SET name = REPLACE(name, ' ', '');
```

-- Utilitzem aquesta funció per a treure els ' , ja que no es poden treure amb la funció TRIM.

```
UPDATE Verifications
SET name = TRIM('[ ]' FROM name);
```

-- Utilitzem el trim, ja que l'array conte les paraules entre cometes i l'array entre [].

```
INSERT INTO Verification(name)
SELECT DISTINCT name
FROM Verifications;
```

```
INSERT INTO Gets (id_host, id_verification)
SELECT DISTINCT id_host, id_verification
FROM Host, Verification, Verifications
WHERE host.host_id = verifications.host_id AND verification.name =
verifications.name;
```

-- Gets: fem el producte cartesià de Host i Verifications utilitzant el host_id, per a que l'id_host sigui el correcte i el de Verification i Verifications utilitzant el nom, per a que l'id_verification sigui correcte.

```
DROP TABLE Verifications;
```

-- Eliminem la taula auxiliar ja que ja no la necessitem.

```
UPDATE Apartments
SET zipcode = TRIM('QWERTYUIOPASDFGHJKLZXCVBNM ' FROM zipcode);
```

-- El zipcode no pot contenir caràcters que no siguin nombres, així que els treiem. Originalment anava a passar el zipcode a INT, però per problemes no he pogut.

```
UPDATE Apartments
SET state = SUBSTRING(state, 1, 3);
```

```
UPDATE Apartments
SET state = UPPER(state);
```

-- Alguns states tenen 3 lletres i altres el nom complet, llavors es tracten com a diferents adreces encara que siguin la mateixa.


```

INSERT INTO Address (street, neighborhood, zipcode, state, city,
country_code, country)
SELECT DISTINCT street, neighbourhood_cleansed, zipcode, state, city,
country_code, country
FROM Apartments;

```

-- Eliminem la taula auxiliar ja que ja no la necessitem.

```

INSERT INTO PropertyType (name)
SELECT DISTINCT property_type
FROM Apartments;

```

```

INSERT INTO Apartment (listing_url, name, description, picture_url,
accommodates, bathrooms, bedrooms, beds, square_feet, price,
weekly_price, monthly_price, security_deposit, cleaning_fee,
minimum_nights, maximum_nights, id_host, id_property, id_address, id)
SELECT ap.listing_url, ap.name, ap.description, ap.picture_url,
accommodates, bathrooms, bedrooms, beds, square_feet, price,
weekly_price, monthly_price, security_deposit, cleaning_fee,
minimum_nights, maximum_nights, id_host, id_property, id_address,
ap.id
FROM Apartments AS ap, Host, PropertyType, Hosts, Address AS ad
WHERE ap.property_type = propertytype.name AND hosts.listing_url =
ap.listing_url AND hosts.host_id = host.host_id AND (ap.street =
ad.street OR (ad.street IS NULL AND ap.street IS NULL)) AND
(ap.neighbourhood_cleansed = ad.neighborhood OR (ad.neighborhood IS
NULL AND ap.neighbourhood_cleansed IS NULL)) AND (ap.city = ad.city OR
(ad.city IS NULL AND ap.city IS NULL)) AND (ap.zipcode = ad.zipcode OR
(ad.zipcode IS NULL AND ap.zipcode IS NULL)) AND (ap.state = ad.state
OR (ad.state IS NULL AND ap.state IS NULL)) AND (ap.country_code =
ad.country_code OR (ad.country_code IS NULL AND ap.country_code IS
NULL)) AND (ap.country = ad.country OR (ad.country IS NULL AND
ap.country IS NULL));

```

-- Apartment: Fem producte cartesià utilitzant els ids auxiliars per a poder ficar a les taules els ids de les altres taules que corresponen. Especifiquem que compari si Address i Apartment tenen algun camp buit, ja que amb la comparació normal (element = element) no te en compte els NULLs.

```

ALTER TABLE Host
DROP host_id;

```

-- Com ja no utilitzem els ids auxiliars, els eliminem.

```

DROP TABLE IF EXISTS Amenities;
CREATE TABLE Amenities (
    name VARCHAR(255),
    id INT
);

```

-- Amenities: es una taula auxiliar que utilitzem per passar el array de amenities a la taula Apartments a una columna amb l'id del apartment corresponent. Fem una taula auxiliar per

dues raons: 1) per poder fer el DISTINCT i treure les verificacions repetides i 2) per a tenir els ids dels hosts corresponents i poder fer el producte cartesià, ja que si el fèiem a la taula original, al estar els ids no repetits, no es pot fer un DISTINCT.

```
INSERT INTO Amenities(name, id)
SELECT DISTINCT REGEXP_SPLIT_TO_TABLE(amenities, E','), id
FROM Apartments;
```

```
UPDATE Amenities
SET name = TRIM('{ ' FROM name);
```

```
INSERT INTO Amenity(name)
SELECT DISTINCT name
FROM Amenities;
```

```
INSERT INTO Has (id_apartment, id_amenity)
SELECT DISTINCT id_apartment, id_amenity
FROM Apartment, Amenity, Amenities
WHERE apartment.id = amenities.id AND amenities.name = amenity.name;
-- Has: fem el producte cartesià de Amenities i Apartment utilitzant l'id, per a que
l'id_apartment sigui el correcte i el de Amenities i Amenity utilitzant el nom, per a que
l'id_amenity sigui correcte.
```

```
ALTER TABLE Apartment
DROP id;
```

```
DROP TABLE Amenities;
-- Eliminem la taula i id auxiliar, ja que ja no el necessitem.
```

```
INSERT INTO Reviewer (name, reviewer_id)
SELECT DISTINCT reviewer_name, reviewer_id
FROM Reviews;
```

```
INSERT INTO Review (id_apartment, id_reviewer, date, comment)
SELECT id_apartment, id_reviewer, date_review, comments
FROM Apartment, Reviewer, Reviews
WHERE apartment.listing_url = reviews.listing_url AND
reviewer.reviewer_id = reviews.reviewer_id;
-- Reviews: Fem producte cartesià utilitzant els ids auxiliars per a poder ficar a les taules els
ids de les altres taules que corresponen.
```

```
ALTER TABLE Reviewer
DROP reviewer_id;
-- Eliminem l'id auxiliar, ja no el necessitem.
```

```
UPDATE Apartment
SET price = REPLACE(price, ',', '');
-- Com el TRIM no treu comes, utilitzem el REPLACE. Trèiem les comes perquè donen
problemes a l'hora de convertir a decimal.
```

```
UPDATE Apartment
SET price = TRIM('$' FROM price);
-- Trec el símbol del dòlar perquè el meu ordinador no reconeix aquest símbol (té el MONEY en €).
```

```
ALTER TABLE Apartment
ALTER COLUMN price TYPE DECIMAL(10,2) USING price::numeric(10,2);
-- Ho passem a DECIMAL, ja que el MONEY de vegades dona problemes, i a més ho mostraria com a €.
```

```
UPDATE Apartment
SET price = 0
WHERE price IS NULL;
-- Posem que els Apartments que tinguin price = NULL sigui igual a 0 per a que sigui més fàcil fer les quèries a la fase 2.
```

```
UPDATE Apartment
SET weekly_price = REPLACE(weekly_price, ',', '');
-- Com el TRIM no treu comes, utilitzem el REPLACE. Trèiem les comes perquè donen problemes a l'hora de convertir a decimal.
```

```
UPDATE Apartment
SET weekly_price = TRIM('$' FROM weekly_price);
-- Trec el símbol del dòlar perquè el meu ordinador no reconeix aquest símbol (té el MONEY en €).
```

```
ALTER TABLE Apartment
ALTER COLUMN weekly_price TYPE DECIMAL(10,2) USING
weekly_price::numeric(10,2);
-- Ho passem a DECIMAL, ja que el MONEY de vegades dona problemes, i a més ho mostraria com a €.
```

```
UPDATE Apartment
SET weekly_price = price * 7
WHERE weekly_price IS NULL;
-- Als Apartments que no tinguin weekly_price, el preu en una setmana serà el preu diari multiplicat pels dies que te una setmana.
```

```
UPDATE Apartment
SET monthly_price = REPLACE(monthly_price, ',', '');
-- Com el TRIM no treu comes, utilitzem el REPLACE. Trèiem les comes perquè donen problemes a l'hora de convertir a decimal.
```

```
UPDATE Apartment
SET monthly_price = TRIM('$' FROM monthly_price);
-- Trec el símbol del dòlar perquè el meu ordinador no reconeix aquest símbol (té el MONEY en €).
```

```
ALTER TABLE Apartment
ALTER COLUMN monthly_price TYPE DECIMAL(10,2) USING
monthly_price::numeric(10,2);
-- Ho passem a DECIMAL, ja que el MONEY de vegades dona problemes, i a més ho mostraria
com a €.
```

```
UPDATE Apartment
SET monthly_price = price * 30
WHERE monthly_price IS NULL;
-- Als Apartments que no tinguin monthly_price, el preu en una mes serà el preu diari
multiplicat pels dies que te un mes. Com el nombre de mesos varia, posem 30 com
aproximació.
```

```
UPDATE Apartment
SET security_deposit = REPLACE(security_deposit, ',', '');
-- Com el TRIM no treu comes, utilitzem el REPLACE. Trèiem les comes perquè donen
problemes a l'hora de convertir a decimal.
```

```
UPDATE Apartment
SET security_deposit = TRIM('$' FROM security_deposit);
-- Trec el símbol del dòlar perquè el meu ordinador no reconeix aquest símbol (té el MONEY
en €).
```

```
ALTER TABLE Apartment
ALTER COLUMN security_deposit TYPE DECIMAL(10,2) USING
security_deposit::numeric(10,2);
-- Ho passem a DECIMAL, ja que el MONEY de vegades dona problemes, i a més ho mostraria
com a €.
```

```
UPDATE Apartment
SET security_deposit = 0
WHERE security_deposit IS NULL;
-- Posem que els Apartments que tinguin security_deposit = NULL sigui igual a 0 per a que sigui
més fàcil fer les quèries a la fase 2.
```

```
UPDATE Apartment
SET cleaning_fee = REPLACE(cleaning_fee, ',', '');
-- Com el TRIM no treu comes, utilitzem el REPLACE. Trèiem les comes perquè donen
problemes a l'hora de convertir a decimal.
```

```
UPDATE Apartment
SET cleaning_fee = TRIM('$' FROM cleaning_fee);
-- Trec el símbol del dòlar perquè el meu ordinador no reconeix aquest símbol (té el MONEY
en €).
```

```
ALTER TABLE Apartment
ALTER COLUMN cleaning_fee TYPE DECIMAL(10,2) USING
cleaning_fee::numeric(10,2);
-- Ho passem a DECIMAL, ja que el MONEY de vegades dona problemes, i a més ho mostraria
com a €.
```

```
UPDATE Apartment
SET cleaning_fee = 0
WHERE cleaning_fee IS NULL;
```

-- Posem que els Apartments que tinguin cleaning_fee = NULL sigui igual a 0 per a que sigui més fàcil fer les quèries a la fase 2.

Importation verification

La taula original Apartments té 22.895 files.

La taula original Hosts té 22.895 files.

La taula original Review té 486.920 files.

La taula Host té 14.988 files, que son menys de les que hi havia a la original, ja que hi ha hosts que tenen més d'un apartment. Els hosts que apareixen no estan repetits.

La taula Verification té 21 files, cap d'elles repetides.

La taula Gets té 73.950 files, que son menys de 21×14.988 files, ja que no tots els hosts tenen totes les verificacions. Alguns no estan verificats tampoc.

La taula Address té 937 files, no repetides, ja que hi ha varis Apartments en un mateix carrer.

La taula PropertyType té 35 files, cap d'elles repetides.

La taula Apartment té 22.895 files, igual que el CSV Apartments.

La taula Amenity té 188 files, cap d'elles repetides.

La taula Has té 564.714 files, que son menys de 188×22.895 , ja que no tots els apartments tenen totes les amenities.

La taula Reviewer té 391061 files, que son menys de les que hi havia a la original, ja que hi ha reviews diferents fetes per la mateixa persona.

La taula Review té 486920 files, que son tantes com al CSV Review.

Queries and their verification

Query 1:

Show the top 3 cities in which the average savings percentage between staying 7 individual days or staying a week on the apartments with a verified host is bigger.

```
SELECT city AS name, AVG(((price - (weekly_price/7))/price) * 100) AS
savings_percentage
FROM Address AS ad, Apartment AS ap, Host AS h
WHERE ad.id_address = ap.id_address AND ap.id_host = h.id_host AND
identity_verified = true AND price <> 0
GROUP BY city
ORDER BY savings_percentage DESC
LIMIT 3;
```

-- He exclòs els preus que siguin igual a 0 ja que sinó no es pot fer la divisió, ja que dividim entre price.

	name character varying (255)	savings_percentage numeric
1	Cranbourne South	79.22077922077922078800
2	Wattle Glen	64.28571428571428570000
3	Pakenham	56.04395604395604400000

-- Els resultats no son iguals als del enunciat, ja que l'ultima ciutat es diferent. Així que he mirat quin era el savings_percentage de la ciutat que sortia a l'enunciat i els seus preus.

```
SELECT city AS name, AVG(((price - (weekly_price/7))/price) * 100) AS
savings_percentage
FROM Address AS ad, Apartment AS ap, Host AS h
WHERE ad.id_address = ap.id_address AND ap.id_host = h.id_host AND
identity_verified = true AND price <> 0
AND city LIKE 'Doncaster East'
GROUP BY city;
```

	name character varying (255)	savings_percentage numeric
1	Doncaster East	-31.40761423569560759144

```
SELECT price, weekly_price
FROM Address AS ad, Apartment AS ap, Host AS h
WHERE ad.id_address = ap.id_address AND ap.id_host = h.id_host AND
identity_verified = true AND price <> 0 AND city LIKE 'Doncaster East'
ORDER BY (price - (weekly_price/7))/price DESC;
```

	price numeric (10,2)	weekly_price numeric (10,2)
1	161.00	469.00
2	48.00	336.00
3	98.00	686.00
4	40.00	280.00
5	71.00	497.00
6	450.00	3150.00
7	98.00	686.00
8	40.00	280.00
9	35.00	245.00

	price numeric (10,2)	weekly_price numeric (10,2)
10	233.00	1631.00
11	30.00	210.00
12	30.00	210.00
13	251.00	1757.00
14	60.00	420.00
15	50.00	350.00
16	148.00	1036.00
17	109.00	2470.00
18	68.00	2380.00

-- Si calculem el descompte més gran que hi ha a Doncaster East, es 58,38% de descompte. El segon descompte més gran es de 0%. Si fem la mitja dels 2 descomptes més grans ja obtenim una mitja més petita que la de Pakenham, així que la mitja de Pakenham es major que la de Doncaster East.

-- Utilitzant el mateix procediment per la resta (però fent totes les operacions) podem comprovar que els resultats són correctes.

Query 2:

Show the Guesthouse that have the most expensive price per square feet and also has at least 200 reviews.

```
SELECT ap.name, CONCAT('$', CAST(price /(square_feet) AS DECIMAL
(10,2))) AS price_m2, COUNT(id_review) AS reviews
FROM Apartment AS ap, Review as r, PropertyType AS pt
WHERE r.id_apartment = ap.id_apartment AND pt.id_property =
ap.id_property AND pt.name LIKE 'Guesthouse' AND square_feet IS NOT
NULL AND square_feet <> 0
GROUP BY ap.id_apartment
HAVING COUNT(id_review) > 200
ORDER BY price_m2
LIMIT 1;
```

-- Com estem dividint entre square_feet, square_feet no pot ser igual a 0 ni NULL.

	name character varying (255)	price_m2 text	reviews bigint
1	The Stables, Island of Richmond	\$0.37	538

-- El resultat es el mateix que mostra l'enunciat.

```
SELECT ap.name, price, square_feet, COUNT(id_review)
FROM Apartment AS ap, Review as r, PropertyType AS pt
WHERE r.id_apartment = ap.id_apartment AND pt.id_property =
ap.id_property AND pt.name LIKE 'Guesthouse' AND square_feet IS NOT
NULL AND square_feet <> 0
GROUP BY ap.id_apartment;
```

	name character varying (255)	price numeric (10,2)	square_feet integer	count bigint
1	The Stables, Island of Richmo...	100.00	269	538
2	Japanese Slipper B&B Midori ...	199.00	269	32
3	Luxurious Self Contained Pool...	136.00	323	186

-- L'Apartment que retorna la query es l'únic que compleix tots els requisits, ja que les altres Guesthouses no tenen suficients Reviews o tenen NULLs o 0 a square_feet. Quan calculem el preu per feet utilitzant les dades retornades, comprovem que $100/269$ es $0,37174...$ que s'arrodoneix a **0,37**.

Query 3:

A group of 6 (accommodates = 6) friends want to rent an apartment for 5 days (and 5 nights) on Porth Phillip neighbourhood. The apartment must meet the following conditions:

- It must have a balcony.
- It must have more than 1.5 bathrooms.
- The host response rate must be more than 90%.

Find the apartment with the cheapest total price. The total price is rent price multiplied by the number of people plus the cleaning fee and 10% of the deposit.

```
SELECT ap.name, listing_url AS url, CONCAT('$', CAST ((price * 6 * 5 +
cleaning_fee + security_deposit * 0.1) AS DECIMAL (10,2))) AS
total_price
FROM Apartment AS ap, Address AS ad, Host AS ho, Amenity AS am, Has AS
ha
WHERE ap.id_address = ad.id_address AND ho.id_host = ap.id_host AND
am.id_amenity = ha.id_amenity AND ha.id_apartment = ap.id_apartment
AND neighborhood LIKE 'Port Phillip' AND am.name LIKE 'Balcony' AND
bathrooms > 1.5 AND response_rate > 0.9 AND accommodates = 6 AND
maximum_nights >= 5 AND minimum_nights <= 5
ORDER BY (price * 6 * 5 + cleaning_fee + security_deposit * 0.1)
LIMIT 1;
```

	name character varying (255)	url character varying (255)	total_price text
1	Spacious Designer Apartment...	https://www.airbnb.com/roo...	\$10050.00

-- El resultat es el mateix que mostra l'enunciat.

```
SELECT ap.name, price, cleaning_fee, security_deposit
FROM Apartment AS ap, Address AS ad, Host AS ho, Amenity AS am, Has AS
ha
WHERE ap.id_address = ad.id_address AND ho.id_host = ap.id_host AND
am.id_amenity = ha.id_amenity AND ha.id_apartment = ap.id_apartment
AND neighborhood LIKE 'Port Phillip' AND am.name LIKE 'Balcony' AND
bathrooms > 1.5 AND response_rate > 0.9 AND accommodates = 6 AND
maximum_nights >= 5 AND minimum_nights <= 5;
```

	name character varying (255)	price numeric (10,2)	cleaning_fee numeric (10,2)	security_deposit numeric (10,2)
1	Spacious Designer Apartment...	325.00	180.00	1200.00
2	Luxury Victorian Terrace near ...	400.00	200.00	1000.00
3	Cosy, Luxurious Home Next to...	379.00	140.00	300.00

-- Només hi ha 3 Apartments que compleixen totes les condicions. Si calculem el preu de cadascun obtenim 10050 per el primer, 12300 per el segon i 11540 per el tercer. Així que com la query diu, el més barat és el primer resultat.

Query 4:

The policy of superhosts has changed on Eirbienbi. Update the database by promoting all users that were registered 5 or more years ago to superhost and demote all the others.

```
UPDATE Host
SET is_superhost = false
WHERE since > CURRENT_DATE - 5 * 365;
```

```
UPDATE Host
SET is_superhost = true
WHERE since <= CURRENT_DATE - 5 * 365;
```

-- Li restem a la data actual els 5 anys pels 365 dies que té un any, ja que las sumes o restes en el tipus DATE es fan en dies.

```
SELECT SUM(is_superhost::INT) AS superhosts, COUNT(is_superhost) -
SUM(is_superhost::int) AS normal_hosts
FROM Host;
```

	superhosts bigint	normal_hosts bigint
1	4902	10083

-- Els resultats son diferents als del enunciat degut a que no sabem la data exacta que es va utilitzar a l'enunciat, per tant no podem saber que van utilitzar com a referencia per a d'aquí 5 anys.

```
SELECT since
FROM Host
WHERE is_superhost = TRUE
ORDER BY since DESC;
```

	since date
1	2014-12-23
2	2014-12-23
3	2014-12-23
4	2014-12-22
5	2014-12-22
6	2014-12-22
7	2014-12-22
8	2014-12-21

```
SELECT since
FROM Host
WHERE is_superhost = FALSE
ORDER BY since;
```

	since date
1	2014-12-24
2	2014-12-24
3	2014-12-24
4	2014-12-25
5	2014-12-25
6	2014-12-26
7	2014-12-26
8	2014-12-26

-- Com podem observar la data més recent de la creació del compte dels superhosts es el 23 de desembre del 2014 i la més antiga de la creació del compte dels que no són superhost es el 24 de desembre de 2014. (Avui es 22/12/19)

Query 5:

Find the 3 streets that have the biggest number of apartments as long as their average price is lower than 100\$.

```
SELECT DISTINCT street, COUNT(id_apartment) AS num, CONCAT('$',  
CAST(AVG(price) AS DECIMAL (10,2))) AS price  
FROM Address AS ad, Apartment AS ap  
WHERE ad.id_address = ap.id_address  
GROUP BY street  
HAVING AVG(price) < 100  
ORDER BY num DESC  
LIMIT 3;
```

	street character varying (255)	num bigint	price text
1	Brunswick, VIC, Australia	437	\$94.80
2	Footscray, VIC, Australia	161	\$90.86
3	Preston, VIC, Australia	142	\$94.37

-- El primer carrer es igual al enunciat, però els altres 2 no.

```
SELECT *  
FROM Address  
WHERE street LIKE 'Footscray, VIC, Australia' OR street LIKE 'Preston,  
VIC, Australia';
```

	id_address [PK] integer	street character varying (255)	neighborhood character varying (255)	zipcode character varying (255)	city character varying (255)	state character (3)	country_code character (2)	country character varying (255)
1	33	Footscray, VIC, Australia	Maribymong	[null]	Footscray	VIC	AU	Australia
2	200	Preston, VIC, Australia	Darebin	3072	Preston	VIC	AU	Australia
3	742	Preston, VIC, Australia	Darebin	[null]	Preston	VIC	AU	Australia
4	933	Footscray, VIC, Australia	Maribymong	3011	Footscray	VIC	AU	Australia

-- Al mirar els carrers que tenen el resultat diferent al enunciat, veiem que hi ha una variació de cada carrer que te el zipcode com a NULL.

```
SELECT DISTINCT street, COUNT(id_apartment) AS num, CONCAT('$',  
CAST(AVG(price) AS DECIMAL (10,2))) AS price  
FROM Address AS ad, Apartment AS ap  
WHERE ad.id_address = ap.id_address AND zipcode IS NOT NULL  
GROUP BY street  
HAVING AVG(price) < 100  
ORDER BY num DESC  
LIMIT 3;
```

	street character varying (255)	num bigint	price text
1	Brunswick, VIC, Australia	437	\$94.80
2	Footscray, VIC, Australia	159	\$90.69
3	Preston, VIC, Australia	140	\$94.29

-- Si provem la mateixa query d'abans, però sense els zipcodes NULLs, veiem que obtenim els mateixos resultats. Així que podem assumir que a l'enunciat no s'han tingut en compte les Address amb valors NULLs.

Query 6:

Eirbienbi wants to eradicate fake reviews. To do so, they want us to find the 3 people that have done more reviews on the same apartment. Check the reviews on the link of the apartment and choose one of the three users who you think is creating fake reviews and why.

```
SELECT rer.name AS name_reviewer, listing_url, COUNT(id_review) AS
num_reviews
FROM Reviewer AS rer, Review AS r, Apartment AS ap
WHERE rer.id_reviewer = r.id_reviewer AND ap.id_apartment =
r.id_apartment
GROUP BY r.id_reviewer, listing_url, rer.name
ORDER BY num_reviews DESC
LIMIT 3;
--It is most likely Cameron since he has the most reviews and they
look too good to be true.
```

	name_reviewer character varying (255)	listing_url character varying (255)	num_reviews bigint
1	Cameron	https://www.airbnb.com/roo...	81
2	Therese	https://www.airbnb.com/roo...	24
3	Laurie	https://www.airbnb.com/roo...	23

-- A l'enunciat l'últim Reviewer es Michael en comptes de Laurie, però els dos tenen el mateix nombre de Reviews, així que la query pot agafar qualsevol dels dos indistintament. La resta esta igual a l'enunciat.

```
SELECT rer.name AS name_reviewer, listing_url, COUNT(id_review) AS
num_reviews
FROM Reviewer AS rer, Review AS r, Apartment AS ap
WHERE rer.id_reviewer = r.id_reviewer AND ap.id_apartment =
r.id_apartment
GROUP BY r.id_reviewer, listing_url, rer.name
ORDER BY num_reviews DESC;
```

	name_reviewer character varying (255)	listing_url character varying (255)	num_reviews bigint
1	Cameron	https://www.airbnb.com/roo...	81
2	Therese	https://www.airbnb.com/roo...	24
3	Laurie	https://www.airbnb.com/roo...	23
4	Michael	https://www.airbnb.com/roo...	23

-- Al fer la query sense el LIMIT, podem observar que el següent a Laurie és Michael.

-- Al mirar les pàgines dels Apartments on han comentat, vaig veure que la pàgina del Apartment on va comentar la Laurie ja no existia, així que vaig descartar a la Laurie. Es possible que l'anunci del Apartment fossi eliminat per les Reviews falses, però com no les puc veure, no puc opinar.

Mirant els altres 2 no tenia molt clar que era exactament el que havia de mirar per saber si eren falses o no. Finalment vaig decidir que si algú estigues pagant a algú per a fer Reviews, voldrien Reviews extenses amb detalls per a que la persona que les llegís volgués quedar-se en aquest Apartment. Així que vaig concloure que era el Cameron. A més, és el que més té.

```
SELECT COUNT(reviewer_id) AS num_reviews, reviewer_name
FROM Reviews
WHERE listing_url = 'https://www.airbnb.com/rooms/7842455'
GROUP BY reviewer_id, reviewer_name
ORDER BY num_reviews DESC;
```

	num_reviews bigint	reviewer_name character varying (255)
1	81	Cameron
2	2	Brian
3	2	Ben
4	2	Jeremy
5	2	Alexander

-- Utilitzant la url del apartament al que s'ha fet la review podem mirar a la taula d'importació i veiem que Cameron ha fet 81 reviews, com ens va sortir a la query.
 -- Si repetim el mateix amb la resta de resultats podem comprovar que el resultat es correcte.

Query 7:

Two friends have a budget of 5000\$ for a trip of 2 days. Show the most expensive apartments they are able to rent taking into account that:

- The apartment must be for two people (for 2 or more accommodates) and at least it must have 2 beds.
- It must be on Saint Kilda.
- It must have kitchen.
- The host must have the phone verification.

The total price is rent price multiplied by the number of people plus the cleaning fee and 10% of the deposit.

```
SELECT ap.id_apartment AS id, ap.name, CONCAT('$', CAST((price * 2 * 2 + cleaning_fee + 0.1 * security_deposit) AS DECIMAL (10,2))) AS total_price
FROM Apartment AS ap, Amenity AS am, Host AS ho, Has AS ha, Gets AS g, Verification AS v, Address AS ad
WHERE (price * 2 * 2 + cleaning_fee + 0.1 * security_deposit) < 5000
AND accommodates >= 2 AND beds >= 2 AND city LIKE 'Saint Kilda' AND
am.id_amenity = ha.id_amenity AND ap.id_apartment = ha.id_apartment
AND ho.id_host = ap.id_host AND am.name LIKE 'Kitchen' AND g.id_host =
ho.id_host AND v.id_verification = g.id_verification AND v.name LIKE
'phone' AND ad.id_address = ap.id_address AND minimum_nights <= 2 AND
maximum_nights >= 2
ORDER BY (price * 2 * 2 + cleaning_fee + 0.1 * security_deposit) DESC;
```

	id integer	name character varying (255)	total_price text
1	21510	★ Beach at your Doorstep: Enjoy Summer in St Kilda	\$4166.00
2	8391	Premium property in great location	\$2000.00
3	5440	Luxury Vintage Melbourne Loft	\$1807.00
4	5812	Gorgeous Stkilda Apartments Sleeps 14 parking	\$1805.00
5	6045	Spacious Designer Apartment near Albert Park	\$1600.00
6	7170	Design House near Saint Kilda	\$1550.00
7	5809	Huge St Kilda Beach PentHouse Beauty	\$1540.00
8	9243	Stylish and Spacious Contemporary Apartment in St Kilda	\$1500.00
9	1434	Family and Large Group Parking WIFI Exc Location	\$1436.00
10	22112	2 Br Apartment fully furnished near albert park	\$1364.00





-- El resultat es casi idèntic al del enunciat, excepte que apareix un Apartment que al enunciat no apareix (el número 10, 2 Br Apartment fully furnished near albert park).

```
SELECT *
FROM Apartment
WHERE id_apartment = 22112
```





accommodates integer	bath real	beds integer	beds integer	squa intec	price numeric (10,2)	weel num	mgn num	security_deposit numeric (10,2)	cleaning_fee numeric (10,2)	minimum_nights integer	maximum_nights integer	id_host integer	id_property integer	id_address integer
6	1	2	2	[null]	341.00	7.00	0.00	0.00	0.00	2	3	13399	1	130

-- Comprovem que l'Apartment tingui 2 o més beds , accomodates 2 o més també, el preu sigui el que es mostra ($1364 = 2 * 2 * 341 + 0 + 0,1 * 0$), i el mínim i màxim de nights també es el correcte. Utilitzem els ids per a mirar la resta.









```
SELECT *
FROM Amenity AS a, Has AS h
WHERE id_apartment = 22112 AND a.id_amenity = h.id_amenity AND name
LIKE 'Kitchen'
```

	 id_amenity integer	 name character varying (255)	 id_apartment integer	 id_amenity integer
1	40	Kitchen	22112	40

```
SELECT *
FROM Gets AS g, Verification AS v
WHERE id_host = 13399 AND v.id_verification = g.id_verification AND
name LIKE 'phone';
```

	 id_host integer	 id_verification integer	 id_verification integer	 name character varying (255)
1	13399	15	15	phone

```
SELECT *
FROM Address
WHERE id_address = 130
```

	 id_address [PK] integer	 street character varying (255)	 neighborhood character varying (255)	 zipcode character varying (255)	 city character varying (255)	 state character (3)	 country_code character (2)	 country character varying (255)
1	130	Saint Kilda, VIC, Australia	Port Phillip	3182	Saint Kilda	VIC	AU	Australia

-- L'Apartment té Kitchen, el Host té la verificació phone i la adreça està a Saint Kilda. Hem comprovat que es correcte.

--Fem el mateix amb la resta i comprovem que son correctes.

Query 8:

Show the top 3 users calculating his score with the next formula:

$$\sum \frac{1}{\text{apartment_price}_i} * (1 + \text{is_superhost}) * \text{num_of_verifications} * \text{num_of_apartments}$$

```
DROP TABLE IF EXISTS Sumatori;  
CREATE TABLE Sumatori (  
    id_host INT,  
    suma REAL  
);
```

```
INSERT INTO Sumatori (id_host, suma)  
SELECT h.id_host, SUM(1/price)  
FROM Apartment AS a, Host AS h  
WHERE a.id_host = h.id_host AND price <> 0  
GROUP BY h.id_host;
```

```
SELECT h.name, (suma * (1 + is_superhost::int) * COUNT(DISTINCT  
v.id_verification) * COUNT(DISTINCT id_apartment)) AS score  
FROM Apartment AS a, Host AS h, Verification AS v, Gets AS g, Sumatori  
AS s  
WHERE g.id_verification = v.id_verification AND g.id_host = h.id_host  
AND h.id_host = a.id_host AND s.id_host = h.id_host AND h.name IS NOT  
NULL  
GROUP BY h.id_host, suma  
ORDER BY score DESC  
LIMIT 3;
```

```
DROP TABLE Sumatori;
```

-- He creat una taula auxiliar perquè a l'hora de ficar el sumatori en la query, com la taula Gets té cada Host repetit per cada verificació que té, el nombre d'apartaments queda multiplicat i el sumatori dona un resultat es major.

-- Posem la condició de que el preu no pugui ser 0 perquè estem dividint entre aquest.

--També posem la condició de que name no sigui NULL perquè si no la taula retorna tres noms NULLs amb score NULLs.

	name character varying (255)	score double precision
1	Valeria	1146.0592679977417
2	Jared SSP	1105.9727313518524
3	Adji	947.2764530181885

-- El resultat es igual al de l'enunciat.

```
SELECT h.id_host, h.name, (suma * (1 + is_superhost::int) *  
COUNT(DISTINCT v.id_verification) * COUNT(DISTINCT id_apartment)) AS  
score  
FROM Apartment AS a, Host AS h, Verification AS v, Gets AS g, Sumatori  
AS s
```

```

WHERE g.id_verification = v.id_verification AND g.id_host = h.id_host
AND h.id_host = a.id_host AND s.id_host = h.id_host AND h.name IS NOT
NULL
GROUP BY h.id_host, suma
ORDER BY score DESC
LIMIT 3;

```

	id_host [PK] integer	name character varying (255)	score double precision
1	14398	Valeria	1146.0592679977417
2	14075	Jared SSP	1105.9727313518524
3	6638	Adji	947.2764530181885

-- Busquem els ids dels Hosts.

```

SELECT h.name, suma, 1 + is_superhost::int, COUNT(DISTINCT
v.id_verification), COUNT(DISTINCT id_apartment)
FROM Apartment AS a, Host AS h, Verification AS v, Gets AS g, Sumatori
AS s
WHERE g.id_verification = v.id_verification AND g.id_host = h.id_host
AND h.id_host = a.id_host AND s.id_host = h.id_host AND h.id_host =
14398
GROUP BY h.id_host, suma;

```

	name character varying (255)	suma real	?column? integer	count bigint	count bigint
1	Valeria	1.6096338	1	8	89

-- Ara que sabem els valors específics, els comparem amb els de la taula d'importació per veure que son correctes

```

SELECT *
FROM Apartment
WHERE id_host = 14398;

```

	id_apartment [PK] integer	listing_url character varying (255)	name character varying (255)	description text	picture_url character varying (255)
1	15128	https://www.airbnb.com/roo...	209BR3 - Executive Double Ro...	Recently renova...	https://a0.muscache.com/in
2	15121	https://www.airbnb.com/roo...	209BR4 - One bedroom Apart...	THE APARTME...	https://a0.muscache.com/in
3	15044	https://www.airbnb.com/roo...	209BR2 - Executive Single Ro...	Recently renova...	https://a0.muscache.com/in

```

SELECT *
FROM Hosts
WHERE name LIKE '209BR3 - Executive Double Room in Brunswick';

```

host_is_superhost...	host_pict...	host_listings_count	host_verifications
boolean	character	integer	character varying (255)
false	https://...	89	['email', 'phone', 'facebook', 'reviews', 'jumio', 'offline_government_id', 'government_id', 'work_email']

-- Com podem veure Valeria no es superhost, té 89 Apartments i 8 Verifications, les dades són correctes a la query inicial.

```

SELECT SUM(1/price)
FROM Apartment AS a, Host AS h
WHERE a.id_host = h.id_host AND price <> 0 AND h.id_host = 14398
GROUP BY h.id_host;

```

	sum numeric
1	1.60963374970787789428

- Com veiem el sumatori també es correcte, i si calculem manualment el resultat de l'operació, també es correcte.
- Si repetim els mateixos passos amb els altres 2 Hosts veurem que també són correctes.

Query 9:

Show the top 10 reviewers with the most points taking into account that each review shorter than 100 characters counts for 10 points and each review longer (or equal) than 100 characters counts for 15 points.

```
SELECT re.id_reviewer, re.name, SUM(CASE WHEN LENGTH(comment) < 100
THEN 10 ELSE 15 END) AS points
FROM Review AS r, Reviewer AS re
WHERE r.id_reviewer = re.id_reviewer
GROUP BY re.id_reviewer
ORDER BY points DESC
LIMIT 10;
```

	id_reviewer [PK] integer	name character varying (255)	points bigint
1	209819	Laurie	1520
2	52444	Cameron	1395
3	162599	Jessica	780
4	250302	Michael	715
5	351025	Thanh	540
6	20543	Andrew	520
7	19191	Anders	475
8	344416	Suzanne	475
9	171080	John	435
10	20534	Andrew	425

-- El resultat es sembla al de l'enunciat, però hi ha diferències. Laurie te 1520 punts en comptes de 1505 i el Cameron, la Jessica i la Suzanne no apareixen a l'enunciat.

```
SELECT COUNT(id_review) AS total_comments
FROM Review
WHERE id_reviewer = 209819;
```

	total_comments bigint
1	105

```
SELECT COUNT(id_review) AS short_comments
FROM Review
WHERE id_reviewer = 209819 AND LENGTH(comment) < 100;
```

	short_comments bigint
1	11

-- Si fem la operació $\text{short_comments} * 10 + (\text{total_comments} - \text{short_comments}) * 15$ obtenim el mateix resultat que ens ha donat la query, 1520.

```
SELECT *
FROM Review
WHERE id_reviewer = 209819;
```

	id_review [PK] integer	id_apartment integer	id_reviewer integer	date date	comment text
1	300983	11577	209819	2018-02-06	Had a brilliant stay with Alision, which is locat...
2	301005	10166	209819	2016-03-17	Had a wonderful time with Bella (and Ben) and...
3	301006	21819	209819	2018-05-09	Fantastic location, excellent apartment and co...
4	301007	21819	209819	2018-01-31	Tana is a brilliant host, made the stay a wonde...
5	301008	8265	209819	2018-01-10	Have stayed with RuRu a number of times and

```
SELECT *
FROM Reviews
WHERE comments LIKE 'Had a brilliant stay with Alision, which is %';
```

(255)	date_review date	reviewer_id integer	reviewer_name character varying (255)	comments text
	2018-02-06	11193188	Laurie	Had a brilliant s...

-- Trobem l'id a la taula d'importació per comprovar que ha estat ben importat.

```
SELECT COUNT (comments) AS total_comments
FROM reviews
WHERE reviewer_id = 11193188
```

	total_comments bigint
1	105

```
SELECT COUNT (comments) AS short_comments
FROM reviews
WHERE reviewer_id = 11193188 AND LENGTH(comments) < 100
```

	short_comments bigint
1	11

-- Obtenim el mateix resultat així que podem assumir que la importació està ben feta.

-- Repetint aquest procediment amb la resta de resultats, podem comprovar que la resta de resultats també concorden amb el que ha retornat la query.

Query 10:

You can build up any query you think it would be interesting to analyze, there are no restrictions. The mark of this query will be proportional to the importance of the analyzed aspect.

```
--What are the 5 most expensive property types and what is their
average price per day.
SELECT pt.name, CONCAT('$', CAST(AVG(price) AS DECIMAL (10,2))) AS
property_price
FROM PropertyType AS pt, Apartment AS ap
WHERE pt.id_property = ap.id_property
GROUP BY pt.id_property
ORDER BY AVG(price) DESC
LIMIT 5;
```

	name character varying (255)	property_price text
1	Boutique hotel	\$355.16
2	Castle	\$335.50
3	Nature lodge	\$332.50
4	Farm stay	\$274.40
5	Boat	\$261.13

-- Com podem observar, el tipus de propietat més car es el Boutique hotel, seguit del Castle, Nature lodge, Farm stay i, finalment, Boat. Aquest resultat crida l'atenció, ja que allotjar-se en un castell sonà més car que allotjar-se en un hotel, encara que sigui un tipus especial d'hotel (Un boutique hotel es tipus d'hotel amb servei personalitzat i disseny sofisticat, que té un menor nombre d'habitacions que un hotel normal, entre 10 i 100. De vegades també són temàtics.)

```
SELECT price
FROM Apartments
WHERE property_type LIKE 'Boutique hotel';
```

	price character varying (255)
1	\$100.00
2	\$100.00
3	\$318.00
4	\$100.00
5	\$199.00

-- Calculem la mitja utilitzant els valors que ens retorna i observem que la mitja es 355,1509, que arrodonit es 355,16 com ens retorna la query.

Conclusions

Al començar aquesta practica, no sabia ben be com separar les taules inicials per normalitzar-les, ja que em donava la impressió que ja estaven normalitzades, excepte per les dades que es repetien en les diferent taules. Tampoc estava segura de com tractar els arrays, ja que no sabia que es podien separar, però sabia que no es devien emmagatzemar arrays a una casella.

Inicialment tenia quatre taules, Apartment, Host, Review i Reviewer. No se'm acudia com separar-les més. Després vaig aprendre que els arrays es podien separar i vaig crear dues taules noves per aquest, encara que tenia la impressió de que aquestes taules no contenien la suficient informació com per a ser la seva pròpia taula al principi.

Després d'aquestes, vaig crear quatre taules més, Country, State, City i Street. Mentre feia la practica, em vaig adonar del atribut `neighbourhood_cleansed`, i vaig crear una nova taula per a Neighborhood. Inicialment totes aquestes taules excepte Country eren febles, el que m'obligava a posar a totes un PK/FK de la taula de la que venia. Més endavant em vaig adonar de que probablement seria més senzill a l'hora de fer quèries si tractés les dades com a una sola taula, ja que separant-les a l'hora de buscar qualsevol dada sobre un carrer en certa ciutat per exemple, hauria de cridar totes les taules anterior, així que vaig crear la taula Address.

Mentre estava fent el model físic, em vaig adonar de que `Property_type` era una manera de classificar els Apartments, així que vaig crear la taula `PropertyType`.

Mentre feia el model físic, vaig haver d'aprendre a separar els arrays en columnes. Quan vaig aprendre, encara tenia el problema de que separant els arrays en columnes, encara tenia tota la informació de cada array, el que volia dir que si dos arrays contenien la mateixa informació, aquesta informació apareixeria dues vegades, així que vaig haver de crear una taula auxiliar.

També va ser un problema el relacionar les taules entre si, ja que no havia conservat cap dels ids originals. Al final, el que vaig fer va ser afegir els ids originals a les taules que ho necessitaven i després eliminar-los.

A més, buscant com tractar un percentatge vaig aprendre que la millor manera de tractar-ho es guardant-ho com un REAL dividit entre 100, ja que un percentatge es en realitat un valor dividit entre 100, i a l'hora de fer operacions amb aquests es més senzill tractar-los així.

Una vegada vaig començar la segona part em vaig adonar d'alguns errors que havia fet al model físic, com que havia posat que la relació entre Address i Apartment era 1:1, així que vaig haver de canviar com importava les dades a aquesta taula, ja que Address tenia el mateix nombre de files que Apartment. Així vaig aprendre que els valors NULLs no es poden comparar entre si, perquè sempre retornen fals. També vaig millorar l'estat, ja que inicialment no el vaig tractar i dues Address idèntiques excepte pel fet que alguns states estaven escrits sencers i altres només amb tres lletres, eren tractades com a diferents. D'aquesta manera vaig aprendre que les dades son case-sensitive, així que vaig utilitzar la funció UPPER.

També he tractat les dades de diners que abans no havia tractat i em vaig adonar que la funció TRIM no serveix per a treure comes i que per a tractar diners es millor tractar les dades com a DECIMAL(10,2) que MONEY, ja que MONEY en ocasions dona problemes. A mi em va donar problemes perquè el meu ordinador no reconeix \$ sinó €.

Fent les quèries em vaig adonar de que quan hi ha varis COUNTs en una sola query es multipliquen entre si. A més he après a utilitzar els CASE, els CONCAT, els CAST i he descobert la funció CURRENT_DATE que retorna la data actual.

En conclusió, aquesta practica m'ha servit per aprendre a organitzar millor les dades, a separar els arrays en una columna, a que per a ajuntar les taules, necessitem tenir alguna informació en comú entre aquestes, encara que sembli obvi, i la millor manera de tractar els percentatges.