**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

# Databases Project – Spring 2019

Team No: 26

Names: Camilla Giaccari, Hédi Sassi, Simon Perriard

## Contents

# Deliverable 1

## *Assumptions*

We made no assumption concerning the correctness of the data, we checked every field of every CSV file. The type of each field has been checked and each line containing a wrong input (i.e. missing mandatory field, negative price,...) has been kicked out of the dataset. The only assumption we made was that all the entries in city_listings.csv where in that city and we overwrote the CITY field in each entry with Barcelona, Madrid or Berlin.

We defined some mandatory fields, listed below.

Listings: listing_id, listing_url, listing_name, host_id, host_url, host_name

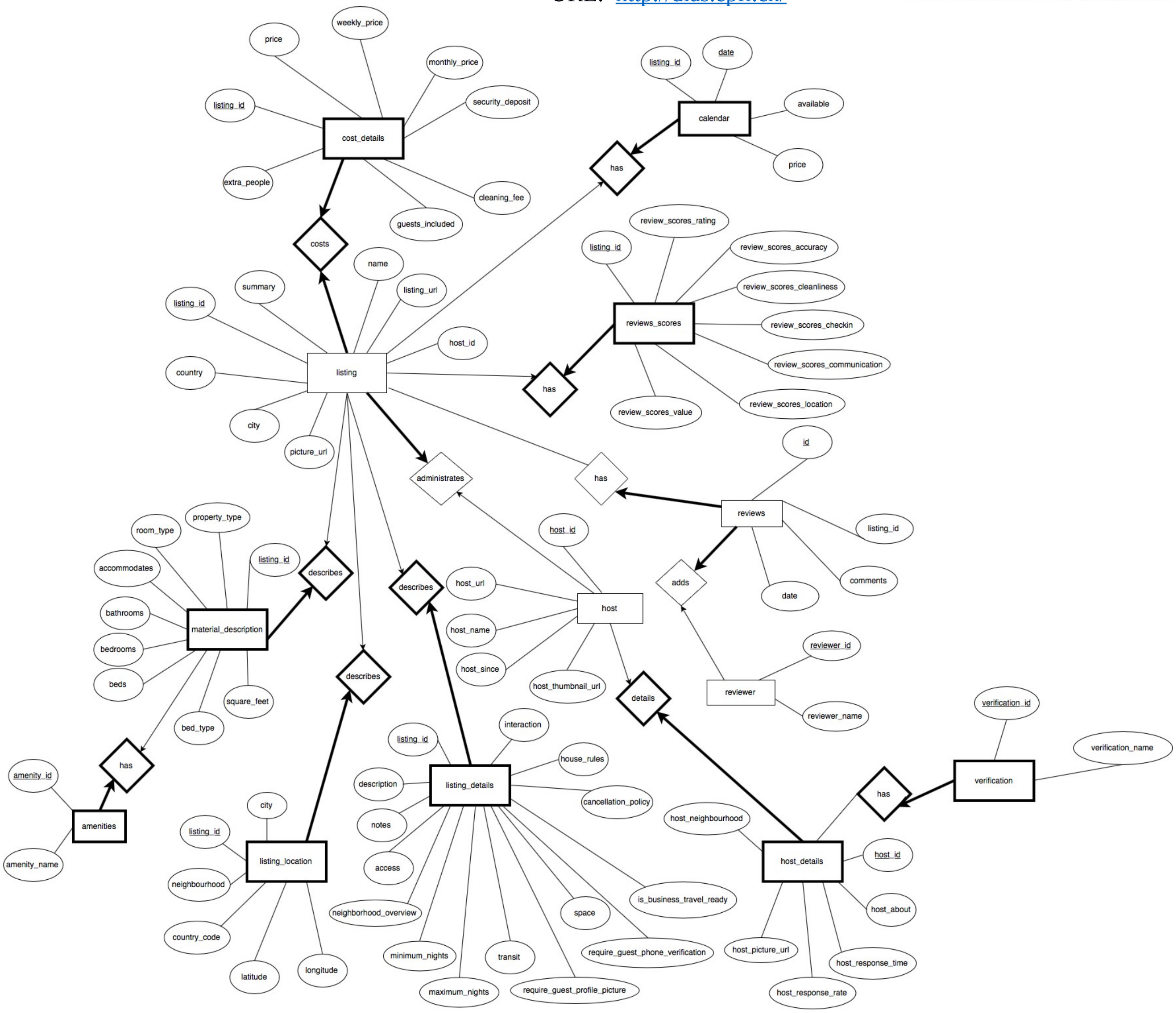
Reviews: all fields are mandatory

Calendar: all fields except price are mandatory

## *Entity Relationship Schema*

### Schema

The schema can be found here :
https://github.com/hedi-sassi/rbnb_db_project/tree/master/ER



## Description

First, we chose to separate the main listing attributes and the listing's details. This way, when we want display a lot of listings, we only query the table with the main information (small description, name, thumbnail etc..) and if someone select the listing, we also query the tables containing the details.

We did the same for the host and host details tables.

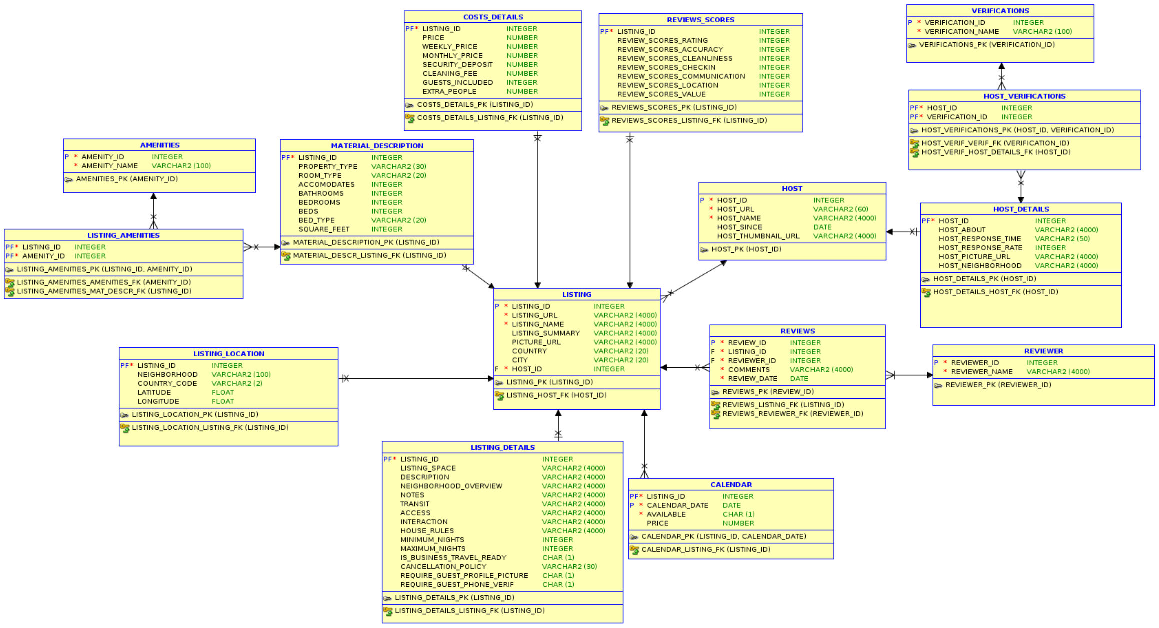
The listing table is connected (one to one mapping) with the calendar, review scores, material description and cost details tables. This implies they are all weak entities with respect to the listing table.

We decided to create special tables to hold the amenities and the host verifications as those are list attributes. We link them to the listing using intermediate tables containing the listing id and the amenities/host verifications id.

# *Relational Schema*

## ER schema to Relational schema

The schema can be found here : https://github.com/hedi-sassi/rbnb_db_project/tree/master/relational_model


COSTS_DETAILS
PF* LISTING_ID INTEGER
PRICE NUMBER
WEEKLY_PRICE NUMBER
MONTHLY_PRICE NUMBER
SECURITY_DEPOSIT NUMBER
CLEANING_FEE NUMBER
GUESTS_INCLUDED INTEGER
EXTRA_PEOPLE NUMBER
COSTS_DETAILS_PK (LISTING_ID)
COSTS_DETAILS_LISTING_FK (LISTING_ID)
REVIEWS_SCORES
PF* LISTING_ID INTEGER
REVIEW_SCORES_RATING INTEGER
REVIEW_SCORES_ACCURACY INTEGER
REVIEW_SCORES_CLEANLINESS INTEGER
REVIEW_SCORES_CHECKIN INTEGER
REVIEW_SCORES_COMMUNICATION INTEGER
REVIEW_SCORES_LOCATION INTEGER
REVIEW_SCORES_VALUE INTEGER
REVIEWS_SCORES_PK (LISTING_ID)
REVIEWS_SCORES_LISTING_FK (LISTING_ID)
VERIFICATIONS
P * VERIFICATION_ID INTEGER
* VERIFICATION_NAME VARCHAR2 (100)
VERIFICATIONS_PK (VERIFICATION_ID)
HOST_VERIFICATIONS
PF* HOST_ID INTEGER
PF* VERIFICATION_ID INTEGER
HOST_VERIFICATIONS_PK (HOST_ID, VERIFICATION_ID)
HOST_VERIF_VERIF_FK (VERIFICATION_ID)
HOST_VERIF_HOST_DETAILS_FK (HOST_ID)
AMENITIES
P * AMENITY_ID INTEGER
* AMENITY_NAME VARCHAR2 (100)
AMENITIES_PK (AMENITY_ID)
MATERIAL_DESCRIPTION
PF* LISTING_ID INTEGER
PROPERTY_TYPE VARCHAR2 (30)
ROOM_TYPE VARCHAR2 (20)
ACCOMODATES INTEGER
BATHROOMS INTEGER
BEDROOMS INTEGER
BEDS INTEGER
BED_TYPE VARCHAR2 (20)
SQUARE_FEET INTEGER
MATERIAL_DESCRIPTION_PK (LISTING_ID)
MATERIAL_DESCR_LISTING_FK (LISTING_ID)
HOST
P * HOST_ID INTEGER
* HOST_URL VARCHAR2 (60)
* HOST_NAME VARCHAR2 (4000)
HOST_SINCE DATE
HOST_THUMBNAIL_URL VARCHAR2 (4000)
HOST_PK (HOST_ID)
HOST_DETAILS
PF* HOST_ID INTEGER
HOST_ABOUT VARCHAR2 (4000)
HOST_RESPONSE_TIME VARCHAR2 (50)
HOST_RESPONSE_RATE INTEGER
HOST_PICTURE_URL VARCHAR2 (4000)
HOST_NEIGHBORHOOD VARCHAR2 (4000)
HOST_DETAILS_PK (HOST_ID)
HOST_DETAILS_HOST_FK (HOST_ID)
LISTING_AMENITIES
PF* LISTING_ID INTEGER
PF* AMENITY_ID INTEGER
LISTING_AMENITIES_PK (LISTING_ID, AMENITY_ID)
LISTING_AMENITIES_AMENITIES_FK (AMENITY_ID)
LISTING_AMENITIES_MAT_DESCR_FK (LISTING_ID)
LISTING
P * LISTING_ID INTEGER
* LISTING_URL VARCHAR2 (4000)
* LISTING_NAME VARCHAR2 (4000)
LISTING_SUMMARY VARCHAR2 (4000)
PICTURE_URL VARCHAR2 (4000)
COUNTRY VARCHAR2 (20)
CITY VARCHAR2 (20)
F * HOST_ID INTEGER
LISTING_PK (LISTING_ID)
LISTING_HOST_FK (HOST_ID)
REVIEWS
P * REVIEW_ID INTEGER
F * LISTING_ID INTEGER
F * REVIEWER_ID INTEGER
* COMMENTS VARCHAR2 (4000)
* REVIEW_DATE DATE
REVIEWS_PK (REVIEW_ID)
REVIEWS_LISTING_FK (LISTING_ID)
REVIEWS_REVIEWER_FK (REVIEWER_ID)
REVIEWER
P * REVIEWER_ID INTEGER
* REVIEWER_NAME VARCHAR2 (4000)
REVIEWER_PK (REVIEWER_ID)
LISTING_LOCATION
PF* LISTING_ID INTEGER
NEIGHBORHOOD VARCHAR2 (100)
COUNTRY_CODE VARCHAR2 (2)
LATITUDE FLOAT
LONGITUDE FLOAT
LISTING_LOCATION_PK (LISTING_ID)
LISTING_LOCATION_LISTING_FK (LISTING_ID)
LISTING_DETAILS
PF* LISTING_ID INTEGER
LISTING_SPACE VARCHAR2 (4000)
DESCRIPTION VARCHAR2 (4000)
NEIGHBORHOOD_OVERVIEW VARCHAR2 (4000)
NOTES VARCHAR2 (4000)
TRANSIT VARCHAR2 (4000)
ACCESS VARCHAR2 (4000)
INTERACTION VARCHAR2 (4000)
HOUSE_RULES VARCHAR2 (4000)
MINIMUM_NIGHTS INTEGER
MAXIMUM_NIGHTS INTEGER
IS_BUSINESS_TRAVEL_READY CHAR (1)
CANCELLATION_POLICY VARCHAR2 (30)
REQUIRE_GUEST_PROFILE_PICTURE CHAR (1)
REQUIRE_GUEST_PHONE_VERIF CHAR (1)
LISTING_DETAILS_PK (LISTING_ID)
LISTING_DETAILS_LISTING_FK (LISTING_ID)
CALENDAR
PF* LISTING_ID INTEGER
P * CALENDAR_DATE DATE
* AVAILABLE CHAR (1)
PRICE NUMBER
CALENDAR_PK (LISTING_ID, CALENDAR_DATE)
CALENDAR_LISTING_FK (LISTING_ID)

Weak entities are accounted for with the help of foreign keys. If the foreign key is not present, it will trigger a “Cascade” deletion policy.

## DDL

The DDL can be found here:
https://github.com/hedi-sassi/rbnb_db_project/tree/master/relational_model

```
CREATE TABLE amenities (

    amenity_id     INTEGER NOT NULL,

    amenity_name   VARCHAR2(100) NOT NULL

);


ALTER TABLE amenities ADD CONSTRAINT amenities_pk PRIMARY KEY ( amenity_id );


CREATE TABLE calendar (

    listing_id      INTEGER NOT NULL,

    calendar_date   DATE NOT NULL,

    available       CHAR(1) NOT NULL,

    price           NUMBER

);


ALTER TABLE calendar ADD CONSTRAINT calendar_pk PRIMARY KEY ( listing_id,

                                                              calendar_date );


CREATE TABLE costs_details (

    listing_id        INTEGER NOT NULL,

    price             NUMBER,
    weekly_price       NUMBER,
    monthly_price      NUMBER,
    security_deposit   NUMBER,
    cleaning_fee       NUMBER,
    guests_included    INTEGER,
    extra_people       NUMBER
);

ALTER TABLE costs_details ADD CONSTRAINT costs_details_pk PRIMARY KEY ( listing_id );

CREATE TABLE host (
    host_id            INTEGER NOT NULL,
    host_url           VARCHAR2(60) NOT NULL,
    host_name          VARCHAR2(4000) NOT NULL,
    host_since         DATE,
    host_thumbnail_url   VARCHAR2(4000)
);

ALTER TABLE host ADD CONSTRAINT host_pk PRIMARY KEY ( host_id );

CREATE TABLE host_details (
    host_id            INTEGER NOT NULL,
    host_about         VARCHAR2(4000),
    host_response_time   VARCHAR2(50),
    host_response_rate   INTEGER,
    host_picture_url     VARCHAR2(4000),
    host_neighborhood    VARCHAR2(4000)
);

ALTER TABLE host_details ADD CONSTRAINT host_details_pk PRIMARY KEY ( host_id );

CREATE TABLE host_verifications (
    host_id           INTEGER NOT NULL,
    verification_id   INTEGER NOT NULL
);

ALTER TABLE host_verifications ADD CONSTRAINT host_verifications_pk PRIMARY KEY
( host_id,
                                                                          verification_id );

CREATE TABLE listing (
    listing_id        INTEGER NOT NULL,
    listing_url       VARCHAR2(4000) NOT NULL,
    listing_name      VARCHAR2(4000) NOT NULL,
    listing_summary   VARCHAR2(4000),
    picture_url       VARCHAR2(4000),
    country          VARCHAR2(20),

    city             VARCHAR2(20),

    host_id          INTEGER NOT NULL

);


ALTER TABLE listing ADD CONSTRAINT listing_pk PRIMARY KEY ( listing_id );


CREATE TABLE listing_amenities (

    listing_id   INTEGER NOT NULL,

    amenity_id   INTEGER NOT NULL

);


ALTER TABLE listing_amenities ADD CONSTRAINT listing_amenities_pk PRIMARY KEY
( listing_id,

                                                                    amenity_id );


CREATE TABLE listing_details (

    listing_id                   INTEGER

        CONSTRAINT nnc_listing_details_listing_id NOT NULL,

    listing_space                 VARCHAR2(4000),

    description                   VARCHAR2(4000),

    neighborhood_overview          VARCHAR2(4000),

    notes                         VARCHAR2(4000),
    transit                        VARCHAR2(4000),
    "ACCESS"                       VARCHAR2(4000),
    interaction                    VARCHAR2(4000),
    house_rules                    VARCHAR2(4000),
    minimum_nights                 INTEGER,
    maximum_nights                 INTEGER,
    is_business_travel_ready       CHAR(1),
    cancellation_policy            VARCHAR2(30),
    require_guest_profile_picture  CHAR(1),
    require_guest_phone_verif      CHAR(1)
);

ALTER TABLE listing_details ADD CONSTRAINT listing_details_pk PRIMARY KEY
( listing_id );

CREATE TABLE listing_location (
    listing_id     INTEGER NOT NULL,
    neighborhood   VARCHAR2(100),
    country_code   VARCHAR2(2),
    latitude       FLOAT,
    longitude      FLOAT
);
```

```
ALTER TABLE listing_location ADD CONSTRAINT listing_location_pk PRIMARY KEY
( listing_id );


CREATE TABLE material_description (

    listing_id      INTEGER NOT NULL,

    property_type   VARCHAR2(30),

    room_type       VARCHAR2(20),

    accomodates     INTEGER,

    bathrooms       INTEGER,

    bedrooms        INTEGER,

    beds            INTEGER,

    bed_type        VARCHAR2(20),

    square_feet     INTEGER

);


ALTER TABLE material_description ADD CONSTRAINT material_description_pk PRIMARY
KEY ( listing_id );


CREATE TABLE reviewer (

    reviewer_id     INTEGER NOT NULL,

    reviewer_name   VARCHAR2(4000) NOT NULL

);
```

```
ALTER TABLE reviewer ADD CONSTRAINT reviewer_pk PRIMARY KEY ( reviewer_id );


CREATE TABLE reviews (
    review_id     INTEGER NOT NULL,
    listing_id    INTEGER NOT NULL,
    reviewer_id   INTEGER NOT NULL,
    comments      VARCHAR2(4000) NOT NULL,
    review_date   DATE NOT NULL
);


ALTER TABLE reviews ADD CONSTRAINT reviews_pk PRIMARY KEY ( review_id );


CREATE TABLE reviews_scores (
    listing_id                    INTEGER NOT NULL,
    review_scores_rating          INTEGER,
    review_scores_accuracy        INTEGER,
    review_scores_cleanliness     INTEGER,
    review_scores_checkin         INTEGER,
    review_scores_communication   INTEGER,
    review_scores_location        INTEGER,
    review_scores_value           INTEGER
);
```

```
ALTER TABLE reviews_scores ADD CONSTRAINT reviews_scores_pk PRIMARY KEY
( listing_id );


CREATE TABLE verifications (
    verification_id     INTEGER NOT NULL,
    verification_name   VARCHAR2(100) NOT NULL
);


ALTER TABLE verifications ADD CONSTRAINT verifications_pk PRIMARY KEY
( verification_id );


ALTER TABLE calendar
    ADD CONSTRAINT calendar_listing_fk FOREIGN KEY ( listing_id )
        REFERENCES listing ( listing_id )
            ON DELETE CASCADE;


ALTER TABLE costs_details
    ADD CONSTRAINT costs_details_listing_fk FOREIGN KEY ( listing_id )
        REFERENCES listing ( listing_id )
            ON DELETE CASCADE;


ALTER TABLE host_details
    ADD CONSTRAINT host_details_host_fk FOREIGN KEY ( host_id )
```

```
        REFERENCES host ( host_id )
            ON DELETE CASCADE;


ALTER TABLE host_verifications
    ADD CONSTRAINT host_verif_host_details_fk FOREIGN KEY ( host_id )
        REFERENCES host_details ( host_id )
            ON DELETE CASCADE;


ALTER TABLE host_verifications
    ADD CONSTRAINT host_verif_verif_fk FOREIGN KEY ( verification_id )
        REFERENCES verifications ( verification_id )
            ON DELETE CASCADE;


ALTER TABLE listing_amenities
    ADD CONSTRAINT listing_amenities_amenities_fk FOREIGN KEY ( amenity_id )
        REFERENCES amenities ( amenity_id )
            ON DELETE CASCADE;


ALTER TABLE listing_amenities
    ADD CONSTRAINT listing_amenities_mat_descr_fk FOREIGN KEY ( listing_id )
        REFERENCES material_description ( listing_id )
            ON DELETE CASCADE;
```

```
ALTER TABLE listing_details
    ADD CONSTRAINT listing_details_listing_fk FOREIGN KEY ( listing_id )
        REFERENCES listing ( listing_id )
            ON DELETE CASCADE;


ALTER TABLE listing
    ADD CONSTRAINT listing_host_fk FOREIGN KEY ( host_id )
        REFERENCES host ( host_id )
            ON DELETE CASCADE;


ALTER TABLE listing_location
    ADD CONSTRAINT listing_location_listing_fk FOREIGN KEY ( listing_id )
        REFERENCES listing ( listing_id )
            ON DELETE CASCADE;


ALTER TABLE material_description
    ADD CONSTRAINT material_descr_listing_fk FOREIGN KEY ( listing_id )
        REFERENCES listing ( listing_id )
            ON DELETE CASCADE;


ALTER TABLE reviews
    ADD CONSTRAINT reviews_listing_fk FOREIGN KEY ( listing_id )
        REFERENCES listing ( listing_id )
```

```
      ON DELETE CASCADE;


ALTER TABLE reviews
  ADD CONSTRAINT reviews_reviewer_fk FOREIGN KEY ( reviewer_id )
    REFERENCES reviewer ( reviewer_id );


ALTER TABLE reviews_scores
  ADD CONSTRAINT reviews_scores_listing_fk FOREIGN KEY ( listing_id )
    REFERENCES listing ( listing_id )
      ON DELETE CASCADE;
```

## *General Comments*

We split the work as followed:

- ER model: Camilla
- Relational Model : Simon
- Data verification (scala program on the repo) : Hédi

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

# Deliverable 2

## *Assumptions*

We made no assumption about the data.

The ER schema and the Relational schema have been updated following the directions given by the TAs for Milestone1.

## *Data Loading*

We organized the data as described in the Relational schema and imported it using SQLDeveloper.

## *Query Implementation*

### Query 1:

What is the average price for a listing with 8 bedrooms?

*Description of logic:*

We take the average value of the price attribute, considering only the listings with 8 bedrooms in their material description.

*SQL statement*

```
select AVG(CD.PRICE)

from COSTS_DETAILS CD

  INNER JOIN MATERIAL_DESCRIPTION M

  ON CD.LISTING_ID = M.LISTING_ID

where M.BEDROOMS = 8;
```

### *Result*

Only one row with value 313,153846153846153846153846153846153846

## Query 2:

What is the average cleaning review score for listings with TV?

### *Description of logic:*

We take the average value of the REVIEW_SCORES_CLEANLINESS attribute, considering only the listings with ‘TV’ in their amenities.

### *SQL statement*

```
select AVG(RS.REVIEW_SCORES_CLEANLINESS)

from REVIEWS_SCORES RS

  INNER JOIN LISTING_AMENITIES LA

  ON RS.LISTING_ID = LA.LISTING_ID

  INNER JOIN AMENITIES AM

  ON AM.AMENITY_ID = LA.AMENITY_ID

where AM.AMENITY_NAME = 'TV';
```

### *Result*

Only one row with value 9,39864565813932902540497477206337965832

## Query 3:

Print all the hosts who have an available property between date 03.2019 and 09.2019.

### *Description of logic:*

We take all the informations about the hosts who have a listing with available date following (>=) 03.2019 and prior (<=) 09.2019.

### *SQL statement*

```
select *

from HOST H

where H.HOST_ID IN (select L.HOST_ID
                from LISTING L, CALENDAR CA

                where L.LISTING_ID = CA.LISTING_ID and CA.AVAILABLE = 't' and
CA.CALENDAR_DATE >= '01-MAR-19' and CA.CALENDAR_DATE <= ’30-SEP-19' );
```

*Result*

```
(HOST_ID; HOST_URL; HOST_NAME; HOST_SINCE; HOST_THUMBNAIL_URL)

71615	https://www.airbnb.com/users/show/71615	Mireia And Maria	19-GEN-10
	https://a0.muscache.com/im/users/71615/profile_pic/1426612511/original.jpg?
aki_policy=profile_small

82522	https://www.airbnb.com/users/show/82522	Meritxell	18-FEB-10
	https://a0.muscache.com/im/pictures/ece65ffd-a798-4209-b1b0-
a51060412b29.jpg?aki_policy=profile_small

108310	https://www.airbnb.com/users/show/108310	Pedro	14-APR-10
	https://a0.muscache.com/im/pictures/user/7f7e9c1a-7274-4e90-a797-
f079ffd9a9a3.jpg?aki_policy=profile_small

134698	https://www.airbnb.com/users/show/134698	Svetlana	29-MAG-10
	https://a0.muscache.com/im/users/134698/profile_pic/1334849467/original.jpg?
aki_policy=profile_small

136853	https://www.airbnb.com/users/show/136853	Fidelio	02-GIU-10
	https://a0.muscache.com/im/users/136853/profile_pic/1312382561/original.jpg?
aki_policy=profile_small
```

## Query 4:

Print how many listing items exist that are posted by two different hosts but the hosts have the same name.

*Description of logic:*

We use COUNT to determine how many different listing with (IN) different host_id having the same host_name exist.

*SQL statement*

```
select COUNT(*)
```

```
from LISTING L, HOST H

where L.HOST_ID = H.HOST_ID and H.HOST_ID IN (Select H1.HOST_ID

                    from HOST H1, Host H2

                    where H1.HOST_NAME = H2.HOST_NAME and H1.HOST_ID != H2.HOST_ID

                    );
```

### *Result*

Only one row with value 30343

## Query 5:

Print all the dates that 'Viajes Eco' has available accommodations for rent.

### *Description of logic:*

We take the calendar_date of all the listings with availability value ’t’ (true) and host_name ‘Viajes Eco’.

### *SQL statement*

```
select CA.CALENDAR_DATE

from CALENDAR CA

  INNER JOIN LISTING L

  ON CA.LISTING_ID = L.LISTING_ID and CA.AVAILABLE = 't'

  INNER JOIN HOST H

  ON L.HOST_ID = H.HOST_ID

where H.HOST_NAME = 'Viajes Eco';
```

### *Result*

```
10-NOV-18

11-NOV-18

12-NOV-18
```

13-NOV-18

14-NOV-18

## Query 6:

Find all the hosts (host_ids, host_names) that have only one listing.

### *Description of logic:*

We take only the ones that are in the group of hosts with only one different host_id per listing.

### *SQL statement*

```
select H.HOST_ID, H.HOST_NAME

from HOST H

where H.HOST_ID IN (select L.HOST_ID

                    from LISTING L

                    group by L.HOST_ID having COUNT(*) = 1

                   );
```

### *Result*

```
(HOST_ID; HOST_NAME)

108310                  Pedro

73163         Andres

158596                  Ester

90417         Etain

280070                  Cristina
```

## Query 7:

What is the difference in the average price of listings with and without Wifi?

*Description of logic:*

We created a view named “wifi” (with all the listings with ‘WiFi’ in their amenities) and then used it in the SQL statement:

```
create view wifi as

  select LA.LISTING_ID

  from AMENITIES AM, LISTING_AMENITIES LA

  where AM.AMENITY_ID = LA.AMENITY_ID and AM.AMENITY_NAME = 'Wifi';
```

*SQL statement*

```
select AVG(CD1.PRICE) - AVG(CD2.PRICE)

from COSTS_DETAILS CD1, COSTS_DETAILS CD2

where CD1.LISTING_ID in (select * from wifi)

and CD2.LISTING_ID not in (select * from wifi);
```

*Result*

Only one row with value 6,66174164496683882662676775669337783597

## Query 8:

How much more (or less) costly to rent a room with 8 beds in Berlin compared to Madrid on average?

*Description of logic:*

We take the subtraction of two average prices: the first one is from the listings with 8 beds in their material_description and Berlin as their city; the second one is from the listings with 8 beds in their material_description and Madrid as their city.

*SQL statement*

```
select AVG(CD1.PRICE) - AVG(CD2.PRICE)

from COSTS_DETAILS CD1, COSTS_DETAILS CD2

where CD1.LISTING_ID IN (select MD.LISTING_ID

                        from MATERIAL_DESCRIPTION MD
```

```
                         where MD.BEDS = 8)


and CD1.LISTING_ID IN (select L.LISTING_ID

                      from LISTING L

                      where L.CITY = 'Berlin')


and CD2.LISTING_ID not in (select MD.LISTING_ID

                         from MATERIAL_DESCRIPTION MD

                         where MD.BEDS = 8)


and CD2.LISTING_ID IN (select L.LISTING_ID

                      from LISTING L

                      where L.CITY = 'Madrid');
```

### *Result*

Only one row with value 44,46580490444090251071110006287744746763

## Query 9:

Find the top-10 (in terms of the number of listings) hosts (host_ids, host_names) in Spain.

### *Description of logic:*

We use “order by COUNT(*) DESC” to determine the hosts with the more listings in a descending order. We take only the listings with Spain as their country. We use “where rownum <= 10” to take only the first 10 rows of the result.

### *SQL statement*

```
select * from

  (select H.HOST_ID , H.HOST_NAME
  from LISTING L, HOST H

  where L.COUNTRY = 'Spain' and L.HOST_ID = H.HOST_ID

  group by L.HOST_ID, H.HOST_NAME, H.HOST_ID

  order by COUNT(*) DESC)

  where rownum <= 10;
```

*Result*

(HOST_ID; HOST_NAME)

| | |
|---|---|
| 4459553 | Eva&Jacques |
| 99018982 | Apartamentos |
| 32046323 | Juan |
| 28038703 | Luxury Rentals Madrid |
| 1391607 | Aline |

## Query 10:

Find the top-10 rated apartments in Barcelona.

*Description of logic:*

As for the previous query, we use “order by RS.REVIEW_SCORES_RATING DESC” to determine the rating scores in a descending order. We take only the listings with Barcelona as their city and ‘Apartment’ as their type. We use “where rownum <= 10” to take only the first 10 rows of the result.

*SQL statement*

```
select * from (

select L.LISTING_ID, L.LISTING_NAME

 from LISTING L

  INNER JOIN REVIEWS_SCORES RS
```

```
    ON L.LISTING_ID = RS.LISTING_ID and L.CITY = 'Barcelona'

    INNER JOIN MATERIAL_DESCRIPTION MD

    ON MD.LISTING_ID = L.LISTING_ID and MD.PROPERTY_TYPE = 'Apartment'

  order by RS.REVIEW_SCORES_RATING DESC)

where rownum <=10;
```

### *Result*

(LISTING_ID; LISTING_NAME)

| | |
|---|---|
| 475786 | Room to rent in beautiful apartment |
| 763465 | Very Nice Room to rent in Raval |
| 783032 | room for rent 10 minutes from center |
| 740113 | Sunny, authentic Sant Antoni Apartment |
| 721510 | 32 Valencia Apartment 2 bedrooms |

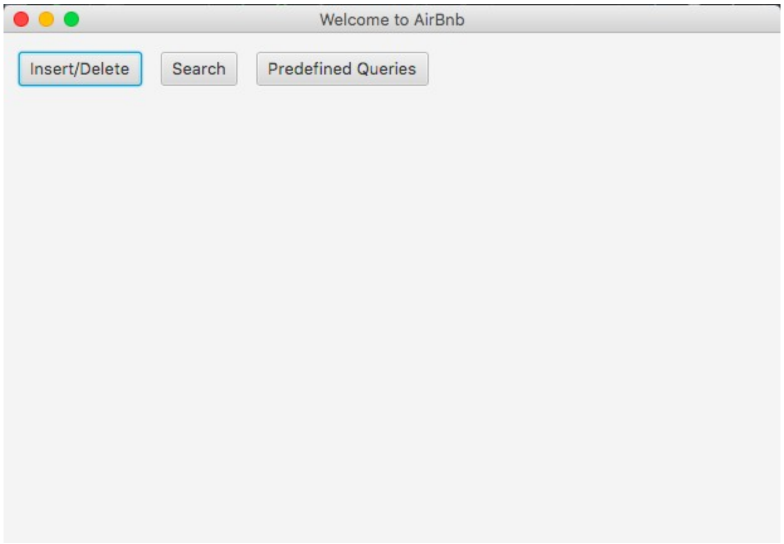
# *Interface*

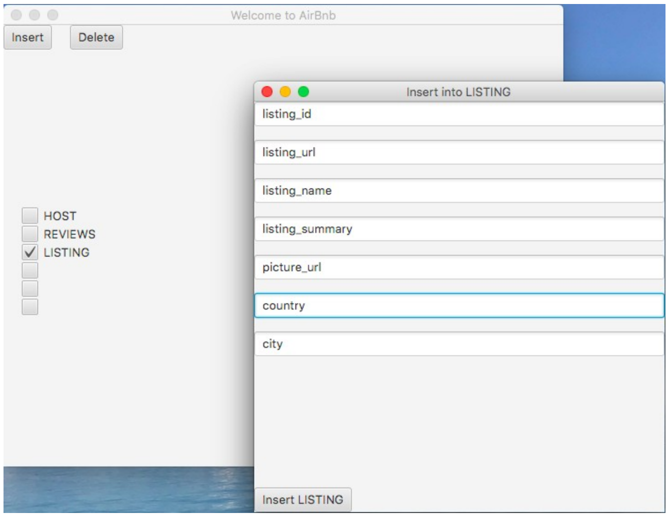
## Design logic Description

We decided to use JavaFX.

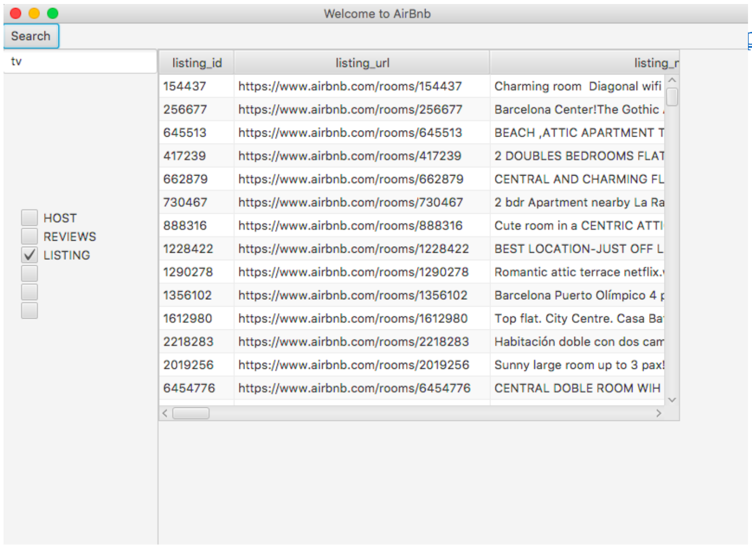
The interface is simple with intuitive buttons: it’s possible to insert/delete data by modifying the attributes of the item; there is a ‘Search’ button to search for a key-word in the database; the 10 queries described above are added as ‘Predefined queries’, so that the program can give the results to the user without showing any SQL language.

## Screenshots





**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

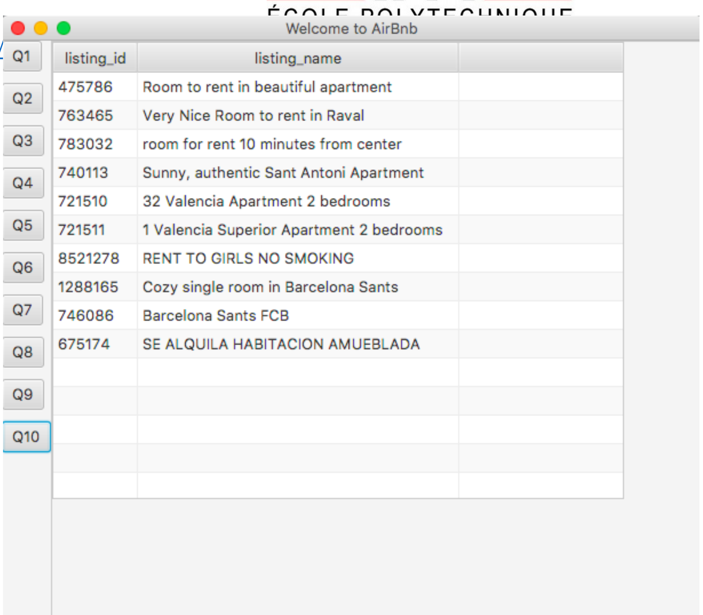


tv

☐ HOST
☐ REVIEWS
☑ LISTING
☐
☐
☐

| listing_id | listing_url | listing_r |
|---|---|---|
| 154437 | https://www.airbnb.com/rooms/154437 | Charming room  Diagonal wifi |
| 256677 | https://www.airbnb.com/rooms/256677 | Barcelona Center!The Gothic |
| 645513 | https://www.airbnb.com/rooms/645513 | BEACH ,ATTIC APARTMENT T |
| 417239 | https://www.airbnb.com/rooms/417239 | 2 DOUBLES BEDROOMS FLAT |
| 662879 | https://www.airbnb.com/rooms/662879 | CENTRAL AND CHARMING FL |
| 730467 | https://www.airbnb.com/rooms/730467 | 2 bdr Apartment nearby La Ra |
| 888316 | https://www.airbnb.com/rooms/888316 | Cute room in a CENTRIC ATTI |
| 1228422 | https://www.airbnb.com/rooms/1228422 | BEST LOCATION-JUST OFF L |
| 1290278 | https://www.airbnb.com/rooms/1290278 | Romantic attic terrace netflix. |
| 1356102 | https://www.airbnb.com/rooms/1356102 | Barcelona Puerto Olímpico 4 |
| 1612980 | https://www.airbnb.com/rooms/1612980 | Top flat. City Centre. Casa Ba |
| 2218283 | https://www.airbnb.com/rooms/2218283 | Habitación doble con dos car |
| 2019256 | https://www.airbnb.com/rooms/2019256 | Sunny large room up to 3 pax! |
| 6454776 | https://www.airbnb.com/rooms/6454776 | CENTRAL DOBLE ROOM WIH |

p://



| listing_id | listing_name |  |
|---|---|---|
| 475786 | Room to rent in beautiful apartment |  |
| 763465 | Very Nice Room to rent in Raval |  |
| 783032 | room for rent 10 minutes from center |  |
| 740113 | Sunny, authentic Sant Antoni Apartment |  |
| 721510 | 32 Valencia Apartment 2 bedrooms |  |
| 721511 | 1 Valencia Superior Apartment 2 bedrooms |  |
| 8521278 | RENT TO GIRLS NO SMOKING |  |
| 1288165 | Cozy single room in Barcelona Sants |  |
| 746086 | Barcelona Sants FCB |  |
| 675174 | SE ALQUILA HABITACION AMUEBLADA |  |

## ***General Comments***

We split the work as followed:

- Data insertion in the DB: Simon and Hédi

- Queries writing: Simon and Hédi

- Queries test and corrections: Camilla

- User interface: Hédi

- Report: Camilla

**DIAS: Data-Intensive Applications and Systems Laboratory**
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne

URL: http://dias.epfl.ch/

# Deliverable 3

# Assumptions

No assumptions where made about the data (as in previous milestones). We wrote a parser to clean, check and regroup the data before insertion.

## *Query Implementation*

All the queries can be found here:
https://github.com/hedi-sassi/rbnb_db_project/blob/master/DB_project.sql

Since the queries are long, it may be better to use the ones on the git repo rather than those written below (not easily readable).

Here is our solutions for the queries of milestone 3:

### Query 1:

*Description of logic:*

First we decided to find the hosts that have listing with the dimension (in square feet) not null and then we grouped those hosts by city, counted those hosts and ordered the result by city

*SQL statement*

```
select count(distinct(h.HOST_ID)), l1.CITY

from HOST h, LISTING l1,  MATERIAL_DESCRIPTION md

where h.HOST_ID = l1.HOST_ID and l1.LISTING_ID = md.LISTING_ID and md.SQUARE_FEET is not null

group by l1.CITY order by l1.CITY asc;
```

Result:

```
345 Barcelona

370 Berlin

249 Madrid
```

## Query 2:

### *Description of logic:*

First we selected all the listings in Madrid with non-null review score rating. Then, we order the listings according to their review score rating per neighborhood (using partition by) and add a column for the rownumber. We then compute the number of listings per neighborhood and we divide it by 2 so we have the median position per neighborhood.

Eventually, we filter the ordered listings to only select the ones that are in the median position per neighborhood and select the results where row number smaller than 5

### *SQL statement*

```
select * from

(select distinct(med_per_ng.NEIGHBORHOOD), REVIEW_SCORES_RATING from

(select distinct(loc.NEIGHBORHOOD), floor((count(*) over(partition by
loc.NEIGHBORHOOD)+1)/2) as median_elem_per_ng
from REVIEWS_SCORES rs, LISTING l, LISTING_LOCATION loc
where rs.LISTING_ID = l.LISTING_ID and loc.LISTING_ID = l.LISTING_ID and
rs.REVIEW_SCORES_RATING is not null and l.CITY = 'Madrid') med_per_ng,

(select loc.NEIGHBORHOOD, rs.REVIEW_SCORES_RATING, ROW_NUMBER() over(partition
by loc.NEIGHBORHOOD order by rs.REVIEW_SCORES_RATING desc) as rnum
from REVIEWS_SCORES rs, LISTING_LOCATION loc, LISTING l
where rs.LISTING_ID = loc.LISTING_ID and l.LISTING_ID = rs.LISTING_ID and l.CITY =
'Madrid' and rs.REVIEW_SCORES_RATING is not null) ranked_by_ng_and_rev

where med_per_ng.NEIGHBORHOOD = ranked_by_ng_and_rev.NEIGHBORHOOD and
median_elem_per_ng = rnum
order by REVIEW_SCORES_RATING desc)
```

```
where rownum <= 5
;
```

Result:

| | |
|---|---|
| Estrella | 100 |
| Tetuán | 100 |
| Hispanoamérica | 98 |
| Vallehermosa | 98 |
| Vicálvaro | 98 |

## Query 3:

### *Description of logic:*

We counted the number of listings per host using the count function and groupy by host_id. We then ranked thoses results and selected the lines with rank = 1 (with ties).

### *SQL statement*

```
select h.HOST_ID, h.HOST_NAME

from

(select HOST_ID, rank() over(order by nbr desc) as rnk

        from

        (select L.HOST_ID, count(*) as nbr

                from LISTING l

group by l.HOST_ID)

) ranked

,

HOST h

where h.HOST_ID = ranked.HOST_ID and ranked.rnk = 1;
```

Result:

```
4459553   Eva&Jacques
```

## Query 4:

### *Description of logic:*

First we filter the concerned listings, then we compute the average price using the calendar and eventually we order and take the top 5 using rownum.

### *SQL statement*

```
select * from

(select AVG(cal.PRICE) as average, cal.LISTING_ID

from
        (select l.LISTING_ID from
LISTING l, MATERIAL_DESCRIPTION md, REVIEWS_SCORES rs, LISTING_DETAILS ld
where l.LISTING_ID = md.LISTING_ID and l.LISTING_ID = rs.LISTING_ID and
l.LISTING_ID = ld.LISTING_ID and l.CITY = 'Berlin'
and md.PROPERTY_TYPE = 'Apartment' and
md.BEDS >= 2 and rs.REVIEW_SCORES_LOCATION >= 8
and ld.CANCELLATION_POLICY = 'flexible' and
l.HOST_ID IN (
                select hv.HOST_ID
                from HOST_VERIFICATIONS hv, VERIFICATIONS v
                where hv.VERIFICATION_ID = v.VERIFICATION_ID and v.VERIFICATION_NAME
                LIKE '%government_id%')
        ) filtered
,
CALENDAR cal
where cal.LISTING_ID = filtered.LISTING_ID and cal.CALENDAR_DATE between date'2019-03-01' and date'2019-04-30' and cal.AVAILABLE = 't'

group by cal.LISTING_ID order by average asc) averaged

where rownum <= 5;
```

Result:

```
        20   1490274
21.0655738  24043706
21.2903226   1368460
        22   7071541
        22   6691656
```

## Query 5:

### *Description of logic:*

We filtered the amenities concerned (from the table with all the amenities for each listings) and then grouped them by listing and selected those with a counted >=2 (thus the listings will have at least 2 of the concerned amenities).

Then we just had to get their review score , partition them by the number of person they can accommodate and rank them.

Eventually we selected those with rank <=5.

### *SQL statement*

```
select * from

(select filtered.LISTING_ID, md.ACCOMODATES, ROW_NUMBER() over(partition by
md.ACCOMODATES order by rs.REVIEW_SCORES_RATING desc) as ranked
from
(select facilities.LISTING_ID from
(select la.LISTING_ID, count(*) as counted
from AMENITIES am, LISTING_AMENITIES la
where la.AMENITY_ID = am.AMENITY_ID and
(am.AMENITY_NAME = 'Wifi' or am.AMENITY_NAME = 'Internet' or
am.AMENITY_NAME = 'TV' or am.AMENITY_NAME = 'Free street parking')
group by la.LISTING_ID) facilities

where facilities.counted >= 2) filtered,

MATERIAL_DESCRIPTION md, REVIEWS_SCORES rs

where filtered.LISTING_ID = md.LISTING_ID and rs.LISTING_ID = filtered.LISTING_ID) rnk

where ranked <= 5
;
```

Result:

```
475786        1

 675175        1

 675174        1

 676924        1

1288165        1

 539349        2
```

## Query 6:

### *Description of logic:*

First we count the number of reviews per listings using a join between the Listing and Reviews tables. Then we order and rank the number of reviews partitioned by host_id. Eventually we select for each  host the top 3 listings.

### *SQL statement*

```
select HOST_ID, LISTING_ID
from(
        select HOST_ID, LISTING_ID, ROW_NUMBER() over(partition by HOST_ID order by
        counted desc) as r
                from
                (select distinct(l.LISTING_ID), l.HOST_ID, count(*) over(partition by
                        l.LISTING_ID) as counted

                        from LISTING l, REVIEWS r
                        where l.LISTING_ID = r.LISTING_ID
                )
        )

where r <= 3;
```

Result:

```
 host id         listing id

   2217      2015

   2217   21315310

   2217   18773184

   3073    6287375

   3718      3176
```

## Query 7:

### *Description of logic:*

First we filter the listings in Berlin that are private rooms. Then we count the amenities partitioned by neighborhood with respect with the amenity name of those listings. Eventually we sort and rank the amenities based on the count and display those with rank <=3.

### *SQL statement*

```
select AMENITY_NAME, NEIGHBORHOOD

from

(select AMENITY_NAME, AMEN_COUNT, NEIGHBORHOOD , row_number() over(partition by
ordered_data.NEIGHBORHOOD order by AMEN_COUNT desc) as rank
from

  (select distinct AMENITY_NAME, AMEN_COUNT, NEIGHBORHOOD
  from

    (select AM.AMENITY_NAME, count(AMENITY_NAME) over(partition by
LOC.NEIGHBORHOOD, AM.AMENITY_NAME) as amen_count, LOC.NEIGHBORHOOD

    from LISTING_LOCATION LOC, AMENITIES AM, LISTING_AMENITIES LA
    where LOC.LISTING_ID = LA.LISTING_ID and LA.AMENITY_ID = AM.AMENITY_ID and
LOC.LISTING_ID in (

      select L.LISTING_ID
      from LISTING L, MATERIAL_DESCRIPTION MD
      where L.LISTING_ID = MD.LISTING_ID and L.CITY = 'Berlin' and MD.ROOM_TYPE =
'Private room'
      )
    ) data_amen

  order by data_amen.NEIGHBORHOOD, data_amen.amen_count desc) ordered_data
  ) ranked_data

where ranked_data.rank <= 3  ;
```

Result:

```
Essentials      Adlershof
Heating          Adlershof
Wifi             Adlershof
Heating          Alt-Hohenschönhausen
Essentials      Alt-Hohenschönhausen
Wifi             Alt-Hohenschönhausen
Wifi             Alt-Treptow
Essentials      Alt-Treptow
```

## Query 8:

### *Description of logic:*

First we created a view with the hosts and their respective verification count using count and group by host id in the host verification table.

Then we order this list by descending order (host with the most diverse way), take the first row and compute (using the review_scores) the average communication review scores for that host.

We do the same (with an inverted ordering) for the host with the least diverse way of verification.

Eventually we compute the difference in a select statement.

### *SQL statement*

```
create view number_of_host_verif as
  select count(*) as verifications, HV.HOST_ID
  from HOST_VERIFICATIONS HV
  group by HV.HOST_ID;

select avg(average_most.avg_m - average_least.avg_l) as diff
from

 (select coalesce(avg(RS1.REVIEW_SCORES_COMMUNICATION),0) as avg_m
 from

    (select h.HOST_ID
    from
      (select n.HOST_ID
      from number_of_host_verif n
      order by n.verifications desc) h
    where rownum = 1) host_most ,

    LISTING L1, REVIEWS_SCORES RS1

    where L1.LISTING_ID = RS1.LISTING_ID and L1.HOST_ID = host_most.HOST_ID and
RS1.REVIEW_SCORES_COMMUNICATION is not null
  ) average_most

  ,
   (select coalesce(avg(RS2.REVIEW_SCORES_COMMUNICATION),0) as avg_l
    from

    (select h2.HOST_ID
    from
      (select n2.HOST_ID
      from number_of_host_verif n2
      order by n2.verifications asc) h2
    where rownum = 1) host_least ,

    LISTING L2, REVIEWS_SCORES RS2

    where L2.LISTING_ID = RS2.LISTING_ID and L2.HOST_ID = host_least.HOST_ID and
RS2.REVIEW_SCORES_COMMUNICATION is not null
  ) average_least;
```

Result:

10 (the second host has apparently no communication review and the first one has 10)

## Query 9:

### *Description of logic:*

First we compute what are the room types that have an average of accommodate > 3 and then we count the number of review per listings and then sum them with respect to the cities.

We then rank the cities based on that sum and select the first one.

### *SQL statement*

```
select * from (

select city from
(select  sum(rev_per_list) over(partition by CITY) as total, CITY from

LISTING l,

(select distinct(rev.LISTING_ID), count(*) over(partition by rev.LISTING_ID) as rev_per_list

from REVIEWS rev) rpl,

(select md.LISTING_ID,  AVG(md.ACCOMODATES) over(partition by md.ROOM_TYPE) as
average_per_room_type

from MATERIAL_DESCRIPTION md) av

where rpl.LISTING_ID = av.LISTING_ID and l.LISTING_ID = av.LISTING_ID and
av.average_per_room_type > 3)

order by total desc)
```

```
where rownum = 1;
```

Result:

```
    Madrid
```

## Query 10:

### *Description of logic:*

First we filter the listings and select only those whose host have been registered before 2017-06-01. Then we remove those who were not occupied in 2019. Finally we compare the ratio of those who were occupied in 2019 vs all the listings per neighborhood (using group by) and display those who have a ratio >= 0.5.

### *SQL statement*

```
select total_listing.NEIGHBORHOOD
from

(select count(*) as total, LOC2.NEIGHBORHOOD

from LISTING_LOCATION LOC2, LISTING L3
where LOC2.LISTING_ID = L3.LISTING_ID and L3.CITY = 'Madrid' group by
LOC2.NEIGHBORHOOD) total_listing

,
(select count(*) as occupied_listings, LOC.NEIGHBORHOOD
from LISTING_LOCATION LOC, LISTING L2
where L2.LISTING_ID = LOC.LISTING_ID and L2.CITY = 'Madrid' and L2.LISTING_ID in (

  select distinct L1.LISTING_ID
  from LISTING L1, CALENDAR CAL
  where CAL.CALENDAR_DATE >= date '2019-01-01' and CAL.AVAILABLE = 'f' and
L1.LISTING_ID = CAL.LISTING_ID
  and L1.LISTING_ID in (

    select L.LISTING_ID
    from LISTING L, HOST H
    where L.CITY = 'Madrid' and L.HOST_ID = H.HOST_ID and H.HOST_SINCE <= date
'2017-06-01'
  )

) group by LOC.NEIGHBORHOOD) filtered_listing

where total_listing.NEIGHBORHOOD = filtered_listing.NEIGHBORHOOD and
(filtered_listing.occupied_listings / total_listing.total) >= 0.5;
```

Result:

```
 Malasaña
```

Prosperidad

Cortes

San Blas

La Chopera

Berruguete

Bellas Vistas

## Query 11:

### *Description of logic:*

We first select the listings that were available in 2018 and then group them by country. Then we compute the total number of listings per country and compute the ratio between those that were available in 2018 and the total number and all of that with respect with the countries (using group by). Eventually, we filter out the countries whose ratio are below 0.2.

### *SQL statement*

```
select filtered.COUNTRY
from

(select count(*) as available, L.COUNTRY
from LISTING L
where L.LISTING_ID in (
 select distinct L1.LISTING_ID
  from LISTING L1, CALENDAR CAL
  where CAL.CALENDAR_DATE >= date '2018-01-01' and CAL.CALENDAR_DATE < date '2019-01-01' and CAL.AVAILABLE = 't' and L1.LISTING_ID = CAL.LISTING_ID
) group by L.COUNTRY) filtered

,

(select count(*) total_listing, L2.COUNTRY
from LISTING L2
group by L2.COUNTRY) total

where filtered.COUNTRY = total.COUNTRY and (filtered.available/ total.total_listing) >=
0.2;
```

Result:

Spain

Germany

## Query 12:

### *Description of logic:*

First we filter the listings that are strict with grace periods per neighborhood and then compute the ratio per neighborhood and filter out if the ratio is smaller than 0.5.

### *SQL statement*

```
select total.NEIGHBORHOOD
from
```

```
(select count(*) total_list, LOC.NEIGHBORHOOD
from LISTING_LOCATION LOC, LISTING L
where LOC.LISTING_ID = L.LISTING_ID and L.CITY = 'Barcelona' group by
LOC.NEIGHBORHOOD) total

,

(
select count(*) strict_count, LOC.NEIGHBORHOOD
from LISTING_LOCATION LOC, LISTING_DETAILS LD, LISTING L2
where LOC.LISTING_ID = LD.LISTING_ID and L2.LISTING_ID = LD.LISTING_ID and L2.CITY
= 'Barcelona' and LD.CANCELLATION_POLICY = 'strict_14_with_grace_period' group by
LOC.NEIGHBORHOOD

) filtered

where total.NEIGHBORHOOD = filtered.NEIGHBORHOOD and (filtered.strict_count /
total.total_list) >= 0.05;
```

Result:

Glòries - El Parc

La Nova Esquerra de l'Eixample

L'Antiga Esquerra de l'Eixample

Sant Pere/Santa Caterina

Sarrià

Sant Gervasi - la Bonanova

# *Query Analysis*

## Selected Queries (and why)

We selected query 12, 3 and 2 because we could easily improve them by putting indexes on the table's access predicates.

The optimized queries toghether with other relevant informations can be found here :

https://github.com/hedi-sassi/rbnb_db_project/blob/master/optimized_queries.sql

### *Query 12*

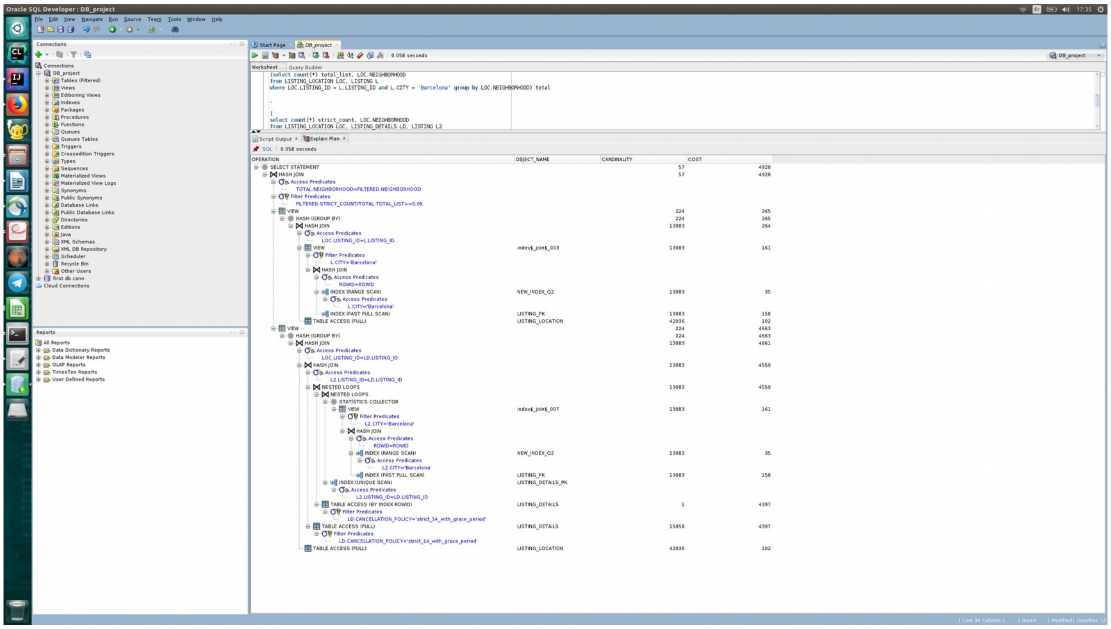
Initial Running time: 0.12 s

Optimized Running time: 0.064 s

Explain the improvement:

We put indexes on the listing city and the cancellation policy. Thus table access predicates are faster since there is an index on the filtered columns.

Initial plan:



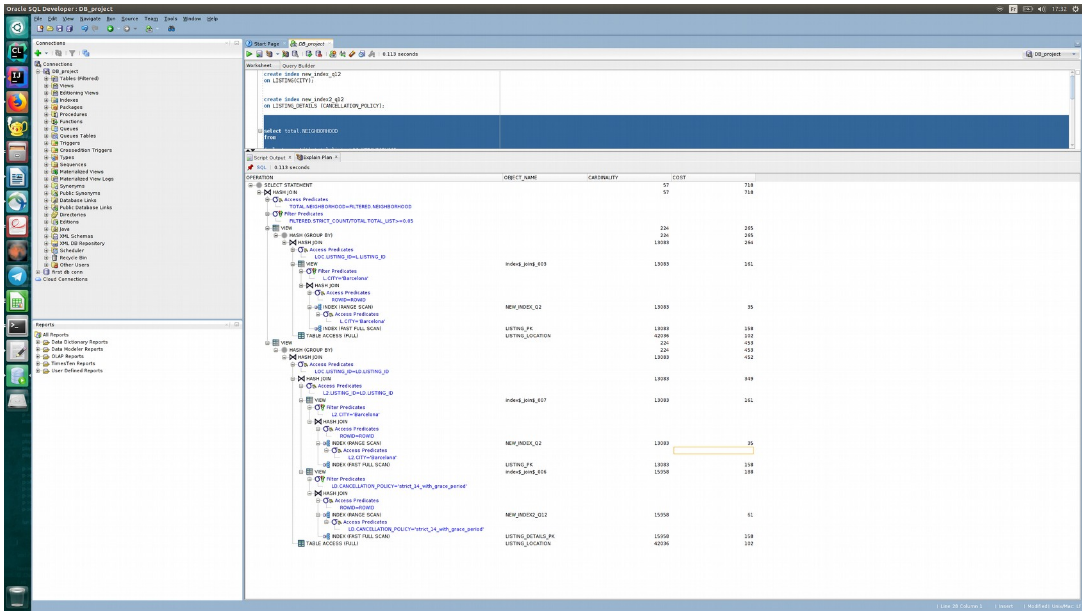
0.058 seconds

```
(select count(*) total_list, LOC.NEIGHBORHOOD
from LISTING_LOCATION LOC, LISTING L
where LOC.LISTING_ID = L.LISTING_ID and L.CITY = 'Barcelona' group by LOC.NEIGHBORHOOD) total

,

(
select count(*) strict_count, LOC.NEIGHBORHOOD
from LISTING_LOCATION LOC, LISTING_DETAILS LD, LISTING L2
```

SQL | 0.058 seconds

```
OPERATION                                                   OBJECT_NAME          CARDINALITY   COST
SELECT STATEMENT                                                                          57   4928
  HASH JOIN                                                                               57   4928
    Access Predicates
      TOTAL.NEIGHBORHOOD=FILTERED.NEIGHBORHOOD
    Filter Predicates
      FILTERED.STRICT_COUNT/TOTAL.TOTAL_LIST>=0.05
    VIEW                                                                                 224    265
      HASH (GROUP BY)                                                                    224    265
        HASH JOIN                                                                      13083    264
          Access Predicates
            LOC.LISTING_ID=L.LISTING_ID
          VIEW                                              index$_join$_003           13083    161
            Filter Predicates
              L.CITY='Barcelona'
            HASH JOIN
              Access Predicates
                ROWID=ROWID
              INDEX (RANGE SCAN)                            NEW_INDEX_Q2               13083     35
                Access Predicates
                  L.CITY='Barcelona'
              INDEX (FAST FULL SCAN)                        LISTING_PK                 13083    158
          TABLE ACCESS (FULL)                               LISTING_LOCATION           42036    102
    VIEW                                                                                 224   4663
      HASH (GROUP BY)                                                                    224   4663
        HASH JOIN                                                                      13083   4661
          Access Predicates
            LOC.LISTING_ID=LD.LISTING_ID
          HASH JOIN                                                                    13083   4559
            Access Predicates
              L2.LISTING_ID=LD.LISTING_ID
            NESTED LOOPS                                                               13083   4559
              NESTED LOOPS
                STATISTICS COLLECTOR
                  VIEW                                      index$_join$_007           13083    161
                    Filter Predicates
                      L2.CITY='Barcelona'
                    HASH JOIN
                      Access Predicates
                        ROWID=ROWID
                      INDEX (RANGE SCAN)                    NEW_INDEX_Q2               13083     35
                        Access Predicates
                          L2.CITY='Barcelona'
                      INDEX (FAST FULL SCAN)                LISTING_PK                 13083    158
                INDEX (UNIQUE SCAN)                         LISTING_DETAILS_PK
                  Access Predicates
                    L2.LISTING_ID=LD.LISTING_ID
              TABLE ACCESS (BY INDEX ROWID)                 LISTING_DETAILS                1   4397
                Filter Predicates
                  LD.CANCELLATION_POLICY='strict_14_with_grace_period'
            TABLE ACCESS (FULL)                             LISTING_DETAILS            15958   4397
              Filter Predicates
                LD.CANCELLATION_POLICY='strict_14_with_grace_period'
          TABLE ACCESS (FULL)                               LISTING_LOCATION           42036    102
```

Improved plan:



Oracle SQL Developer : DB_project
File Edit View Navigate Run Source Team Tools Window Help
Start Page
DB_project
Worksheet
Query Builder
Script Output
Explain Plan

```
create index new_index_q12
on LISTING(CITY);


create index new_index2_q12
on LISTING_DETAILS (CANCELLATION_POLICY);


select total.NEIGHBORHOOD
from
```

SQL | 0.113 seconds

```
OPERATION                                              OBJECT_NAME          CARDINALITY   COST
SELECT STATEMENT                                                                     57    718
  HASH JOIN                                                                          57    718
    Access Predicates
      TOTAL.NEIGHBORHOOD=FILTERED.NEIGHBORHOOD
    Filter Predicates
      FILTERED.STRICT_COUNT/TOTAL.TOTAL_LIST>=0.05
    VIEW                                                                            224    265
      HASH (GROUP BY)                                                               224    265
        HASH JOIN                                                                 13083    264
          Access Predicates
            LOC.LISTING_ID=L.LISTING_ID
          VIEW                                         index$_join$_003           13083    161
            Filter Predicates
              L.CITY='Barcelona'
            HASH JOIN
              Access Predicates
                ROWID=ROWID
              INDEX (RANGE SCAN)                       NEW_INDEX_Q2               13083     35
                Access Predicates
                  L.CITY='Barcelona'
              INDEX (FAST FULL SCAN)                   LISTING_PK                 13083    158
          TABLE ACCESS (FULL)                          LISTING_LOCATION           42036    102
    VIEW                                                                            224    453
      HASH (GROUP BY)                                                               224    453
        HASH JOIN                                                                 13083    452
          Access Predicates
            LOC.LISTING_ID=LD.LISTING_ID
          HASH JOIN                                                               13083    349
            Access Predicates
              L2.LISTING_ID=LD.LISTING_ID
            VIEW                                       index$_join$_007           13083    161
              Filter Predicates
                L2.CITY='Barcelona'
              HASH JOIN
                Access Predicates
                  ROWID=ROWID
                INDEX (RANGE SCAN)                     NEW_INDEX_Q2               13083     35
                  Access Predicates
                    L2.CITY='Barcelona'
                INDEX (FAST FULL SCAN)                 LISTING_PK                 13083    158
            VIEW                                       index$_join$_006           15958    188
              Filter Predicates
                LD.CANCELLATION_POLICY='strict_14_with_grace_period'
              HASH JOIN
                Access Predicates
                  ROWID=ROWID
                INDEX (RANGE SCAN)                     NEW_INDEX2_Q12             15958     61
                  Access Predicates
                    LD.CANCELLATION_POLICY='strict_14_with_grace_period'
                INDEX (FAST FULL SCAN)                 LISTING_DETAILS_PK         15958    158
          TABLE ACCESS (FULL)                          LISTING_LOCATION           42036    102
```

## *Query 3*

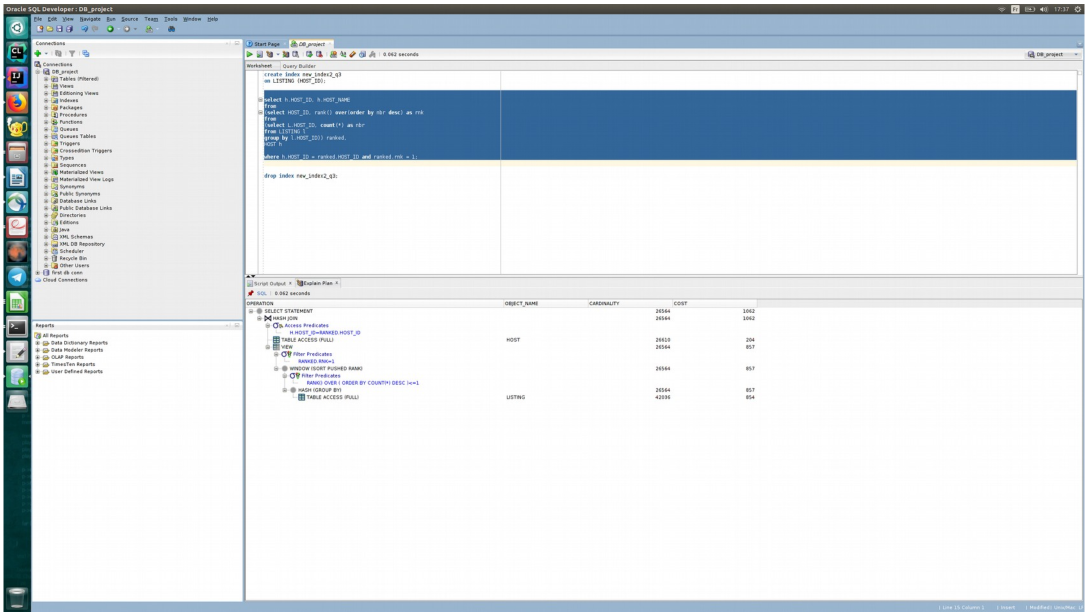
Initial Running time: 0.044 s

Optimized Running time: 0.035 s

Explain the improvement: the join predicate is h.host_id = ranked(listing).host _id but since listings don't have an index on the host id => lose performance.

We only have to add an index on listing.host_id.

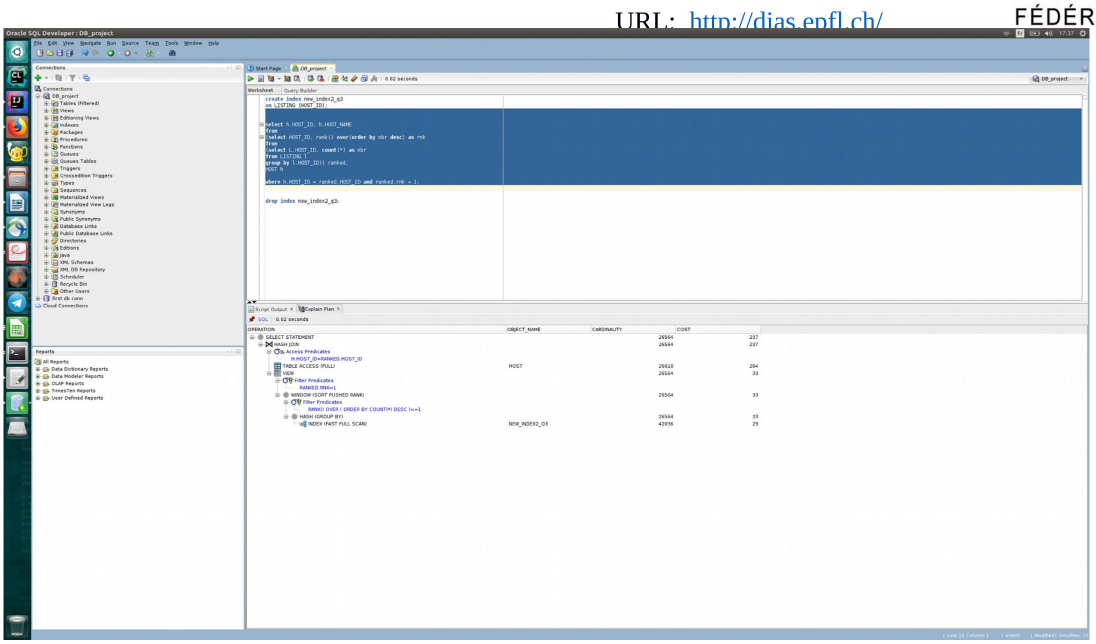
Initial plan:



```
create index new_index2_q3
on LISTING (HOST_ID);


select h.HOST_ID, h.HOST_NAME
from
(select HOST_ID, rank() over(order by nbr desc) as rnk
from
(select L.HOST_ID, count(*) as nbr
from LISTING l
group by l.HOST_ID)) ranked,
HOST h

where h.HOST_ID = ranked.HOST_ID and ranked.rnk = 1;


drop index new_index2_q3;
```

```
OPERATION                                     OBJECT_NAME      CARDINALITY      COST
SELECT STATEMENT                                                     26564                1062
  HASH JOIN                                                          26564                1062
    Access Predicates
      H.HOST_ID=RANKED.HOST_ID
    TABLE ACCESS (FULL)                       HOST                   26610                 204
    VIEW                                                             26564                 857
      Filter Predicates
        RANKED.RNK=1
      WINDOW (SORT PUSHED RANK)                                      26564                 857
        Filter Predicates
          RANK() OVER ( ORDER BY COUNT(*) DESC )<=1
        HASH (GROUP BY)                                              26564                 857
          TABLE ACCESS (FULL)                 LISTING                42036                 854
```

Improved plan:



```
create index new_index2_q3
on LISTING (HOST_ID);


select h.HOST_ID, h.HOST_NAME
from
(select HOST_ID, rank() over(order by nbr desc) as rnk
from
(select L.HOST_ID, count(*) as nbr
from LISTING l
group by l.HOST_ID)) ranked,
HOST h

where h.HOST_ID = ranked.HOST_ID and ranked.rnk = 1;


drop index new_index2_q3;
```

```
SQL | 0.02 seconds
OPERATION                                      OBJECT_NAME      CARDINALITY    COST
SELECT STATEMENT                                                      26564     237
  HASH JOIN                                                           26564     237
    Access Predicates
      H.HOST_ID=RANKED.HOST_ID
    TABLE ACCESS (FULL)                        HOST                   26610     204
    VIEW                                                              26564      33
      Filter Predicates
        RANKED.RNK=1
      WINDOW (SORT PUSHED RANK)                                       26564      33
        Filter Predicates
          RANK() OVER ( ORDER BY COUNT(*) DESC )<=1
        HASH (GROUP BY)                                               26564      33
          INDEX (FAST FULL SCAN)               NEW_INDEX2_Q3          42036      29
```

## *Query 2*

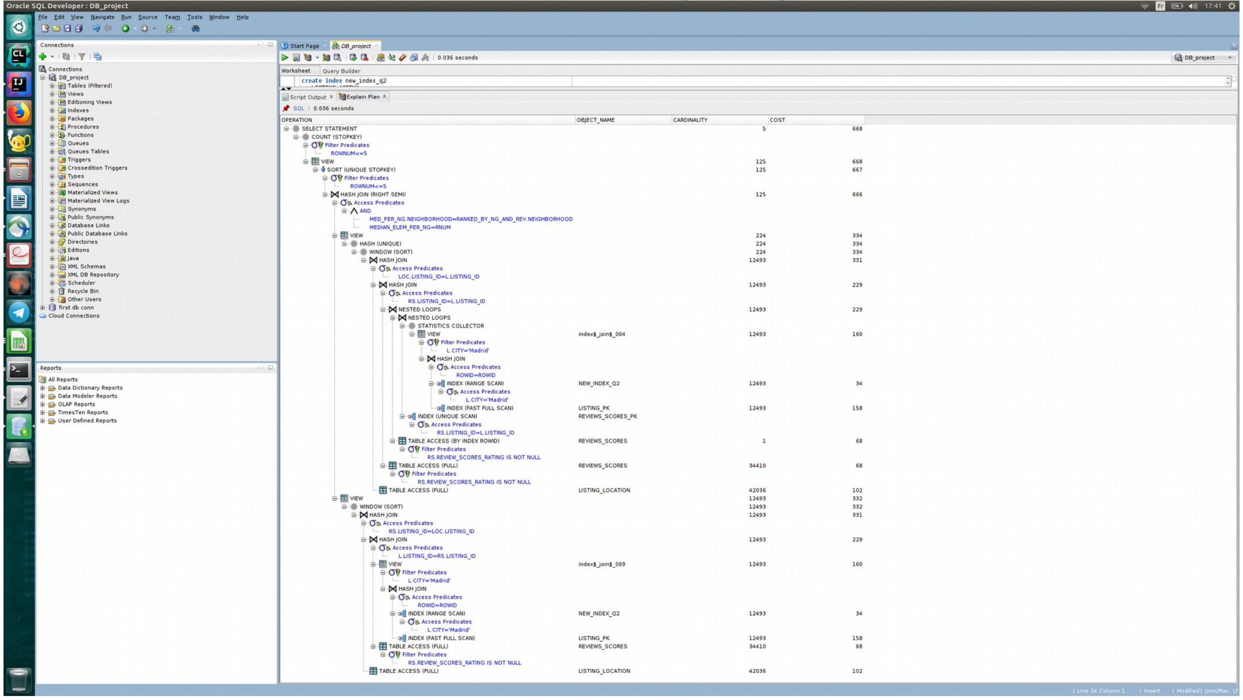
Initial Running time: 0.08 s

Optimized Running time: 0.065 s

Explain the improvement:

Putting an index on the city facilitates the access to the listing table.

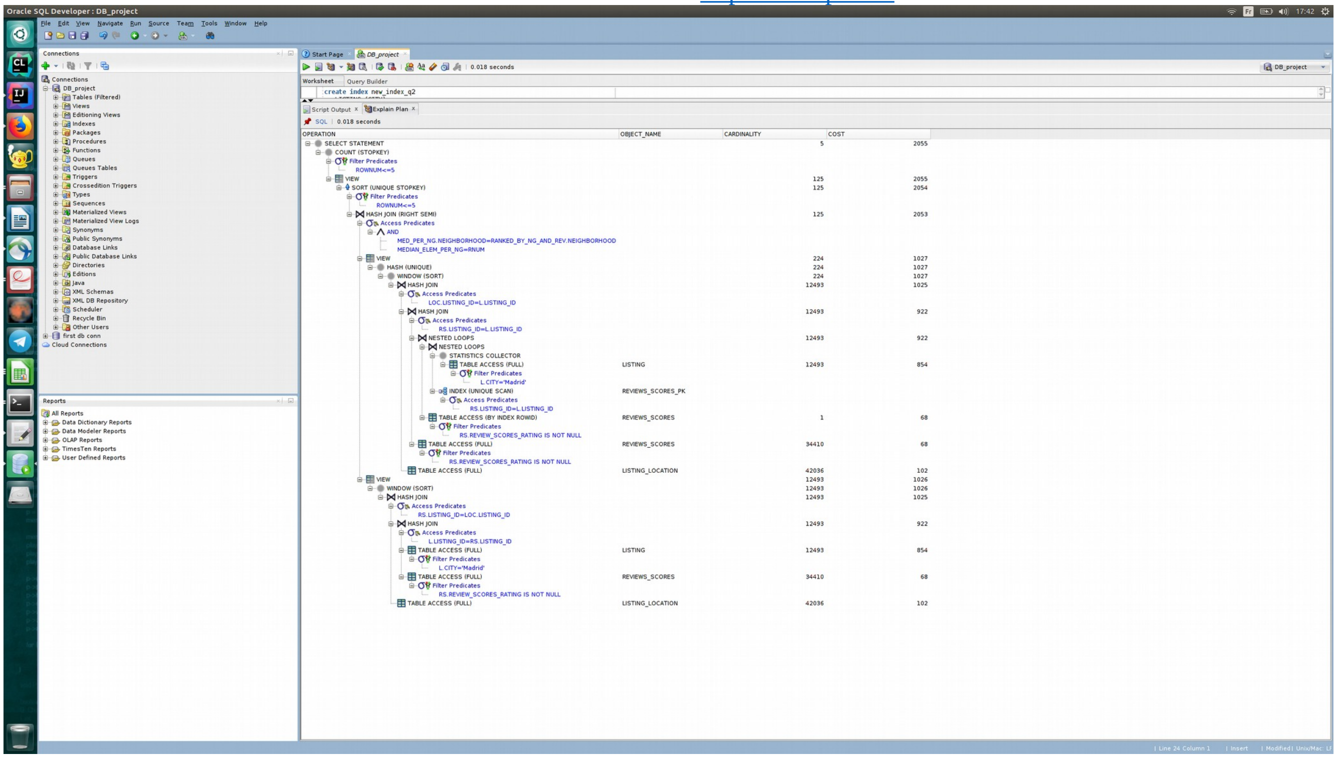
Initial plan:



```
create index new_index_q2
```

```
OPERATION                                                  OBJECT_NAME          CARDINALITY   COST
SELECT STATEMENT                                                                          5    668
  COUNT (STOPKEY)
    Filter Predicates
      ROWNUM<=5
    VIEW                                                                                125    668
      SORT (UNIQUE STOPKEY)                                                             125    667
        Filter Predicates
          ROWNUM<=5
        HASH JOIN (RIGHT SEMI)                                                          125    666
          Access Predicates
            AND
              MED_PER_NG.NEIGHBORHOOD=RANKED_BY_NG_AND_REV.NEIGHBORHOOD
              MEDIAN_ELEM_PER_NG=RNUM
          VIEW                                                                          224    334
            HASH (UNIQUE)                                                               224    334
              WINDOW (SORT)                                                             224    334
                HASH JOIN                                                             12493    331
                  Access Predicates
                    LOC.LISTING_ID=L.LISTING_ID
                  HASH JOIN                                                           12493    229
                    Access Predicates
                      RS.LISTING_ID=L.LISTING_ID
                    NESTED LOOPS                                                      12493    229
                      NESTED LOOPS
                        STATISTICS COLLECTOR
                          VIEW                                     index$_join$_004   12493    160
                            Filter Predicates
                              L.CITY='Madrid'
                            HASH JOIN
                              Access Predicates
                                ROWID=ROWID
                              INDEX (RANGE SCAN)                   NEW_INDEX_Q2       12493     34
                                Access Predicates
                                  L.CITY='Madrid'
                              INDEX (FAST FULL SCAN)               LISTING_PK         12493    158
                        INDEX (UNIQUE SCAN)                        REVIEWS_SCORES_PK
                          Access Predicates
                            RS.LISTING_ID=L.LISTING_ID
                      TABLE ACCESS (BY INDEX ROWID)                REVIEWS_SCORES         1     68
                        Filter Predicates
                          RS.REVIEW_SCORES_RATING IS NOT NULL
                    TABLE ACCESS (FULL)                            REVIEWS_SCORES     34410     68
                      Filter Predicates
                        RS.REVIEW_SCORES_RATING IS NOT NULL
                  TABLE ACCESS (FULL)                              LISTING_LOCATION   42036    102
          VIEW                                                                        12493    332
            WINDOW (SORT)                                                             12493    332
              HASH JOIN                                                               12493    331
                Access Predicates
                  RS.LISTING_ID=LOC.LISTING_ID
                HASH JOIN                                                             12493    229
                  Access Predicates
                    L.LISTING_ID=RS.LISTING_ID
                  VIEW                                             index$_join$_009   12493    160
                    Filter Predicates
                      L.CITY='Madrid'
                    HASH JOIN
                      Access Predicates
                        ROWID=ROWID
                      INDEX (RANGE SCAN)                           NEW_INDEX_Q2       12493     34
                        Access Predicates
                          L.CITY='Madrid'
                      INDEX (FAST FULL SCAN)                       LISTING_PK         12493    158
                  TABLE ACCESS (FULL)                              REVIEWS_SCORES     34410     68
                    Filter Predicates
                      RS.REVIEW_SCORES_RATING IS NOT NULL
                TABLE ACCESS (FULL)                                LISTING_LOCATION   42036    102
```

## Improved plan:



```
create index new_index_q2
```

```
0.018 seconds
OPERATION                                              OBJECT_NAME          CARDINALITY   COST
SELECT STATEMENT                                                                      5   2055
  COUNT (STOPKEY)
    Filter Predicates
      ROWNUM<=5
    VIEW                                                                            125   2055
      SORT (UNIQUE STOPKEY)                                                         125   2054
        Filter Predicates
          ROWNUM<=5
        HASH JOIN (RIGHT SEMI)                                                      125   2053
          Access Predicates
            AND
              MED_PER_NG.NEIGHBORHOOD=RANKED_BY_NG_AND_REV.NEIGHBORHOOD
              MEDIAN_ELEM_PER_NG=RNUM
          VIEW                                                                      224   1027
            HASH (UNIQUE)                                                           224   1027
              WINDOW (SORT)                                                         224   1027
                HASH JOIN                                                         12493   1025
                  Access Predicates
                    LOC.LISTING_ID=L.LISTING_ID
                  HASH JOIN                                                       12493    922
                    Access Predicates
                      RS.LISTING_ID=L.LISTING_ID
                    NESTED LOOPS                                                  12493    922
                      NESTED LOOPS
                        STATISTICS COLLECTOR
                          TABLE ACCESS (FULL)                  LISTING            12493    854
                            Filter Predicates
                              L.CITY='Madrid'
                        INDEX (UNIQUE SCAN)                    REVIEWS_SCORES_PK
                          Access Predicates
                            RS.LISTING_ID=L.LISTING_ID
                      TABLE ACCESS (BY INDEX ROWID)            REVIEWS_SCORES         1     68
                        Filter Predicates
                          RS.REVIEW_SCORES_RATING IS NOT NULL
                    TABLE ACCESS (FULL)                        REVIEWS_SCORES     34410     68
                      Filter Predicates
                        RS.REVIEW_SCORES_RATING IS NOT NULL
                  TABLE ACCESS (FULL)                          LISTING_LOCATION   42036    102
          VIEW                                                                    12493   1026
            WINDOW (SORT)                                                         12493   1026
              HASH JOIN                                                           12493   1025
                Access Predicates
                  RS.LISTING_ID=LOC.LISTING_ID
                HASH JOIN                                                         12493    922
                  Access Predicates
                    L.LISTING_ID=RS.LISTING_ID
                  TABLE ACCESS (FULL)                          LISTING            12493    854
                    Filter Predicates
                      L.CITY='Madrid'
                  TABLE ACCESS (FULL)                          REVIEWS_SCORES     34410     68
                    Filter Predicates
                      RS.REVIEW_SCORES_RATING IS NOT NULL
                TABLE ACCESS (FULL)                            LISTING_LOCATION   42036    102
```

# Interface

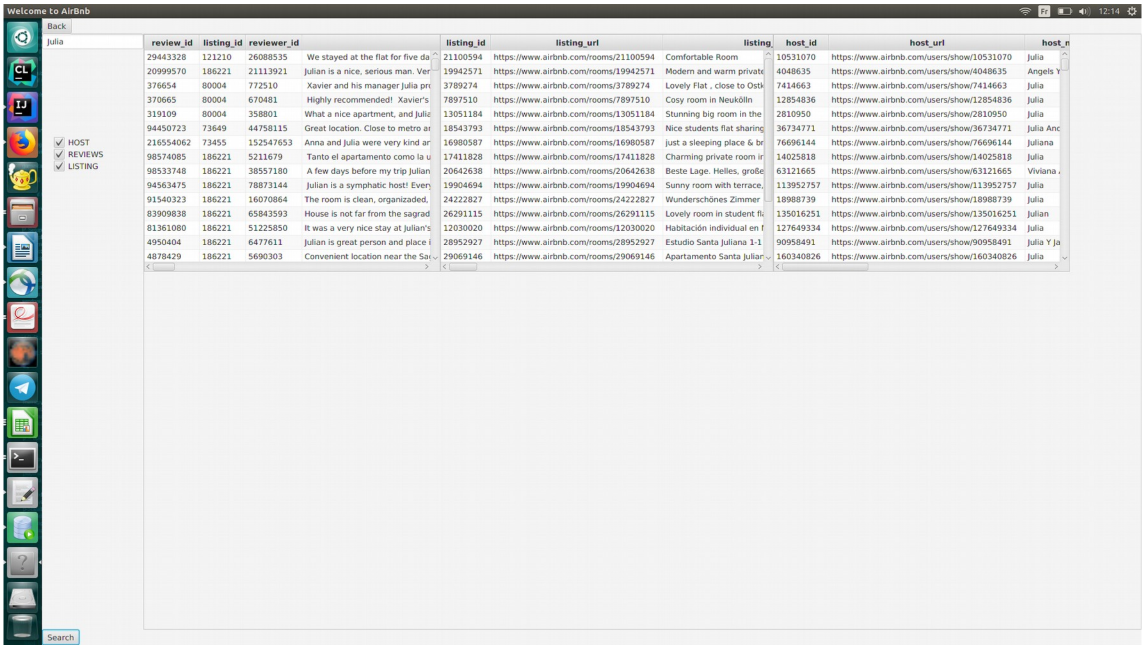
## Design logic Description

This part is similar to the one in milestone 2.

We used JavaFx and JDBC. The design is simple: a main page with buttons to switch to action-specific windows (insert/delete, search and predefined queries).

The code can be found here: https://github.com/hedi-sassi/rbnb_db_project/tree/master/UI

## Screenshots



Julia

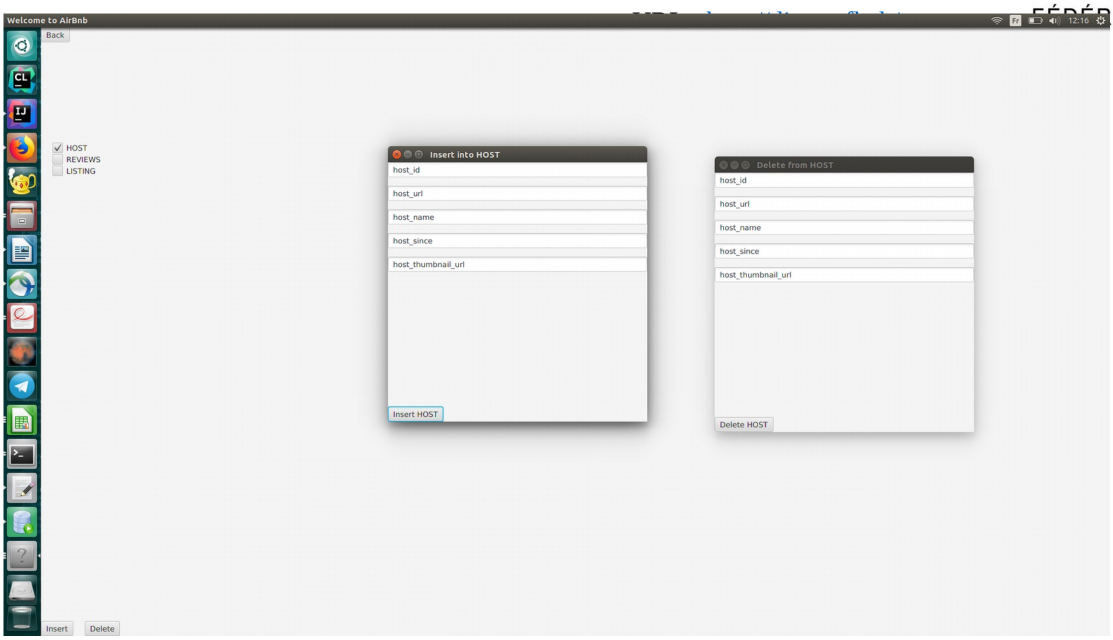
☑ HOST
☑ REVIEWS
☑ LISTING

| review_id | listing_id | reviewer_id | |
|---|---|---|---|
| 29443328 | 121210 | 26088535 | We stayed at the flat for five da |
| 20999570 | 186221 | 21113921 | Julian is a nice, serious man. Ver |
| 376654 | 80004 | 772510 | Xavier and his manager Julia pro |
| 370665 | 80004 | 670481 | Highly recommended! Xavier's |
| 319109 | 80004 | 358801 | What a nice apartment, and Julia |
| 94450723 | 73649 | 44758115 | Great location. Close to metro ar |
| 216554062 | 73455 | 152547653 | Anna and Julia were very kind ar |
| 98574085 | 186221 | 5211679 | Tanto el apartamento como la u |
| 98533748 | 186221 | 38557180 | A few days before my trip Julian |
| 94563475 | 186221 | 78873144 | Julian is a symphatic host! Every |
| 91540323 | 186221 | 16070864 | The room is clean, organizaded, |
| 83909838 | 186221 | 65843593 | House is not far from the sagrad |
| 81361080 | 186221 | 51225850 | It was a very nice stay at Julian's |
| 4950404 | 186221 | 6477611 | Julian is great person and place i |
| 4878429 | 186221 | 5690303 | Convenient location near the Sa |

| listing_id | listing_url | listing_ |
|---|---|---|
| 21100594 | https://www.airbnb.com/rooms/21100594 | Comfortable Room |
| 19942571 | https://www.airbnb.com/rooms/19942571 | Modern and warm private |
| 3789274 | https://www.airbnb.com/rooms/3789274 | Lovely Flat , close to Ostk |
| 7897510 | https://www.airbnb.com/rooms/7897510 | Cosy room in Neukölln |
| 13051184 | https://www.airbnb.com/rooms/13051184 | Stunning big room in the |
| 18543793 | https://www.airbnb.com/rooms/18543793 | Nice students flat sharing |
| 16980587 | https://www.airbnb.com/rooms/16980587 | just a sleeping place & br |
| 17411828 | https://www.airbnb.com/rooms/17411828 | Charming private room ir |
| 20642638 | https://www.airbnb.com/rooms/20642638 | Beste Lage. Helles, große |
| 19904694 | https://www.airbnb.com/rooms/19904694 | Sunny room with terrace, |
| 24222827 | https://www.airbnb.com/rooms/24222827 | Wunderschönes Zimmer |
| 26291115 | https://www.airbnb.com/rooms/26291115 | Lovely room in student fla |
| 12030020 | https://www.airbnb.com/rooms/12030020 | Habitación individual en I |
| 28952927 | https://www.airbnb.com/rooms/28952927 | Estudio Santa Juliana 1-1 |
| 29069146 | https://www.airbnb.com/rooms/29069146 | Apartamento Santa Julian |

| host_id | host_url | host_n |
|---|---|---|
| 10531070 | https://www.airbnb.com/users/show/10531070 | Julia |
| 4048635 | https://www.airbnb.com/users/show/4048635 | Angels Y |
| 7414663 | https://www.airbnb.com/users/show/7414663 | Julia |
| 12854836 | https://www.airbnb.com/users/show/12854836 | Julia |
| 2810950 | https://www.airbnb.com/users/show/2810950 | Julia |
| 36734771 | https://www.airbnb.com/users/show/36734771 | Julia Anc |
| 76696144 | https://www.airbnb.com/users/show/76696144 | Juliana |
| 14025818 | https://www.airbnb.com/users/show/14025818 | Julia |
| 63121665 | https://www.airbnb.com/users/show/63121665 | Viviana |
| 113952757 | https://www.airbnb.com/users/show/113952757 | Julia |
| 18988739 | https://www.airbnb.com/users/show/18988739 | Julia |
| 135016251 | https://www.airbnb.com/users/show/135016251 | Julian |
| 127649334 | https://www.airbnb.com/users/show/127649334 | Julia |
| 90958491 | https://www.airbnb.com/users/show/90958491 | Julia Y Ja |
| 160340826 | https://www.airbnb.com/users/show/160340826 | Julia |

Here we



This is the insertion/deletion window.

# General Comments

We split the work as follows:

Simon, Hédi & Camilla : writing queries and queries optimization

Hédi : Report and UI

Camilla: Correction of queries from milestone 2