

## The Requirements specification

### **1) The roles and their actions:**

#### **- THE HOST :**

- post an announce(picture, description ...)
- update an announce
- delete an announce
- rate the guest
- get paid after 24 hours after guest arrival
- create a profile page
- upload at least one picture
- post his/her phone number
- calculate his/her expected income from his/her own accomodation
- connect his/her profile with social networks

#### **- THE GUEST :**

- pay by credit card
- Book an accomodation
- rate the host

- create an account
- provide more personal information (pictures, phone number, job ..)
- connect his/her profile with social networks

- **THE COMPANY :**

- establish the contact between host and guest
- handle the orders
- handle the transactions
- get 6-12% from guests and 3% from hosts after each deal done
- provide the access to various accomodations

## **1) Which data and functions are required?**

- **THE HOST :**

- HostID {ID}
- First Name
- Last Name
- Email Address
- Phone Number
- Password
- Nationality

- Picture link
- Accomodation address ( country, city, street and zip code)
- Accomodation description
- Accomodation picture(s)
- Accomodation price per night
- Accomodation availability
- Accomodation category(category, vue, space)
- Rate
- Invoice
- Invoice Date
- Bank account number
- Bank name
- \* postAds()
- \* deleteAds()
- \* updateAds()
- \* rate()
- \* createProfile()
- \* uploadPhoto()
- \* postPhoneNumber()
- \* calculateIncome()

\* shareProfil()

- THE GUEST :

- GuestID {ID}
- First Name
- Last Name
- Email Address
- Phone Number
- Job
- Marital Status
- Nationality
- Password
- Rate
- Invoice
- Invoice Date
- PhotoID {ID}
- Photo link
- CardID {ID}
- Card Number
- BookingID {ID}
- nbr of nights
- CheckIn Date
- CheckOut Date
- Total Cost
- nbr of persons
- BStatusID {ID}

- isBooked
  - ExpirationDate
  - \* rate()
  - \* createProfile()
  - \* uploadPhoto()
  - \* postPhoneNumber()
  - \* shareProfil()
  - \* book()
  - \* pay()
- THE COMPANY :
- AdminID {ID}
  - Name
  - Email Address
  - Phone Number
  - Password
  - InvoiceID {ID}
  - Invoice
  - Date
  - OrderID
  - isConfirm
  - Confirmation Date
  - CardID {ID}
  - Card Number

- BankAccountID {ID}
- Account Number
- Bank Name
- TransactionID {ID}
- Method
- Status
- AdminComission
- Host Comission
- Date
- CurrencyID {ID}
- Currency
- \* handleTransaction()
- \* handleOrders()
- \* establishContact()
- \* calculateComission()

### **3) Data dictionary:**

<b>The table name</b>	<b>Attribute name</b>	<b>Attribute description</b>	<b>Data type</b>	<b>Data range</b>
Host	HostID	Assign a primary key to each host	INT	NOT NULL UNIQUE
	FirstName	Includes the first name of the host	VARCHAR	(50)
	LastName	Includes the last name of the host	VARCHAR	(50)
	EmailAddress	Hold the email address of the host	VARCHAR	(100) UNIQUE
	PhoneNumber	Describes the phone number	INT	
	Password	Includes the password of the host	CHAR	(100)

		profile		NOT NULL
	Nationality	Includes the origin of the host	CHAR	(200)
	Photo	Includes a link of the host photo	VARBINARY	(1500)
<b>Accommodation</b>	AccommodationID	Assign a primary key to each accommodation	INT	NOT NULL UNIQUE
	Description	Provide the accommodation with its description	TEXT	
	Price/night	Hold the price of one night of the described accommodation	REAL	
	HostID	Identify the owner of the accommodation as a foreign key	INT	
<b>Rating</b>	RatingID	Assign a primary key to every rating	INT	NOT NULL UNIQUE
	Rating	Include the rating assigned by the guests	Decimal	(2,1)
	HostID	Identify the owner of the rating as a foreign key	INT	
<b>Host Commission</b>	CommissionID	Assign a primary key for each commission	INT	NOT NULL UNIQUE
	Commission	Includes the amount of the commission	REAL	
	Date	When the host gets his commission from the admin	DATE	
	IncomeID	Identify the income as a foreign key	INT	
	HostID	Identify the owner of the commission as a foreign key	INT	
	CardID	Identify which card paid the commission as a foreign key	INT	
<b>Host BankAccount</b>	BankAccountID	Assign a primary key for each bank account	INT	NOT NULL UNIQUE
	Account Number	Describes the number of the bank account	INT	NOT NULL UNIQUE
	Bank Name	Describes the name of the bank account	CHAR	(100)

	HostID	Identify the owner of the bank account as a foreign key	INT	
	TransactionID	Identify the owner of the transaction as a foreign key	INT	
<b>Pictures</b>	PictureID	Assign a primary key for each picture	INT	NOT NULL UNIQUE
	Link	Includes the link for the photos	VARBINARY	(1500)
	AccommodationID	Identify the ownership of the pictures as a foreign key	INT	
<b>Category</b>	CategoryID	Assign a primary key for accommodation category	INT	NOT NULL UNIQUE
	Category	Describes the category of the accommodation	CHAR	(100)
	Space	Includes the space of the visited place	REAL	
	Vue	Mentions the vue of the place	CHAR	(80)
	AccommodationID	Identify the possessor of the category as a foreign key	INT	
<b>Status</b>	StatusID	Assign a primary key for the status of the accommodation	INT	NOT NULL UNIQUE
	isAvailable	Describes the status of the reserved accommodation	BOOLEAN	
	AccommodationID	Identify the possessor of the category as a foreign key	INT	
<b>Country</b>	CountryID	Assign a primary key for the country of the accommodation	INT	NOT NULL UNIQUE
	Name	Includes the name of the country	VARCHAR	(500)
	AccommodationID	Identify the possessor of the category as a foreign key	INT	
<b>City</b>	CityID	Assign a primary key for the city	INT	NOT NULL UNIQUE
	City	Includes the name of the city	VARCHAR	(500)
	Zip Code	Includes the zip code of the city	INT	
	CountryID	Identify the country primary key as a foreign key	INT	

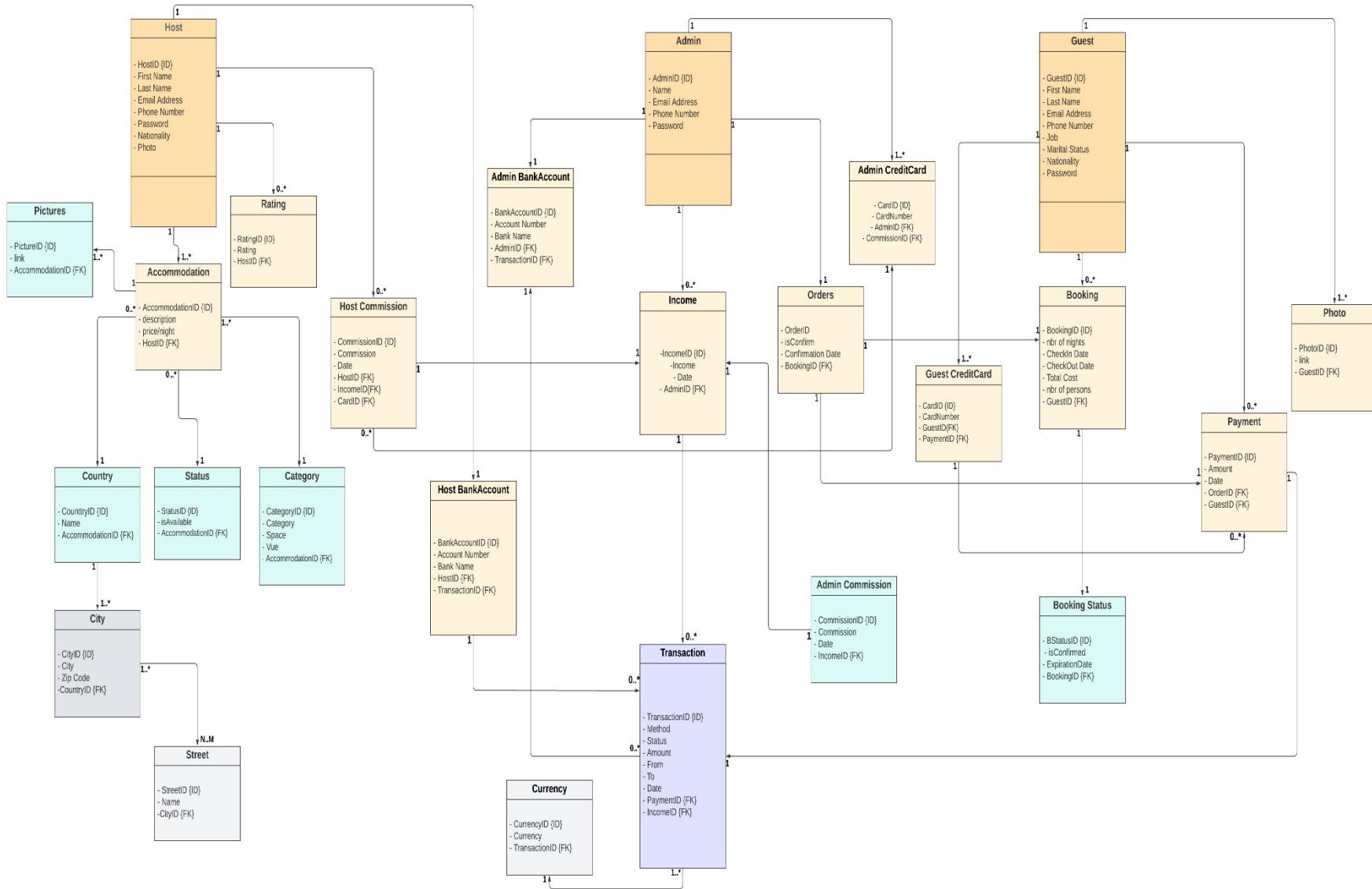
<b>Admin</b>	AdminID	Assigns a primary key for the platform	INT	NOT NULL UNIQUE
	FullName	Includes the name of the platform	VARCHAR	(200)
	Email Address	Hold the email address of the admin	CHAR	(200)
	Phone Number	Describes the phone number	INT	
	Password	Includes the password of the admin profile	CHAR	(100) NOT NULL
<b>Admin BankAccount</b>	BankAccountID	Assign a primary key for each bank account	INT	NOT NULL UNIQUE
	Account Number	Describes the number of the bank account	INT	NOT NULL UNIQUE
	Bank Name	Describes the name of the bank account	CHAR	(80)
	AdminID	Identify the owner of the bank account as a foreign key	INT	
	TransactionID	Identify the owner of the transaction as a foreign key	INT	
<b>Income</b>	IncomeID	Assign a primary key for each income	INT	NOT NULL UNIQUE
	Income	Includes the cost of the reservation	REAL	
	Date	Identifies when the admin gets the income	DATE	
	AdminID	Identify the owner of the income as a foreign key	INT	
<b>Admin Commission</b>	CommissionID	Assign a primary key for each commission	INT	NOT NULL UNIQUE
	Commission	Includes the amount of the commission	REAL	
	Date	When the admin gets his commission from one income	DATE	
	IncomeID	Identify the income as a foreign key	INT	

<b>Orders</b>	OrderID	Assign a primary key for order	INT	NOT NULL UNIQUE
	isConfirm	Describes the status of the order	BOOLEAN	
	Confirmation Date	The date of the confirmation	DATE	
	BookingID	Identify the order is for which booking as a foreign key	INT	
<b>Admin CreditCard</b>	CardID	Assign a primary key for each credit card	INT	NOT NULL UNIQUE
	CardNumber	Holds the number of the credit card	INT	
	AdminID	Identify the owner of the credit card as a foreign key	INT	
	InvoiceID	Identify which credit card paid for which invoice as a foreign key	INT	
<b>Guest</b>	GuestID	Assign a primary key to each guest	INT	NOT NULL UNIQUE
	First Name	Includes the first name of the guest	VARCHAR	(50)
	Last Name	Includes the last name of the guest	VARCHAR	(50)
	Email Address	Hold the email address of the guest	VARCHAR	(100) UNIQUE
	Phone number	Describes the phone number	INT	
	Password	Includes the password of the guest profile	CHAR	(100) NOT NULL
	Nationality	Includes the origin of the guest	CHAR	(200)
	Marital status	Includes the marital status of the guest	VARCHAR	(40)
	Job	Describes the guest's job	CHAR	(60)
<b>Guest CreditCard</b>	CardID	Assign a primary key for each credit card	INT	NOT NULL UNIQUE
	CardNumber	Holds the number of the credit card	INT	

	GuestID	Identify the owner of the credit card as a foreign key	INT	
	PaymentID	Identify which credit card paid for which invoice as a foreign key	INT	
<b>Booking</b>	BookingID	Assign a primary key for each credit card	INT	NOT NULL UNIQUE
	nbrOfNights	The number of nights consumed by guests	DECIMAL(2,1)	
	ChekInDate	The check in date	DATE	
	ChekOutDate	The check out date	DATE	
	Total	The total cost of one reservation	REAL	
	nbrOfPersons	The number of persons of one reservation	INT	NOT NULL
	GuestID	Identify the owner of the booking as a foreign key	INT	
<b>Payment</b>	Payment ID	Assign a primary key for each payment	INT	NOT NULL UNIQUE
	Amount	Includes the amount of the transactions	REAL	
	Date	The date of the invoice	DATE	
	GuestID	Identify the owner of the invoice as a foreign key	INT	
	OrderID	Identify the order of the invoice as a foreign key	INT	
<b>Photo</b>	Photoid	Assign a primary key for each user photo	INT	
	Link	Includes the link for the photos	VARBINARY	(1500)
	GuestID	Identify the owner of the photo as a foreign key	INT	
<b>Booking Status</b>	BStatusID	Assign a primary key for the booking status	INT	NOT NULL UNIQUE
	isConfirmed	Describes the status of the booking	BOOLEAN	
	ExpirationDate	The date of the booking's expiration	DATE	

	BookingID	Identify the booking as a foreign key	INT	
<b>Transaction</b>	TransactionID	Assign a primary key for the booking status	INT	NOT NULL UNIQUE
	Method	The method of the payment could be for example by credit card or by bank	CHAR	(20)
	Status	Is transaction done or in progress ?	CHAR	(20)
	Amount	The amount of the transaction	REAL	
	From	From who's guest	VARCHAR	(100)
	To	To the company account	VARCHAR	(100)
	Date	The date of the transaction	DATE	
	PaymentID	Identify the transaction is for which payment as a foreign key	INT	
	IncomeID	Identify the income of the admin as a foreign key	INT	
<b>Currency</b>	CurrencyID	Assign a primary key for each currency	INT	NOT NULL UNIQUE
	Currency	The currency could be euros or dollars or others	CHAR	(20)
	TransactionID	Identify the owner of the transaction as a foreign key	INT	

- The ER Model



# **PROJECT: BUILD A DATA MART IN SQL**

**FIRST NAME : DORRA**

**LAST NAME : KAHLA**

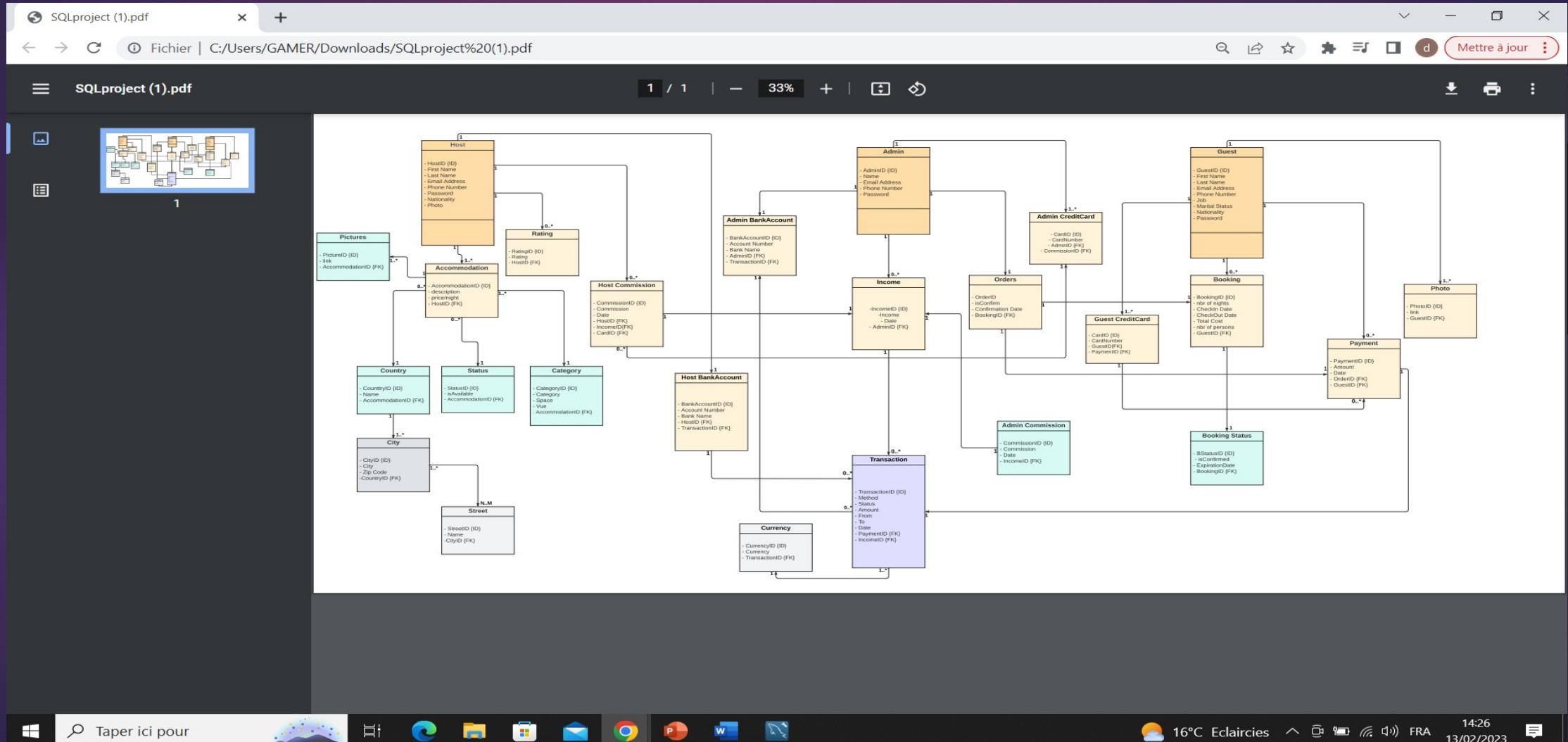
**MATRICULATIO N° : 92108575**



# A SUMMARY

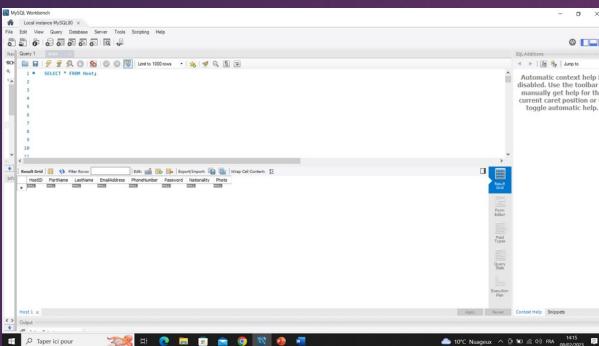
- ▶ In this phase, I used MySQL as a database management system to implement the my SQL queries according to the modeling phase .
- ▶ In the following slides you will find the ER model of the project then the implementation of each entity separated each in its own slide .
- ▶ The slides include the creation of the tables then the insertion query followed by a test case , each with a screenshot of the corresponding sql statement .

# The ER Diagram



- CREATE TABLE Host (HostID INT NOT NULL UNIQUE PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), EmailAddress VARCHAR(100) UNIQUE, PhoneNumber INT, Password CHAR(100) NOT NULL, Nationality CHAR(200), Photo VARBINARY(1500));

- 
- 
- 
- 
- 



The screenshot shows the MySQL Workbench interface with the 'Host' table selected. The SQL editor pane contains a SELECT query: 'SELECT \* FROM Host;'. The results grid displays 20 rows of data, each representing a host with a unique ID, first name, last name, email address, phone number, password, nationality, and a binary photo.

HostID	FirstName	LastName	EmailAddress	PhoneNumber	Password	Nationality
1	John	Doe	john.doe@gmail.com	123456789	password1	USA
2	Jane	Doe	jane.doe@gmail.com	987654321	password2	UK
3	Jack	Smith	jack.smith@gmail.com	121212121	password3	Canada
4	Jill	Smith	jill.smith@gmail.com	343434343	password4	Australia
5	James	Johnson	james.johnson@gmail.com	565656565	password5	France
6	Emily	Johnson	emily.johnson@gmail.com	787878787	password6	Germany
7	Michael	Williams	michael.williams@gmail.com	919191919	password7	Italy
8	Sarah	Williams	sarah.williams@gmail.com	101010101	password8	Spain
9	William	Brown	william.brown@gmail.com	111111111	password9	Russia
10	Elizabeth	Brown	elizabeth.brown@gmail.com	121212121	password10	Japan
11	David	Jones	david.jones@gmail.com	1313131313	password11	China
12	Karen	Jones	karen.jones@gmail.com	1414141414	password12	India
13	Richard	Miller	richard.miller@gmail.com	1515151515	password13	Brazil
14	Jessica	Miller	jessica.miller@gmail.com	1616161616	password14	Mexico
15	Catherine	Davis	catherine.davis@gmail.com	2020202020	password15	South Africa
16	Sophia	Wilson	sophia.wilson@gmail.com	2121212121	password16	Canada
17	Charles	Wilson	charles.wilson@gmail.com	2222222222	password17	France
18	Thomas	Davis	thomas.davis@gmail.com	2323232323	password18	Germany
19	Matthew	Anderson	matthew.anderson@gmail.com	2424242424	password19	Australia
20	Owen	King	owenking@example.com	2525252525	password20	UK

- INSERT INTO Host (HostID, FirstName, LastName, EmailAddress, PhoneNumber, Password, Nationality, Photo)VALUES (1, 'John', 'Doe', 'john.doe@gmail.com', 123456789, 'password1', 'USA', 0x01),(2, 'Jane', 'Doe', 'jane.doe@gmail.com', 987654321, 'password2', 'UK', 0x02),(3, 'Jack', 'Smith', 'jack.smith@gmail.com', 121212121, 'password3', 'Canada', 0x03),(4, 'Jill', 'Smith', 'jill.smith@gmail.com', 343434343, 'password4', 'Australia', 0x04),(5, 'James', 'Johnson', 'james.johnson@gmail.com', 565656565, 'password5', 'France', 0x05),(6, 'Emily', 'Johnson', 'emily.johnson@gmail.com', 787878787, 'password6', 'Germany', 0x06),(7, 'Michael', 'Williams', 'michael.williams@gmail.com', 919191919, 'password7', 'Italy', 0x07),(8, 'Sarah', 'Williams', 'sarah.williams@gmail.com', 101010101, 'password8', 'Spain', 0x08),(9, 'William', 'Brown', 'william.brown@gmail.com', 111111111, 'password9', 'Russia', 0x09),(10, 'Elizabeth', 'Brown', 'elizabeth.brown@gmail.com', 121212121, 'password10', 'Japan', 0x0A),(11, 'David', 'Jones', 'david.jones@gmail.com', 1313131313, 'password11', 'China', 0x0B),(12, 'Karen', 'Jones', 'karen.jones@gmail.com', 1414141414, 'password12', 'India', 0x0C),(13, 'Richard', 'Miller', 'richard.miller@gmail.com', 1515151515, 'password13', 'Brazil', 0x0D),(14, 'Jessica', 'Miller', 'jessica.miller@gmail.com', 1616161616, 'password14', 'Mexico', 0x0E),(15, 'Thomas', 'Davis', 'thomas.davis@gmail.com', 1717171717, 'password15', 'South Africa', 0x0F),(16, 'Catherine', 'Davis', 'catherine.davis@gmail.com', 1818181818, 'password16', 'Canada', 0x10),(17, 'Charles', 'Wilson', 'charles.wilson@gmail.com', 1919191919, 'password17', 'France', 0x11),(18, 'Sophia', 'Wilson', 'sophia.wilson@gmail.com', 2020202020, 'password18', 'Germany', 0x12),(19, 'Matthew', 'Anderson', 'matthew.anderson@gmail.com', 2121212121, 'password19', 'Australia', 0x13),(20, 'Owen', 'King', 'owenking@example.com', 2222222222, 'password20', 'UK', null);

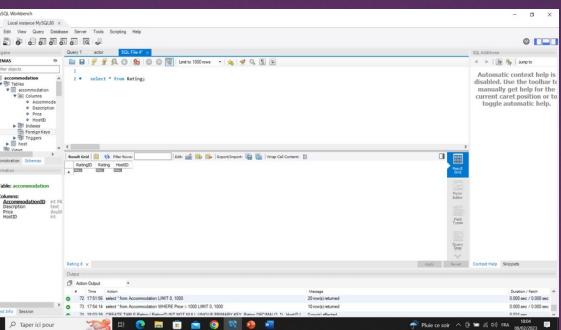
- 

- Test Case : SELECT COUNT(FisrtName), Nationality FROM Host GROUP BY Nationality;

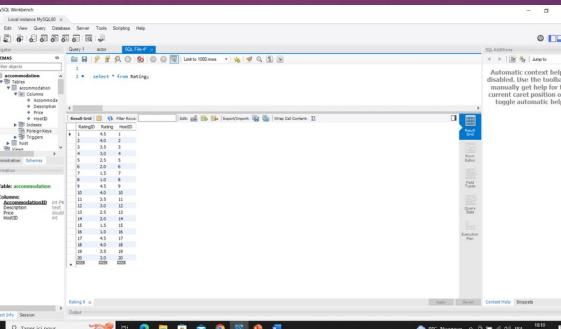
The screenshot shows the MySQL Workbench interface with the 'Host' table selected. The SQL editor pane contains the following query: 'SELECT COUNT(FisrtName), Nationality FROM Host GROUP BY Nationality;'. The results grid displays the count of hosts for each nationality.

Nationality	Count
USA	1
Canada	1
Australia	1
France	1
Germany	1
Italy	1
Spain	1
Russia	1
Japan	1
China	1
India	1
Brazil	1
Mexico	1
South Africa	1
Canada	1
France	1
Germany	1
Italy	1
Spain	1
Russia	1
Japan	1
China	1
India	1
Brazil	1
Mexico	1
South Africa	1

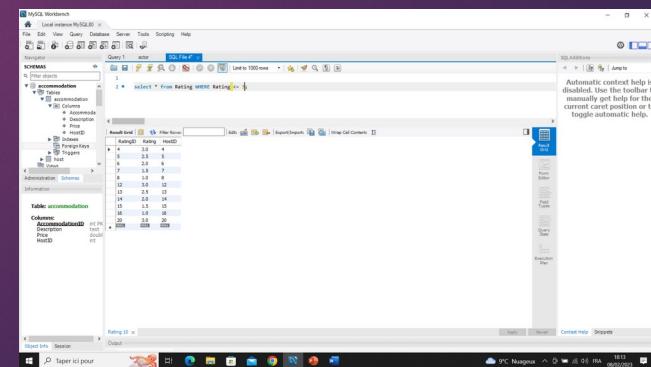
- CREATE TABLE Rating (RatingID INT NOT NULL UNIQUE PRIMARY KEY, Rating DECIMAL(2, 1), HostID INT REFERENCES Host(HostID) )



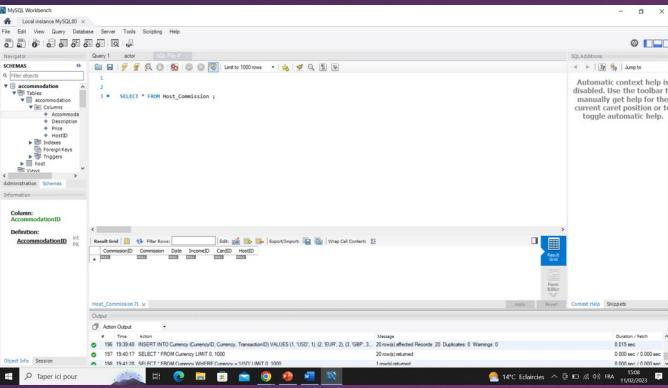
- INSERT INTO Rating (RatingID, Rating, HostID) VALUES (1, 4.5, 1), (2, 4.0, 2), (3, 3.5, 3), (4, 3.0, 4), (5, 2.5, 5), (6, 2.0, 6), (7, 1.5, 7), (8, 1.0, 8), (9, 4.5, 9), (10, 4.0, 10), (11, 3.5, 11), (12, 3.0, 12), (13, 2.5, 13), (14, 2.0, 14), (15, 1.5, 15), (16, 1.0, 16), (17, 4.5, 17), (18, 4.0, 18), (19, 3.5, 19), (20, 3.0, 20);



- Test case : select \* from Rating WHERE Rating <= 3;



- CREATE TABLE Host\_Commission (CommissionID INT NOT NULL UNIQUE PRIMARY KEY, Commission REAL, Date DATE, IncomeID INT REFERENCES Income(IncomeID), CardID INT REFERENCES Card(CardID), HostID INT REFERENCES Host(HostID));



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following SQL code:

```
INSERT INTO Host_Commission (CommissionID, Commission, Date, IncomeID, CardID, HostID) VALUES(1, 10.0, '2022-12-05', 1, 1, 1), (2, 15.0, '2022-12-06', 2, 2, 2), (3, 20.0, '2022-12-07', 3, 3, 3), (4, 25.0, '2022-12-08', 4, 4, 4), (5, 30.0, '2022-12-09', 5, 5, 5), (6, 35.0, '2022-12-10', 6, 6, 6), (7, 40.0, '2022-12-11', 7, 7, 7), (8, 45.0, '2022-12-12', 8, 8, 8), (9, 50.0, '2022-12-13', 9, 9, 9), (10, 55.0, '2022-12-14', 10, 10, 10), (11, 60.0, '2022-12-15', 11, 11, 11), (12, 65.0, '2022-12-16', 12, 12, 12), (13, 70.0, '2022-12-17', 13, 13, 13), (14, 75.0, '2022-12-18', 14, 14, 14), (15, 80.0, '2022-12-19', 15, 15, 15), (16, 85.0, '2022-12-20', 16, 16, 16), (17, 90.0, '2022-12-21', 17, 17, 17), (18, 95.0, '2022-12-22', 18, 18, 18), (19, 100.0, '2022-12-23', 19, 19, 19), (20, 105.0, '2022-12-24', 20, 20, 20);
```

The results pane shows the inserted data:

CommissionID	Commission	Date	IncomeID	CardID	HostID
1	10.0	2022-12-05	1	1	1
2	15.0	2022-12-06	2	2	2
3	20.0	2022-12-07	3	3	3
4	25.0	2022-12-08	4	4	4
5	30.0	2022-12-09	5	5	5
6	35.0	2022-12-10	6	6	6
7	40.0	2022-12-11	7	7	7
8	45.0	2022-12-12	8	8	8
9	50.0	2022-12-13	9	9	9
10	55.0	2022-12-14	10	10	10
11	60.0	2022-12-15	11	11	11
12	65.0	2022-12-16	12	12	12
13	70.0	2022-12-17	13	13	13
14	75.0	2022-12-18	14	14	14
15	80.0	2022-12-19	15	15	15
16	85.0	2022-12-20	16	16	16
17	90.0	2022-12-21	17	17	17
18	95.0	2022-12-22	18	18	18
19	100.0	2022-12-23	19	19	19
20	105.0	2022-12-24	20	20	20

- INSERT INTO Host\_Commission (CommissionID, Commission, Date, IncomeID, CardID, HostID)VALUES(1, 10.0, '2022-12-05', 1, 1, 1),(2, 15.0, '2022-12-06', 2, 2, 2),(3, 20.0, '2022-12-07', 3, 3, 3),(4, 25.0, '2022-12-08', 4, 4, 4),(5, 30.0, '2022-12-09', 5, 5, 5),(6, 35.0, '2022-12-10', 6, 6, 6),(7, 40.0, '2022-12-11', 7, 7, 7),(8, 45.0, '2022-12-12', 8, 8, 8),(9, 50.0, '2022-12-13', 9, 9, 9),(10, 55.0, '2022-12-14', 10, 10, 10),(11, 60.0, '2022-12-15', 11, 11, 11),(12, 65.0, '2022-12-16', 12, 12, 12),(13, 70.0, '2022-12-17', 13, 13, 13),(14, 75.0, '2022-12-18', 14, 14, 14),(15, 80.0, '2022-12-19', 15, 15, 15),(16, 85.0, '2022-12-20', 16, 16, 16),(17, 90.0, '2022-12-21', 17, 17, 17),(18, 95.0, '2022-12-22', 18, 18, 18),(19, 100.0, '2022-12-23', 19, 19, 19),(20, 105.0, '2022-12-24', 20, 20, 20);
- Test case : SELECT \* FROM Host\_Commission WHERE Commission = 105;

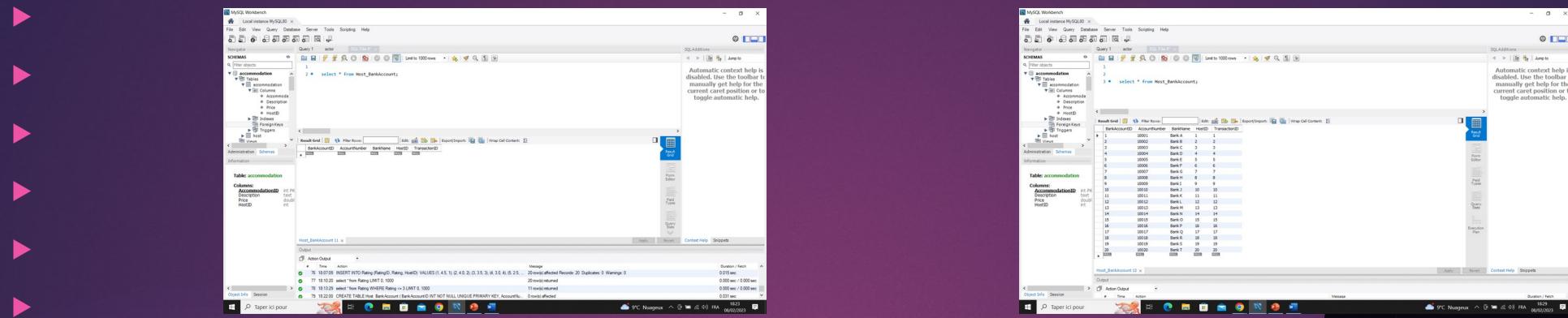
The screenshot shows the MySQL Workbench interface. The SQL editor contains the following SQL code:

```
SELECT * FROM Host_Commission WHERE Commission = 105;
```

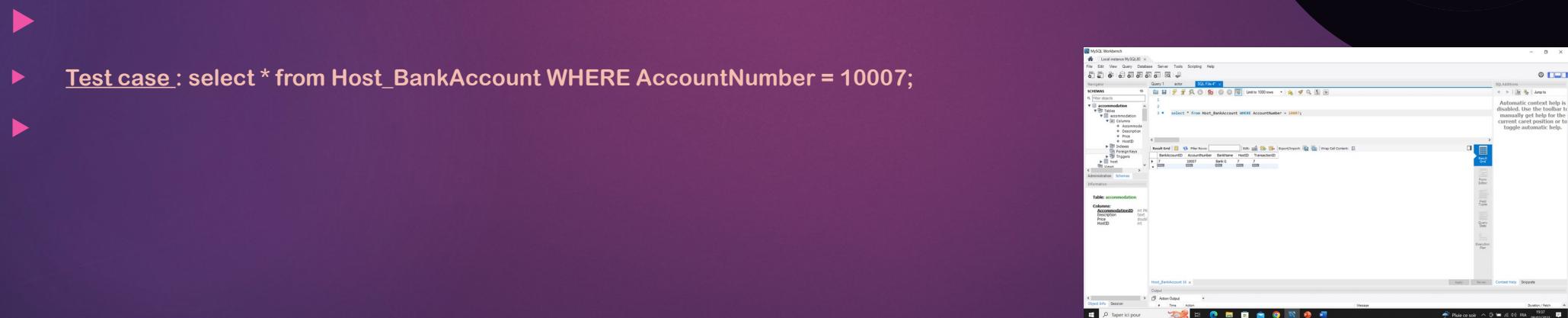
The results pane shows the single row returned by the query:

CommissionID	Commission	Date	IncomeID	CardID	HostID
20	105.0	2022-12-24	20	20	20

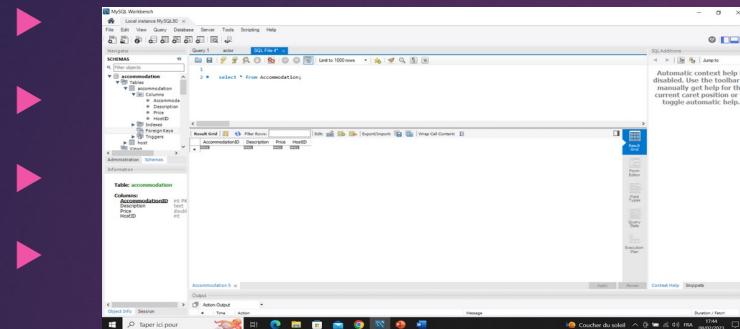
- CREATE TABLE Host\_BankAccount (BankAccountId INT NOT NULL UNIQUE PRIMARY KEY, AccountNumber INT NOT NULL UNIQUE, BankName CHAR(100), HostID INT REFERENCES Host(HostID), TransactionID INT REFERENCES Transaction(TransactionID) );



- INSERT INTO Host\_BankAccount (BankAccountId, AccountNumber, BankName, HostID, TransactionID)VALUES (1, 10001, 'Bank A', 1, 1), (2, 10002, 'Bank B', 2, 2), (3, 10003, 'Bank C', 3, 3), (4, 10004, 'Bank D', 4, 4), (5, 10005, 'Bank E', 5, 5), (6, 10006, 'Bank F', 6, 6), (7, 10007, 'Bank G', 7, 7), (8, 10008, 'Bank H', 8, 8), (9, 10009, 'Bank I', 9, 9), (10, 10010, 'Bank J', 10, 10), (11, 10011, 'Bank K', 11, 11), (12, 10012, 'Bank L', 12, 12), (13, 10013, 'Bank M', 13, 13), (14, 10014, 'Bank N', 14, 14), (15, 10015, 'Bank O', 15, 15), (16, 10016, 'Bank P', 16, 16), (17, 10017, 'Bank Q', 17, 17), (18, 10018, 'Bank R', 18, 18), (19, 10019, 'Bank S', 19, 19), (20, 10020, 'Bank T', 20, 20);



- CREATE TABLE Accommodation (AccommodationID INT NOT NULL UNIQUE PRIMARY KEY, Description TEXT, Price REAL, HostID int REFERENCES Host(HostID))

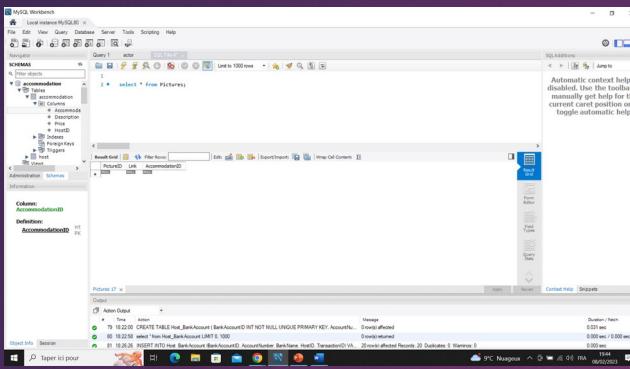


AccommodationID	Description	Price	HostID
1	Nice and cozy apartment located in the city center	1000	1
2	Beautiful villa with a private pool and large garden	1500	2
3	Spacious room in a shared house	500	3
4	Comfortable studio with a view of the city	800	4
5	Stylish apartment with a rooftop terrace	1200	5
6	Charming cottage with a fireplace	700	6
7	Modern apartment with a large kitchen	1100	7
8	Luxury apartment with a Jacuzzi	1700	8
9	Elegant apartment with a balcony	1300	9
10	Rustic cabin in the woods	600	10
11	Stylish penthouse with a rooftop pool	2000	11
12	Cozy room in a bed and breakfast	400	12
13	Vintage apartment in a historic building	900	13
14	Bright and airy apartment with a balcony	1000	14
15	Comfortable room in a shared apartment	450	15
16	Spacious apartment with a large living room	1200	16
17	Elegant apartment with a fireplace	1300	17
18	Luxury villa with a private pool	2000	18
19	Stylish apartment with a roof garden	1500	19
20	Comfortable apartment with a view of the park	900	20

- INSERT INTO Accommodation (AccommodationID, Description, Price, HostID) VALUES (1, 'Nice and cozy apartment located in the city center', 1000, 1), (2, 'Beautiful villa with a private pool and large garden', 1500, 2), (3, 'Spacious room in a shared house', 500, 3), (4, 'Comfortable studio with a view of the city', 800, 4), (5, 'Stylish apartment with a rooftop terrace', 1200, 5), (6, 'Charming cottage with a fireplace', 700, 6), (7, 'Modern apartment with a large kitchen', 1100, 7), (8, 'Luxury apartment with a Jacuzzi', 1700, 8), (9, 'Elegant apartment with a balcony', 1300, 9), (10, 'Rustic cabin in the woods', 600, 10), (11, 'Stylish penthouse with a rooftop pool', 2000, 11), (12, 'Cozy room in a bed and breakfast', 400, 12), (13, 'Vintage apartment in a historic building', 900, 13), (14, 'Bright and airy apartment with a balcony', 1000, 14), (15, 'Comfortable room in a shared apartment', 450, 15), (16, 'Spacious apartment with a large living room', 1200, 16), (17, 'Elegant apartment with a fireplace', 1300, 17), (18, 'Luxury villa with a private pool', 2000, 18), (19, 'Stylish apartment with a roof garden', 1500, 19), (20, 'Comfortable apartment with a view of the park', 900, 20);
- Test case : select \* from Accommodation WHERE Price > 1000;

AccommodationID	Description	Price	HostID
2	Beautiful villa with a private pool and large garden	1500	2
3	Spacious room in a shared house	500	3
4	Comfortable studio with a view of the city	800	4
5	Stylish apartment with a rooftop terrace	1200	5
6	Charming cottage with a fireplace	700	6
7	Modern apartment with a large kitchen	1100	7
8	Luxury apartment with a Jacuzzi	1700	8
9	Elegant apartment with a balcony	1300	9
11	Stylish penthouse with a rooftop pool	2000	11
12	Cozy room in a bed and breakfast	400	12
13	Vintage apartment in a historic building	900	13
14	Bright and airy apartment with a balcony	1000	14
15	Comfortable room in a shared apartment	450	15
16	Spacious apartment with a large living room	1200	16
17	Elegant apartment with a fireplace	1300	17
18	Luxury villa with a private pool	2000	18
19	Stylish apartment with a roof garden	1500	19
20	Comfortable apartment with a view of the park	900	20

- CREATE TABLE Pictures (PictureID INT NOT NULL UNIQUE PRIMARY KEY,Link VARBINARY(1500), AccommodationID INT REFERENCES Accommodation(AccommodationID));

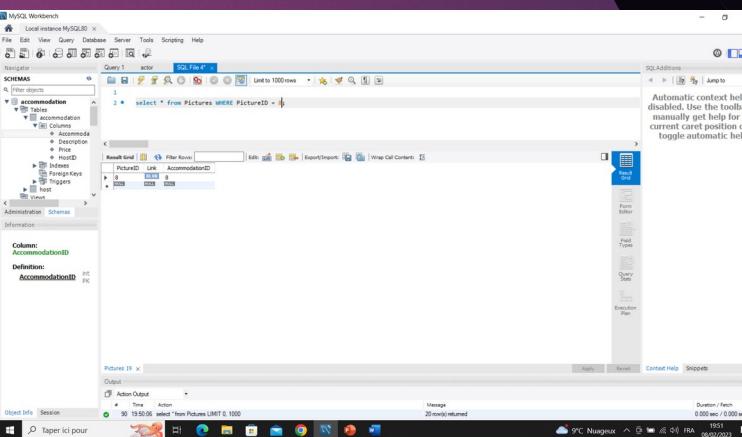


The screenshot shows the MySQL Workbench interface with two panes. The left pane displays the 'Schemas' tree, and the right pane shows the 'Query 1' editor with the following SQL command:

```
CREATE TABLE Pictures (PictureID INT NOT NULL UNIQUE PRIMARY KEY,Link VARBINARY(1500), AccommodationID INT REFERENCES Accommodation(AccommodationID));
```

The status bar at the bottom indicates the command was executed successfully.

- INSERT INTO Pictures (PictureID, Link, AccommodationID) VALUES (1, 0x0101010101, 1), (2, 0x0202020202, 2), (3, 0x0303030303, 3), (4, 0x0404040404, 4), (5, 0x0505050505, 5), (6, 0x0606060606, 6), (7, 0x0707070707, 7), (8, 0x0808080808, 8), (9, 0x0909090909, 9), (10, 0x1010101010, 10), (11, 0x1111111111, 11), (12, 0x1212121212, 12), (13, 0x1313131313, 13), (14, 0x1414141414, 14), (15, 0x1515151515, 15), (16, 0x1616161616, 16), (17, 0x1717171717, 17), (18, 0x1818181818, 18), (19, 0x1919191919, 19), (20, 0x2020202020, 20);

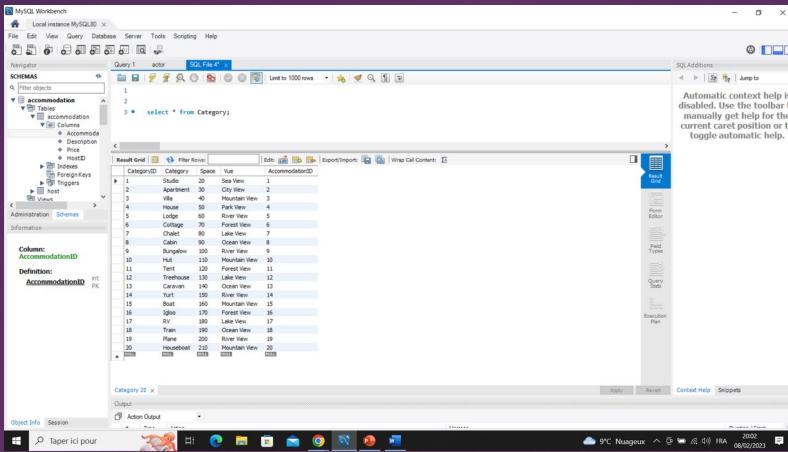


The screenshot shows the MySQL Workbench interface with two panes. The left pane displays the 'Schemas' tree, and the right pane shows the 'Query 1' editor with the following SQL command:

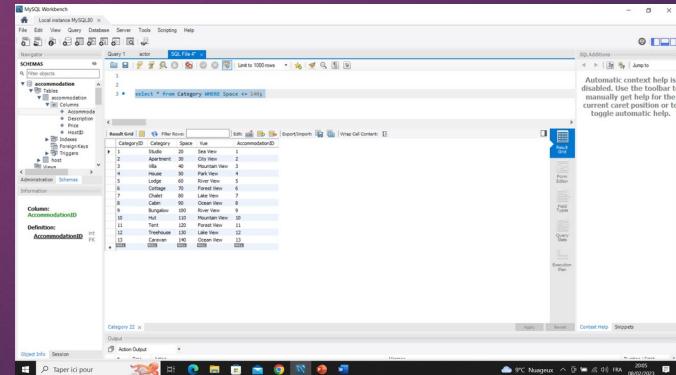
```
INSERT INTO Pictures (PictureID, Link, AccommodationID) VALUES (1, 0x0101010101, 1), (2, 0x0202020202, 2), (3, 0x0303030303, 3), (4, 0x0404040404, 4), (5, 0x0505050505, 5), (6, 0x0606060606, 6), (7, 0x0707070707, 7), (8, 0x0808080808, 8), (9, 0x0909090909, 9), (10, 0x1010101010, 10), (11, 0x1111111111, 11), (12, 0x1212121212, 12), (13, 0x1313131313, 13), (14, 0x1414141414, 14), (15, 0x1515151515, 15), (16, 0x1616161616, 16), (17, 0x1717171717, 17), (18, 0x1818181818, 18), (19, 0x1919191919, 19), (20, 0x2020202020, 20);
```

The status bar at the bottom indicates the command was executed successfully.

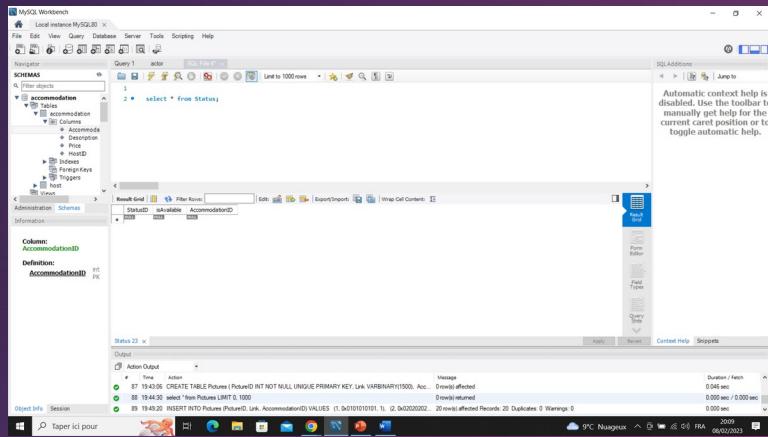
- CREATE TABLE Category (CategoryID INT NOT NULL UNIQUE PRIMARY KEY, Category CHAR(100), Space REAL, Vue CHAR(80), AccommodationID INT REFERENCES Accommodation(AccommodationID) );



- `INSERT INTO Category (CategoryID, Category, Space, Vue, AccommodationID)VALUES(1, 'Studio', 20, 'Sea View', 1),(2, 'Apartment', 30, 'City View', 2),(3, 'Villa', 40, 'Mountain View', 3),(4, 'House', 50, 'Park View', 4),(5, 'Lodge', 60, 'River View', 5),(6, 'Cottage', 70, 'Forest View', 6),(7, 'Chalet', 80, 'Lake View', 7),(8, 'Cabin', 90, 'Ocean View', 8),(9, 'Bungalow', 100, 'River View', 9),(10, 'Hut', 110, 'Mountain View', 10),(11, 'Tent', 120, 'Forest View', 11),(12, 'Treehouse', 130, 'Lake View', 12),(13, 'Caravan', 140, 'Ocean View', 13),(14, 'Yurt', 150, 'River View', 14),(15, 'Boat', 160, 'Mountain View', 15),(16, 'Igloo', 170, 'Forest View', 16),(17, 'RV', 180, 'Lake View', 17),(18, 'Train', 190, 'Ocean View', 18),(19, 'Plane', 200, 'River View', 19),(20, 'Houseboat', 210, 'Mountain View', 20);`
  - Test case : select \* from Category WHERE Space <= 140;



- ▶ CREATE TABLE Status (StatusID INT NOT NULL UNIQUE PRIMARY KEY, isAvailable BOOLEAN, AccommodationID INT REFERENCES Accommodation(AccommodationID));



This screenshot shows the MySQL Workbench interface with the 'Status' table populated with data. The table has three columns: StatusID, isAvailable, and AccommodationID. The data is as follows:

StatusID	isAvailable	AccommodationID
2	0	2
3	1	3
5	0	4
6	1	5
7	1	6
9	1	9
10	0	20
11	1	11
12	0	12
14	1	14
15	1	15
16	0	16
17	1	17
19	1	19
20	0	20

- ▶ INSERT INTO Status (StatusID, isAvailable, AccommodationID) VALUES (1, 1, 1), (2, 0, 2), (3, 1, 3), (4, 0, 4), (5, 1, 5), (6, 0, 6), (7, 1, 7), (8, 0, 8), (9, 1, 9), (10, 0, 10), (11, 1, 11), (12, 0, 12), (13, 1, 13), (14, 0, 14), (15, 1, 15), (16, 0, 16), (17, 1, 17), (18, 0, 18), (19, 1, 19), (20, 0, 20);
- ▶ Test case : select \* from Status WHERE isAvailable = 1;

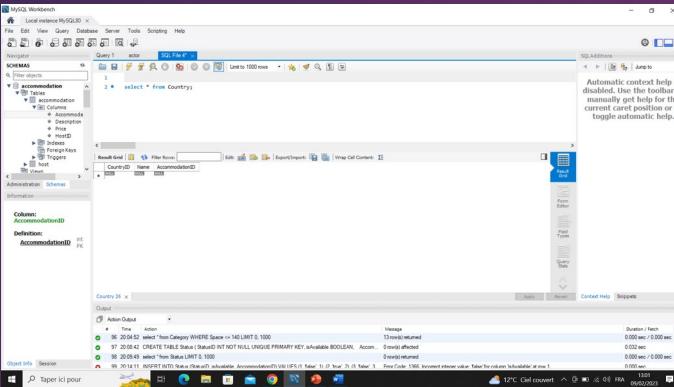
This screenshot shows the MySQL Workbench interface with the results of the test query. The query window contains the command:

```
select * from Status WHERE isAvailable = 1;
```

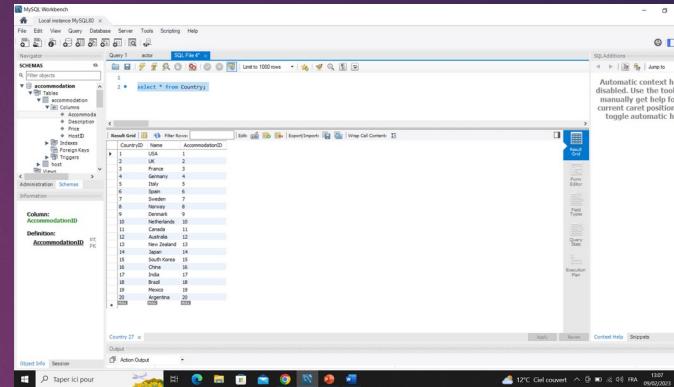
The results show the rows where 'isAvailable' is 1:

StatusID	isAvailable	AccommodationID
1	1	1
3	1	3
5	1	5
6	1	6
7	1	7
9	1	9
11	1	11
13	1	13
15	1	15
17	1	17
19	1	19
20	0	20

- ▶ CREATE TABLE Country (CountryID INT NOT NULL UNIQUE PRIMARY KEY, Name VARCHAR(500), AccommodationID INT REFERENCES Accommodation(AccommodationID));

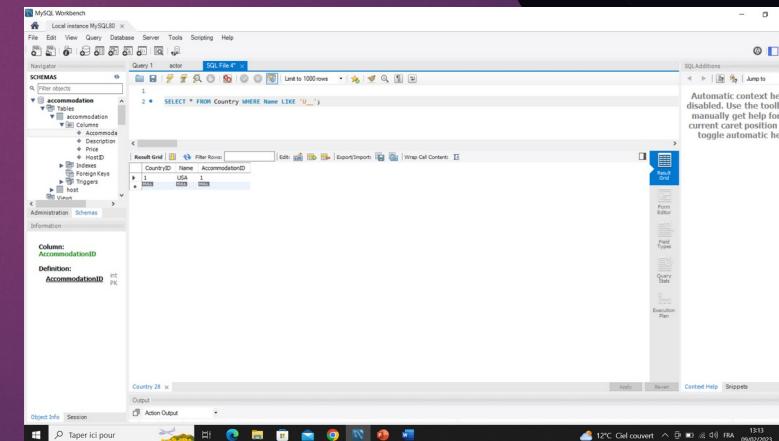


The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', there is a tree view for the 'accommodation' schema. A new table named 'Country' is being created. The table structure includes a primary key 'CountryID' (INT) and a column 'Name' (VARCHAR(500)). A foreign key constraint 'AccommodationID' (INT) references the 'AccommodationID' column in the 'Accommodation' table. The 'Definition' pane shows the SQL code for the table creation.

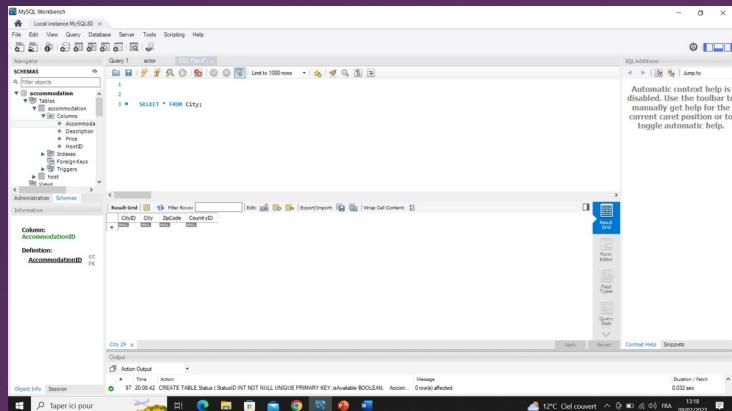
The screenshot shows the MySQL Workbench interface after running a query. The results grid displays 20 rows of country data, each with a unique 'CountryID' from 1 to 20 and a corresponding 'Name'. The columns are labeled 'CountryID' and 'Name'.

- ▶ INSERT INTO Country (CountryID, Name, AccommodationID) VALUES (1, 'USA', 1), (2, 'UK', 2), (3, 'France', 3), (4, 'Germany', 4), (5, 'Italy', 5), (6, 'Spain', 6), (7, 'Sweden', 7), (8, 'Norway', 8), (9, 'Denmark', 9), (10, 'Netherlands', 10), (11, 'Canada', 11), (12, 'Australia', 12), (13, 'New Zealand', 13), (14, 'Japan', 14), (15, 'South Korea', 15), (16, 'China', 16), (17, 'India', 17), (18, 'Brazil', 18), (19, 'Mexico', 19), (20, 'Argentina', 20);
- ▶ Test case : SELECT \* FROM Country WHERE Name LIKE 'U\_\_';



The screenshot shows the MySQL Workbench interface after running a query. The results grid displays 2 rows of country data where the name starts with 'U'. The first row is 'USA' with 'CountryID' 1 and 'AccommodationID' 1. The second row is 'UK' with 'CountryID' 2 and 'AccommodationID' 2.

- CREATE TABLE City (CityID INT NOT NULL UNIQUE PRIMARY KEY,City VARCHAR(500), ZipCode INT,CountryID INT REFERENCES Country(CountryID) );



The screenshot shows the MySQL Workbench interface with the City table populated with 20 rows of data. The columns are CityID, City, ZipCode, and CountryID. The data includes cities like New York, London, Marseille, etc., with their respective ZipCodes and CountryIDs.

CityID	City	ZipCode	CountryID
1	New York	12345	1
2	London	23456	2
3	Marseille	34567	3
4	Hambourg	45678	4
5	Venise	56789	5
6	Barcelona	67890	6
7	Umea	78901	7
8	Arendal	89012	8
9	Ribe	90123	9
10	La Haye	10124	10
11	Ottawa	11125	11
12	Sydney	12126	12
13	Napier	13127	13
14	Kobe	14128	14
15	Pusan	15129	15
16	Nankin	16130	16
17	Jaipur	17131	17
18	Manaus	18132	18
19	Mérida	19133	19
20	Salta	20144	20

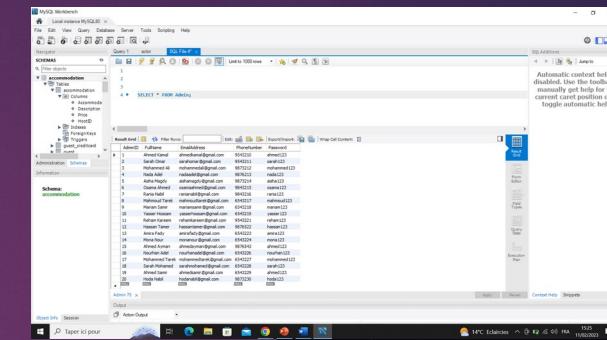
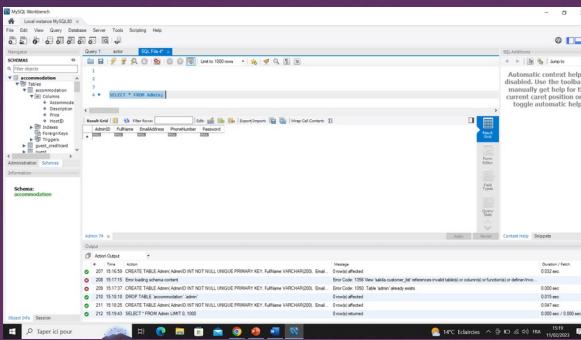
- INSERT INTO City (CityID, City, ZipCode, CountryID) VALUES (1, 'New York', 12345, 1), (2, 'London', 23456, 2), (3, 'Marseille', 34567, 3), (4, 'Hambourg', 45678, 4), (5, 'Venise', 56789, 5), (6, 'Barcelona', 67890, 6), (7, 'Umea', 78901, 7), (8, 'Arendal', 89012, 8), (9, 'Ribe', 90123, 9), (10, 'La Haye', 10124, 10), (11, 'Ottawa', 11125, 11), (12, 'Sydney', 12126, 12), (13, 'Napier', 13127, 13), (14, 'Kobe', 14128, 14), (15, 'Pusan', 15129, 15), (16, 'Nankin', 16130, 16), (17, 'Jaipur', 17131, 17), (18, 'Manaus', 18132, 18), (19, 'Mérida', 19133, 19), (20, 'Salta', 20144, 20);

- Test case : SELECT \* FROM City WHERE ZipCode BETWEEN 50000 AND 100000;

The screenshot shows the MySQL Workbench interface with the results of the SELECT query. The results table contains 5 rows of data where the ZipCode is between 50000 and 100000.

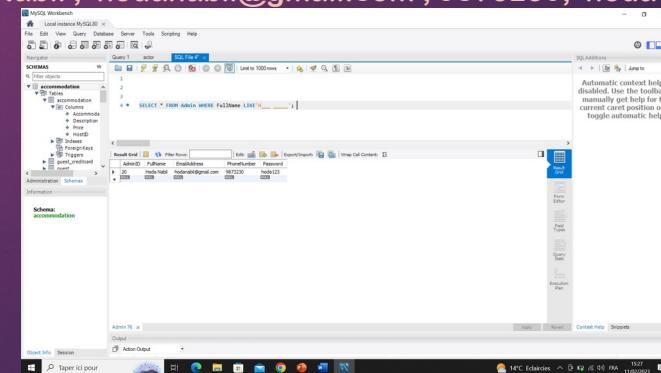
CityID	City	ZipCode	CountryID
5	Venise	56789	5
6	Barcelona	67890	6
7	Umea	78901	7
8	Arendal	89012	8
9	Ribe	90123	9

- ▶ CREATE TABLE Admin( AdminID INT NOT NULL UNIQUE PRIMARY KEY, FullName VARCHAR(200), EmailAddress CHAR(200), PhoneNumber INT, Password CHAR(100) NOT NULL );

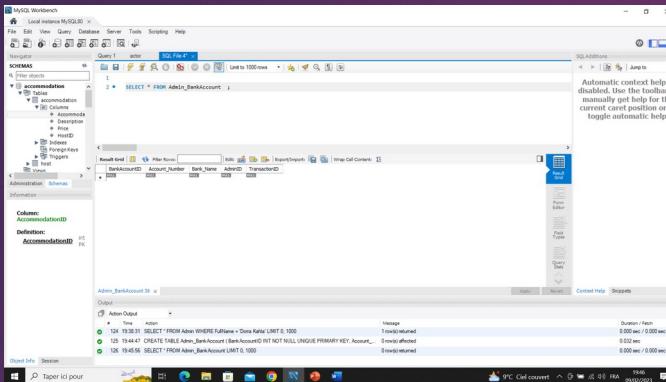


- ▶ INSERT INTO Admin (AdminID, FullName, EmailAddress, PhoneNumber, Password)VALUES(1, 'Ahmed Kamal', 'ahmedkamal@gmail.com', 9543210, 'ahmed123'),(2, 'Sarah Omar', 'sarahomar@gmail.com', 9543211, 'sarah123'),(3, 'Mohammed Ali', 'mohammedali@gmail.com', 9873212, 'mohammed123'),(4, 'Nada Adel', 'nadaadel@gmail.com', 9876213, 'nada123'),(5, 'Aisha Magdy', 'aishamagdy@gmail.com', 9873214, 'aisha123'),(6, 'Osama Ahmed', 'osamaahmed@gmail.com', 9843215, 'osama123'),(7, 'Rania Nabil', 'ranianabil@gmail.com', 9843216, 'rania123'),(8, 'Mahmoud Tarek', 'mahmoudtarek@gmail.com', 6543217, 'mahmoud123'),(9, 'Mariam Samir', 'mariamsamir@gmail.com', 6543218, 'mariam123'),(10, 'Yasser Hossam', 'yasserhossam@gmail.com', 6543219, 'yasser123'),(11, 'Reham Kareem', 'rehamkareem@gmail.com', 9543221, 'reham123'),(12, 'Hassan Tamer', 'hassantamer@gmail.com', 9876522, 'hassan123'),(13, 'Amira Fady', 'amirafady@gmail.com', 6543223, 'amira123'),(14, 'Mona Nour', 'monanour@gmail.com', 6543224, 'mona123'),(15, 'Ahmed Ayman', 'ahmedayman@gmail.com', 9876543, 'ahmed123'),(16, 'Nourhan Adel', 'nourhanadel@gmail.com', 6543226, 'nourhan123'),(17, 'Mohammed Tarek', 'mohammedtarek@gmail.com', 6543227, 'mohammed123'),(18, 'Sarah Mohamed', 'sarahmohamed@gmail.com', 6543228, 'sarah123'),(19, 'Ahmed Samir', 'ahmedsamir@gmail.com', 6543229, 'ahmed123'),(20, 'Hoda Nabil', 'hodanabil@gmail.com', 9873230, 'hoda123');

- ▶ Test case :SELECT \* FROM Admin WHERE FullName LIKE'H\_\_\_\_\_';



- ▶ CREATE TABLE Admin\_BankAccount (BankAccountId INT NOT NULL UNIQUE PRIMARY KEY, Account\_Number INT NOT NULL UNIQUE, Bank\_Name CHAR(80), AdminID INT REFERENCES Admin/AdminID), TransactionID INT REFERENCES Transaction(TransactionID);



The screenshot shows the MySQL Workbench interface with the results of a SELECT \* query on the Admin\_BankAccount table. The results grid displays the following columns: BankAccountId, Account\_Number, Bank\_Name, AdminID, and TransactionID. The data consists of 20 rows, each representing a different bank account with unique values for all columns.

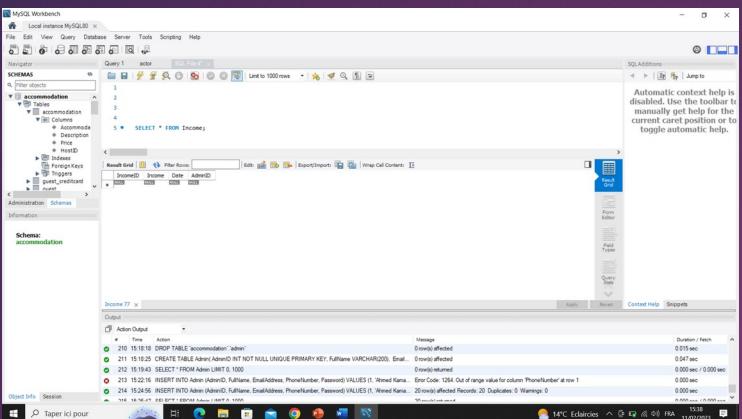
BankAccountId	Account_Number	Bank_Name	AdminID	TransactionID
1	18901231	'ABC Bank'	1	
2	18901232	'oTTAWA Bank'	2	2
3	18901233	'ATB Bank'	3	3
4	18901234	'DEF Bank'	4	4
5	18901235	'FAHED Bank'	5	5
6	18901236	'GHI Bank'	6	6
7	18901237	'SAID Bank'	7	7
8	18901238	'JKL Bank'	8	8
9	18901239	'Emarr Bank'	9	9
10	18901240	'MNO Bank'	10	10
11	18901244	'QATAR Bank'	11	11
12	18901255	'PQR Bank'	12	12
13	18901266	'TUNIS Bank'	13	13
14	18901277	'STU Bank'	14	14
15	18901288	'VWX Bank'	15	15
16	18901299	'WVN Bank'	16	16
17	18901300	'ZAYTOUNA Bank'	17	17
18	17890174	'YZ Bank'	18	18
19	1234572	'UTB Bank'	19	19
20	1234567	'ABCDEF Bank'	20	20

- ▶ INSERT INTO Admin\_BankAccount (BankAccountId, Account\_Number, Bank\_Name, AdminID, TransactionID)VALUES(1, 18901231, 'ABC Bank', 1, 1),(2, 18901232, 'oTTAWA Bank', 2, 2),(3, 18901233, 'ATB Bank', 3, 3),(4, 18901234, 'DEF Bank', 4, 4),(5, 18901235, 'FAHED Bank', 5, 5),(6, 18901236, 'GHI Bank', 6, 6),(7, 18901237, 'SAID Bank', 7, 7),(8, 18901238, 'JKL Bank', 8, 8),(9, 18901239, 'Emarr Bank', 9, 9),(10, 18901240, 'MNO Bank', 10, 10),(11, 18901244, 'QATAR Bank', 11, 11),(12, 18901255, 'PQR Bank', 12, 12),(13, 18901266, 'TUNIS Bank', 13, 13),(14, 18901277, 'STU Bank', 14, 14),(15, 18901288, 'VWX Bank', 15, 15),(16, 18901299, 'WVN Bank', 16, 16),(17, 18901300, 'ZAYTOUNA Bank', 17, 17),(18, 17890174, 'YZ Bank', 18, 18),(19, 1234572, 'UTB Bank', 19, 19),(20, 1234567, 'ABCDEF Bank', 20, 20);
- ▶ Test case : SELECT \* FROM Admin\_BankAccount WHERE Account\_Number > 1890000 ;

The screenshot shows the MySQL Workbench interface with the results of a SELECT query filtering by Account\_Number > 1890000. The results grid displays the following columns: BankAccountId, Account\_Number, Bank\_Name, AdminID, and TransactionID. The data consists of 18 rows, representing bank accounts with account numbers greater than 1890000.

BankAccountId	Account_Number	Bank_Name	AdminID	TransactionID
2	18901232	'oTTAWA Bank'	2	2
3	18901233	'ATB Bank'	3	3
4	18901234	'DEF Bank'	4	4
5	18901235	'FAHED Bank'	5	5
6	18901236	'GHI Bank'	6	6
7	18901237	'SAID Bank'	7	7
8	18901238	'JKL Bank'	8	8
9	18901239	'Emarr Bank'	9	9
10	18901240	'MNO Bank'	10	10
11	18901244	'QATAR Bank'	11	11
12	18901255	'PQR Bank'	12	12
13	18901266	'TUNIS Bank'	13	13
14	18901277	'STU Bank'	14	14
15	18901288	'VWX Bank'	15	15
16	18901299	'WVN Bank'	16	16
17	18901300	'ZAYTOUNA Bank'	17	17
18	17890174	'YZ Bank'	18	18
19	1234572	'UTB Bank'	19	19

- CREATE TABLE Income (IncomeID INT NOT NULL UNIQUE PRIMARY KEY, Income REAL, Date DATE, AdminID INT REFERENCES Admin(AdminID) );



The screenshot shows the MySQL Workbench interface with the 'Query' tab selected. A query window displays the SQL command:

```
SELECT * FROM Income;
```

The results grid shows the following data:

IncomeID	Income	Date	AdminID
1	1000	2022-05-10	1
2	2000	2022-05-12	2
3	1500	2022-06-08	3
4	2500	2022-06-11	4
5	3500	2022-07-05	5
6	1000	2022-07-12	6
7	6000	2022-08-08	7
8	2500	2022-08-12	8
9	7500	2022-09-05	9
10	3000	2022-09-12	10
11	8000	2022-09-18	11
12	5000	2022-10-05	12
13	9000	2022-10-09	13
14	6000	2022-10-12	14
15	7000	2022-10-18	15
16	2000	2022-10-22	16
17	9000	2023-01-05	17
18	5000	2023-01-12	18
19	6000	2023-01-18	19
20	7000	2023-02-12	20

- INSERT INTO Income (IncomeID, Income, Date, AdminID) VALUES(1, 1000, '2022-05-10', 1),(2, 2000, '2022-05-12', 2),(3, 1500, '2022-06-08', 3),(4, 2500, '2022-06-11', 4),(5, 3500, '2022-07-05', 5),(6, 1000, '2022-07-12', 6),(7, 6000, '2022-08-08', 7),(8, 2500, '2022-08-12', 8),(9, 7500, '2022-09-05', 9),(10, 3000, '2022-09-12', 10),(11, 8000, '2022-10-08', 11),(12, 5000, '2022-10-12', 12),(13, 9000, '2022-11-05', 13),(14, 6000, '2022-11-12', 14),(15, 6000, '2022-12-08', 15),(16, 7000, '2022-12-12', 16),(17, 9000, '2023-01-05', 17),(18, 5000, '2023-01-12', 18),(19, 6000, '2023-02-08', 19),(20, 7000, '2023-02-12', 20);

- Test case : SELECT \* FROM Income WHERE Income = 1000;

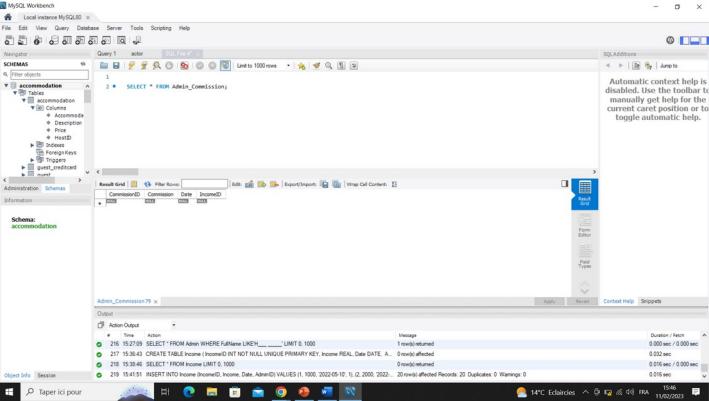
The screenshot shows the MySQL Workbench interface with the 'Query' tab selected. A query window displays the SQL command:

```
SELECT * FROM Income WHERE Income = 1000;
```

The results grid shows the following data:

IncomeID	Income	Date	AdminID
1	1000	2022-05-10	1
2	2000	2022-05-12	2

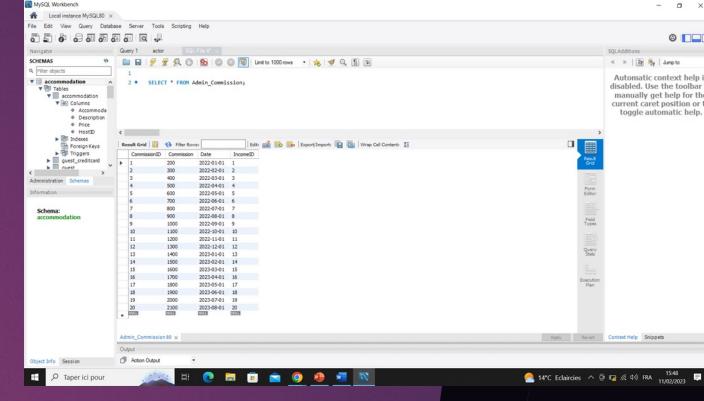
- CREATE TABLE Admin\_Commission (CommissionID INT NOT NULL UNIQUE PRIMARY KEY, Commission REAL, Date DATE, IncomeID INT REFERENCES Income(IncomeID));



The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel:** Shows the schema structure for the 'accommodation' database, including tables like 'Accommodation', 'Income', and 'Commission'.
- Central Area:** A query editor window titled 'Admin\_Commission79' containing the SQL command:
 

```
CREATE TABLE Admin_Commission (CommissionID INT NOT NULL UNIQUE PRIMARY KEY, Commission REAL, Date DATE, IncomeID INT REFERENCES Income(IncomeID));
```
- Bottom Area:** An 'Object Info' panel and a 'Session' tab.



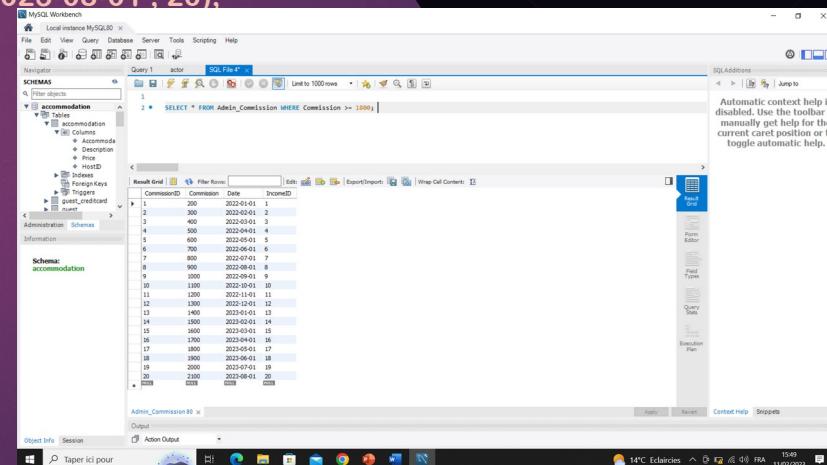
The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel:** Shows the schema structure for the 'accommodation' database, including tables like 'Accommodation', 'Income', and 'Commission'.
- Central Area:** A query editor window titled 'Admin\_Commission' containing the SQL command:
 

```
CREATE TABLE Admin_Commission (CommissionID INT NOT NULL UNIQUE PRIMARY KEY, Commission REAL, Date DATE, IncomeID INT REFERENCES Income(IncomeID));
```
- Bottom Area:** An 'Object Info' panel and a 'Session' tab.

- INSERT INTO Admin\_Commission (CommissionID, Commission, Date, IncomeID) VALUES (1, 200.0, '2022-01-01', 1),(2, 300.0, '2022-02-01', 2),(3, 400.0, '2022-03-01', 3),(4, 500.0, '2022-04-01', 4),(5, 600.0, '2022-05-01', 5),(6, 700.0, '2022-06-01', 6),(7, 800.0, '2022-07-01', 7),(8, 900.0, '2022-08-01', 8),(9, 1000.0, '2022-09-01', 9),(10, 1100.0, '2022-10-01', 10),(11, 1200.0, '2022-11-01', 11),(12, 1300.0, '2022-12-01', 12),(13, 1400.0, '2023-01-01', 13),(14, 1500.0, '2023-02-01', 14),(15, 1600.0, '2023-03-01', 15),(16, 1700.0, '2023-04-01', 16),(17, 1800.0, '2023-05-01', 17),(18, 1900.0, '2023-06-01', 18),(19, 2000.0, '2023-07-01', 19),(20, 2100.0, '2023-08-01', 20);

- Test case : SELECT \* FROM Admin\_Commission WHERE Commission >= 1800;

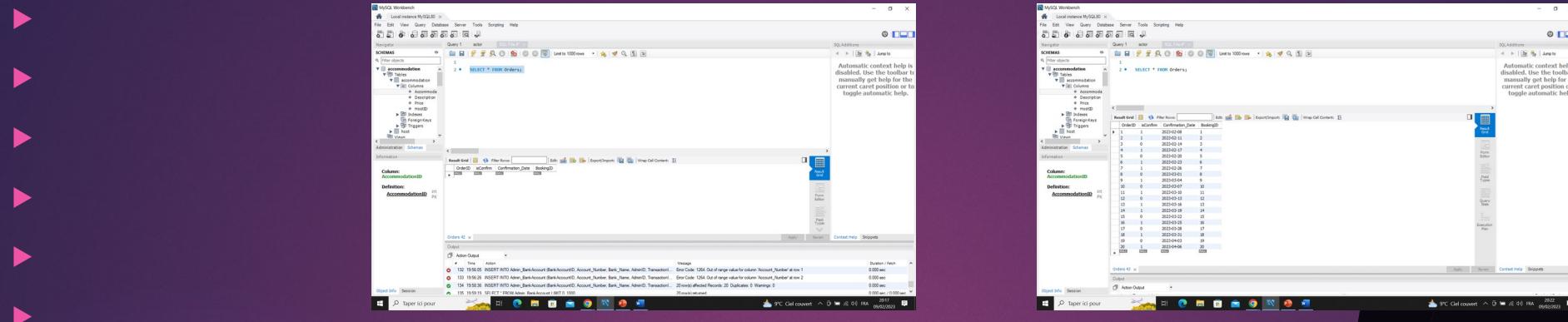


The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel:** Shows the schema structure for the 'accommodation' database, including tables like 'Accommodation', 'Income', and 'Commission'.
- Central Area:** A query editor window titled 'Admin\_Commission 80' containing the SQL command:
 

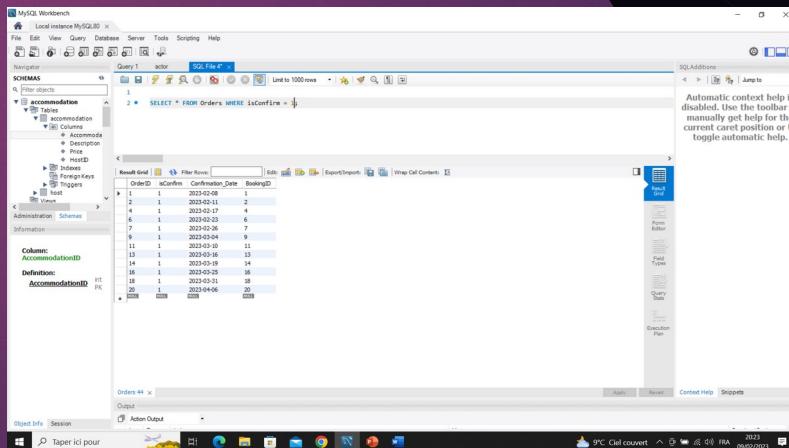
```
SELECT * FROM Admin_Commission WHERE Commission >= 1800;
```
- Bottom Area:** An 'Object Info' panel and a 'Session' tab.

- ▶ CREATE TABLE Orders (OrderID INT NOT NULL UNIQUE PRIMARY KEY, isConfirm BOOLEAN, Confirmation\_Date DATE, BookingID INT REFERENCES Booking(BookingID) );



- ▶ INSERT INTO Orders (OrderID, isConfirm, Confirmation\_Date, BookingID)VALUES (1, 1, '2023-02-08', 1), (2, 1, '2023-02-11', 2), (3, 0, '2023-02-14', 3), (4, 1, '2023-02-17', 4), (5, 0, '2023-02-20', 5), (6, 1, '2023-02-23', 6), (7, 1, '2023-02-26', 7), (8, 0, '2023-03-01', 8), (9, 1, '2023-03-04', 9), (10, 0, '2023-03-07', 10), (11, 1, '2023-03-10', 11), (12, 0, '2023-03-13', 12), (13, 1, '2023-03-16', 13), (14, 1, '2023-03-19', 14), (15, 0, '2023-03-22', 15), (16, 1, '2023-03-25', 16), (17, 0, '2023-03-28', 17), (18, 1, '2023-03-31', 18), (19, 0, '2023-04-03', 19), (20, 1, '2023-04-06', 20);

- ▶ Test case : SELECT \* FROM Orders WHERE isConfirm = 1;



- CREATE TABLE Admin\_CreditCard (CardID INT NOT NULL UNIQUE PRIMARY KEY, Card\_Number INT, AdminID INT REFERENCES Admin(AdminID), InvoiceID INT REFERENCES Invoice(InvoiceID));

The screenshot shows two instances of MySQL Workbench. The left instance displays the SQL Editor with the command:

```
CREATE TABLE Admin_CreditCard (CardID INT NOT NULL UNIQUE PRIMARY KEY, Card_Number INT, AdminID INT REFERENCES Admin(AdminID), InvoiceID INT REFERENCES Invoice(InvoiceID));
```

The right instance shows the results of an INSERT operation into the Admin\_Invoice table, which has populated the Admin\_CreditCard table with 20 rows of sample data.

CardID	Card_Number	AdminID	InvoiceID
1	11111111	1	1
2	22222222	2	2
3	33333333	3	3
4	44444444	4	4
5	55555555	5	5
6	66666666	6	6
7	77777777	7	7
8	88888888	8	8
9	99999999	9	9
10	10101010	10	10
11	11111111	11	11
12	12121212	12	12
13	13131313	13	13
14	14141414	14	14
15	15151515	15	15
16	16161616	16	16
17	17171717	17	17
18	18181818	18	18
19	19191919	19	19
20	20202020	20	20

- INSERT INTO Admin\_CreditCard (CardID, Card\_Number, AdminID, InvoiceID) VALUES (1, 11111111, 1, 1), (2, 22222222, 2, 2), (3, 33333333, 3, 3), (4, 44444444, 4, 4), (5, 55555555, 5, 5), (6, 66666666, 6, 6), (7, 77777777, 7, 7), (8, 88888888, 8, 8), (9, 99999999, 9, 9), (10, 10101010, 10, 10), (11, 11111111, 11, 11), (12, 12121212, 12, 12), (13, 13131313, 13, 13), (14, 14141414, 14, 14), (15, 15151515, 15, 15), (16, 16161616, 16, 16), (17, 17171717, 17, 17), (18, 18181818, 18, 18), (19, 19191919, 19, 19), (20, 20202020, 20, 20);
- Test case : SELECT \* FROM Admin\_CreditCard WHERE CardID != 10 AND Card\_Number != 55555555 ;

The screenshot shows MySQL Workbench with the following SQL query in the SQL Editor:

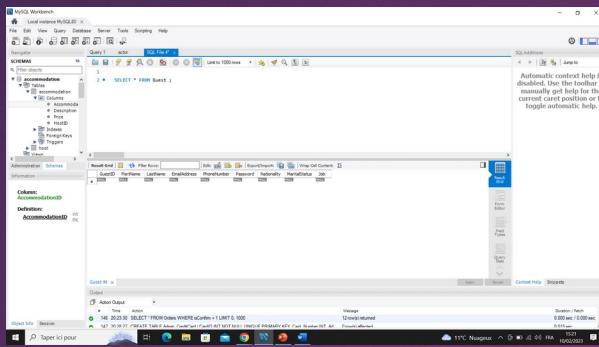
```
SELECT * FROM Admin_CreditCard WHERE CardID != 10 AND Card_Number != 55555555 ;
```

The results pane displays the filtered data, excluding the row where CardID is 10 and Card\_Number is 55555555.

CardID	Card_Number	AdminID	InvoiceID
1	11111111	1	1
2	22222222	2	2
3	33333333	3	3
4	44444444	4	4
5	55555555	5	5
6	66666666	6	6
7	77777777	7	7
8	88888888	8	8
9	99999999	9	9
11	11111111	11	11
12	12121212	12	12
13	13131313	13	13
14	14141414	14	14
15	15151515	15	15
16	16161616	16	16
17	17171717	17	17
18	18181818	18	18
19	19191919	19	19
20	20202020	20	20

- CREATE TABLE Guest (GuestID INT NOT NULL UNIQUE PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), EmailAddress VARCHAR(100) UNIQUE, PhoneNumber INT, Password CHAR(100) NOT NULL, Nationality CHAR(200), MaritalStatus VARCHAR(40), Job CHAR(60) );

- 
- 
- 
- 
- 
- 



The screenshot shows the MySQL Workbench interface with the 'SCHEMAS' tree and 'Guest' table selected. A 'Query' tab is active, displaying the following SQL code:

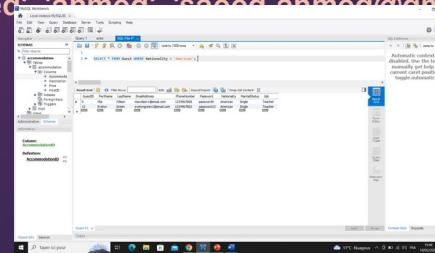
```
SELECT * FROM Guest;
```

The results grid shows 20 rows of data, each representing a guest with columns: GuestID, FirstName, LastName, EmailAddress, PhoneNumber, Password, Nationality, MaritalStatus, and Job.

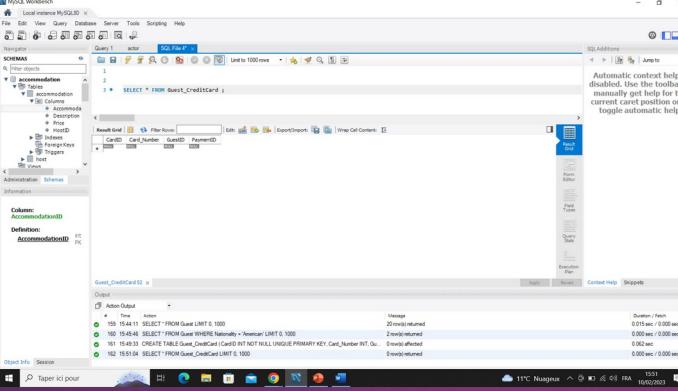
GuestID	FirstName	LastName	EmailAddress	PhoneNumber	Password	Nationality	MaritalStatus	Job
1	ahmed	abdullah	ahmed.abdullah@gmail.com	123456789	password1	Egyptian	Married	Doctor
2	muhammad	ahmed	muhammad.ahmed@gmail.com	987654321	password2	Moroccan	Single	Engineer
3	ali	muhammad	ali.muhammad@gmail.com	111111111	password3	Tunisian	Married	Teacher
4	nour	abdulrahman	nour.abdulrahman@gmail.com	222222222	password4	Algerian	Single	Lawyer
5	saeed	muhammad	saeed.muhammad@gmail.com	333333333	password5	Sudanese	Married	Journalist
6	osama	ali	osama.ali@gmail.com	444444444	password6	Libyan	Single	Architect
7	Ava	Miller	avamiller1@email.com	1234567896	password7	British	Single	Engineer
8	Isabella	Davis	isabelladavis1@email.com	1234567897	password8	Australian	Married	Nurse
9	Mia	Wilson	miawilson1@email.com	1234567898	password9	American	Single	Teacher
10	Charlotte	Wood	charlottewood1@email.com	1234567899	password10	Canadian	Married	Doctor
11	Amelia	Hill	ameliahill1@email.com	1234567800	password11	British	Single	Engineer
12	Harper	Lewis	harperlewis1@email.com	1234567801	password12	Australian	Married	Nurse
13	Evelyn	Green	evelyngreen1@email.com	1234567802	password13	American	Single	Teacher
14	Abigail	Adams	abigailadams1@email.com	1234567803	password14	Canadian	Married	Doctor
15	Emily	Baker	emilybaker1@email.com	1234567804	password15	British	Single	Engineer
16	Elizabeth	Barnes	elizabethbarnes1@email.com	1234567805	password16	Australian	Married	Nurse
17	muhammad	lahmer	muhammad.lahmer@gmail.com	888888888	password17	Ghanaian	Single	Painter
18	ali	abdulrahman	ali.abdulrahman@gmail.com	999999999	password18	Cameroonian	Married	Scientist
19	nour	muhammad	nour.muhammad@gmail.com	101010101	password19	Kenyan	Single	Programmer
20	saeed	ahmed	saeed.ahmed@gmail.com	202020202	password20	Zimbabwean	Married	Banker

- INSERT INTO Guest (GuestID, FirstName, LastName, EmailAddress, PhoneNumber, Password, Nationality, MaritalStatus, Job)VALUES (1, 'ahmed', 'abdullah', 'ahmed.abdullah@gmail.com', 123456789, 'password1', 'Egyptian', 'Married', 'Doctor'),(2, 'muhammad', 'ahmed', 'muhammad.ahmed@gmail.com', 987654321, 'password2', 'Moroccan', 'Single', 'Engineer'),(3, 'ali', 'muhammad', 'ali.muhammad@gmail.com', 111111111, 'password3', 'Tunisian', 'Married', 'Teacher'),(4, 'nour', 'abdulrahman', 'nour.abdulrahman@gmail.com', 222222222, 'password4', 'Algerian', 'Single', 'Lawyer'),(5, 'saeed', 'muhammad', 'saeed.muhammad@gmail.com', 333333333, 'password5', 'Sudanese', 'Married', 'Journalist'),(6, 'osama', 'ali', 'osama.ali@gmail.com', 444444444, 'password6', 'Libyan', 'Single', 'Architect'),(7, 'Ava', 'Miller', 'avamiller1@email.com', 1234567896, 'password7', 'British', 'Single', 'Engineer'),(8, 'Isabella', 'Davis', 'isabelladavis1@email.com', 1234567897, 'password8', 'Australian', 'Married', 'Nurse'),(9, 'Mia', 'Wilson', 'miawilson1@email.com', 1234567898, 'password9', 'American', 'Single', 'Teacher'),(10, 'Charlotte', 'Wood', 'charlottewood1@email.com', 1234567899, 'password10', 'Canadian', 'Married', 'Doctor'),(11, 'Amelia', 'Hill', 'ameliahill1@email.com', 1234567800, 'password11', 'British', 'Single', 'Engineer'),(12, 'Harper', 'Lewis', 'harperlewis1@email.com', 1234567801, 'password12', 'Australian', 'Married', 'Nurse'),(13, 'Evelyn', 'Green', 'evelyngreen1@email.com', 1234567802, 'password13', 'American', 'Single', 'Teacher'),(14, 'Abigail', 'Adams', 'abigailadams1@email.com', 1234567803, 'password14', 'Canadian', 'Married', 'Doctor'),(15, 'Emily', 'Baker', 'emilybaker1@email.com', 1234567804, 'password15', 'British', 'Single', 'Engineer'),(16, 'Elizabeth', 'Barnes', 'elizabethbarnes1@email.com', 1234567805, 'password16', 'Australian', 'Married', 'Nurse'),(17, 'muhammad', 'lahmer', 'muhammad.lahmer@gmail.com', 888888888, 'password17', 'Ghanaian', 'Single', 'Painter'),(18, 'ali', 'abdulrahman', 'ali.abdulrahman@gmail.com', 999999999, 'password18', 'Cameroonian', 'Married', 'Scientist'),(19, 'nour', 'muhammad', 'nour.muhammad@gmail.com', 101010101, 'password19', 'Kenyan', 'Single', 'Programmer'),(20, 'saeed', 'ahmed', 'saeed.ahmed@gmail.com', 202020202, 'password20', 'Zimbabwean', 'Married', 'Banker');

- Test case : SELECT \* FROM Guest WHERE Nationality = 'American';



- ▶ CREATE TABLE Guest\_CreditCard (CardID INT NOT NULL UNIQUE PRIMARY KEY, Card\_Number INT, GuestID INT REFERENCES Guest(GuestID), PaymentID INT REFERENCES Payment( PaymentID ) );

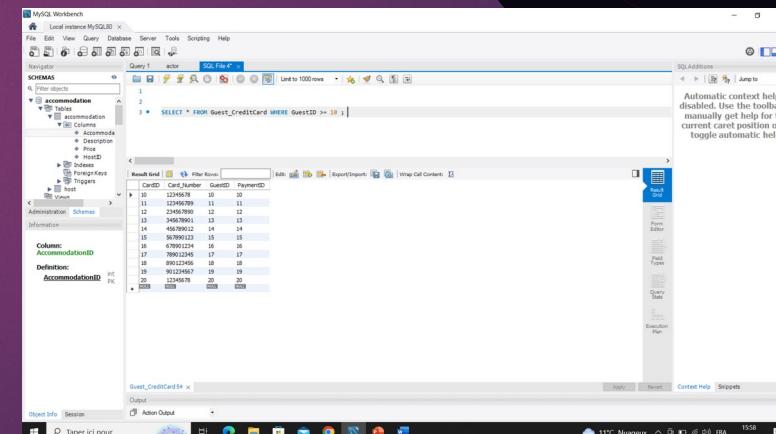


The screenshot shows the MySQL Workbench interface with two panes. The left pane displays the schema structure for the 'accommodation' table, specifically the 'Guest\_CreditCard' table definition. The right pane shows the results of a SELECT query on the same table, displaying 20 rows of data.

CardID	Card_Number	GuestID	PaymentID
1	1245623456	1	1
2	220123456	2	2
3	345678901	3	
4	456789012	4	4
5	567890123	5	5
6	678901234	6	6
7	789012345	7	7
8	890123456	8	8
9	901234567	9	9
10	912345678	10	10
11	123456789	11	11
12	123456790	12	12
13	134567901	13	13
14	145679012	14	14
15	156790123	15	15
16	167890123	16	
17	178901234	17	17
18	189012345	18	18
19	199012345	19	19
20	223456789	20	20

- ▶ INSERT INTO Guest\_CreditCard (CardID, Card\_Number, GuestID, PaymentID) VALUES (1, 1245623456, 1, 1), (2, 220123456, 2, 2), (3, 345678901, 3, 3), (4, 456789012, 4, 4), (5, 567890123, 5, 5), (6, 678901234, 6, 6), (7, 789012345, 7, 7), (8, 890123456, 8, 8), (9, 901234567, 9, 9), (10, 012345678, 10, 10), (11, 123456789, 11, 11), (12, 234567890, 12, 12), (13, 345678901, 13, 13), (14, 456789012, 14, 14), (15, 567890123, 15, 15), (16, 678901234, 16, 16), (17, 789012345, 17, 17), (18, 890123456, 18, 18), (19, 901234567, 19, 19), (20, 012345678, 20, 20);

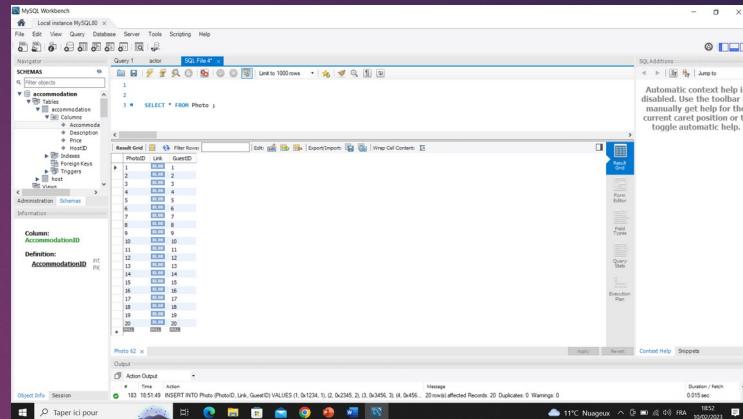
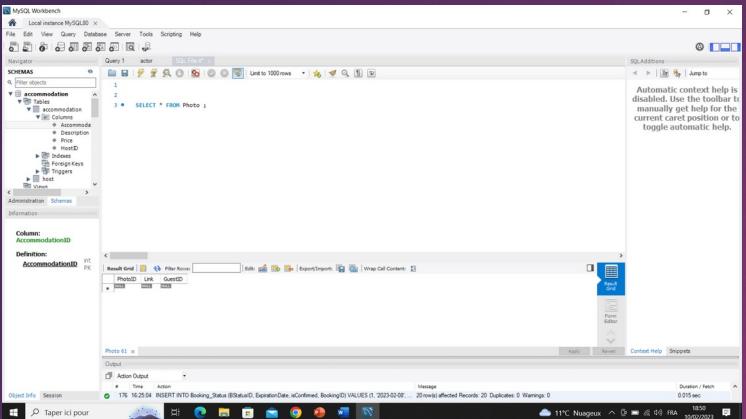
- ▶ Test case : SELECT \* FROM Guest\_CreditCard WHERE GuestID >= 10 ;



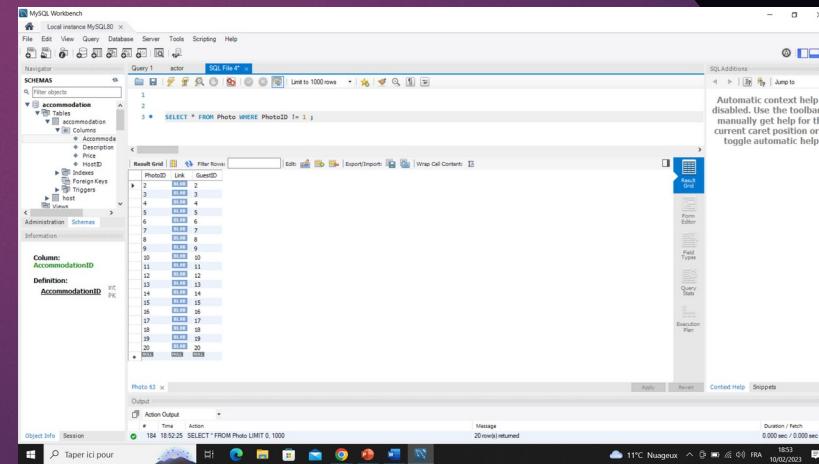
The screenshot shows the MySQL Workbench interface with two panes. The left pane displays the schema structure for the 'accommodation' table, specifically the 'Guest\_CreditCard' table definition. The right pane shows the results of a SELECT query on the table, filtering for GuestID values of 10 or higher.

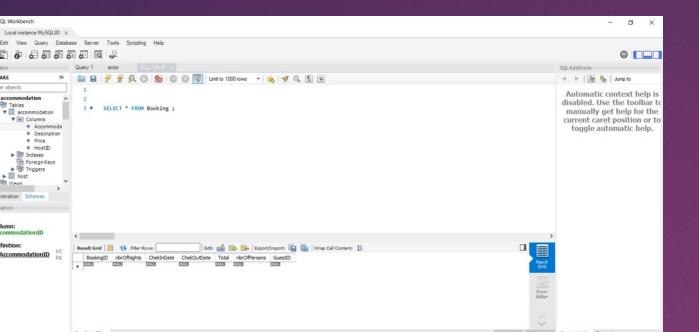
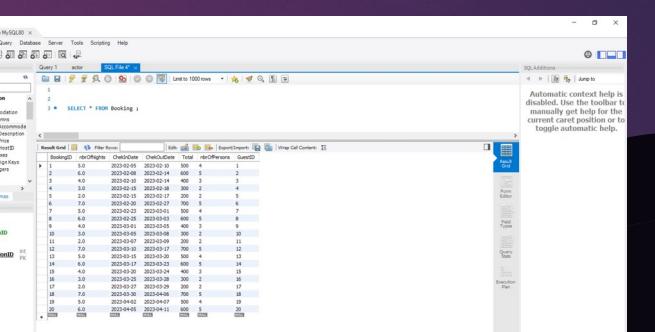
CardID	Card_Number	GuestID	PaymentID
10	123456789	10	10
11	123456790	11	11
12	134567901	12	12
13	145679012	13	13
14	145679012	14	14
15	156790123	15	15
16	167890123	16	
17	178901234	17	17
18	189012345	18	18
19	199012345	19	19
20	223456789	20	20

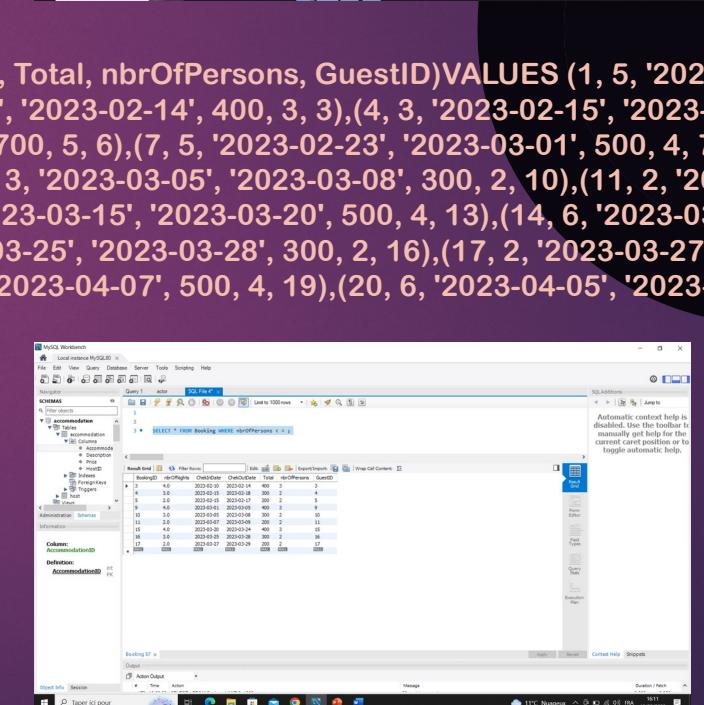
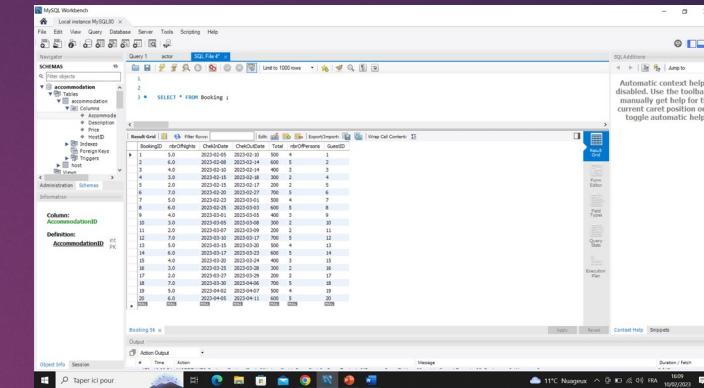
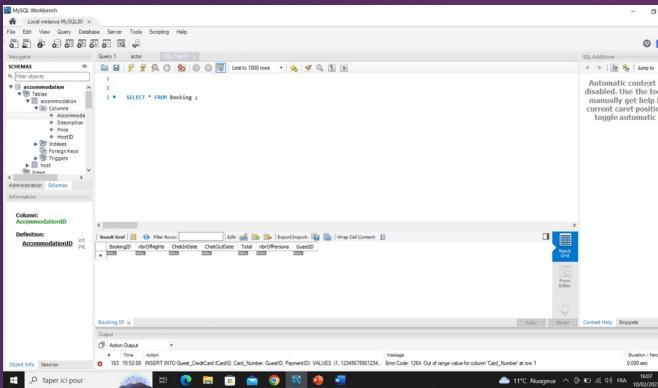
- ▶ CREATE TABLE Photo (PhotoID INT NOT NULL UNIQUE PRIMARY KEY,Link VARBINARY(1500), GuestID INT REFERENCES Guest(GuestID));



- ▶ INSERT INTO Photo (PhotoID, Link, GuestID) VALUES(1, 0x1234, 1),(2, 0x2345, 2),(3, 0x3456, 3),(4, 0x4567, 4),(5, 0x5678, 5),(6, 0x6789, 6),(7, 0x7890, 7),(8, 0x8901, 8),(9, 0x9012, 9),(10, 0x0123, 10),(11, 0x1234, 11),(12, 0x2345, 12),(13, 0x3456, 13),(14, 0x4567, 14),(15, 0x5678, 15),(16, 0x6789, 16),(17, 0x7890, 17),(18, 0x8901, 18),(19, 0x9012, 19),(20, 0x0123, 20);
- ▶ Test case : SELECT \* FROM Photo WHERE PhotoID != 1 ;

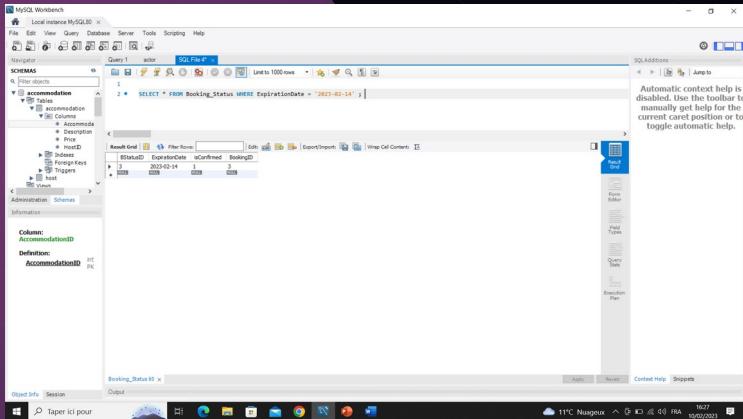
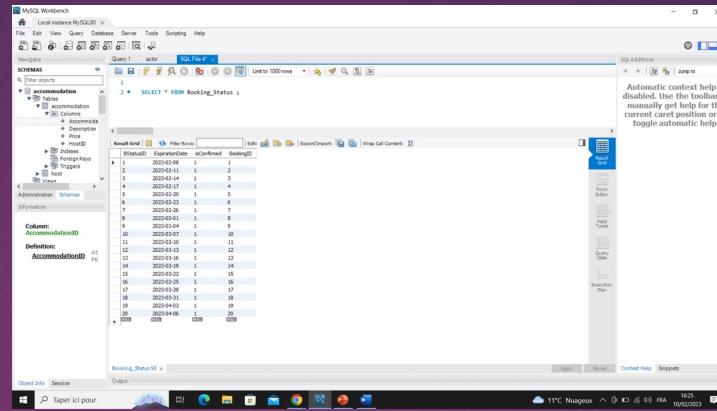


- CREATE TABLE Booking (BookingID INT NOT NULL UNIQUE PRIMARY KEY,nbrOfNights DECIMAL(2,1),ChekInDate DATE,ChekOutDate DATE,Total REAL,nbrOfPersons INT NOT NULL,GuestID INT REFERENCES Guest(GuestID) );
  - 
  - 
  - INSERT INTO Booking (BookingID, nbrOfNights, ChekInDate, ChekOutDate, Total, nbrOfPersons, GuestID)VALUES (1, 5, '2023-02-05', '2023-02-10', 500, 4, 1),(2, 6, '2023-02-08', '2023-02-14', 600, 5, 2),(3, 4, '2023-02-10', '2023-02-14', 400, 3, 3),(4, 3, '2023-02-15', '2023-02-18', 300, 2, 4),(5, 2, '2023-02-15', '2023-02-17', 200, 2, 5),(6, 7, '2023-02-20', '2023-02-27', 700, 5, 6),(7, 5, '2023-02-23', '2023-03-01', 500, 4, 7),(8, 6, '2023-02-25', '2023-03-03', 600, 5, 8),(9, 4, '2023-03-01', '2023-03-05', 400, 3, 9),(10, 3, '2023-03-05', '2023-03-08', 300, 2, 10),(11, 2, '2023-03-07', '2023-03-09', 200, 2, 11),(12, 7, '2023-03-10', '2023-03-17', 700, 5, 12),(13, 5, '2023-03-15', '2023-03-20', 500, 4, 13),(14, 6, '2023-03-17', '2023-03-23', 600, 5, 14),(15, 4, '2023-03-20', '2023-03-24', 400, 3, 15),(16, 3, '2023-03-25', '2023-03-28', 300, 2, 16),(17, 2, '2023-03-27', '2023-03-29', 200, 2, 17),(18, 7, '2023-03-30', '2023-04-06', 700, 5, 18),(19, 5, '2023-04-02', '2023-04-07', 500, 4, 19),(20, 6, '2023-04-05', '2023-04-11', 600, 5, 20);
  - Test case : SELECT \* FROM Booking WHERE nbrOfPersons < 4 ;

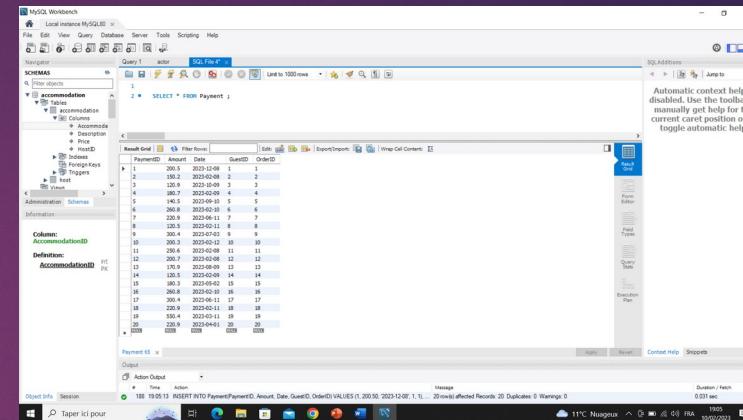
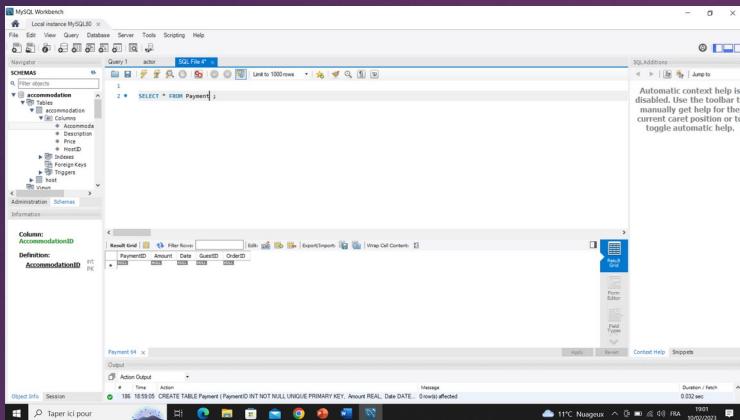


- CREATE TABLE Booking\_Status (BStatusID INT NOT NULL UNIQUE PRIMARY KEY,ExpirationDate DATE,isConfirmed BOOLEAN,BookingID INT REFERENCES Booking(BookingID) );

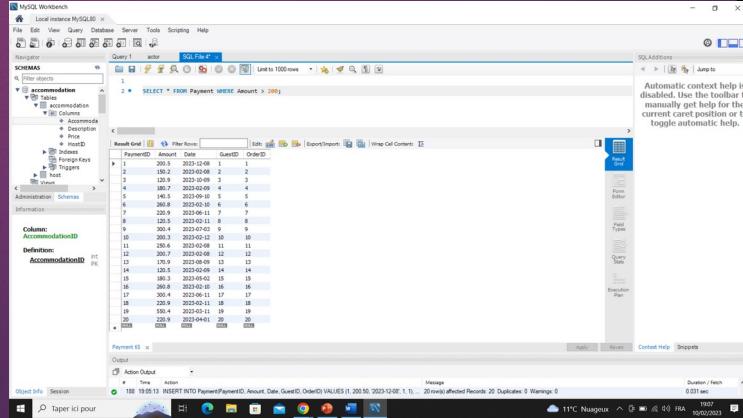
BStatusID	ExpirationDate	isConfirmed	BookingID
1	2023-02-08	1	1
2	2023-02-11	1	2
3	2023-02-14	1	3
4	2023-02-17	1	4
5	2023-02-20	1	5
6	2023-02-23	1	6
7	2023-02-26	1	7
8	2023-03-01	1	8
9	2023-03-04	1	9
10	2023-03-07	1	10
11	2023-03-10	1	11
12	2023-03-13	1	12
13	2023-03-16	1	13
14	2023-03-19	1	14
15	2023-03-22	1	15
16	2023-03-25	1	16
17	2023-03-28	1	17
18	2023-03-31	1	18
19	2023-04-03	1	19
20	2023-04-06	1	20



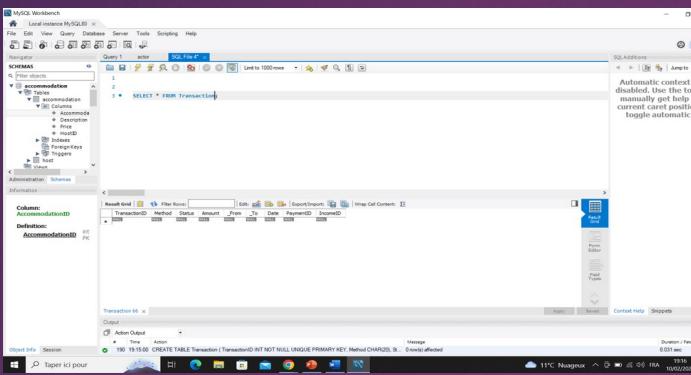
- CREATE TABLE Payment (PaymentID INT NOT NULL UNIQUE PRIMARY KEY, Amount REAL, Date DATE, GuestID INT REFERENCES Guest(GuestID), OrderID INT REFERENCES Orders(OrderID) );



- INSERT INTO Payment(PaymentID, Amount, Date, GuestID, OrderID)VALUES(1, 200.50, '2023-12-08', 1, 1),(2, 150.20, '2023-02-08', 2, 2),(3, 120.90, '2023-10-09', 3, 3),(4, 180.70, '2023-02-09', 4, 4),(5, 140.50, '2023-09-10', 5, 5),(6, 260.80, '2023-02-10', 6, 6),(7, 220.90, '2023-06-11', 7, 7),(8, 120.50, '2023-02-11', 8, 8),(9, 300.40, '2023-07-03', 9, 9),(10, 200.30, '2023-02-12', 10, 10),(11, 250.60, '2023-02-08', 11, 11),(12, 200.70, '2023-02-08', 12, 12),(13, 170.90, '2023-08-09', 13, 13),(14, 120.50, '2023-02-09', 14, 14),(15, 180.30, '2023-05-2', 15, 15),(16, 260.80, '2023-02-10', 16, 16),(17, 300.40, '2023-06-11', 17, 17),(18, 220.90, '2023-02-11', 18, 18),(19, 550.40, '2023-03-11', 19, 19),(20, 220.90, '2023-04-01', 20, 20);
- Test case : SELECT \* FROM Payment WHERE Amount > 200;



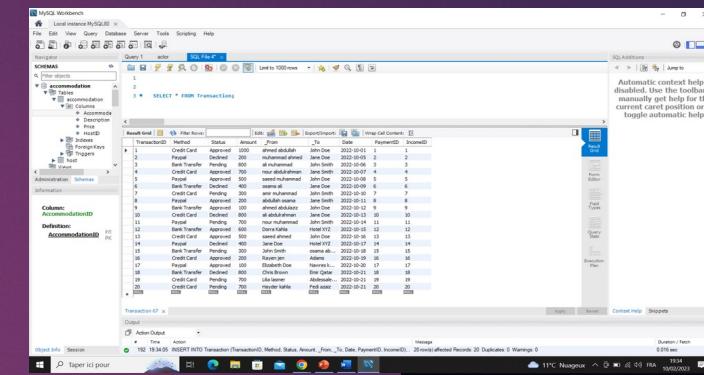
- ▶ CREATE TABLE Transaction (TransactionID INT NOT NULL UNIQUE PRIMARY KEY, Method CHAR (20), Status CHAR (20), Amount REAL, \_From VARCHAR(100), \_To VARCHAR(100), Date DATE, PaymentID INT REFERENCES Payment(PaymentID), IncomeID INT REFERENCES Income(IncomeID));



The screenshot shows the MySQL Workbench interface. A query window displays the SQL command for creating the Transaction table:

```
CREATE TABLE Transaction (TransactionID INT NOT NULL UNIQUE PRIMARY KEY, Method CHAR(20), Status CHAR(20), Amount REAL, _From VARCHAR(100), _To VARCHAR(100), Date DATE, PaymentID INT REFERENCES Payment(PaymentID), IncomeID INT REFERENCES Income(IncomeID));
```

The transaction table has been successfully created, as indicated by the message "1 row in set (0.00 sec)".



The screenshot shows the MySQL Workbench interface with a query window displaying the results of the SELECT \* FROM Transaction query. The results grid shows 20 rows of transaction data, including various payment methods like Credit Card, Paypal, and Bank Transfer, with amounts ranging from 100.00 to 800.00.

- ▶ INSERT INTO Transaction (TransactionID, Method, Status, Amount, \_From, \_To, Date, PaymentID, IncomeID) VALUES (1, 'Credit Card', 'Approved', 1000.0, 'ahmed abdullah', 'John Doe', '2022-10-01', 1, 1),(2, 'Paypal', 'Declined', 200.0, 'muhammad ahmed', 'Jane Doe', '2022-10-05', 2, 2),(3, 'Bank Transfer', 'Pending', 800.0, 'ali muhammad', 'John Smith', '2022-10-06', 3, 3),(4, 'Credit Card', 'Approved', 700.0, 'nour abdulrahman', 'Jane Smith', '2022-10-07', 4, 4),(5, 'Paypal', 'Approved', 500.0, 'saeed muhammad', 'John Doe', '2022-10-08', 5, 5),(6, 'Bank Transfer', 'Declined', 400.0, 'osama ali', 'Jane Doe', '2022-10-09', 6, 6),(7, 'Credit Card', 'Pending', 300.0, 'amir muhammad', 'John Smith', '2022-10-10', 7, 7),(8, 'Paypal', 'Approved', 200.0, 'abdullah osama', 'Jane Smith', '2022-10-11', 8, 8),(9, 'Bank Transfer', 'Approved', 100.0, 'ahmed abdulaziz', 'John Doe', '2022-10-12', 9, 9),(10, 'Credit Card', 'Declined', 800.0, 'ali abdulrahman', 'Jane Doe', '2022-10-13', 10, 10),(11, 'Paypal', 'Pending', 700.0, 'nour muhammad', 'John Smith', '2022-10-14', 11, 11),(12, 'Bank Transfer', 'Approved', 600.0, 'Dorra Kahla', 'Hotel XYZ', '2022-10-15', 12, 12),(13, 'Credit Card', 'Approved', 500.0, 'saeed ahmed', 'John Doe', '2022-10-16', 13, 13),(14, 'Paypal', 'Declined', 400.0, 'Jane Doe', 'Hotel XYZ', '2022-10-17', 14, 14),(15, 'Bank Transfer', 'Pending', 300.0, 'John Smith', 'osama abdullah', '2022-10-18', 15, 15),(16, 'Credit Card', 'Approved', 200.0, 'Rayen jen', 'Adams', '2022-10-19', 16, 16),(17, 'Paypal', 'Approved', 100.0, 'Elizabeth Doe', 'Nawres kahla', '2022-10-20', 17, 17),(18, 'Bank Transfer', 'Declined', 800.0, 'Chris Brown', 'Emir Qatar', '2022-10-21', 18, 18),(19, 'Credit Card', 'Pending', 700.0, 'Lilia lasmer', 'Abdessalem lahmer', '2022-10-21', 19, 19),(20, 'Credit Card', 'Pending', 700.0, 'Hayder kahla', 'Fedi azaiz', '2022-10-21', 20, 20);

- CREATE TABLE Currency (CurrencyID INT NOT NULL UNIQUE PRIMARY KEY,Currency CHAR(20),TransactionID INT REFERENCES Transaction(TransactionID) );

The left screenshot shows the MySQL Workbench interface with the SQL tab active. A query window displays the creation of the Currency table:

```
CREATE TABLE Currency (CurrencyID INT NOT NULL UNIQUE PRIMARY KEY,Currency CHAR(20),TransactionID INT REFERENCES Transaction(TransactionID) );
```

The right screenshot shows the MySQL Workbench interface with the Results tab active. A query window displays the data inserted into the Currency table:

CurrencyID	Currency	TransactionID
1	USD	1
2	EUR	2
3	GBP	3
4	JPY	4
5	CHF	5
6	CAD	6
7	AUD	7
8	NZD	8
9	CNY	9
10	INR	10
11	BRL	11
12	MXN	12
13	RUB	13
14	ZAR	14
15	HKD	15
16	SAR	16
17	AED	17
18	SGD	18
19	THB	19
20	TRY	20

INSERT INTO Currency (CurrencyID, Currency, TransactionID)VALUES(1, 'USD', 1),(2, 'EUR', 2),(3, 'GBP', 3),(4, 'JPY', 4),(5, 'CHF', 5),(6, 'CAD', 6),(7, 'AUD', 7),(8, 'NZD', 8),(9, 'CNY', 9),(10, 'INR', 10),(11, 'BRL', 11),(12, 'MXN', 12),(13, 'RUB', 13),(14, 'ZAR', 14),(15, 'HKD', 15),(16, 'SAR', 16),(17, 'AED', 17),(18, 'SGD', 18),(19, 'THB', 19),(20, 'TRY', 20);

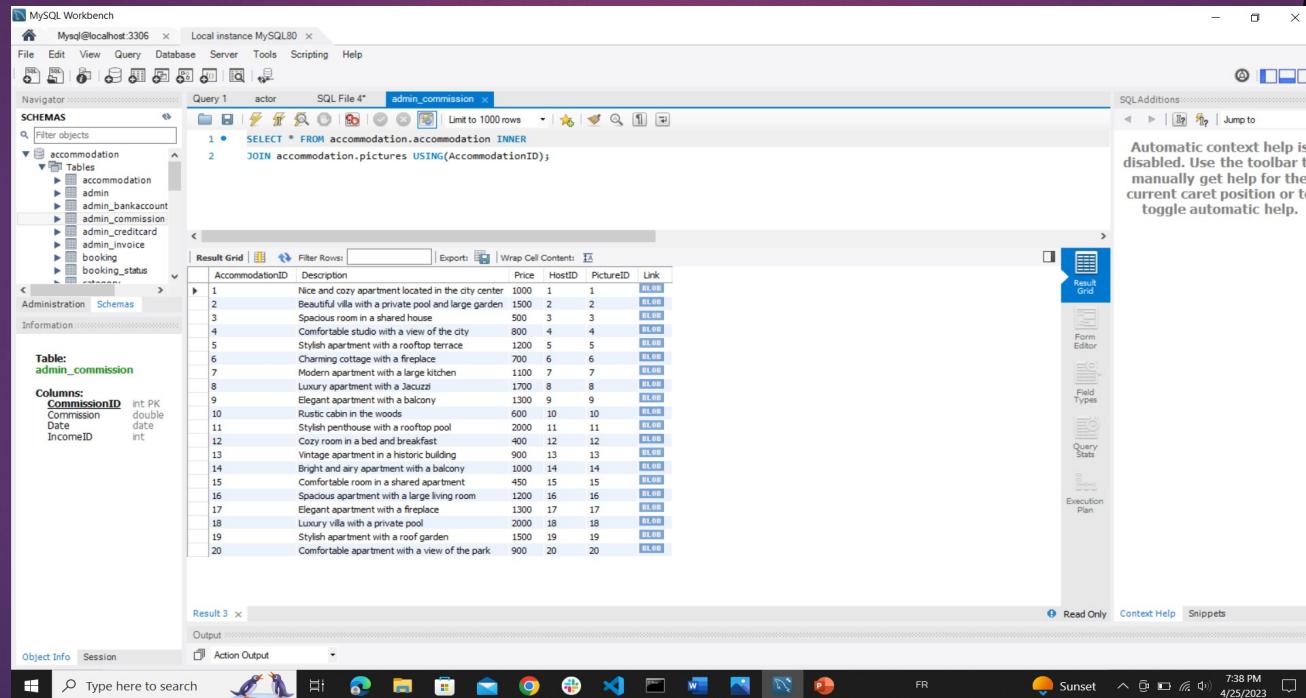
Test case: SELECT \* FROM Currency WHERE Currency = 'USD';

The screenshot shows the MySQL Workbench interface with the Results tab active. A query window displays the results of the test query:

CurrencyID	Currency	TransactionID
1	USD	1
200	USD	200

# Complex Test Cases

- ▶ `SELECT * FROM accommodation.accommodation INNERJOIN accommodation.pictures USING(AccommodationID);`



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:`1 • SELECT * FROM accommodation.accommodation INNER  
2 JOIN accommodation.pictures USING(AccommodationID);`

The results grid displays 20 rows of data from the joined tables. The columns are: AccommodationID, Description, Price, HostID, PictureID, and Link. The data includes various apartment descriptions and their corresponding details.

AccommodationID	Description	Price	HostID	PictureID	Link
1	Nice and cozy apartment located in the city center	1000	1	1	\$1.00
2	Beautiful villa with a private pool and large garden	1500	2	2	\$1.00
3	Spacious room in a shared house	500	3	3	\$1.00
4	Comfortable studio with a view of the city	800	4	4	\$1.00
5	Stylish apartment with a rooftop terrace	1200	5	5	\$1.00
6	Charming cottage with a fireplace	700	6	6	\$1.00
7	Modern apartment with a large kitchen	1100	7	7	\$1.00
8	Luxury apartment with a jacuzzi	1700	8	8	\$1.00
9	Elegant apartment with a balcony	1300	9	9	\$1.00
10	Rustic cabin in the woods	600	10	10	\$1.00
11	Stylish penthouse with a rooftop pool	2000	11	11	\$1.00
12	Cozy room in a bed and breakfast	400	12	12	\$1.00
13	Vintage apartment in a historic building	900	13	13	\$1.00
14	Bright and airy apartment with a balcony	1000	14	14	\$1.00
15	Comfortable room in a shared apartment	450	15	15	\$1.00
16	Spacious apartment with a large living room	1200	16	16	\$1.00
17	Elegant apartment with a fireplace	1300	17	17	\$1.00
18	Luxury villa with a private pool	2000	18	18	\$1.00
19	Stylish apartment with a roof garden	1500	19	19	\$1.00
20	Comfortable apartment with a view of the park	900	20	20	\$1.00

▶ SELECT \* FROM accommodation.booking INNERJOIN accommodation.booking\_status ON isConfirmed = 0;



The screenshot shows the MySQL Workbench interface. The left pane displays the database schema with the 'accommodation' schema selected, showing tables like booking, booking\_status, and admin\_commission. The central pane contains a query editor with the following SQL code:

```
1 • SELECT * FROM accommodation.booking INNER
2 JOIN accommodation.booking_status ON isConfirmed = 0;
```

The right pane shows the results grid with columns: BookingID, nbrOfNights, CheckInDate, CheckOutDate, Total, nbrOfPersons, GuestID, BStatusID, ExpirationDate, isConfirmed, and BookingID. Below the results, there is a note about context help being disabled and a sidebar with various tools and tabs.

▶ SELECT \* FROM accommodation.country LEFTJOIN accommodation.city ON country.CountryID = city.CountryID AND country.Name = 'USA';



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • SELECT * FROM accommodation.country LEFT
2 JOIN accommodation.city ON country.CountryID = city.CountryID AND country.Name = 'USA';
```

The results grid displays the following data:

CountryID	Name	AccommodationID	CityID	City	ZipCode	CountryID
1	USA	1	1	New York	12345	1
2	UK	2	HULL	HULL	HULL	HULL
3	France	3	HULL	HULL	HULL	HULL
4	Germany	4	HULL	HULL	HULL	HULL
5	Italy	5	HULL	HULL	HULL	HULL
6	Spain	6	HULL	HULL	HULL	HULL
7	Sweden	7	HULL	HULL	HULL	HULL
8	Norway	8	HULL	HULL	HULL	HULL
9	Denmark	9	HULL	HULL	HULL	HULL
10	Netherlands	10	HULL	HULL	HULL	HULL
11	Canada	11	HULL	HULL	HULL	HULL
12	Australia	12	HULL	HULL	HULL	HULL
13	New Zealand	13	HULL	HULL	HULL	HULL
14	Japan	14	HULL	HULL	HULL	HULL
15	South Korea	15	HULL	HULL	HULL	HULL
16	China	16	HULL	HULL	HULL	HULL
17	India	17	HULL	HULL	HULL	HULL
18	Brazil	18	HULL	HULL	HULL	HULL
19	Mexico	19	HULL	HULL	HULL	HULL
20	Argentina	20	HULL	HULL	HULL	HULL

The context help panel on the right side of the interface provides information about automatic context help being disabled and how to manually get help for the current caret position or to toggle automatic help.

▶ SELECT \* FROM accommodation.transaction INNERJOIN accommodation.currency ON transaction.Date = '2022-10-01' AND currency.Currency = 'USD';



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL query:

```
1 •  SELECT * FROM accommodation.transaction INNERJOIN
2 JOIN accommodation.currency ON transaction.Date = '2022-10-01' AND currency.Currency = 'USD';
```
- Result Grid:** Shows the resulting data from the query:

TransactionID	Method	Status	Amount	_From	_To	Date	PaymentID	IncomeID	CurrencyID	Currency	TransactionID
1	Credit Card	Approved	1000	ahmed abdullah	John Doe	2022-10-01	1	1	1	USD	1
- Table Information:** Shows details for the `admin_commission` table:
  - Table:** admin\_commission
  - Columns:**
    - `CommissionID` int PK
    - `Commission` double
    - `Date` date
    - `IncomeID` int
- Right Panel:** A tooltip message states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

# Database Management for Airbnb

Airbnb is an online marketplace for short-term rentals of homes, apartments, and other lodgings. In this project, I tried to develop a database for storing and processing information related to the Airbnb use case.

The developed database for the Airbnb-like platform serves as the backbone for managing and storing data related to user profiles, accommodations, bookings, and ratings. The latter is responsible for facilitating the transaction between hosts and guests by handling the booking process and managing payments. In one hand, The database enables hosts to create a textual and visual representation of their accommodations, and in the other hand, it enables guests to browse and book accommodations based on their preferences.

To ensure data integrity, constraints such as data type and data range were set up to prevent invalid data entry. The database is also supporting querying and reporting functionality, allowing for valuable insights to be extracted from the data. Proper documentation of the database schema and data dictionary was implemented to help other developers to understand the structure of the database and how to interact with it. In short, the database was developed with high focus on small details to ensure the efficient and effective functioning of it, facilitating smooth transactions between hosts and guests and enabling valuable insights to be extracted from the data.

## Entity Relationship Model

The entity relationship model (ERM) was built to describe the structure of the database, including the entities (tables), their attributes, and the relationships between them. Here is the main components of the database :

- Accommodations: includes information about each accommodation, such as its category, description, price, location and availability.
- Users: includes information about each user, such as their name, email address, phone number, bank account etc .
- Orders: includes information about each order, such as its status (if it is confirmed or not ), the confirmation date and the booking it belongs to.
- Transactions: includes information about each transaction, such as the amount of the transaction, the currency, the method used etc .
- The commission: This component was specifically implemented to accurately identify the commission token from both the guest and the host, ensuring efficient and error-free processing of commission payments.

We then defined the relationships between these entities. For example, each accommodation is associated with a user who owns it, and each transaction needs one payment to be processed and then forwarded to the company bank account .

## Database Management System

I chose to use MySQL as the database management system for this project. MySQL is a popular open-source database system that supports SQL as its basic language.

I defined the structure of the database in MySQL, including the creation of tables and their corresponding attributes. I also created dummy data to ensure that the database was appropriately populated.

## Normalization

I normalized the database in an appropriate way to ensure that only necessary data was stored. This involved breaking down larger tables into smaller ones and ensuring that each table had a primary key.

## Metadata

The database consists of 25 tables: Host, Pictures, Rating, Accommodation, HostCommission, Category, Status, Country, City, Street, Host BankAccount, Transaction, Currency, Admin, Admin BankAccount, Income, Orders, Admin CreditCard, Admin Commission, Guest, Booking, Payment, Photo, Guest CreditCard and Booking Status. Each table has 20 entries . The total size of the database is approximately 15 MB.

Overall, the developed database provides an efficient and effective way to store and process information related to the Airbnb use case.