Hotel Database

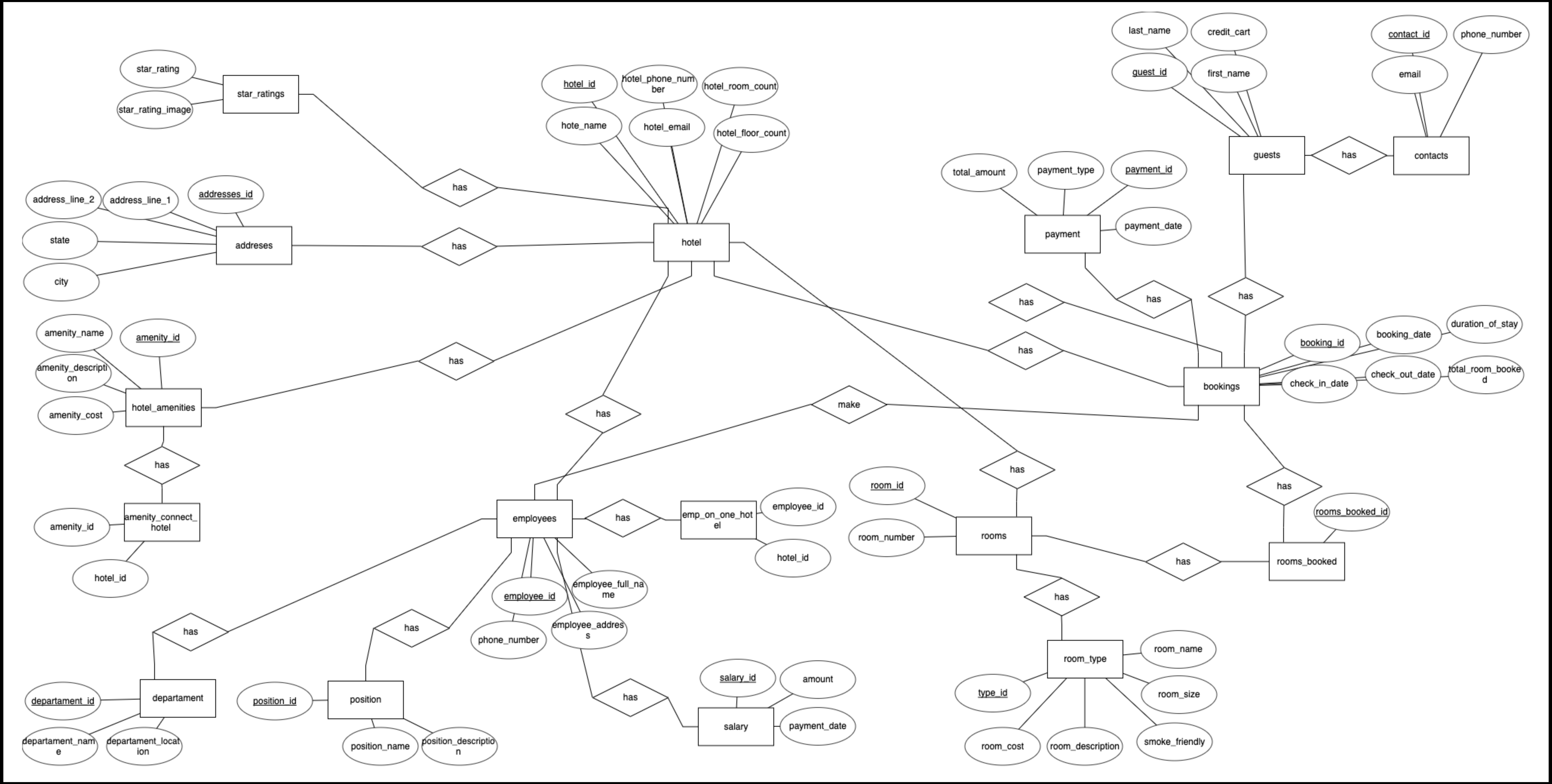
Team members

Akhmetov Abylay

id (210103019)

Amir Mamytbekov

id (210103034)


star_rating
star_ratings
star_rating_image
hotel_id
hotel_phone_number
hotel_room_count
hote_name
hotel_email
hotel_floor_count
has
address_line_2
address_line_1
addresses_id
state
addreses
city
has
hotel
amenity_name
amenity_id
amenity_description
amenity_cost
hotel_amenities
has
has
has
amenity_id
amenity_connect_hotel
hotel_id
employees
has
emp_on_one_hotel
employee_id
hotel_id
make
employee_id
employee_full_name
phone_number
employee_address
has
departament
departament_id
departament_name
departament_location
has
position
position_id
position_name
position_description
has
salary
salary_id
amount
payment_date
last_name
credit_cart
guest_id
first_name
contact_id
phone_number
email
guests
has
contacts
total_amount
payment_type
payment_id
payment
payment_date
has
has
has
has
bookings
booking_id
booking_date
duration_of_stay
check_in_date
check_out_date
total_room_booked
has
has
room_id
room_number
rooms
has
rooms_booked
rooms_booked_id
has
room_type
room_name
type_id
room_size
room_cost
room_description
smoke_friendly

ER Diagram

Functional Dependencies

addresses_id -> addresses_line1 , addresses_line2 , state , zipcode , city

amenity_id -> amenity_name , amenity_description , amenity_cost

department_id -> department_name , department_location

position_id -> position_name , position_description

employee_id -> phone_number , employee_address_id , employee_full_name

salary_id -> amount , payment_date

room_id -> room_number

type_id -> room_cost , room_description , smoke_friendly , room_size , room_name

booking_id -> check_in_date , booking_date , duration_of_stay , check_out_date , total_room_booked , payment_id

payment_id -> payment_date , payment_type , total_amount

guest_id -> last_name , first_name , credit_cart

contact_id -> email , phone_number

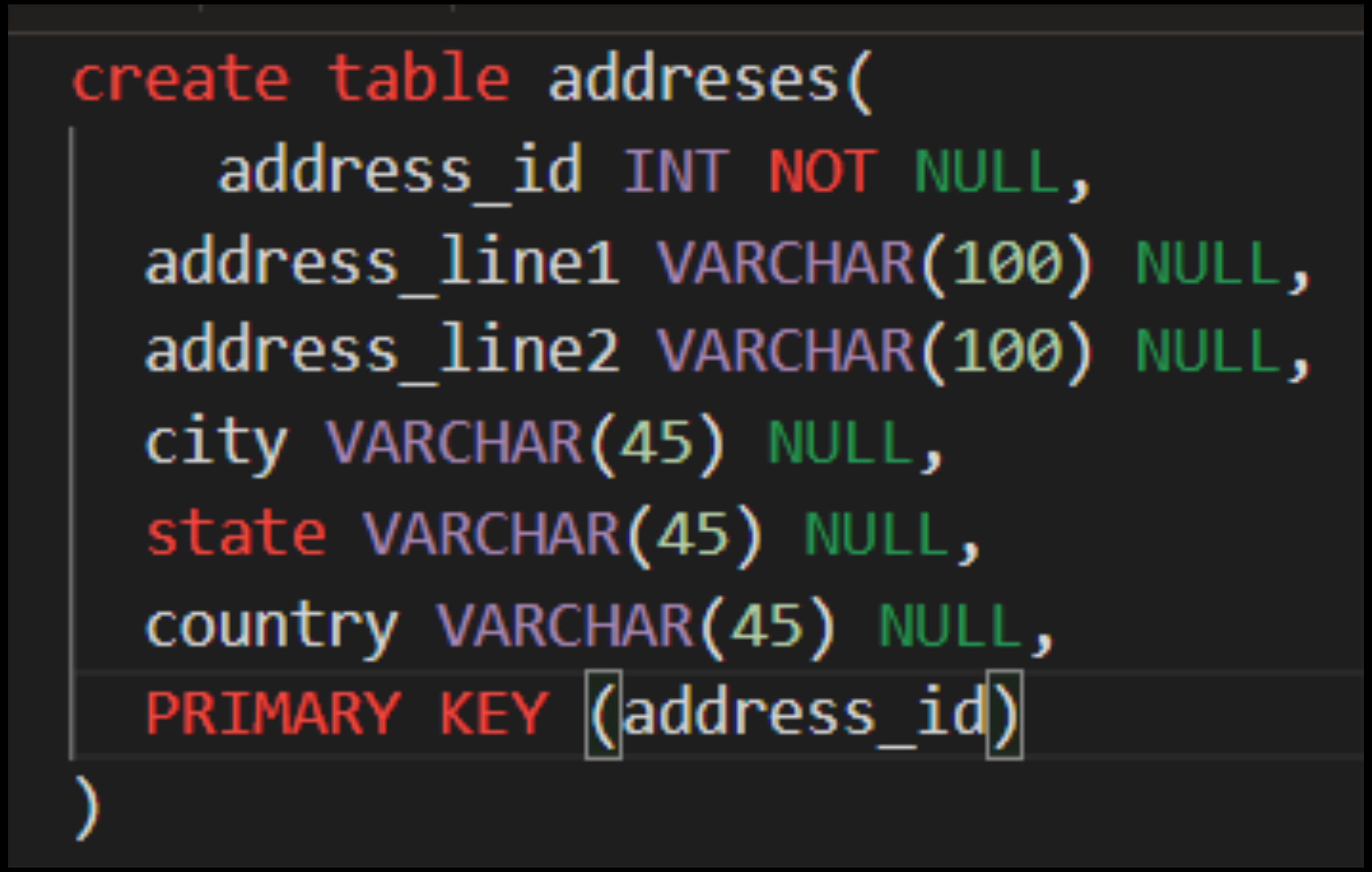
hotel_id -> hotel_phone_number , hotel_room_count , hotel_name , hotel_email , hotel_floor_count

Normalization

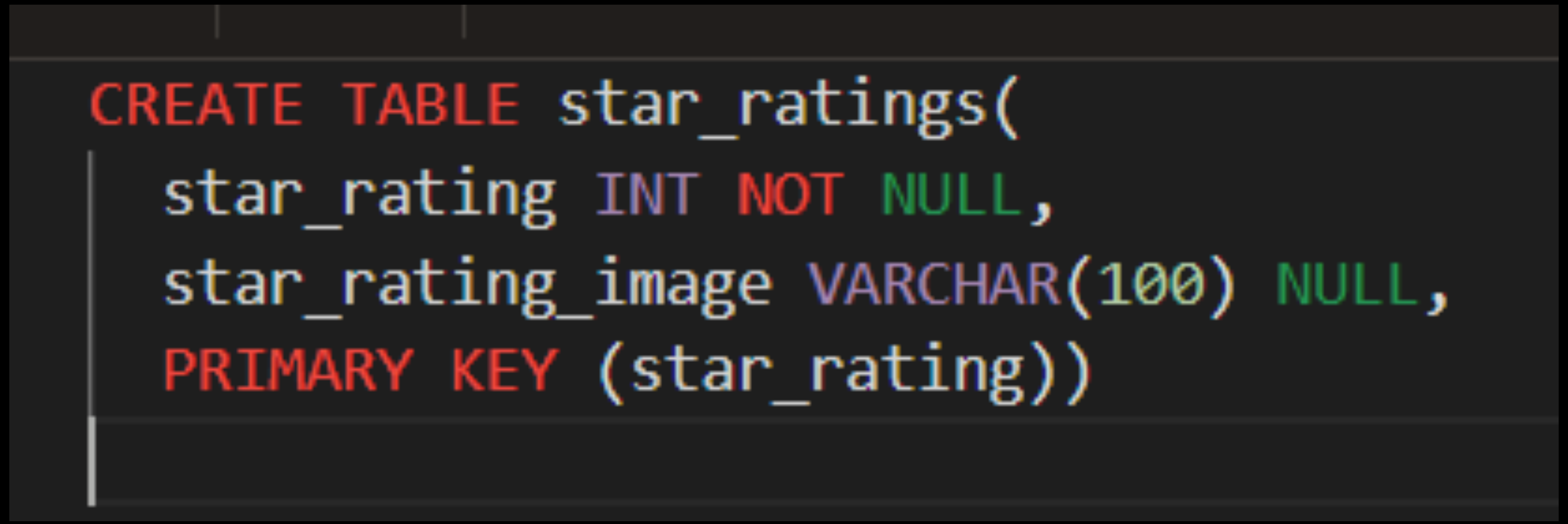
Bring tables into the correct form

Conditions to avoid or minimize redundancy, update anomalies, insert and deletion anomalies

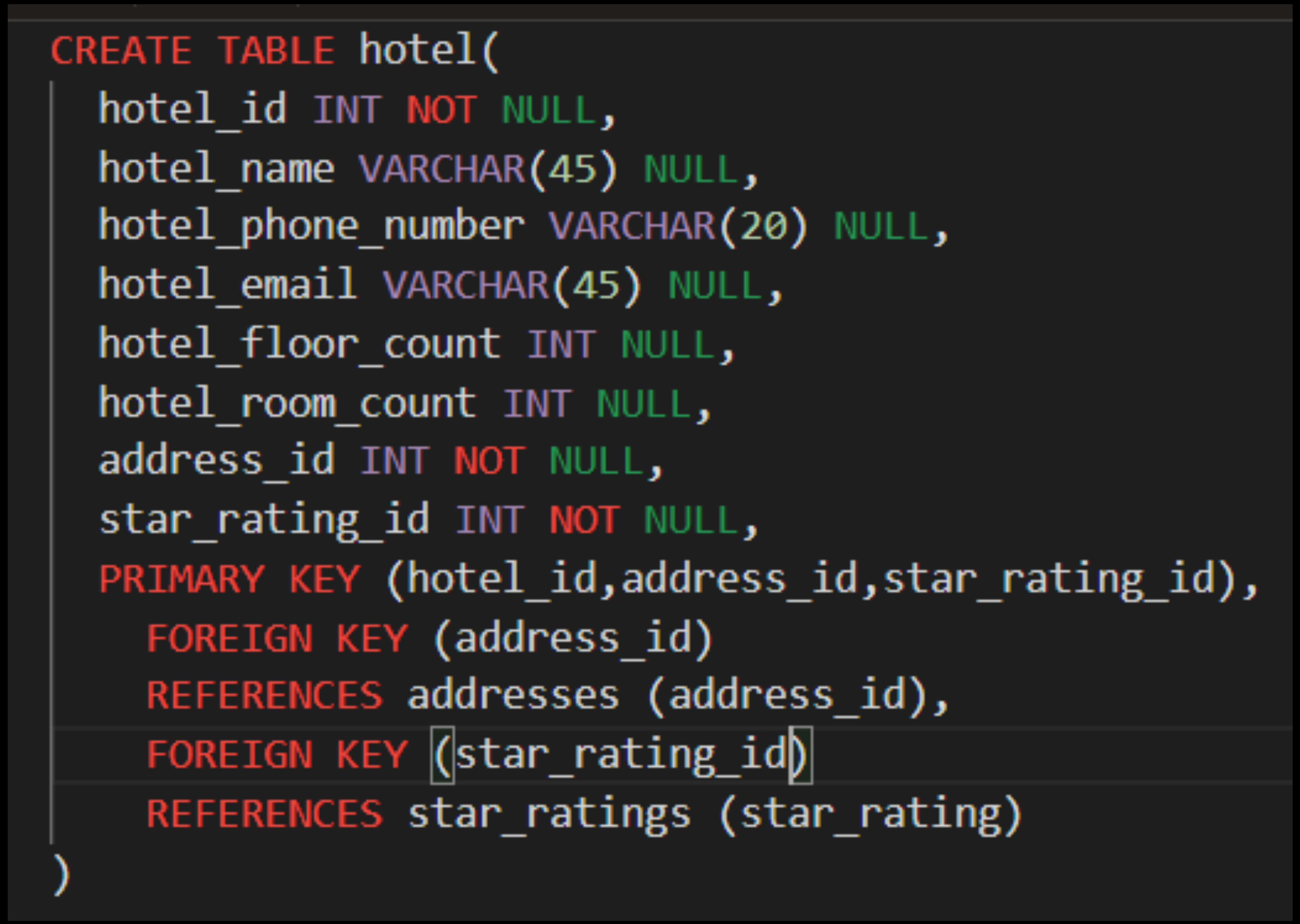
Creating Tables



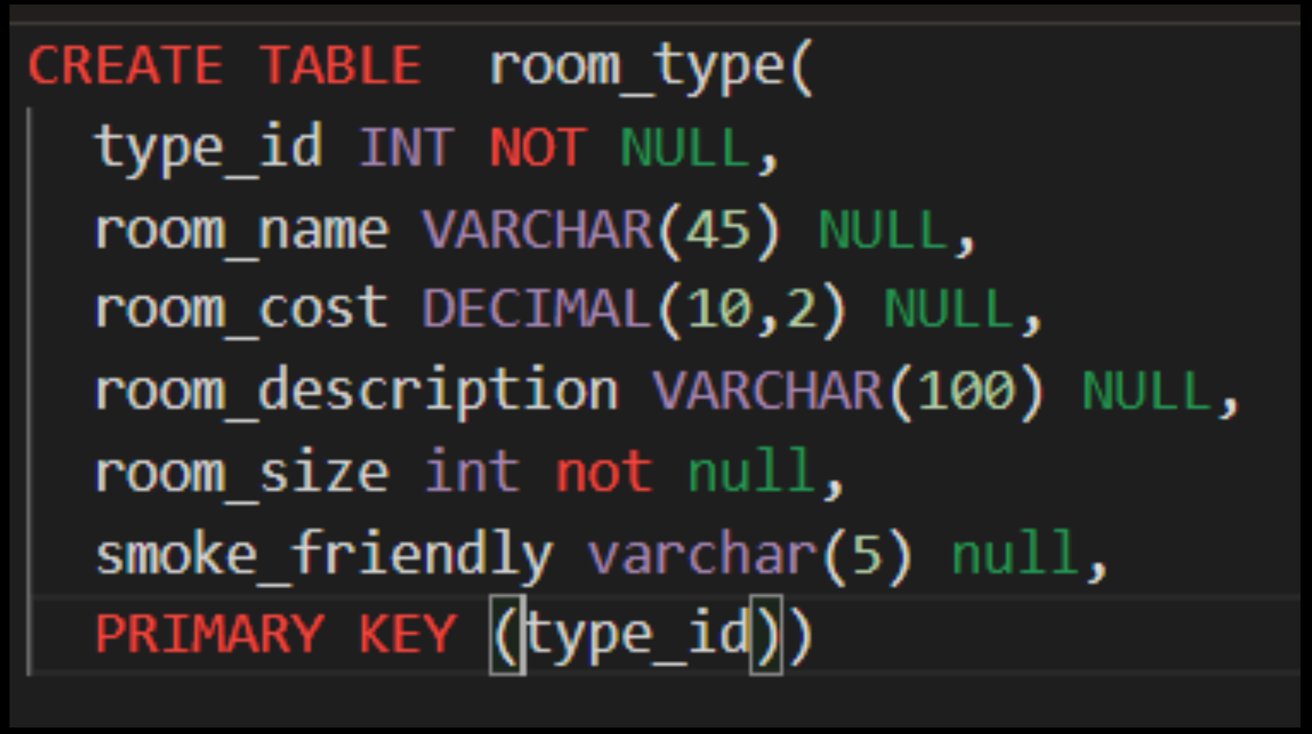
```sql
create table addreses(
      address_id INT NOT NULL,
   address_line1 VARCHAR(100) NULL,
   address_line2 VARCHAR(100) NULL,
   city VARCHAR(45) NULL,
   state VARCHAR(45) NULL,
   country VARCHAR(45) NULL,
   PRIMARY KEY (address_id)
)
```



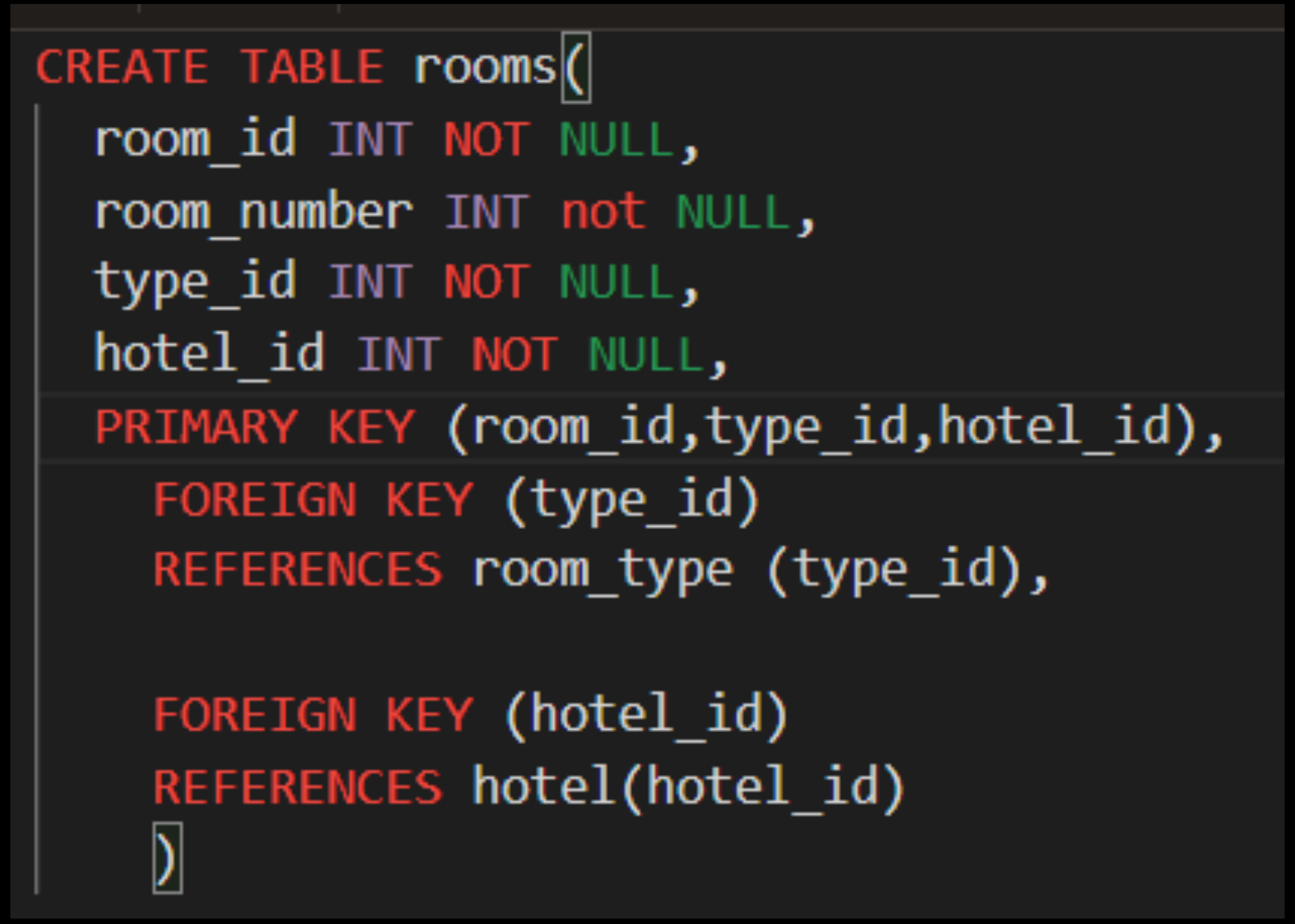
```sql
CREATE TABLE star_ratings(
  star_rating INT NOT NULL,
  star_rating_image VARCHAR(100) NULL,
  PRIMARY KEY (star_rating))
```



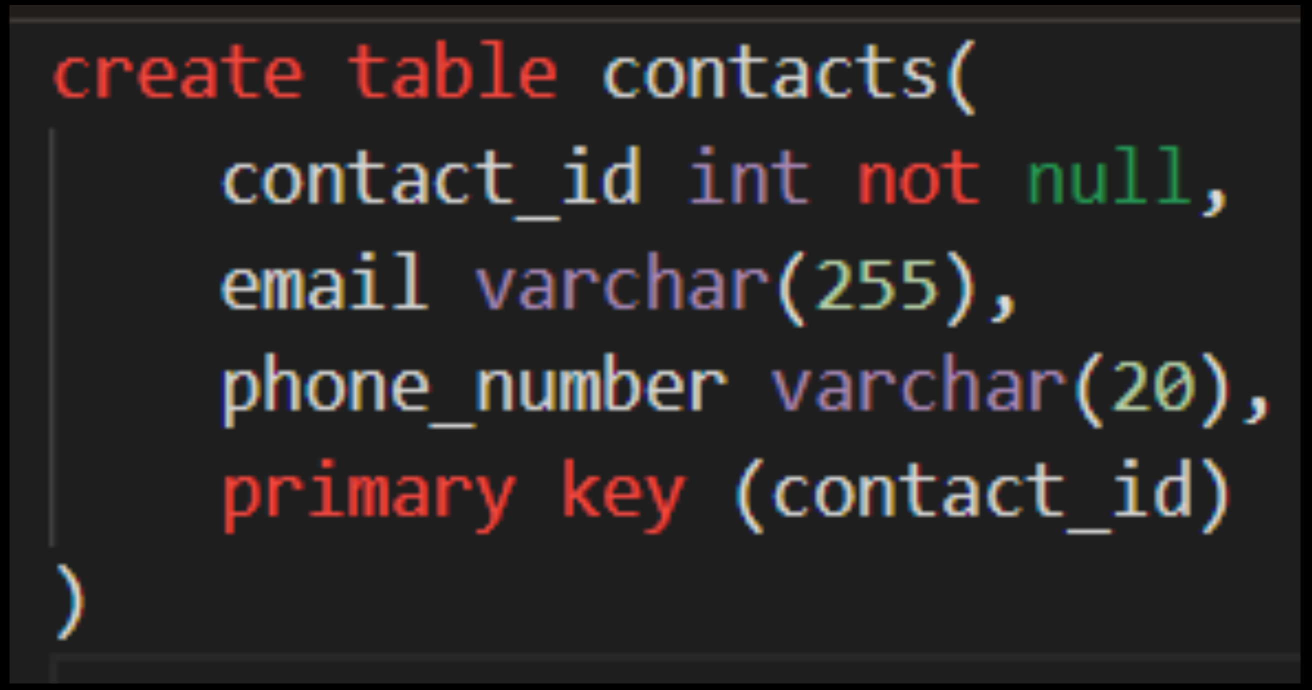
```sql
CREATE TABLE hotel(
  hotel_id INT NOT NULL,
  hotel_name VARCHAR(45) NULL,
  hotel_phone_number VARCHAR(20) NULL,
  hotel_email VARCHAR(45) NULL,
  hotel_floor_count INT NULL,
  hotel_room_count INT NULL,
  address_id INT NOT NULL,
  star_rating_id INT NOT NULL,
  PRIMARY KEY (hotel_id,address_id,star_rating_id),
    FOREIGN KEY (address_id)
    REFERENCES addresses (address_id),
    FOREIGN KEY (star_rating_id)
    REFERENCES star_ratings (star_rating)
)
```



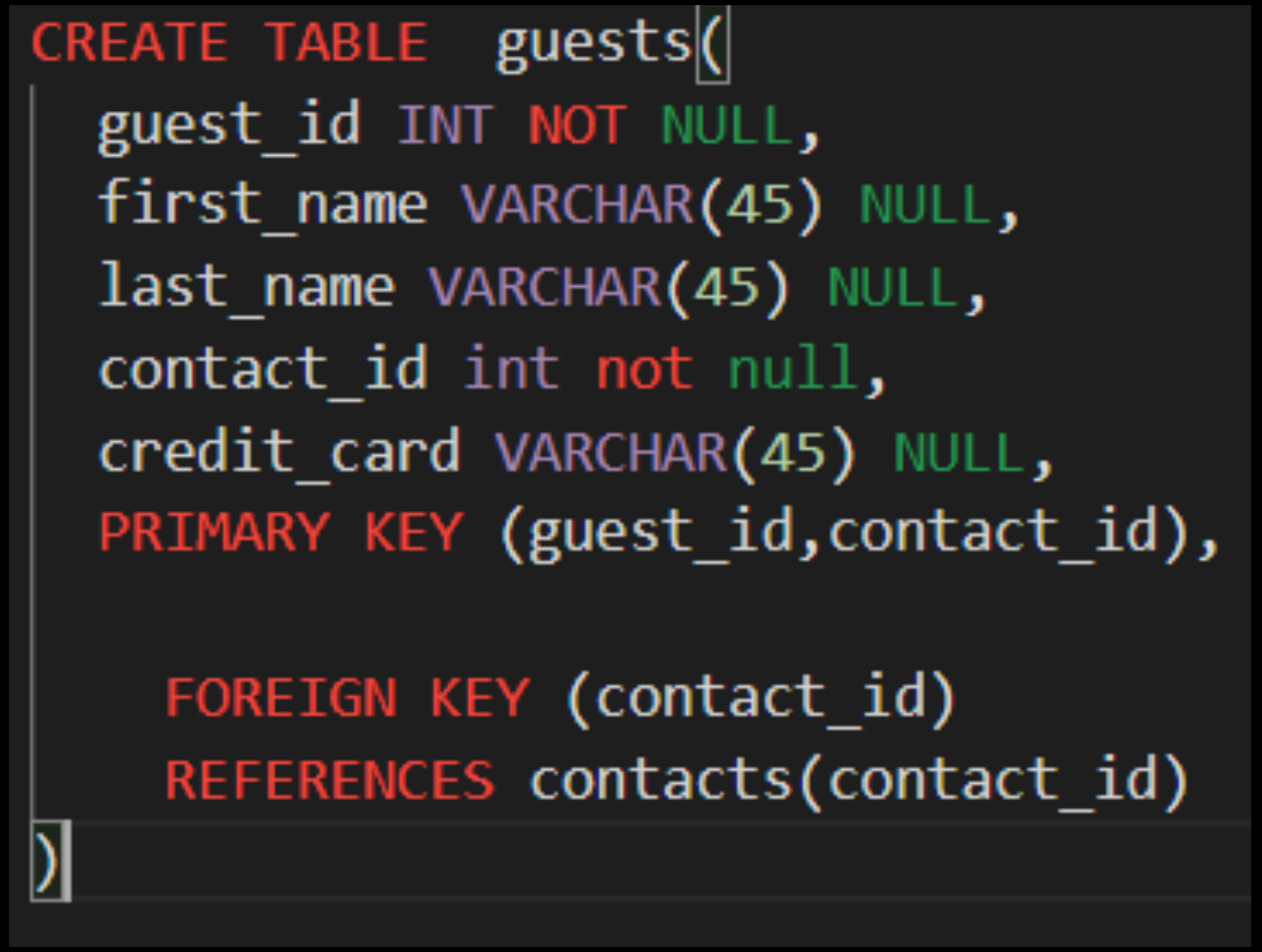
```sql
CREATE TABLE  room_type(
  type_id INT NOT NULL,
  room_name VARCHAR(45) NULL,
  room_cost DECIMAL(10,2) NULL,
  room_description VARCHAR(100) NULL,
  room_size int not null,
  smoke_friendly varchar(5) null,
  PRIMARY KEY (type_id))
```



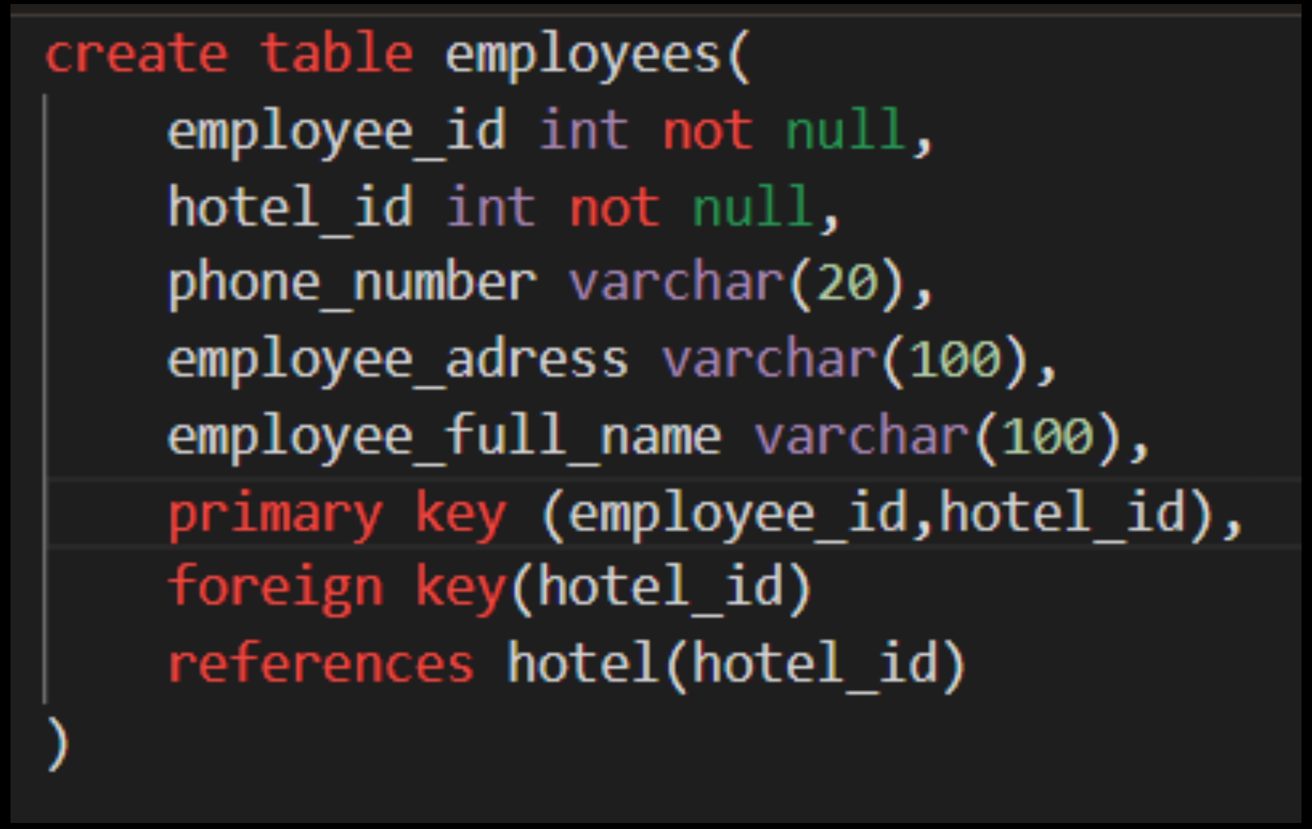
```sql
CREATE TABLE rooms(
  room_id INT NOT NULL,
  room_number INT not NULL,
  type_id INT NOT NULL,
  hotel_id INT NOT NULL,
  PRIMARY KEY (room_id,type_id,hotel_id),
    FOREIGN KEY (type_id)
    REFERENCES room_type (type_id),

    FOREIGN KEY (hotel_id)
    REFERENCES hotel(hotel_id)
    )
```



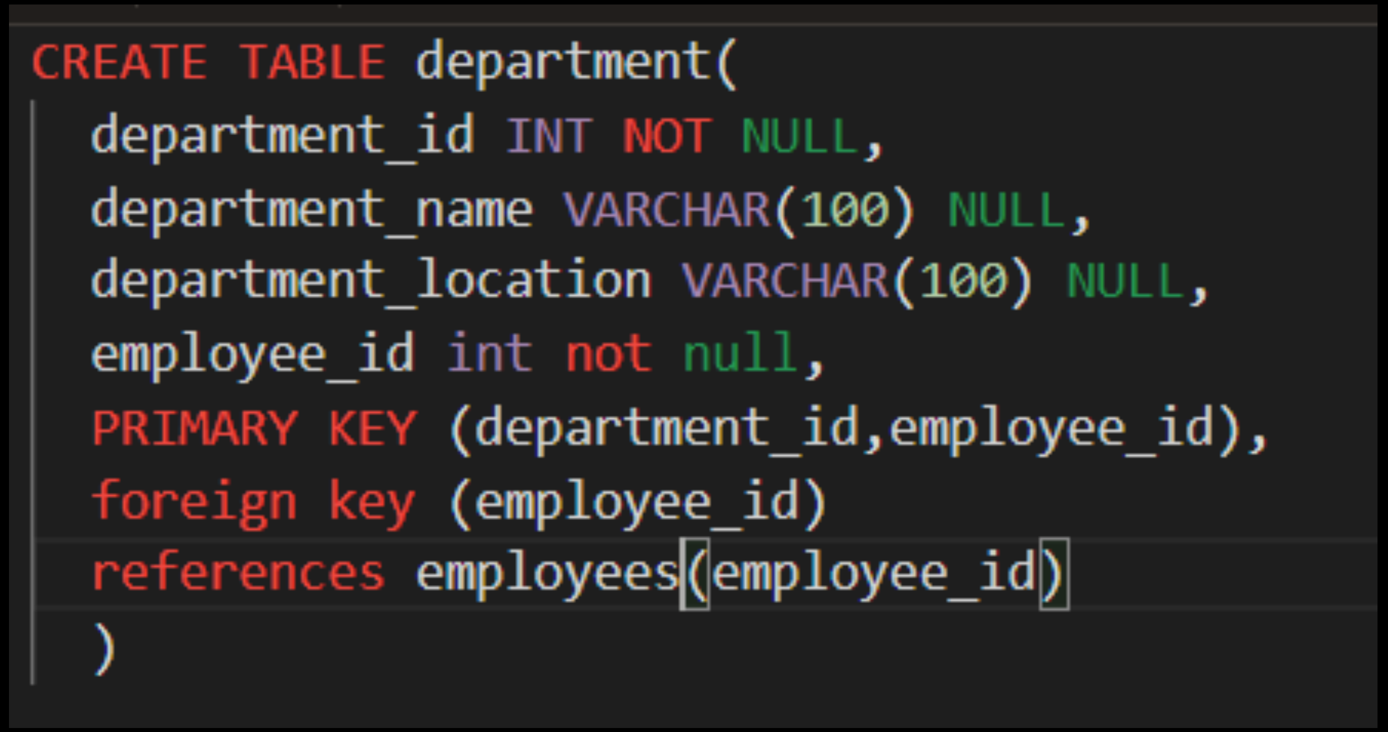
```sql
create table contacts(
    contact_id int not null,
    email varchar(255),
    phone_number varchar(20),
    primary key (contact_id)
)
```



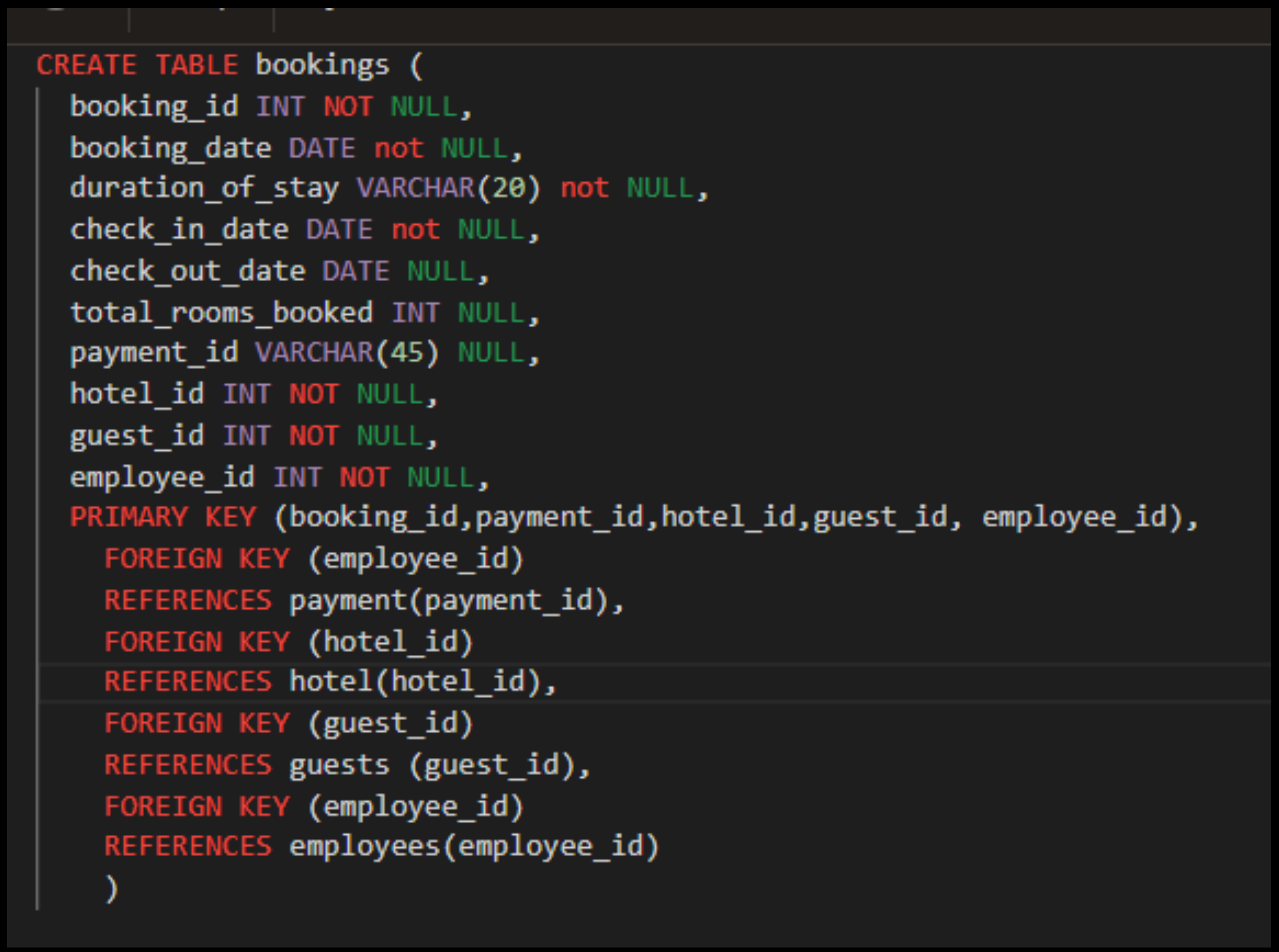
```sql
CREATE TABLE  guests(
  guest_id INT NOT NULL,
  first_name VARCHAR(45) NULL,
  last_name VARCHAR(45) NULL,
  contact_id int not null,
  credit_card VARCHAR(45) NULL,
  PRIMARY KEY (guest_id,contact_id),

    FOREIGN KEY (contact_id)
    REFERENCES contacts(contact_id)
)
```



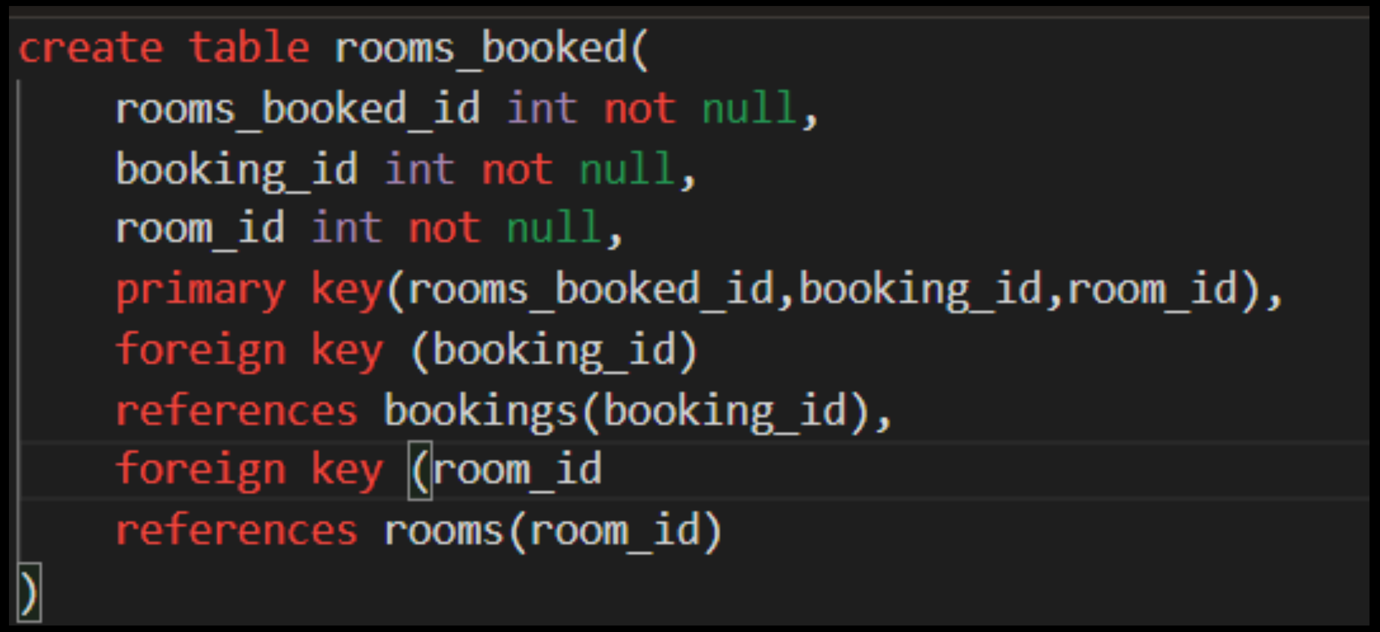
```sql
create table employees(
    employee_id int not null,
    hotel_id int not null,
    phone_number varchar(20),
    employee_adress varchar(100),
    employee_full_name varchar(100),
    primary key (employee_id,hotel_id),
    foreign key(hotel_id)
    references hotel(hotel_id)
)
```



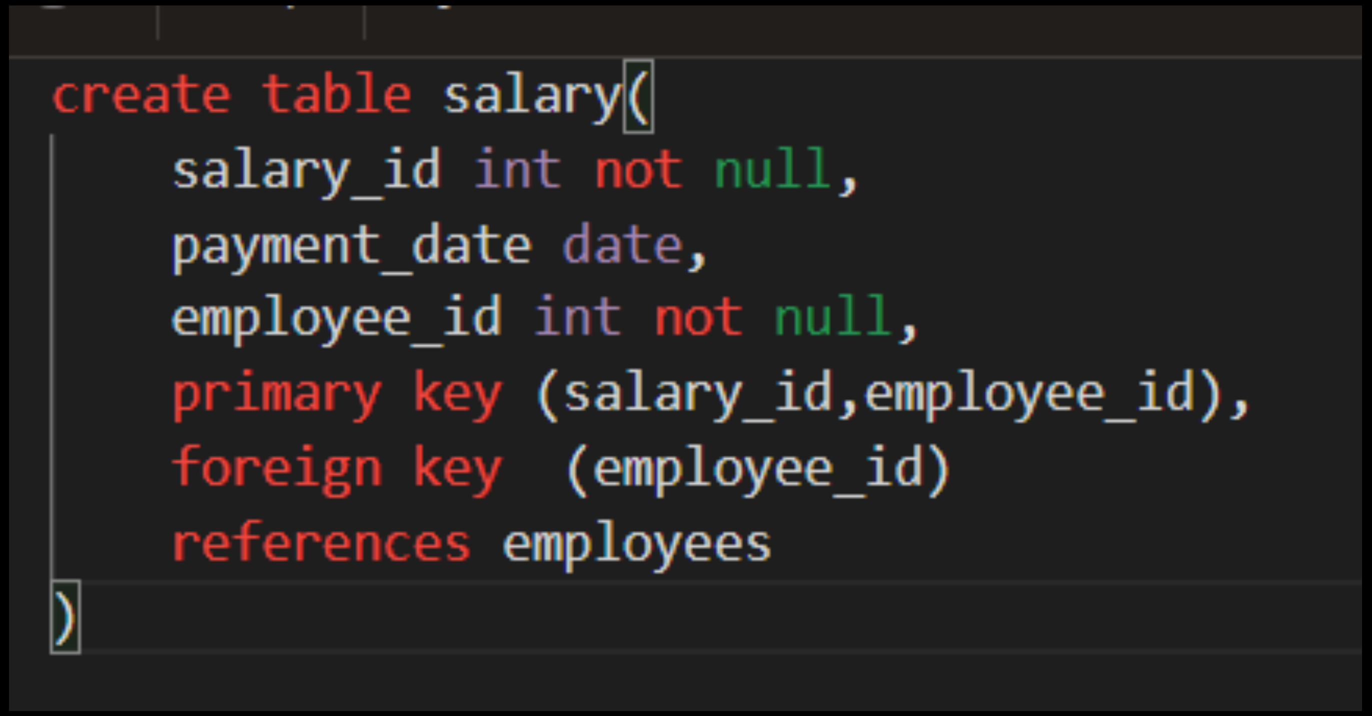
```sql
CREATE TABLE department(
  department_id INT NOT NULL,
  department_name VARCHAR(100) NULL,
  department_location VARCHAR(100) NULL,
  employee_id int not null,
  PRIMARY KEY (department_id,employee_id),
  foreign key (employee_id)
  references employees(employee_id)
  )
```



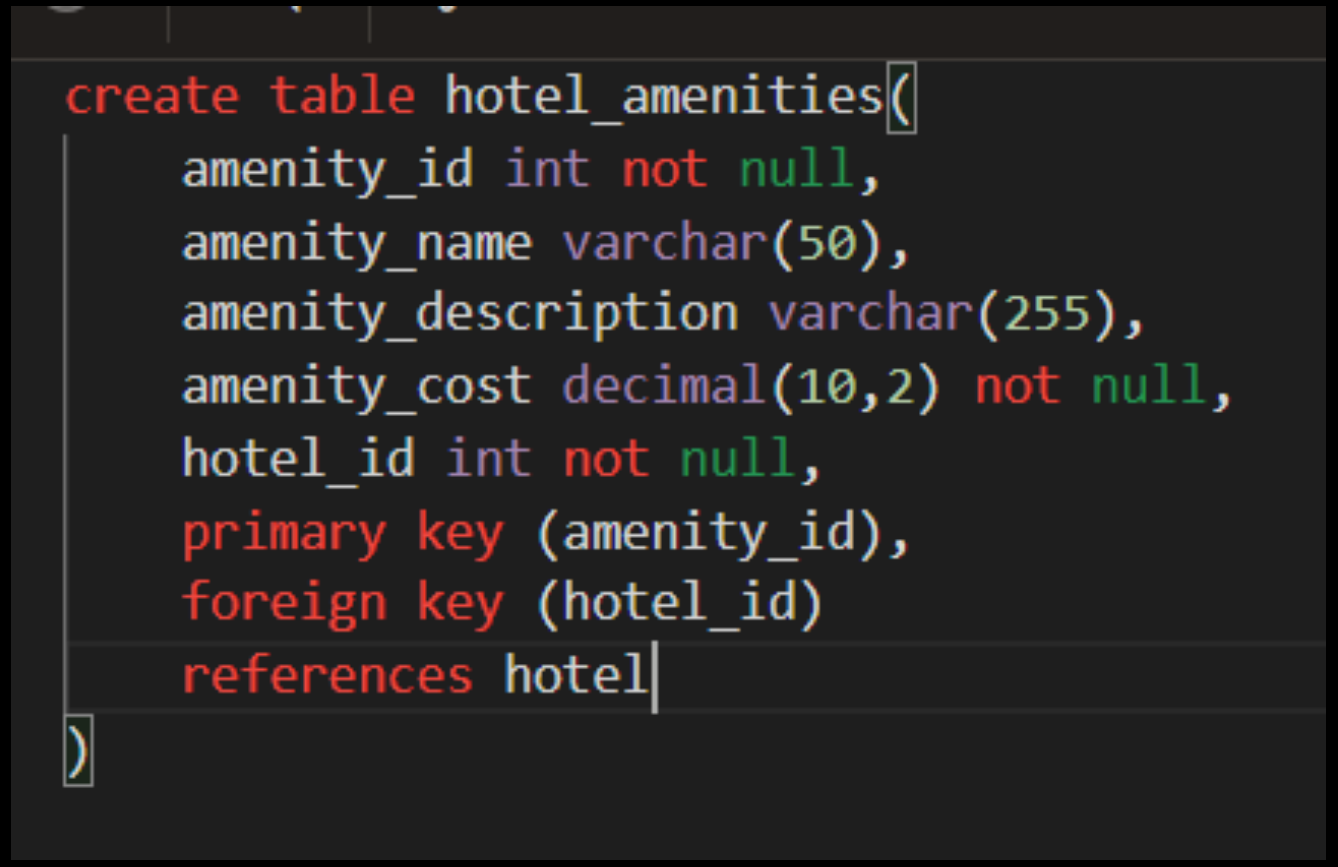
```
CREATE TABLE bookings (
  booking_id INT NOT NULL,
  booking_date DATE not NULL,
  duration_of_stay VARCHAR(20) not NULL,
  check_in_date DATE not NULL,
  check_out_date DATE NULL,
  total_rooms_booked INT NULL,
  payment_id VARCHAR(45) NULL,
  hotel_id INT NOT NULL,
  guest_id INT NOT NULL,
  employee_id INT NOT NULL,
  PRIMARY KEY (booking_id,payment_id,hotel_id,guest_id, employee_id),
    FOREIGN KEY (employee_id)
    REFERENCES payment(payment_id),
    FOREIGN KEY (hotel_id)
    REFERENCES hotel(hotel_id),
    FOREIGN KEY (guest_id)
    REFERENCES guests (guest_id),
    FOREIGN KEY (employee_id)
    REFERENCES employees(employee_id)
    )
```



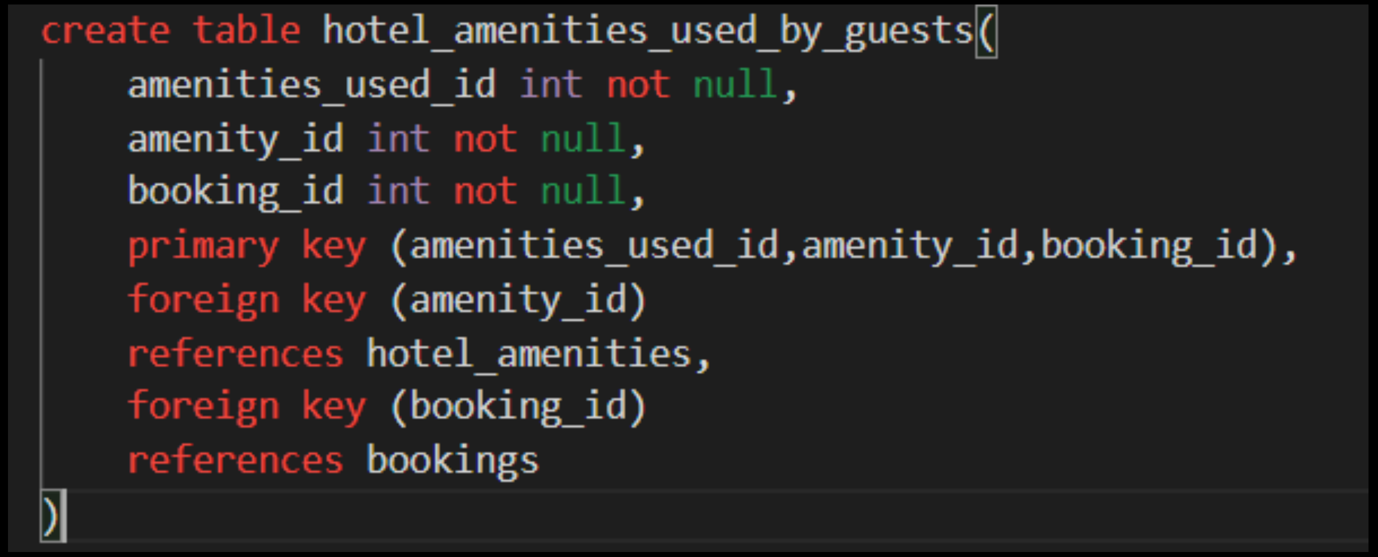
```
create table rooms_booked(
    rooms_booked_id int not null,
    booking_id int not null,
    room_id int not null,
    primary key(rooms_booked_id,booking_id,room_id),
    foreign key (booking_id)
    references bookings(booking_id),
    foreign key (room_id
    references rooms(room_id)
)
```



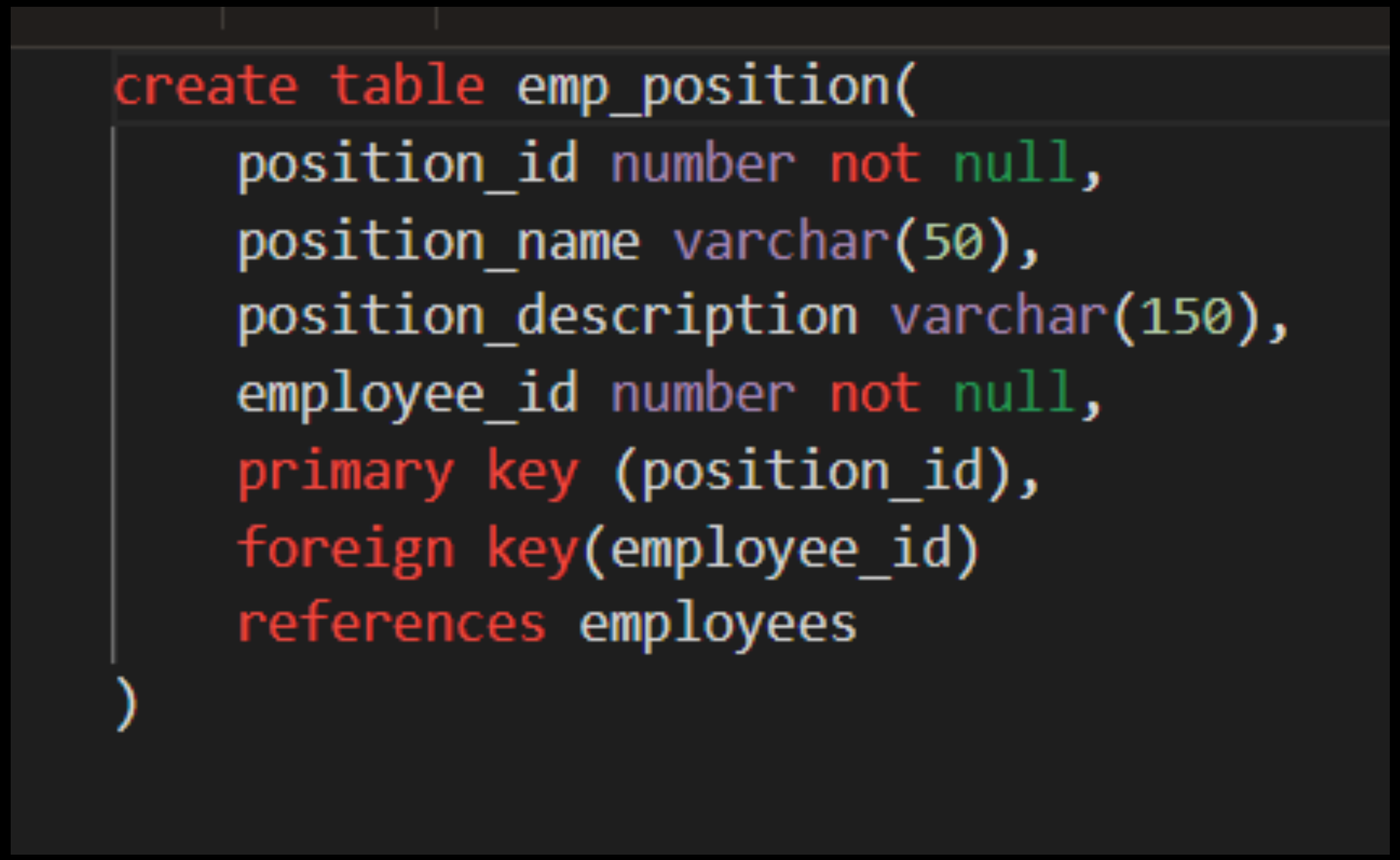
```
create table salary(
    salary_id int not null,
    payment_date date,
    employee_id int not null,
    primary key (salary_id,employee_id),
    foreign key  (employee_id)
    references employees
)
```



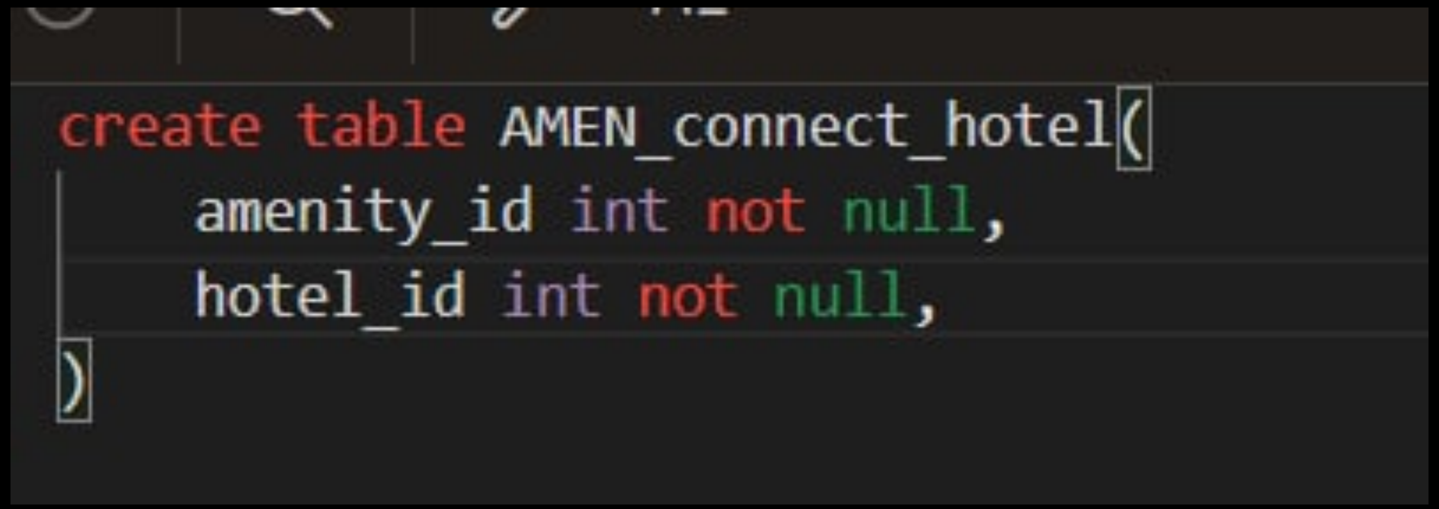
```
create table hotel_amenities(
    amenity_id int not null,
    amenity_name varchar(50),
    amenity_description varchar(255),
    amenity_cost decimal(10,2) not null,
    hotel_id int not null,
    primary key (amenity_id),
    foreign key (hotel_id)
    references hotel
)
```



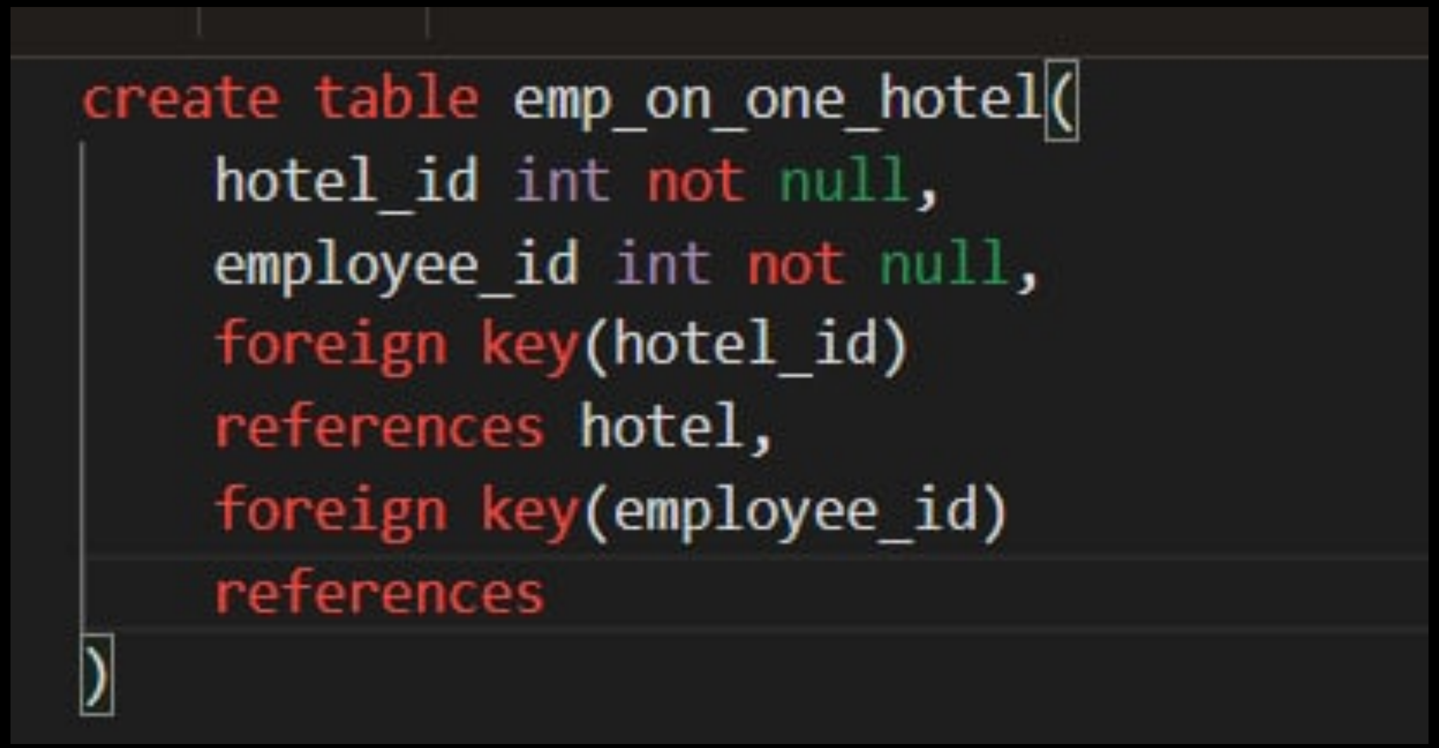
```
create table hotel_amenities_used_by_guests(
    amenities_used_id int not null,
    amenity_id int not null,
    booking_id int not null,
    primary key (amenities_used_id,amenity_id,booking_id),
    foreign key (amenity_id)
    references hotel_amenities,
    foreign key (booking_id)
    references bookings
)
```



```
create table emp_position(
    position_id number not null,
    position_name varchar(50),
    position_description varchar(150),
    employee_id number not null,
    primary key (position_id),
    foreign key(employee_id)
    references employees
)
```



```
create table AMEN_connect_hotel(
    amenity_id int not null,
    hotel_id int not null,
)
```



```
create table emp_on_one_hotel(
    hotel_id int not null,
    employee_id int not null,
    foreign key(hotel_id)
    references hotel,
    foreign key(employee_id)
    references
)
```

Queries

Сколько номеров забронировано в конкретном отеле на определенную дату

```
SELECT count(booking_id) AS "Total Rooms Booked"

FROM bookings

WHERE booking_date LIKE :booking_date;
```

π "Total Rooms Booked" (count(booking_id))

(σ booking_date LIKE :booking_date (bookings))

Выводить имена , номера телефонов , день въезда и день выселения гостей на определенную дату

```
SELECT last_name, first_name, phone_number, booking_date, check_out_date

FROM guests

JOIN contacts ON guests.contact_id = contacts.contact_id

JOIN bookings ON guests.guest_id = bookings.guest_id and bookings.booking_date = :booking_date
```

Test dates (12/28/2021 , 03/12/2022)

π last_name, first_name, phone_number, booking_date, check_out_date(guests ⋈ contacts[guests.contact_id = contacts.contact_id] ⋈ bookings[guests.guest_id = bookings.guest_id and bookings.booking_date = :booking_date])

Выводит имя отеля, контактный номер отеля , его рейтинг, в каком штате находится, количество комнат

```
SELECT hotel.hotel_name,
hotel.hotel_phone_number, addresses.state,
star_ratings.star_rating_image,
hotel.hotel_room_count

FROM hotel

JOIN addresses ON hotel.address_id =
addresses.address_id

JOIN star_ratings ON hotel.star_rating_id =
star_ratings.star_rating;
```

π hotel_name, hotel_phone_number, state, star_rating_image, hotel_room_count(hotel ⋈ addresses[hotel.address_id = addresses.address_id] ⋈ star_ratings[hotel.star_rating_id = star_ratings.star_rating])

Выводит имя отеля, имя его комнат, его КВ м, описание комнаты и его стоимость

```
select
hotel_name,room_name,room_size,room_description,room_cost

from hotel,rooms,room_type

where room_type.type_id = rooms.type_id

and hotel.hotel_id = rooms.hotel_id
```

π hotel_name, room_name, room_size, room_description, room_cost(hotel ⋈ rooms ⋈ room_type[room_type.type_id = rooms.type_id and hotel.hotel_id = rooms.hotel_id])

Сколько сколько раз в год клиент сделал резервирование номера

SELECT count(*)

FROM bookings

WHERE booking_date LIKE '%2022' AND guest_id = 501;

π count(*)(σ booking_date LIKE '%2022' AND guest_id = 501 (bookings))

Сколько номеров забронировано в конкретном отеле на определенную дату

SELECT count(booking_id) AS "Total Rooms Booked"

FROM bookings

WHERE booking_date LIKE :booking_date;

π "Total Rooms Booked" (count(booking_id))(σ booking_date LIKE :booking_date (bookings))

Сколько номеров доступно в данном отеле на конкретную дату

```
SELECT h.hotel_id, h.hotel_room_count -
SUM(b.total_rooms_booked)

FROM bookings b JOIN hotel h ON b.hotel_id =
h.hotel_id

WHERE b.booking_date LIKE '12/28/2021'
AND h.hotel_id = 1

GROUP BY h.hotel_id, h.hotel_room_count
```

π h.hotel_id, h.hotel_room_count - SUM(b.total_rooms_booked)(bookings ⋈ hotel[bookings.hotel_id = hotel.hotel_id]WHERE booking_date LIKE '12/28/2021' AND h.hotel_id = 1 GROUP BY h.hotel_id, h.hotel_room_count)

Выводить сотрудника , отель в котором он работает , номер телефона и название должности

```
select
hotel_name,employee_full_name,phone_number,position_name

from employees,hotel, emp_on_one_hotel,emp_position

where hotel.hotel_id = emp_on_one_hotel.hotel_id

and employees.employee_id =
emp_on_one_hotel.employee_id

and employees.employee_id = emp_position.employee_id
```

π hotel_name, employee_full_name, phone_number, position_name

(hotel ⋈ emp_on_one_hotel[hotel.hotel_id = emp_on_one_hotel.hotel_id] ⋈ employees[emp_on_one_hotel.employee_id = employees.employee_id] ⋈ emp_position[employees.employee_id = emp_position.employee_id])

Вывести список комнат в которые
заселяются гости на конкретную дату

```
SELECT room_number

FROM rooms

WHERE room_number IN (

  SELECT room_number

  FROM bookings

  WHERE check_in_date = '04/28/2022'

);
```

π room_number (rooms ⋈ (σ check_in_date
='04/28/2022' (bookings[room_number])

Triggers

Trigger который запрещает тип оплаты ‘cash’

```
CREATE OR REPLACE TRIGGER
prevent_cash_payments

BEFORE INSERT ON payments

FOR EACH ROW

BEGIN

  IF :new.payment_type = 'cash' THEN

    RAISE_APPLICATION_ERROR(-20001, 'Cash
payments are not allowed');

  END IF;

END;
```

Trigger который удаляет строку когда проходит ‘check_out_date’

```
CREATE OR REPLACE TRIGGER
delete_expired_bookings

AFTER INSERT OR UPDATE ON bookings

FOR EACH ROW

BEGIN

  IF :new.check_out_date < SYSDATE THEN

    DELETE FROM bookings WHERE booking_id
= :new.booking_id;

  END IF;

END;
```

Если длина вводимого номера карточки не равна 16 ти то выводит ошибку

```
CREATE OR REPLACE TRIGGER
credit_card_length_trg

BEFORE INSERT OR UPDATE ON guests

FOR EACH ROW

BEGIN

  IF LENGTH(:NEW.CREDIT_CARD) != 16 THEN

    RAISE_APPLICATION_ERROR(-20002, 'The
credit card number must be exactly 16 characters
long');

  END IF;

END;
```

VIEWS

View который объединяет employees и emp_position tables

```
CREATE OR REPLACE VIEW employee_position AS

SELECT e.employee_id,
e.EMPLOYEE_FULL_NAME,
e.EMPLOYEE_ADRESS, e.PHONE_NUMBER ,
p.position_name , p.POSITION_DESCRIPTION

FROM employees e

JOIN emp_position p ON e.employee_id =
p.employee_id;
```

π(employee_id, employee_full_name, employee_adress, phone_number, position_name, position_description)(employees ⋈ emp_position)

View который объединяет guests и contacts tables

```
CREATE VIEW combined_view AS

SELECT g.GUEST_ID AS GUEST_ID,
g.FIRST_NAME AS guest_name, c.CONTACT_ID
AS contact_id, c.PHONE_NUMBER AS
contact_phone

FROM guests g

JOIN contacts c ON g.contact_id =
c.CONTACT_ID;
```

π(guest_id, guest_name, contact_id, contact_phone) (guests ⋈ contacts)

View в который объединяет rooms и room_type tables

```
CREATE VIEW room_roomtype_view AS

SELECT r.ROOM_ID AS room_id,
r.ROOM_NUMBER AS room_number,
rt.TYPE_ID AS type_id, rt.ROOM_NAME AS
type_name

FROM rooms r

JOIN room_type rt ON r.TYPE_ID =
rt.TYPE_ID;
```

π(room_id, room_number, type_id, type_name)
(rooms $\bowtie$ room_type)

Transactions

Обновление информации о комнате

```
BEGIN

  UPDATE rooms

  SET room_type = :new_room_type,

      rate = :new_rate

  WHERE room_number = :room_number;

  COMMIT;

EXCEPTION

  WHEN OTHERS THEN

    ROLLBACK;

    RAISE;

END;
```

Проверка на существующий guest_id

```
BEGIN

  INSERT INTO guests (guest_id, first_name,
last_name,contact_id,credit_card)

  VALUES
(:guest_id, :first_name, :last_name, :contact_id,:credit_c
ard);

  EXCEPTION

    WHEN dup_val_on_index THEN

      raise_application_error (-20001, 'Error: product
already exists');

END;
```

Indexes

Индекс на имя и фамилию постояльцев

```
CREATE INDEX guest_room_index ON guests
(guest_id, first_name);
```

Индекс для поля check_in_date таблицы booking :

```
CREATE INDEX check_in_date_index ON
booking (check_in_date);
```

Индекс для поля room_number таблицы rooms:

```
CREATE INDEX room_number_index ON
rooms (room_number);
```

Индекс для поля email таблицы guests, с проверкой уникальности

```
CREATE UNIQUE INDEX email_index ON
guests (email);
```

Thank you for attention!