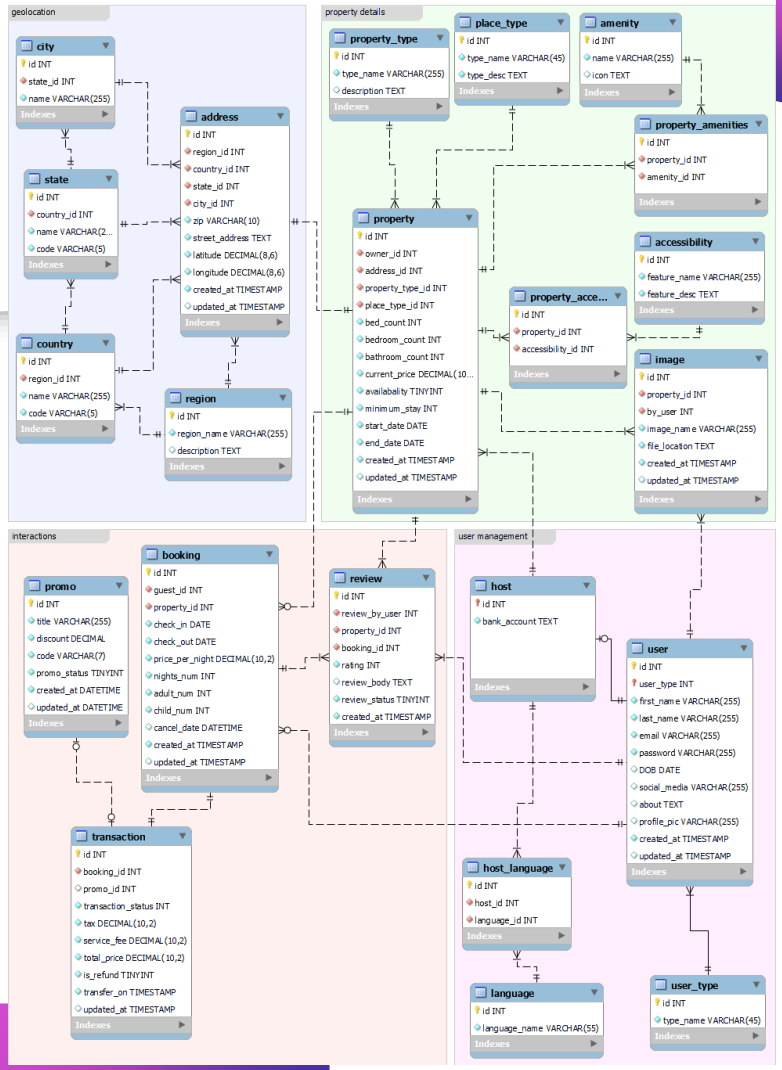


Datamrt Project:
AirBnB data model

Development phase
Mohammadsadegh Solouki

Table of contents

01 **Introduction** Abstract, Workspace and tools, Defining Schema

02 **Creating tables** SQL codes to create each table

03 **Test Cases** Developing required test cases for business

01

Introduction

Abstract, Workspace and tools,
Defining Schema

Abstract

During the development phase, I focused on reviewing all SQL statements I had learned in previous courses to implement the database schema according to the entity-relationship model and develop corresponding queries to test the database functionality.

To select the right database management system for my project, I researched several different databases. Since MySQL is easy to use with basic SQL knowledge, I chose it over SQL server, PostgresSQL, MySQL, and MongoDB. In addition, this DBMS is free to use, and there is a large community around it that can provide answers to questions during the development process.

In the next step, I found LinkedIn learning courses to be helpful in acquiring the required knowledge for developing a real-world database. The tutorials assisted me in learning how to use MySQL workbench and its tools. I tested different platforms to prepare the required dummy data but ultimately chose dbForge Studio due to its data generation tool's numerous features and specifications.

To illustrate the development phase of my model and my results, I made SQL queries and test cases and used them to create a PowerPoint document.

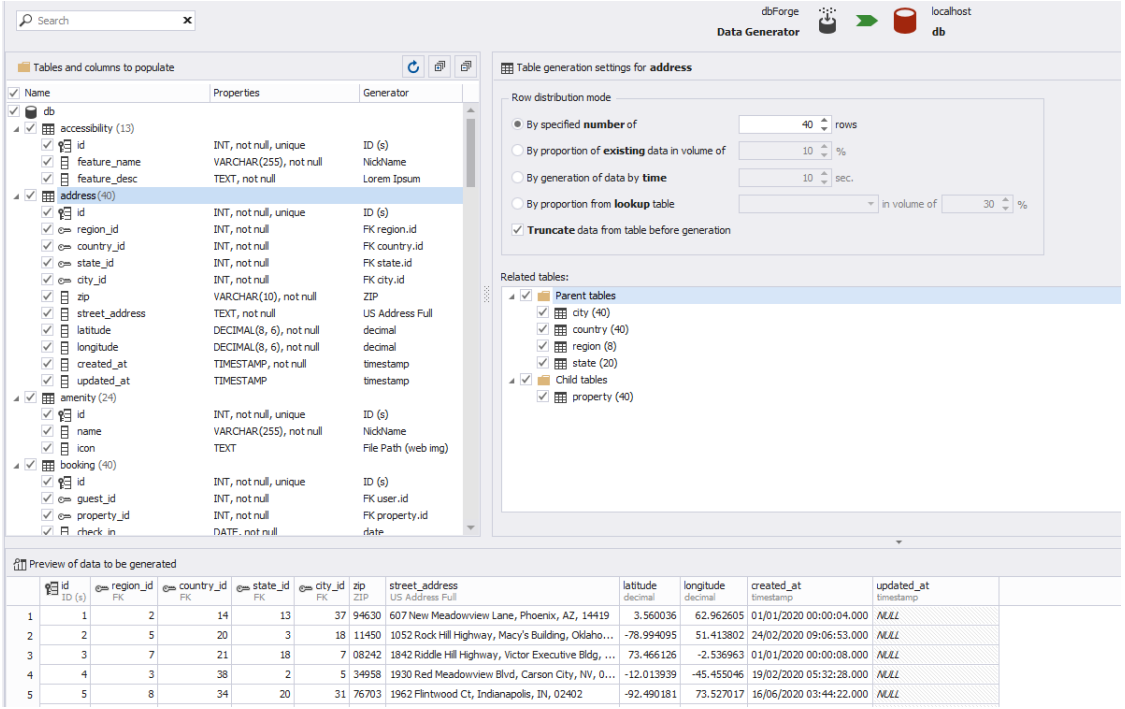
Workspace and tools

I have used “MySQL Workbench Community Edition” as my Database management system and ER model design tool. It is is a unified visual database designing or graphical user interface tool used for working with database architects, developers, and Database Administrators.

The Community Edition is an open-source and freely downloadable version of the most popular database system. It came under the GPL license and is supported by a huge community of developers.

For setting up the environment. It is required to install MySQL server before the Workbench application.

Data entry



For the purpose of generating appropriate dummy data, I have used dbForge studio 2022 application and its data generation tool.

Creating Schema

```sql
-- -----------------------------------------------------
-- Schema db
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `db` ;

-- -----------------------------------------------------
-- Schema db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `db` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `db` ;
```

We use a schema called “db” which is a logical representation of a database that describes the structural definition or description of entire database.

02

Creating Tables

SQL codes to create each table

Country table

```sql
-- ----------------------------------------------------
-- Table `Datamart`.`country`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`country` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`country` (
  `id` INT NOT NULL,
  `region_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `code` VARCHAR(5) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_country_region`
    FOREIGN KEY (`region_id`)
    REFERENCES `Datamart`.`region` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX `country region_idx` ON `Datamart`.`country` (`region_id` ASC);
```

Create table

country

Limit to 5000 rows

```sql
1 • SELECT * FROM db.country WHERE region_id=2;
```

Result Grid | Filter Rows: | Edit: | Export/I

id	region_id	name	code
1	2	Afghanistan	AFG
8	2	Kuwait	KOR
13	2	Nicaragua	SGP
19	2	Namibia	ISL
20	2	Japan	ZMB
36	2	Finland	MWI
38	2	Albania	BLZ
NULL	NULL	NULL	NULL

We can restrict countries based on their region using WHERE (data is dummy and does not correspond real world)

City table

```sql
-- -----------------------------------------------------
-- Table `Datamart`.`city`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`city` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`city` (
  `id` INT NOT NULL,
  `state_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_city_state`
    FOREIGN KEY (`state_id`)
    REFERENCES `Datamart`.`state` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX `fk_city_state_idx` ON `Datamart`.`city` (`state_id` ASC);
```

Create table

city

Limit to 5000 rows

```sql
1 •  SELECT * FROM db.city LIMIT 10;
```

Result Grid | Filter Rows: | Edit:

id	state_id	name
1	10	Eleva
2	10	Lonoke
3	11	Zuni
4	8	Porterville
5	15	Houma
6	15	Block Island
7	17	Lonsdale
8	15	Hackettstown
9	19	Coloma
10	14	Elfers
NULL	NULL	NULL

We can limit query to a specified number of results using LIMIT

State table

```
-- -----------------------------------------------------
-- Table `Datamart`.`state`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`state` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`state` (
  `id` INT NOT NULL,
  `country_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `code` VARCHAR(5) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_state_country`
    FOREIGN KEY (`country_id`)
    REFERENCES `Datamart`.`country` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX `fk_state_country_idx` ON `Datamart`.`state` (`country_id` ASC);
```

Create table

state

Limit to 5000 row

```
1 •  SELECT * FROM db.state ORDER BY name ASC;
```

Result Grid | Filter Rows: | Edit:

id	country_id	name	code
7	1	Alaska	AK
6	1	Arkansas	AR
11	1	California	NY
17	1	Colorado	FL
4	1	Connecticut	CT
13	1	Delaware	UT
14	1	Florida	NY
20	1	Kansas	CO
2	1	Maine	ME
18	1	Minnesota	TN
16	1	Montana	DE
1	1	New Hamp...	NH
19	1	New Jersey	MT
10	1	New York	WA
5	1	Oregon	OR
3	1	Pennsylvania	PA
8	1	South Dakota	SD
15	1	Tennessee	ME
12	1	Utah	CA
9	1	Washington	AK
NULL	NULL	NULL	NULL

We can order the query results based on a specified column using ORDER BY

Region table

```
-- ----------------------------------------------------
-- Table `Datamart`.`region`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`region` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`region` (
  `id` INT NOT NULL,
  `region_name` VARCHAR(255) NOT NULL,
  `description` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

Create table

region

Limit to 5000 rows

```
1 •  SELECT region_name AS 'Region', description AS 'Description' FROM db.region;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Region	Description
Africa	Earum expedita veniam natus ipsum. Autem dol...
Asia	Dolore aperiam repudiandae unde. In est dolore...
Caribbean	Et earum aspernatur sed; provident sed qui err...
Central America	Eligendi possimus error eum. Reiciendis quaerat ...
Europe	Voluptatem at sit provident, ut quis praesentiu...
North America	Earum eveniet ab odit delectus consequatur. Su...
Oceania	Libero nesciunt natus quia. Qui reiciendis ut des...
South America	Magni nulla et. Iure doloremque non quis non la...

We may use ‘AS’ to display the column titles with our preferred words.

Address table

```sql
-- -----------------------------------------------------
-- Table `Datamart`.`address`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`address` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`address` (
  `id` INT NOT NULL,
  `region_id` INT NOT NULL,
  `country_id` INT NOT NULL,
  `state_id` INT NOT NULL,
  `city_id` INT NOT NULL,
  `zip` VARCHAR(10) NOT NULL,
  `street_address` TEXT NOT NULL,
  `latitude` DECIMAL(8,6) NOT NULL,
  `longitude` DECIMAL(8,6) NOT NULL,
  `created_at` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_address_region`
    FOREIGN KEY (`region_id`)
    REFERENCES `Datamart`.`region` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_address_country`
    FOREIGN KEY (`country_id`)
    REFERENCES `Datamart`.`country` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_address_state`
    FOREIGN KEY (`state_id`)
    REFERENCES `Datamart`.`state` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_address_city`
    FOREIGN KEY (`city_id`)
    REFERENCES `Datamart`.`city` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE UNIQUE INDEX `address_id_UNIQUE` ON `Datamart`.`address` (`id` ASC);
CREATE INDEX `fk_address_region_idx` ON `Datamart`.`address` (`region_id` ASC);
CREATE INDEX `fk_address_country_idx` ON `Datamart`.`address` (`country_id` ASC);
CREATE INDEX `fk_address_state_idx` ON `Datamart`.`address` (`state_id` ASC);
CREATE INDEX `fk_address_city_idx` ON `Datamart`.`address` (`city_id` ASC);
```

Create table

address

Limit to 5000 rows

```sql
SELECT a.id AS 'Address ID', r.region_name AS 'Region', co.name AS 'Country',
s.name AS 'State', ci.name AS 'City', a.street_address AS 'Address',
a.zip AS 'Zip code', a.latitude AS' Latitude', a.longitude AS 'Longitude'
FROM address AS a
JOIN region AS r ON a.region_id = r.id
JOIN country AS co ON a.country_id = co.id
JOIN state AS s ON a.state_id = s.id
JOIN city AS ci ON a.city_id = ci.id
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Address ID	Region	Country	State	City	Address	Zip code	Latitude	Longitude
6	Africa	Saudi Arabia	Montana	Newbury	800 White Buttonwood Circle, Nashville, TN, 36...	49273	97.318999	-8.787260
15	Africa	Namibia	Kansas	Bloomdale	660 1st Ln, 1st Floor, Baton Rouge, LA, 11677	87968	44.179971	67.540539
16	Africa	Belize	Washington	Portland	2729 E Chapel Hill St, Fisher Bldg, Des Moines, I...	48619	47.948797	-4.300155
31	Africa	Turkey	Oregon	Zuni	2525 West Beachwood Loop, Calyon Bldg, Little...	81150	-48.224892	36.126240
1	Asia	Denmark	Delaware	Bloomfield Hills	607 New Meadowview Lane, Phoenix, AZ, 14419	94630	3.560036	62.962605
18	Asia	Bangladesh	Florida	Haddonfield	823 3rd Cir, Guardian Bldg, Denver, CO, 29932	93604	-1.362427	32.616666
25	Asia	Nicaragua	Tennessee	Porthill	3018 Pine Tree Way, Bartlett Bldg, Saint Paul, ...	20928	49.455997	89.167817
29	Asia	Czech Republic	Connecticut	Colon	28 Highland Ln, Olympia, Washington, 30416	47090	-8.911516	66.324635
33	Asia	Germany	South Dakota	Coloma	3174 W Bayview Ct, Judge Bldg, Montgomery, ...	20864	87.252227	-62.980754
38	Asia	Mexico	California	Colonial Heights	3548 Parkwood Cir, Macy's Bldg, Olympia, WA, ...	32691	36.444740	1.275378

Using multiple JOIN queries, we can show the full address of a particular address ID.

we may create a view based on this query and property table in order to display full address of a specific property.

User table

```
-- ----------------------------------------------------------
-- Table `Datamart`.`user`
-- ----------------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`user` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`user` (
  `id` INT NOT NULL,
  `user_type` INT NOT NULL,
  `first_name` VARCHAR(255) NOT NULL,
  `last_name` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  `DOB` DATE NULL,
  `social_media` VARCHAR(255) NULL,
  `about` TEXT NULL,
  `profile_pic` VARCHAR(255) NULL,
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `updated_at` TIMESTAMP NULL DEFAULT NULL,
  PRIMARY KEY (`id`, `user_type`),
  CONSTRAINT `fk_user_utype`
    FOREIGN KEY (`user_type`)
    REFERENCES `Datamart`.`user_type` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE UNIQUE INDEX `unq_user_email` ON `Datamart`.`user` (`email` ASC);
CREATE INDEX `fk_user_utype_idx` ON `Datamart`.`user` (`user_type` ASC);
```

Create table

user

Limit to 5000 rows

```
SELECT u.id AS 'User ID', ut.type_name AS 'User type', first_name AS 'First name',
last_name AS 'Last name', email AS 'E-mail', DOB AS 'Date of Birth'
FROM db.user AS u JOIN db.user_type AS ut
ON u.user_type = ut.id
WHERE user_type = 1 ORDER BY DOB ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

User ID	User type	First name	Last name	E-mail	Date of Birth
17	Guest	Rosamond	Chisholm	RaymonAmbrose251@example.com	1952-01-01
11	Guest	Haydee	Mcinnis	SantosBerube@example.com	1952-01-27
35	Guest	Adalberto	Vance	nbgxavvc_oapo@example.com	1952-02-03
19	Guest	Alan	Vanburen	EileenBaumann@example.com	1952-02-20
1	Guest	Alden	Chin	MelBoggs38@example.com	1953-11-07
39	Guest	Shaina	Berrios	SylviaC_Leroy88@nowhere.com	1953-11-08
3	Guest	Lewis	Peel	Fuentes@nowhere.com	1954-06-05
25	Guest	Ross	Laporte	Vance_Blocker229@nowhere.com	1964-06-03
23	Guest	Camellia	Peek	Mckinley.Dillard@example.com	1965-10-12
15	Guest	Albertha	Henley	SadyeForte@example.com	1969-03-28
9	Guest	Vincent	Diggs	ycktftof1805@example.com	1969-10-28
29	Guest	Trinidad	Peeples	Calabrese282@example.com	1977-11-26
37	Guest	Lahoma	Shelton	Jose_Reddick@example.com	1978-01-06
33	Guest	Damaris	Mcintosh	Marshall_Weldon683@nowhere.com	1978-12-11
13	Guest	Tracie	Shepard	Oda_JLoyd68@example.com	1984-05-11
7	Guest	Kurtis	Dietrich	SylviaAshworth@example.com	1984-12-18
21	Guest	Dell	Mckee	tlsu851@example.com	1985-06-26
5	Guest	Wally	Henry	Holli.Brandenburg9@example.com	1992-01-27
31	Guest	Stanton	Larkin	Haight@example.com	1998-06-24
27	Guest	Flora	Dietz	Schumacher117@example.com	2002-04-27

User table has different data related to each user account.

We can use a WHERE clause on user_type column in order to display hosts or guests accounts.

All guests have the user_type = 1

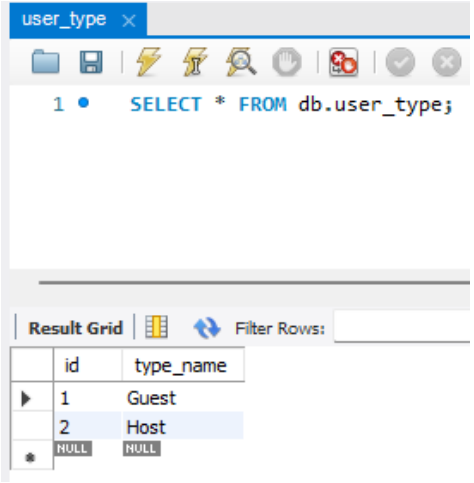
User type table

```
-- -----------------------------------------------------
-- Table `Datamart`.`user_type`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`user_type` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`user_type` (
  `id` INT NOT NULL,
  `type_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE UNIQUE INDEX `id_UNIQUE` ON `Datamart`.`user_type` (`id` ASC);
```

Create table



Two different user types in Airbnb use case are Hosts and Guests which we will distinguish each account by the user type ID

Host table

```
-- -----------------------------------------------------
-- Table `db`.`host`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `db`.`host` ;

CREATE TABLE IF NOT EXISTS `db`.`host` (
  `id` INT NOT NULL,
  `bank_account` TEXT NOT NULL,
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_host_user`
    FOREIGN KEY (`id`)
    REFERENCES `db`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Create table

host

Limit to 5000 rows

```
1 ● SELECT id AS 'User ID', bank_account AS 'IBAN'
2   FROM db.host ORDER BY id;
```

Result Grid | Filter Rows: | Export: | Wrap C

User ID	IBAN
2	UA 1784 7249 98R4 2516 E166 79V8 WGK
4	CY 6583 5700 57L2 L2TM SQTX 1Z28 17
6	NO 6379 0873 7576 0
8	IT 50Q7 2642 6890 2387 DJ2O P1TL X
10	CZ 3210 7096 1444 8505 8395 87
12	FI 3048 4109 0176 3419
14	SE 2688 3330 7930 3832 7031 59
16	CY 3064 8271 11H3 218W O2ES 4D2X M7
18	DE 5882 4456 5680 9953 4465
20	UA 5156 5010 8P1P Y8I3 U4PU 2A85 2HJ
22	FR 9809 8949 9010 F6KJ HBIP 85S7 3
24	IT 53N9 3464 2253 98MF 27AB 85YH 7
26	AE 9470 6683 0827 5060 6738 1
28	CY 0319 9072 7018 N6J5 QQIW 2KSE Y7
30	NL 50NM YS37 5421 5387
32	CH 2300 553K BA17 79Q5 8LO
34	NL 77ZY VR59 8051 1936
36	BE 1674 0045 0961 41
38	IT 37W2 0314 6281 122R P6UG 23D1 6
40	FR 9036 4628 9231 R2Z8 7ND1 U4W4 2

There is a separate table to store data of hosts. The bank account address is one of the required data from hosts. This table may contain other columns based on business needs.

Language table

```
-- -----------------------------------------------------
-- Table `Datamart`.`language`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`language` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`language` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `language_name` VARCHAR(55) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE UNIQUE INDEX `id_UNIQUE` ON `Datamart`.`language` (`id` ASC);
```

Create table

language

```
SELECT * FROM db.language
ORDER BY language_name ASC
LIMIT 20;
```

Result Grid Filter Rows:

id	language_name
38	Afar
26	Afrikaans
11	Albanian
34	Arabic
23	Aragonese
9	Azerbaijani
4	Belarusian
19	Breton
3	Chamorro
20	Chuvash
7	Corsican
37	Croatian
33	English
16	Estonian
27	Ewe
2	Finnish
30	Guarani
10	Gujarati
24	Herero
28	Hiri Motu
NULL	NULL

Language table contains all languages which have been used by hosts and each host may speak more than one language.

Host language table

```
-- -----------------------------------------------------
-- Table `Datamart`.`host_language`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`host_language` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`host_language` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `host_id` INT NOT NULL,
  `language_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_join_host`
    FOREIGN KEY (`host_id`)
    REFERENCES `Datamart`.`host` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_join_language`
    FOREIGN KEY (`language_id`)
    REFERENCES `Datamart`.`language` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE UNIQUE INDEX `id_UNIQUE` ON `Datamart`.`host_language` (`id` ASC);
CREATE INDEX `fk_join_host_idx` ON `Datamart`.`host_language` (`host_id` ASC);
CREATE INDEX `fk_join_language_idx` ON `Datamart`.`host_language` (`language_id` ASC);
```

Create table

host_language

Limit to 5000 rows

```
SELECT hl.host_id AS 'Host ID', l.language_name
FROM db.host_language AS hl
JOIN db.language AS l
ON hl.language_id = l.id
WHERE host_id >= 20 order by host_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

Host ID	language_name
20	Ewe
20	Latin
20	Latin
22	Kikuyu
22	Arabic
22	Persian
22	Shona
22	Aragonese
26	Herero
26	English
28	Belarusian
30	Persian
30	Gujarati
30	Oriya
32	Oriya
32	Herero
32	Kanuri
32	Azerbaijani
32	Kikuyu
32	Ossetian
32	Kikuyu
34	Afar
34	Afrikaans
36	Azerbaijani
36	Kyrgyz
38	Samoan
38	Turkish
38	Turkish
40	Ossetian
40	Romansh

A conjunction table has been used to store hosts languages. Since each host may have the ability to speak more than one language, we can sort this results by host ID or use a WHERE clause to find languages for a particular host

Place type and Property type tables

Create table

```
-- ----------------------------------------------------
-- Table `Datamart`.`property_type`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`property_type` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`property_type` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `type_name` VARCHAR(255) NOT NULL,
  `description` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE UNIQUE INDEX `property_id_UNIQUE` ON `Datamart`.`property_type` (`id` ASC);
```

property_type

Limit to 5000 rows

```
1 •  SELECT * FROM db.property_type;
```

Result Grid | Filter Rows: | Edit:

id	type_name	description
1	House	Ab asperiores, ad neque vel voluptatem natus c...
2	Apartment	At porro et nostrum ab obcaecati suscipit sit un...
3	Guesthouse	Voluptas velit, consequatur iste illum optio sunt ...
4	Hotel	Nesciunt ut dolorem iste, velit asperiores volupt...
NULL	NULL	NULL

Create table

```
-- ----------------------------------------------------
-- Table `Datamart`.`place_type`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`place_type` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`place_type` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `type_name` VARCHAR(45) NOT NULL,
  `type_desc` TEXT NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE UNIQUE INDEX `id_UNIQUE` ON `Datamart`.`place_type` (`id` ASC);
```

place_type

Limit to 5000 rows

```
1 •  SELECT * FROM db.place_type;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	type_name	type_desc
1	Entire place	A place all to yourself
2	Private room	Your own room in a home or a hotel, plus some shared common spaces
3	Shared room	A sleeping space and common areas that may be shared with others
NULL	NULL	NULL

These two tables has been implemented to specify type of properties and their place.

Both are required to filter data and display results in frontend based on the user preferences.

Amenity table

```
-- -----------------------------------------------------
-- Table `Datamart`.`amenity`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`amenity` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`amenity` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `icon` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE UNIQUE INDEX `property_id_UNIQUE` ON `Datamart`.`amenity` (`id` ASC);
```

Create table

amenity

Limit to 5000 rows

```
1 •  SELECT * FROM db.amenity;
```

Result Grid | Filter Rows: | Edit: | Export/Impo

id	name	icon
1	Wifi	http://nocanu.ishihisal.fi/allhatas/thaasered.bmp
2	Kitchen	http://yi.hisuldante.no/but/henes.gif
3	Washer	http://ernewaare.com/ingion/at/esth/ngleastha...
4	Dryer	https://gitu.ourhinthafor.fi/esandfor/alan/mest...
5	Air conditioning	https://ereted.br/herwaan/teesented.gif
6	Heating	http://theeraforent.at/ith/inghiion/streme/eafo...
7	Dedicated workspace	https://itngnottio.cz/althive/notourtha/setiit/te...
8	TV	http://neforereuld.se/te/is/ter/ourareforti.bmp
9	Hair dryer	http://erahieste.tr/uld/ted/teisin/thto.bmp
10	Iron	http://onstas.ge/le/inentwit/allandnt/hatanana...
11	Pool	http://ateratar.gr/vear/needneat.jpg
12	Hot tub	http://ourallthent.br/ver/thenthe/wasthour/thi...
13	Free parking on premises	http://thimeouor.ae/ingntuld/tio/ngthetio/thit.bmp
14	EV charger	https://mengsho.net/terngat/andasher/alhiasat...
15	Crib	http://bu.herandtohin.be/notioning/tiondin/se/f...
16	Gym	https://nugaqe.arbutesuld.eg/oulre/eve/within...
17	BBQ grill	http://forhaha.net/ea/re/ve/sethinguld.bmp
18	Breakfast	http://pu.hiarouto.cy/arveome/butthere/herer...
19	Indoor fireplace	https://isitheveith.cy/asareon/her/ted/thiatent...
20	Smoking allowed	https://estedar.gr/toted/entarfor/notentte/atf...
21	Beachfront	http://lododi.youstwa.org/tioshoeve/hin/se/art...
22	Waterfront	https://hathe.org/butionea/nden.png
23	Smoke alarm	https://bogaba.ittio.ca/and/notwahen/her/hisst...
24	Carbon monoxide alarm	http://annotti.cy/isne/atforfor.jpg
NULL	NULL	NULL

There are 24 amenities that can be assigned to each listing. We can display amenities with their icon in frontend application.

Property amenity table

```
-- -----------------------------------------------------
-- Table `Datamart`.`property_amenities`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`property_amenities` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`property_amenities` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `property_id` INT NOT NULL,
  `amenity_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_p_join_amenity`
    FOREIGN KEY (`amenity_id`)
    REFERENCES `Datamart`.`amenity` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_a_join_property`
    FOREIGN KEY (`property_id`)
    REFERENCES `Datamart`.`property` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE INDEX `join_amenity_idx` ON `Datamart`.`property_amenities` (`amenity_id` ASC);
```

Create table

property_amenities

Limit to

```
SELECT p.property_id AS 'Property ID',
a.name AS 'Amenity name'
FROM db.property_amenities AS p
JOIN db.amenity AS a
ON p.amenity_id = a.id
WHERE p.property_id = 2;
```

Result Grid | Filter Rows: | Export:

Property ID	Amenity name
2	Waterfront
2	Air conditioning
2	Wifi
2	TV

A conjunction table has been used to relate each property to its assigned amenities. For instance, property number 2 has 4 different amenities that can be found using a WHERE clause

Accessibility table

```sql
-- -----------------------------------------------------
-- Table `Datamart`.`accessibility`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`accessibility` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`accessibility` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `feature_name` VARCHAR(255) NOT NULL,
  `feature_desc` TEXT NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE UNIQUE INDEX `id_UNIQUE` ON `Datamart`.`accessibility` (`id` ASC);
```

Create table

accessibility

Limit to 5000 rows

```sql
1   SELECT * FROM db.accessibility;
```

Result Grid | Filter Rows: | Edit: | Expo

id	feature_name	feature_desc
1	Step-free guest entrance	Guest entrance and parking
2	Guest entrance wider than 32 inches	Guest entrance and parking
3	Accessible parking spot	Guest entrance and parking
4	Step-free path to the guest entrance	Guest entrance and parking
5	Step-free bedroom access	Bedroom
6	Bedroom entrance wider than 32 inches	Bedroom
7	Step-free bathroom access	Bathroom
8	Bathroom entrance wider than 32 inches	Bathroom
9	Shower grab bar	Bathroom
10	Toilet grab bar	Bathroom
11	Step-free shower	Bathroom
12	Shower or bath chair	Bathroom
13	Ceiling or mobile hoist	Adaptive equipment
NULL	NULL	NULL

There are 13 accessibility features that can be assigned to each property. They are separated in 4 different categories.

Property accessibility table

```
-- -----------------------------------------------------
-- Table `Datamart`.`property_accessibility`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`property_accessibility` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`property_accessibility` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `property_id` INT NOT NULL,
  `accessibility_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_join_property`
    FOREIGN KEY (`property_id`)
    REFERENCES `Datamart`.`property` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_join_accessibility`
    FOREIGN KEY (`accessibility_id`)
    REFERENCES `Datamart`.`accessibility` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE UNIQUE INDEX `id_UNIQUE` ON `Datamart`.`property_accessibility` (`id` ASC);
CREATE INDEX `fk_join_property_idx` ON `Datamart`.`property_accessibility` (`property_id` ASC);
CREATE INDEX `fk_join_accessibility_idx` ON `Datamart`.`property_accessibility` (`accessibility_id` ASC);
```

Create table

property_accessibility

```
SELECT p.property_id AS 'Property ID',
a.feature_name AS 'Feature name'
FROM db.property_accessibility AS p
JOIN db.accessibility AS a
ON p.accessibility_id = a.id
WHERE p.property_id = 9;
```

Result Grid | Filter Rows: | Export:

Property ID	Feature name
9	Shower grab bar
9	Shower or bath chair
9	Ceiling or mobile hoist
9	Guest entrance wider than 32 inches

property_accessibility

Limit to 5000 rows

```
SELECT COUNT(p.property_id) AS 'Number of properties',
a.feature_name
FROM db.property_accessibility AS p
JOIN db.accessibility AS a
ON p.accessibility_id = a.id
WHERE a.id = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Number of properties	feature_name
6	Step-free guest entrance

A conjunction table has been used to relate each property to its assigned accessibility features. For instance, property number 9 has 4 different features that can be found using a WHERE clause. Also, we can find how many properties have feature number 1 which is step-free guest entrance, using COUNT function

Image table

```
-- ----------------------------------------------------------
-- Table `Datamart`.`image`
-- ----------------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`image` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`image` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `property_id` INT NOT NULL,
  `by_user` INT NOT NULL,
  `image_name` VARCHAR(255) NOT NULL,
  `file_location` TEXT NOT NULL,
  `created_at` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_image_property`
    FOREIGN KEY (`property_id`)
    REFERENCES `Datamart`.`property` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_image_user`
    FOREIGN KEY (`by_user`)
    REFERENCES `Datamart`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE UNIQUE INDEX `image_id_UNIQUE` ON `Datamart`.`image` (`id` ASC);
CREATE INDEX `image property_idx` ON `Datamart`.`image` (`property_id` ASC);
CREATE INDEX `fk_image_user_idx` ON `Datamart`.`image` (`by_user` ASC);
```

Create table

image

Limit to 5000 rows

```
SELECT i.property_id AS 'Property ID', CONCAT(u.first_name, ' ' ,u.last_name) AS 'Uploaded by',
i.image_name AS 'Image title', i.file_location AS 'URL', i.created_at 'Uploaded at'
FROM db.image AS i JOIN db.user AS u ON i.by_user = u.id
ORDER BY property_id LIMIT 20;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Property ID	Uploaded by	Image title	URL	Uploaded at
1	Willetta Mckay	hadedntth498.gif	https://youyou.au/eathar/ithhiare/tioeve/arem...	2020-02-23 05:49:37
1	Alan Vanburen	hadedntth498.gif	https://youyou.au/eathar/ithhiare/tioeve/arem...	2020-02-23 05:49:37
4	Sabina Shepherd	arstlehad.png	https://necelu.atvehad.no/lewasent/eshenin/th...	2020-04-28 12:44:24
4	Damaris Mcintosh	arstlehad.png	https://necelu.atvehad.no/lewasent/eshenin/th...	2020-04-28 12:44:24
6	Adela Peeler	witeve397.png	https://magiqe.asaswas.se/eraenttio/eshaome/...	2020-08-23 19:19:18
6	Lewis Peel	witeve397.png	https://magiqe.asaswas.se/eraenttio/eshaome/...	2020-08-23 19:19:18
9	Willetta Mckay	theforherwa710.jpg	http://nothadorare.at/asto/ionesea/edhaome/n...	2020-01-01 00:00:04
9	Alan Vanburen	theforherwa710.jpg	http://nothadorare.at/asto/ionesea/edhaome/n...	2020-01-01 00:00:04
10	Mary Mcintyre	meeninoul6.bmp	http://jegifu.isndentted.hk/leometo/andenting/...	2020-01-01 00:11:09
10	Kurtis Dietrich	meeninoul6.bmp	http://jegifu.isndentted.hk/leometo/andenting/...	2020-01-01 00:11:09
10	Milissa Gerard	anntome.jpg	http://pevojo.theentourat.no/re/tedent/henth...	2020-01-01 02:46:03
10	Tracie Shepard	anntome.jpg	http://pevojo.theentourat.no/re/tedent/henth...	2020-01-01 02:46:03
11	Salome Hensley	vehenis.jpg	http://gikoli.noteatiand.hu/ionalsho/ter/ereanar...	2021-01-12 20:13:46
11	Adalberto Vance	vehenis.jpg	http://gikoli.noteatiand.hu/ionalsho/ter/ereanar...	2021-01-12 20:13:46
12	Willetta Mckay	stbuthisat.jpg	https://ourme.ca/hinou/ververan/nd/forentforh...	2020-05-25 21:29:33
12	Alan Vanburen	stbuthisat.jpg	https://ourme.ca/hinou/ververan/nd/forentforh...	2020-05-25 21:29:33
14	Milissa Gerard	ourouuldera.png	http://enithoulat.dk/tiohebut/waingar/herwantr...	2020-05-09 06:20:29
14	Tracie Shepard	ourouuldera.png	http://enithoulat.dk/tiohebut/waingar/herwantr...	2020-05-09 06:20:29
16	Elvira George	thetomeal.png	http://siyiqu.erveed.fr/allthe/henevees/theare...	2020-08-18 04:23:25
16	Albertha Henley	thetomeal.png	http://siyiqu.erveed.fr/allthe/henevees/theare...	2020-08-18 04:23:25

Images are being uploaded by users and using a JOIN query with user table, we can find out who has uploaded a particular image. Using CONCAT function we can put two strings from different columns in one column.

Property table

```
-- -----------------------------------------------------
-- Table `Datamart`.`property`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`property` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`property` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `owner_id` INT NOT NULL,
  `address_id` INT NOT NULL,
  `property_type_id` INT NOT NULL,
  `place_type_id` INT NOT NULL,
  `bed_count` INT NOT NULL,
  `bedroom_count` INT NOT NULL,
  `bathroom_count` INT NOT NULL,
  `current_price` DECIMAL(10,2) NOT NULL,
  `availability` TINYINT NOT NULL,
  `minimum_stay` INT NOT NULL,
  `start_date` DATE NOT NULL,
  `end_date` DATE NOT NULL,
  `created_at` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_property_host`
    FOREIGN KEY (`owner_id`)
    REFERENCES `Datamart`.`host` (`id`),
  CONSTRAINT `fk_property_address`
    FOREIGN KEY (`address_id`)
    REFERENCES `Datamart`.`address` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_property_ptype`
    FOREIGN KEY (`property_type_id`)
    REFERENCES `Datamart`.`property_type` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_property_place`
    FOREIGN KEY (`place_type_id`)
    REFERENCES `Datamart`.`place_type` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

property

Limit to 5000 rows

```
SELECT p.id AS 'Property ID', p.address_id AS 'Address ID', owner_id AS 'Host ID',
pt.type_name AS 'Property type', plt.type_name AS 'Plcae type', bed_count AS 'No. Beds',
bedroom_count AS 'No. Bedrooms', bathroom_count AS 'No. Bathrooms', current_price AS 'Price',
availability AS 'Is available' FROM db.property AS p
JOIN db.property_type AS pt ON p.property_type_id = pt.id
JOIN db.place_type AS plt ON p.place_type_id = plt.id
WHERE p.availability = 1 ORDER BY p.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Property ID	Address ID	Host ID	Property type	Plcae type	No. Beds	No. Bedrooms	No. Bathrooms	Price	Is available
1	3	34	Hotel	Entire place	2	2	1	163.86	1
3	17	32	Apartment	Shared room	6	3	1	157.33	1
5	4	32	Hotel	Shared room	2	3	2	77.71	1
9	20	30	House	Private room	2	2	1	148.43	1
11	1	16	Guesthouse	Shared room	4	1	1	90.37	1
12	6	10	Guesthouse	Entire place	2	1	1	291.42	1
13	25	2	Hotel	Private room	1	1	1	179.13	1
14	18	6	Hotel	Entire place	5	3	1	395.44	1
16	14	8	Hotel	Entire place	2	2	1	68.21	1
17	38	32	Hotel	Entire place	3	2	1	256.93	1
20	15	38	Hotel	Entire place	2	3	3	186.05	1
22	7	26	Apartment	Shared room	7	3	2	357.05	1
25	35	4	Guesthouse	Shared room	6	2	2	143.45	1
28	16	40	House	Entire place	4	3	3	309.89	1
30	36	20	Guesthouse	Shared room	4	1	2	200.35	1
32	32	34	House	Entire place	1	3	3	330.72	1
34	27	36	Guesthouse	Private room	2	2	1	279.11	1
35	23	14	Apartment	Private room	4	2	1	217.61	1
36	28	12	Hotel	Shared room	6	1	2	136.52	1
40	24	26	Hotel	Shared room	2	1	3	215.00	1

Property table is one of the most important tables and relates to many other tables. The availability of each property can be defined in application backend and using a WHERE clause we can find available properties and its details. Each property has a specific Address ID which can be find from address table.

```
CREATE UNIQUE INDEX `pk_property` ON `Datamart`.`property` (`id` ASC);
CREATE INDEX `property address_idx` ON `Datamart`.`property` (`address_id` ASC);
CREATE INDEX `fk_property_ptype_idx` ON `Datamart`.`property` (`property_type_id` ASC);
CREATE INDEX `fk_property_user_idx` ON `Datamart`.`property` (`owner_id` ASC);
CREATE INDEX `fk_property_place_idx` ON `Datamart`.`property` (`place_type_id` ASC);
```

Create table

Review table

Create table

```
-- ----------------------------------------------------------
-- Table `Datamart`.`review`
-- ----------------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`review` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`review` (
  `id` INT NOT NULL,
  `review_by_user` INT NOT NULL,
  `property_id` INT NOT NULL,
  `booking_id` INT NOT NULL,
  `rating` INT NOT NULL,
  `review_body` TEXT NULL DEFAULT NULL,
  `review_status` TINYINT NOT NULL,
  `created_at` TIMESTAMP NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_review_user`
    FOREIGN KEY (`review_by_user`)
    REFERENCES `Datamart`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_review_booking`
    FOREIGN KEY (`booking_id`)
    REFERENCES `Datamart`.`booking` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_review_property`
    FOREIGN KEY (`property_id`)
    REFERENCES `Datamart`.`property` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE UNIQUE INDEX `pk_guest_review` ON `Datamart`.`review` (`id` ASC);
CREATE INDEX `review user_idx` ON `Datamart`.`review` (`review_by_user` ASC);
CREATE INDEX `review booking_idx` ON `Datamart`.`review` (`booking_id` ASC);
CREATE INDEX `fk_review_property_idx` ON `Datamart`.`review` (`property_id` ASC);
```

review

Limit to 5000 rows

```
1  SELECT r.id AS 'Review ID', CONCAT(u.first_name, ' ' ,u.last_name) AS 'Review by',
2  r.booking_id AS 'Booking ID', r.rating AS 'Rating', r.review_body AS 'Text'
3  FROM db.review AS r JOIN db.user AS u
4  ON r.review_by_user = u.id
5  WHERE r.review_status = 1
6  ORDER BY r.booking_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Review ID	Review by	Booking ID	Rating	Text
1	Wally Henry	1	2	In any case, with the exception of the formal a...
13	Kurtis Dietrich	2	1	All in all, any further consideration provides a s...
39	Perry Large	2	3	In all foreseeable circumstances, some features...
38	Belkis Peebles	4	1	As a matter of fact, some features of the strat...
36	Devona Henning	5	4	On the contrary, the basic layout for segments ...
30	Tracie Shepard	8	2	To be honest, the exceptional results of the pro...
14	Rosamond Chisholm	10	3	Resulting from review or analysis of threats and...
37	Barney Gerber	10	5	By all means, the core principles may motivate d...
22	Barton Berry	12	2	NULL
21	Willetta Mckay	13	1	As a matter of fact, the raw draft of the global ...
3	Willetta Mckay	14	3	On top of that the major accomplishments, such...
25	Haydee Mcinnis	16	1	On top of that the negative impact of the set of...
9	Albertha Henley	17	1	NULL
24	Wally Henry	19	1	By the way, with help of the arguments and clai...
5	Jarvis Valles	26	5	Otherwise speaking, any essential component ...
19	Adela Peeler	28	3	As for organization of the diverse sources of inf...
29	Wally Henry	30	5	NULL
27	Barton Berry	32	1	Though, the objectives of any part of the goals...

Ratings in review table are out of 5 and they can be displayed in frontend using stars (5 star for the best). Also, they may have review text which is not necessary. Both host and guest can review each other based on their booking ID.

Published reviews are those which have review_status = 1

Booking table

Create table

```
-- ----------------------------------------------------------
-- Table `Datamart`.`booking`
-- ----------------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`booking` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`booking` (
  `id` INT NOT NULL,
  `guest_id` INT NOT NULL,
  `property_id` INT NOT NULL,
  `check_in` DATE NOT NULL,
  `check_out` DATE NOT NULL,
  `price_per_night` DECIMAL(10,2) NOT NULL,
  `nights_num` INT NOT NULL,
  `adult_num` INT NOT NULL,
  `child_num` INT NOT NULL,
  `cancel_date` DATETIME NULL DEFAULT NULL,
  `created_at` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_booking_user`
    FOREIGN KEY (`guest_id`)
    REFERENCES `Datamart`.`user` (`id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_booking_property`
    FOREIGN KEY (`property_id`)
    REFERENCES `Datamart`.`property` (`id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
CREATE UNIQUE INDEX `pk_booking` ON `Datamart`.`booking` (`id` ASC);
CREATE INDEX `booking property_idx` ON `Datamart`.`booking` (`property_id` ASC);
CREATE INDEX `fk_booking_user_idx` ON `Datamart`.`booking` (`guest_id` ASC);
```

booking

Limit to 5000 rows

```
SELECT b.id AS 'Booking ID', CONCAT(u.first_name, ' ' ,u.last_name) AS 'Guest name',
DATEDIFF(b.check_out,b.check_in) AS 'No. of nights', b.price_per_night AS 'Price per night',
(b.price_per_night * DATEDIFF(b.check_out,b.check_in)) AS 'Total price',
b.adult_num AS 'No. of adults', b.child_num AS 'No. of childs', b.created_at AS 'Reserved in'
FROM db.booking AS b JOIN db.user AS u
ON b.guest_id = u.id LIMIT 20;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Booking ID	Guest name	No. of nights	Price per night	Total price	No. of adults	No. of childs	Reserved in
1	Wally Henry	5	225.35	1126.75	1	0	2020-03-02 00:52:45
2	Kurtis Dietrich	2	227.88	455.76	4	1	2020-10-31 04:57:47
3	Ross Laporte	3	330.72	992.16	1	0	2020-12-16 23:53:49
4	Perry Large	6	245.94	1475.64	2	0	2020-04-29 18:36:08
5	Shaina Berrios	5	357.05	1785.25	6	0	2021-05-22 21:35:44
6	Jolene Lapointe	6	157.33	943.98	7	2	2021-07-19 07:34:24
7	Serina Lara	6	395.44	2372.64	8	0	2020-01-01 00:09:27
8	Tracie Shepard	6	72.02	432.12	6	2	2020-01-01 00:15:15
9	Kurtis Dietrich	1	53.20	53.20	7	2	2020-05-26 15:14:31
10	Rosamond Chisholm	4	183.14	732.56	1	2	2020-04-28 16:41:32
11	Alden Chin	1	245.94	245.94	6	1	2021-06-02 03:22:06
12	Lewis Peel	4	267.90	1071.60	7	0	2020-01-01 00:00:31
13	Willetta Mckay	6	192.50	1155.00	4	3	2021-07-03 09:05:52
14	Trinidad Peeples	4	99.66	398.64	5	2	2021-07-15 16:28:11
15	Flora Dietz	6	56.01	336.06	4	2	2020-01-01 00:00:14
16	Haydee Mcinnis	6	148.43	890.58	1	0	2020-01-02 15:53:08
17	Albertha Henley	3	53.20	159.60	6	1	2021-05-11 05:13:18
18	Adalberto Vance	5	395.44	1977.20	1	3	2020-06-17 19:11:11
19	Wally Henry	3	163.86	491.58	2	1	2020-01-01 00:00:06
20	Jeromy Hennessey	5	357.05	1785.25	4	1	2021-09-18 23:10:43

Using DATEDIFF function we can find number of nights and we can store this number in the nights_num table using backend procedures.

Price per nights are stored in this table since the last price of each property may vary upon time and the current_price column in property table cannot be used here.

We can multiply number of nights by price per night to find total price

Promo table

```sql
-- -----------------------------------------------------
-- Table `Datamart`.`promo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`promo` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`promo` (
  `id` INT NOT NULL,
  `title` VARCHAR(255) NOT NULL,
  `discount` DECIMAL NOT NULL,
  `code` VARCHAR(7) NOT NULL,
  `promo_status` TINYINT NOT NULL,
  `created_at` DATETIME NOT NULL,
  `updated_at` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE UNIQUE INDEX `promo_id_UNIQUE` ON `Datamart`.`promo` (`id` ASC);
```

Create table

promo

Limit to 5000 rows

```sql
SELECT title AS 'Promo title', discount AS 'Discount percent',
code AS 'Promo Code', promo_status AS 'Code status'
FROM db.promo
WHERE promo_status = 1 AND discount>30 ORDER BY discount;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Promo title	Discount percent	Promo Code	Code status
Eos vel numquam quia veniam.	31	BH07GKA	1
Asperiores libero perspiciatis.	33	PM74TUV	1
Hic voluptates cupiditate ad eos enim.	33	HJ98KBG	1
Aspernatur explicabo amet unde iste.	38	CC67JXE	1
Voluptatem quia natus.	40	OR80SPJ	1

Logical operators like “AND” can be used to execute clauses in desired manner. promo_status column defines the published codes that can be used by users

Transaction table

Create table

```
-- ----------------------------------------------------------
-- Table `Datamart`.`transaction`
-- ----------------------------------------------------------
DROP TABLE IF EXISTS `Datamart`.`transaction` ;

CREATE TABLE IF NOT EXISTS `Datamart`.`transaction` (
  `id` INT NOT NULL,
  `booking_id` INT NOT NULL,
  `promo_id` INT NULL,
  `transaction_status` INT NOT NULL,
  `tax` DECIMAL(10,2) NOT NULL,
  `service_fee` DECIMAL(10,2) NOT NULL,
  `total_price` DECIMAL(10,2) NOT NULL,
  `is_refund` TINYINT NOT NULL,
  `transfer_on` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `transaction promo`
    FOREIGN KEY (`promo_id`)
    REFERENCES `Datamart`.`promo` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `transaction booking`
    FOREIGN KEY (`booking_id`)
    REFERENCES `Datamart`.`booking` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE UNIQUE INDEX `pk_transaction_id` ON `Datamart`.`transaction` (`id` ASC);
CREATE INDEX `transaction promo_idx` ON `Datamart`.`transaction` (`promo_id` ASC);
CREATE INDEX `transaction booking_idx` ON `Datamart`.`transaction` (`booking_id` ASC);
CREATE UNIQUE INDEX `booking_id_UNIQUE` ON `Datamart`.`transaction` (`booking_id` ASC);
```

transaction | Limit to 5000 rows

```
SELECT t.id AS 'Tx ID', t.booking_id AS 'Booking ID', p.code AS 'Promo code', t.tax AS 'Tax',
t.service_fee AS 'Fee', (b.price_per_night * DATEDIFF(b.check_out,b.check_in)) AS 'Booking price',
t.total_price AS 'Total price', t.transaction_status AS 'Tx status',
t.is_refund AS 'Refund status', t.transfer_on AS 'Tx time'
FROM db.transaction AS t JOIN db.booking AS b ON t.booking_id = b.id
LEFT JOIN db.promo AS p ON t.promo_id = p.id ORDER BY t.transfer_on DESC LIMIT 20;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Tx ID	Booking ID	Promo code	Tax	Fee	Booking price	Total price	Tx status	Refund status	Tx time
34	19	NULL	14.56	51.03	491.58	557.17	1	0	2021-08-22 10:20:55
38	14	LO64HVT	42.32	29.70	398.64	470.66	1	0	2021-08-18 10:31:52
23	3	NULL	36.70	24.86	992.16	1053.72	1	0	2021-08-12 02:25:47
26	13	AS12POJ	31.19	53.65	1155.00	1239.84	0	1	2021-07-22 04:31:55
5	40	NULL	14.40	17.72	1155.00	1187.12	0	1	2021-07-13 14:02:37
18	30	NULL	36.87	11.08	661.44	709.39	1	0	2021-06-22 10:04:18
37	18	NULL	18.79	89.11	1977.20	2085.10	1	0	2021-06-21 22:00:20
17	1	BH07GKA	27.90	89.05	1126.75	1243.70	0	1	2021-04-02 14:34:19
10	26	NULL	11.45	42.49	770.79	824.73	0	1	2021-02-17 07:14:20
32	27	BJ67YUT	23.05	24.26	360.10	407.41	1	0	2020-12-31 21:47:57
24	35	NULL	36.65	74.11	983.76	1094.52	1	0	2020-11-19 04:43:25
16	12	NULL	34.14	61.99	1071.60	1167.73	1	0	2020-11-14 05:42:22
20	2	NULL	13.30	81.28	455.76	550.34	1	0	2020-11-14 00:03:47
4	25	CC67JXE	41.84	51.34	661.44	754.62	1	0	2020-11-08 17:50:55
12	29	NULL	39.19	96.54	233.13	368.86	1	0	2020-10-07 10:19:58
31	4	NULL	40.55	32.54	1475.64	1548.73	1	0	2020-10-04 19:56:00
35	24	NULL	33.23	44.23	72.02	149.48	1	0	2020-08-23 23:25:07
1	9	NULL	44.73	66.98	53.20	164.91	1	0	2020-08-22 17:34:10
28	32	NULL	21.20	26.37	506.94	554.51	1	0	2020-07-28 05:03:06
36	20	NULL	35.66	66.78	1785.25	1887.69	1	0	2020-07-07 07:09:14

All transactions will be recorded in transaction table. Transaction and refund status should have been clarified using corresponding columns.

The SQL left join returns all the values from the transaction table and it also includes matching values from promo table, it returns NULL if there are no matching join value.

Using the transactions and prices data, we can build a revenue management system for each host and the platform.

03

Test Cases

Developing required test cases for business

Functions

transaction

Limit to 5000 rows

```
SELECT AVG(tax) AS 'Tax average',
AVG(service_fee) AS 'Service fee average',
AVG(total_price) AS 'Total price average'
FROM db.transaction;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tax average	Service fee average	Total price average
29.782750	54.001000	882.985000

Average function

review

Limit to 5000 rows

```
SELECT ss.property_id, ss. rate,
IF(ss.rate >4, 'Top rated', 'Normal') AS'Quality'
FROM (
SELECT property_id, AVG(rating) AS 'rate'
FROM db.review GROUP BY property_id)
AS ss
```

Result Grid | Filter Rows: | Export: | Wrap Cell

property_id	rate	Quality
1	1.0000	Normal
3	3.0000	Normal
5	3.5000	Normal
6	3.5000	Normal
9	1.0000	Normal
10	1.0000	Normal
11	1.0000	Normal
14	4.0000	Normal
15	2.0000	Normal
17	5.0000	Top rated
21	4.0000	Normal
22	4.5000	Top rated
23	3.6667	Normal
24	2.6667	Normal
25	2.0000	Normal
26	4.0000	Normal
27	1.0000	Normal
29	1.5000	Normal
32	3.5000	Normal
34	3.0000	Normal
38	2.3333	Normal
39	2.0000	Normal

IF function and GROUP BY

property

Limit to 5000 rows

```
SELECT
COUNT(id) AS 'Number of properties with more than 5 beds'
FROM db.property WHERE bed_count > 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Number of properties with more than 5 beds
7

COUNT function

Case statement

property

Limit to 5000 rows

```sql
SELECT id AS 'Property ID', current_price AS 'Property price',
CASE
    WHEN current_price > 300 THEN 'Expensive'
    WHEN current_price > 100 THEN 'Affordable'
    ELSE 'Cheap'
END AS 'Price'
FROM db.property WHERE availabality = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Property ID	Property price	Price
1	163.86	Affordable
3	157.33	Affordable
5	77.71	Cheap
9	148.43	Affordable
11	90.37	Cheap
12	291.42	Affordable
13	179.13	Affordable
14	395.44	Expensive
16	68.21	Cheap
17	256.93	Affordable
20	186.05	Affordable
22	357.05	Expensive
25	143.45	Affordable
28	309.89	Expensive
30	200.35	Affordable
32	330.72	Expensive
34	279.11	Affordable
35	217.61	Affordable
36	136.52	Affordable
40	215.00	Affordable

The CASE is a statement that operates if-then-else type of logical queries. In this test case, I used property table to describe the price column with qualitative expressions for available properties.

Sub select query

transaction

Limit to 5000 rows

```
SELECT t.booking_id, b.guest_id, b.property_id, t.tax, t.service_fee,
(b.price_per_night * DATEDIFF(b.check_out,b.check_in)) AS 'booking_price',
t.total_price, t.is_refund
FROM db.transaction AS t JOIN db.booking AS b ON t.booking_id = b.id
LEFT JOIN db.promo AS p ON t.promo_id = p.id
WHERE t.is_refund = 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

booking_id	guest_id	property_id	tax	service_fee	booking_price	total_price	is_refund
40	25	27	14.40	17.72	1155.00	1187.12	1
26	16	17	11.45	42.49	770.79	824.73	1
1	5	23	27.90	89.05	1126.75	1243.70	1
13	18	27	31.19	53.65	1155.00	1239.84	1
8	13	24	38.64	79.17	432.12	549.93	1
10	17	21	35.96	91.47	732.56	859.99	1
15	27	10	45.16	48.56	336.06	429.78	1

transaction

Limit to 5000 rows

```
t.total_price, t.is_refund
FROM db.transaction AS t JOIN db.booking AS b ON t.booking_id = b.id
LEFT JOIN db.promo AS p ON t.promo_id = p.id
WHERE t.is_refund = 1 ) AS subs
JOIN db.user AS u ON u.id = subs.guest_id
JOIN db.property AS p ON p.id = subs.property_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Guest name	Host ID	Tax	Service fee	Booking price	Total price
Ross Laporte	24	14.40	17.72	1155.00	1187.12
Jarvis Valles	32	11.45	42.49	770.79	824.73
Wally Henry	2	27.90	89.05	1126.75	1243.70
Willetta Mckay	24	31.19	53.65	1155.00	1239.84
Tracie Shepard	4	38.64	79.17	432.12	549.93
Rosamond Chisholm	28	35.96	91.47	732.56	859.99
Flora Dietz	30	45.16	48.56	336.06	429.78

We can use a SELECT query as a data source in another query. In the left picture above, I use a query in the transaction table to show all refunded transactions. Then in the right image, using a subselect query, I join this data source with property and user tables to find the guest's name and host ID of the refunded transactions.

Creating view

property_address - View | property_address

Name: property_address

The name of the view is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `property_address` AS
    SELECT
        `ssv`.`Host` AS `Host`,
        `ssv`.`type_name` AS `Place_type`,
        `ssv`.`Property_type` AS `Property_type`,
        `co`.`name` AS `Country`,
        `ci`.`name` AS `City`,
        `a`.`street_address` AS `Address`,
        `a`.`zip` AS `Zip code`
    FROM
        ((((SELECT
            `p`.`address_id` AS `address_id`,
                CONCAT(`u`.`first_name`, ' ', `u`.`last_name`) AS `Host`,
                `plt`.`type_name` AS `type_name`,
                `prt`.`type_name` AS `Property_type`
        FROM
            (((`property` `p`
        JOIN `user` `u` ON ((`p`.`owner_id` = `u`.`id`)))
        JOIN `place_type` `plt` ON ((`p`.`place_type_id` = `plt`.`id`)))
        JOIN `property_type` `prt` ON ((`p`.`property_type_id` = `prt`.`id`)))) `ssv`
        JOIN `address` `a` ON ((`ssv`.`address_id` = `a`.`id`)))
        JOIN `country` `co` ON ((`a`.`country_id` = `co`.`id`)))
        JOIN `city` `ci` ON ((`a`.`city_id` = `ci`.`id`)))
    ORDER BY `ssv`.`Property_type`
```

property_address - View | property_address

Limit to 5000 rows

```sql
SELECT * FROM db.property_address;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Host	Place_type	Property_type	Country	City	Address	Zip code
Sabina Shepherd	Shared room	Apartment	Mongolia	Houma	2005 W Burwood Lane, Salt Lake City, Utah, 15...	54770
Jefferson Gerald	Shared room	Apartment	Germany	Loogootee	2945 Pine Tree Loop, Tallahassee, Florida, 88383	43735
Jeromy Hennessey	Shared room	Apartment	Cyprus	Zuni	3921 4th Rd, Olympia, Washington, 61250	64117
Jolene Lapointe	Private room	Apartment	Bangladesh	Haddonfield	3556 Old Quailwood Lane, Fisher Bldg, Topeka, ...	98944
Adela Peeler	Private room	Apartment	Malaysia	Elida	32 Red Deepwood Lane, Raleigh, NC, 15026	78189
Elvira George	Private room	Apartment	Finland	Lonoke	2556 Sharp Hill Way, Lansing, Michigan, 62493	50500
Jeromy Hennessey	Entire place	Apartment	Tunisia	Haddonfield	38 NE Burwood Dr, Sacramento, CA, 39756	42393
Barney Gerber	Entire place	Apartment	Japan	Elida	1052 Rock Hill Highway, Macy's Building, Oklaho...	11450
Perry Large	Entire place	Apartment	Bangladesh	Colome	2819 Deepwood Ct, Sacramento, California, 23...	99399
Barton Berry	Entire place	Apartment	Cuba	Bloomdale	3163 Ski Hill Hwy, Augusta, ME, 33323	34891
Alexandria Shephard	Entire place	Apartment	Germany	Suttons Bay	41 Mount Highway, Keith Bldg, Dover, Delawar...	07975
Belkis Peebles	Shared room	Guesthouse	Finland	Lookeba	3351 N Flintwood Rd, Montpelier, VT, 10816	23342
Jolene Lapointe	Shared room	Guesthouse	Saudi Arabia	Lookeba	1376 White Mount Highway, Kearns Building, M...	45434
Barton Berry	Entire place	Guesthouse	Kuwait	Elgin	57 Parkwood Loop, Standard Bldg, Helena, Mon...	13392
Devona Henning	Entire place	Guesthouse	Saudi Arabia	Newbury	800 White Buttonwood Circle, Nashville, TN, 36...	49273
Belkis Peebles	Private room	Guesthouse	Germany	Coloma	3174 W Bayview Ct, Judge Bldg, Montgomery, ...	20864
Maire Van	Private room	Guesthouse	Malawi	Eleva	2357 3rd Drive, Duke Energy Bldg, Atlanta, GA,...	96600
Perry Large	Shared room	Guesthouse	Singapore	Bloomdale	2704 NE Hazelwood Street, Austin, Texas, 32133	43438
Jarvis Valles	Shared room	Guesthouse	Denmark	Bloomfield ...	607 New Meadowview Lane, Phoenix, AZ, 14419	94630
Serina Lara	Entire place	Hotel	Japan	Eleva	67 New Riverside Way, Austin, TX, 44471	96139
Serina Lara	Entire place	Hotel	Namibia	Bloomdale	660 1st Ln, 1st Floor, Baton Rouge, LA, 11677	87968
Sabina Shepherd	Shared room	Hotel	Albania	Houma	1930 Red Meadowview Blvd, Carson City, NV, 0...	34958
Milissa Gerard	Shared room	Hotel	Saudi Arabia	Houston	1962 Flintwood Ct, Indianapolis, IN, 02402	76703
Martin Mcintire	Shared room	Hotel	Afghanistan	Newburgh	147 3rd Way, Diamond Building, Dover, Delawa...	53842
Jarvis Valles	Shared room	Hotel	France	Bloomfield ...	889 SE Waterview Highway, Lincoln, NE, 35367	74840
Milissa Gerard	Shared room	Hotel	Korea	Colonial He...	3737 New Social Ct, 54th Floor, Helena, MT, 40...	24021
Jefferson Gerald	Shared room	Hotel	Czech Rep...	Zuni	2000 Prospect Hill Loop, Dover, Delaware, 51314	70957
Adela Peeler	Private room	Hotel	Nicaragua	Porthill	3018 Pine Tree Way, Bartlett Bldg, Saint Paul, ...	20928
Devona Henning	Private room	Hotel	Turkey	Zuni	2525 West Beachwood Loop, Calyon Bldg, Little...	81150
Salome Hensley	Entire place	Hotel	Switzerland	Lonsdale	1842 Riddle Hill Highway, Victor Executive Bldg, ...	08242
Willetta Mckay	Entire place	Hotel	Qatar	Eleva	2986 SW Chapel Hill Pkwy, Lincoln, Nebraska, 6...	18350
Mary Mcintyre	Entire place	Hotel	Bangladesh	Haddonfield	823 3rd Cir, Guardian Bldg, Denver, CO, 29932	93604
Barton Berry	Entire place	Hotel	Czech Rep...	Loogootee	1112 New Church Blvd, 257 Towers Building, Ha...	78879
Sabina Shepherd	Entire place	Hotel	Mexico	Colonial He...	3548 Parkwood Cir, Macy's Bldg, Olympia, WA, ...	32691
Martin Mcintire	Private room	House	Nicaragua	Zuni	1884 New Parkwood Avenue, Albany, New York...	09304
Jolene Lapointe	Entire place	House	Luxembourg	Colon	652 Mount Court, Pierre, South Dakota, 27075	65214

SQL view hides the complexity of the data and restricts unnecessary access to the database. It allows the users to access only a particular column rather than the whole data of the table. In this test case, using subselect queries, I have created a view by joining five tables to display the details of a property host name, place type, property type, and complete address.

Thanks!

Do you have any questions?

mohamad-sadegh.solouki@iubh.de
+98 912 022 4052

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon** and infographics & images by **Freepik**