Final Project Report

Airbnb and Airbnb Online Experiences

Team Number – 20

Team Members – 1) Nikhilesh Gorrepati (NXG220004)

2) Shravya Sheela (SXS220053)

3) Venkata Satya Vyshnavi Ryali (VXR210056)

Step 1: Data Requirements

As part of the data requirements collection and analysis, below are the observations:

1. A person can be a host/owner who will list the properties owned by them on the site or a person can be a guest who can book them through reservation/booking page based on the availability and preferences.
2. The properties may have several rooms which can be completely booked or shared by the guests and they are managed by the employees who are recruited by the owner. The separation of duties allows us to categorize the employees as managers, maintenance staff and cab drivers. The guests can choose to avail the taxi services based on their need.
3. The booking or reservation initiated by a guest can either go to auto approval or further approval by the host.
4. There is a payment gateway service after which booking is confirmed. They payment methods include debit card, credit card or other source of payment methods.
5. The properties listed have its price, availability, room details and several amenities such as cab services, gym, swimming pool, washer and dryers etc.
6. Based on the experience, a guest or the host can post ratings and reviews.

Entities and Attributes Identified:

Host/Owner: Name, Mobile Number, Email Address, SSN, Address.

Guest: Guest ID, Name, Phone number, Email Address, Address

Employees (maintenance staff, drivers, managers): Employee ID, Name, Mobile Number, Email Address, SSN, Salary

Feedback: Review, Ratings

Property: Property ID, Address, Owner Details, Amenities, Price, Room Details, Available Date

Payment: Payment Id, Date, Amount, Mode of Payment, Payment Status

Booking/Reservation: Booking ID, Booking date, Start date, End date, Price, Number of guests, Amenities

Note: Mobile Number is a multivalued attribute i.e. a person can have more than one mobile number, Name is a complex attribute i.e. it contains first name, middle name, last name, Amenities is multivalued i.e. it encloses swimming pool, gym, air conditioning, complementary breakfast, washer dryer etc. Room details/Rooms is a complex attribute that has number of bathrooms and number of bedrooms details.

Relationships between entities:

1. Host LISTS Property: 1:N → Host can list multiple properties and a property can belisted by only one host/owner

2. Host RECRUITS Employees: 1:N → Host can recruit multiple employees and an employee can be recruited only by one host (Assumption: Employees work only for one particular property owned by the host hence they can be recruited only by one host)

3. Host ACCEPTS Booking: 1:N → Host can accept multiple bookings and a bookingpertaining to a property can be accepted by that particular host only

4. Host HOST_RATING Feedback: 1:N → Host can give multiple feedbacks but afeedback belongs or will be given by one host only

5. Guest MAKES Payment: 1:N → Guest can make multiple payments but a paymentmade belongs to only one guest

6. Guest GUEST_RATING Feedback: 1:N → Guest can give multiple feedbacks but afeedback belongs or will be given by one guest only

7. Guest BROWSES Property: M:N → Guest can browse multiple properties and aproperty can also be browsed by multiple guests

8. Guest REQUESTS Booking: 1:N → Guest can requests multiple bookings but abooking belongs to only one guest

9. Manager MANAGES Property: 1:1 → Only one manager can manage a particular property and a property is managed by only one manager (Assumption: Since different properties of a host are located in different locations, a manager can only manage one property)

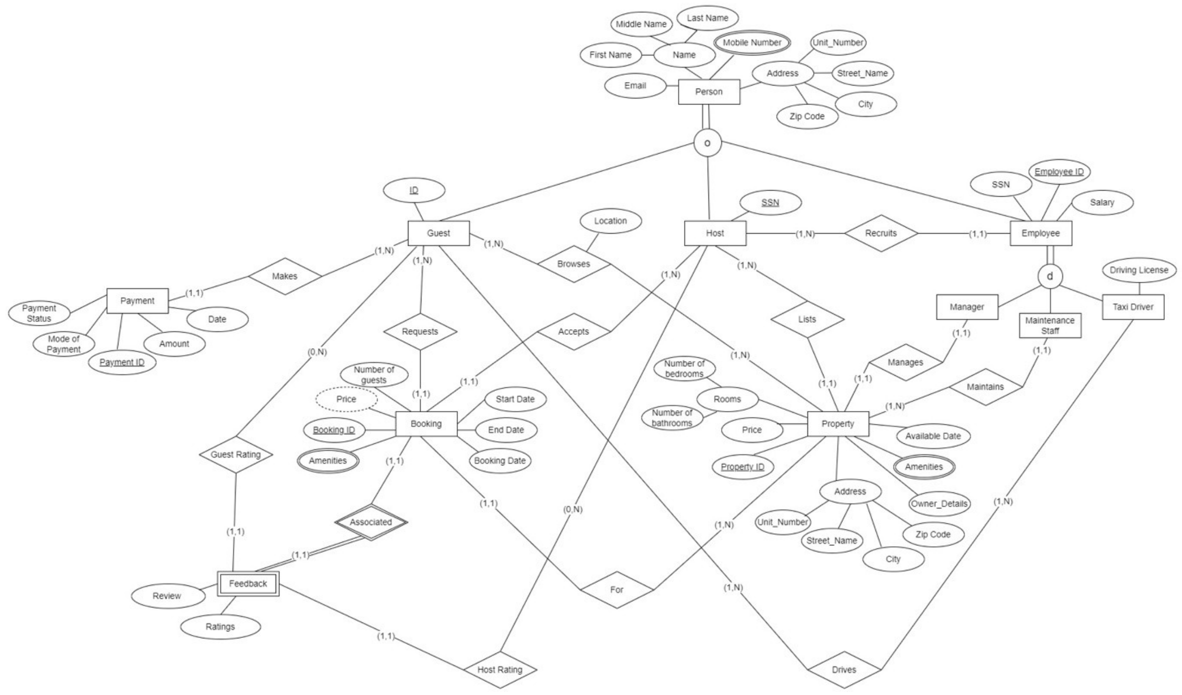
10. Maintenance staff MAINTAINS Property: N:1 → A property can be maintained bymultiple maintenance staff but the staff maintains only one property

11. Taxi driver DRIVES Guest: M:N → Cab drivers can drive multiple guests and guestscan book multiple drivers to navigate through places during their stay

12. Booking ASSOCIATED Feedback: 1:1 → A feedback belongs to a particular bookingand a booking has only one feedback associated to that booking

13. Booking FOR Property: N:1 → A booking can be made for one property where as a property can have multiple bookings. (Assumption: A property can be shared by different guests hence it can have multiple bookings)

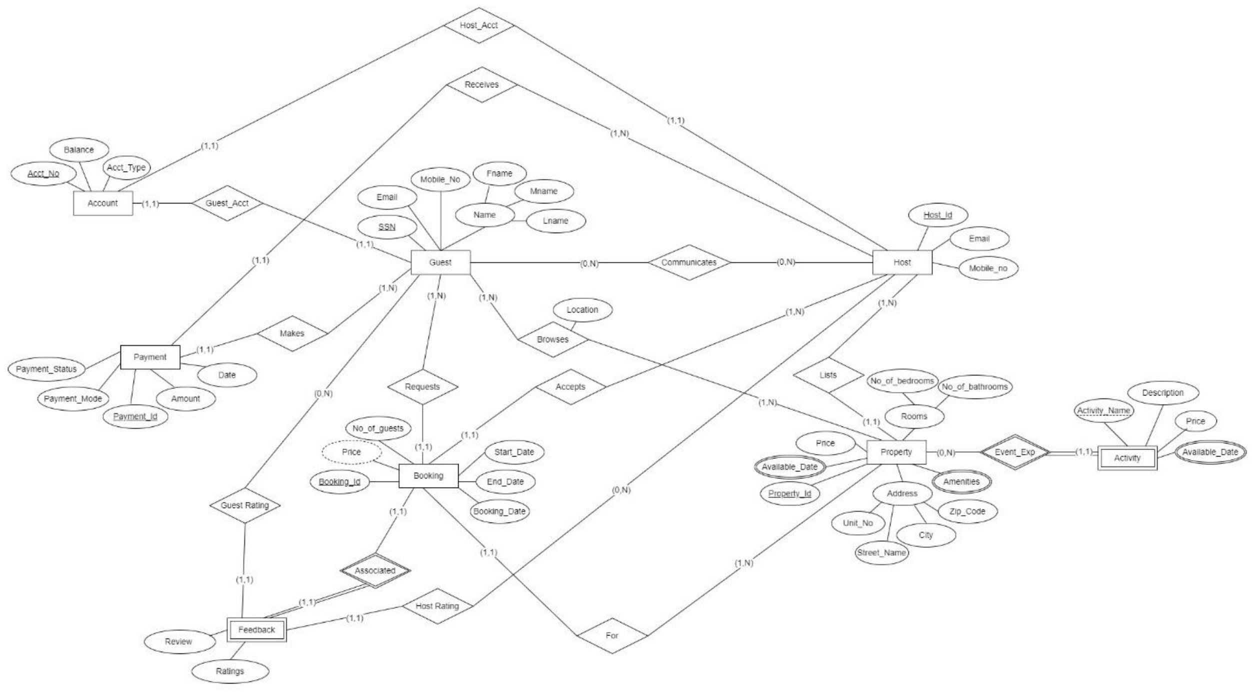
Step 2: EER Diagram



Feedback for Steps-1,2:

- Missing Experience and Online Experience
- Guest communicates with the host
- Host will have a bank account (Guest and Host will have the bank account for payment/transactions)
- Can ignore Employee

Below is the updated ER Diagram based on the feedback provided:



Modifications for ER Diagram:

1) Missing Experience and Online Experience

    - Added new entity ACTIVITY with attributes as Activity_Name, Description, Price, Available Date

    - Created new relationship Events_Exp between PROPERTY and ACTIVITY to relate activities hosted near the property

2) Guest communicates with the host

    - Added new relationship communicates between guest and host

3) Host will have a bank account (Guest and Host will have the bank account for payment/transactions)

    - Created new entity ACCOUNT with attributes as Account Number, Account Type and Balance

    - Added two new relationships (Guest_Acct, Host_Acct) between host, account and guest, account

4) Can ignore Employee

    - Removed employee part and added new relationships to make sure that the minimum requirement of relationships is met (two one-to-one binary relationships, two one-to-many binary relationships, two many-to-many binary relationships)

New set of Relationships between entities:

1. Host LISTS Property: 1:N → Host can list multiple properties and a property can belisted by only one host/owner

2. Host ACCEPTS Booking: 1:N → Host can accept multiple bookings and a bookingpertaining to a property can be accepted by that particular host only

3. Host HOST_RATING Feedback: 1:N → Host can give multiple feedbacks but a feedback belongs or will be given by one host only

4. Guest MAKES Payment: 1:N → Guest can make multiple payments but a paymentmade belongs to only one guest

5. Host Receives Payment: 1:N → Host can receive multiple payments for multiple bookings but a payment can be received by only one host

6. Guest GUEST_RATING Feedback: 1:N → Guest can give multiple feedbacks but afeedback belongs or will be given by one guest only

7. Guest BROWSES Property: M:N → Guest can browse multiple properties and aproperty can also be browsed by multiple guests

8. Guest REQUESTS Booking: 1:N → Guest can requests multiple bookings but abooking belongs to only one guest

9. Booking ASSOCIATED Feedback: 1:1 → A feedback belongs to a particular bookingand a booking has only one feedback associated to that booking

10. Booking FOR Property: N:1 → A booking can be made for one property where as a property can have multiple bookings. (Assumption: A property can be shared by different guests hence it can have multiple bookings)

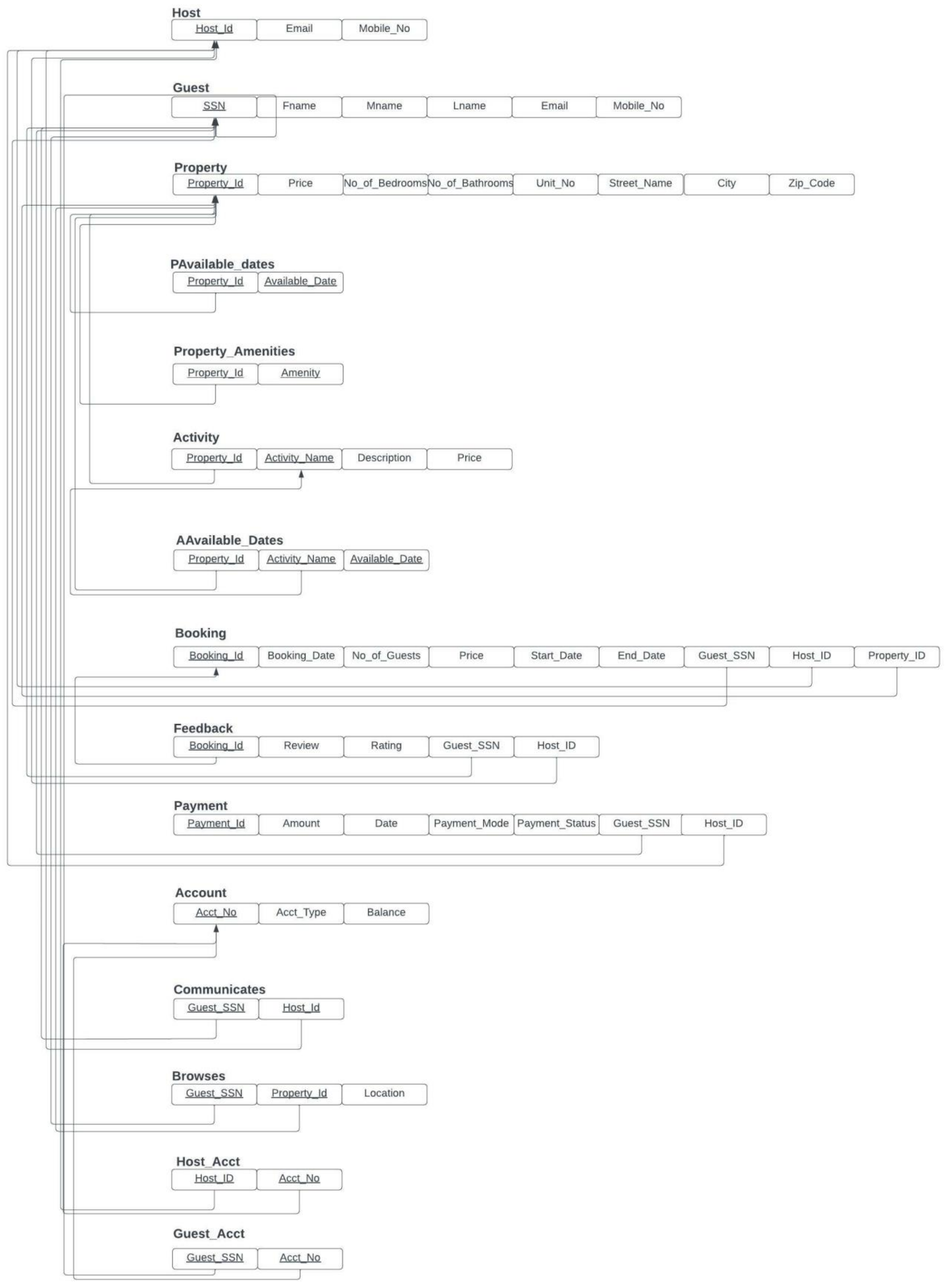
11. Property EVENT_EXP Activity: 1:N → A property can have many activities hosted nearby whereas an activity is related to one property which is near one

12. Guest GUEST_ACCT Account 1:1 → Guest can have only one account added at one time and an account belongs to one guest

13. Host HOST_ACCT Account 1:1 → Host can have only one account added at one time and an account belongs to one host

14. Guest COMMUNICATES Host M:N → A guest can communicate with multiple hosts and vice-versa because of multiple bookings

Step 3: ER Diagram to Relational Schema


Host
Host_Id | Email | Mobile_No
Guest
SSN | Fname | Mname | Lname | Email | Mobile_No
Property
Property_Id | Price | No_of_Bedrooms | No_of_Bathrooms | Unit_No | Street_Name | City | Zip_Code
PAvailable_dates
Property_Id | Available_Date
Property_Amenities
Property_Id | Amenity
Activity
Property_Id | Activity_Name | Description | Price
AAvailable_Dates
Property_Id | Activity_Name | Available_Date
Booking
Booking_Id | Booking_Date | No_of_Guests | Price | Start_Date | End_Date | Guest_SSN | Host_ID | Property_ID
Feedback
Booking_Id | Review | Rating | Guest_SSN | Host_ID
Payment
Payment_Id | Amount | Date | Payment_Mode | Payment_Status | Guest_SSN | Host_ID
Account
Acct_No | Acct_Type | Balance
Communicates
Guest_SSN | Host_Id
Browses
Guest_SSN | Property_Id | Location
Host_Acct
Host_ID | Acct_No
Guest_Acct
Guest_SSN | Acct_No

Step 4: Database Normalization Rules

Normalization - The process of decomposing unsatisfactory relations by breaking up their attributes into smaller relations.

Normal Form - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.

- 1st Normal Form Violations:

  Every relation in the above follows the 1NF rule as they do not contain any multivalued or composite attributes.

- 2nd Normal Form Violations:

  In ACTIVITY relation, primary key is (Property_Id, Activity_Name). Other attributes are Description and Price. Out of both non key attributes, Description is only dependent on Activity_Name but not on Property_Id. Hence there is partial dependency and this violates 2NF. In order to resolve this issue, we need to split the relation Activity to two tables i.e

  - Activity (Property_Id, Activity_Name, Price) with Property_Id, Activity_Name as primary key
  - Activity_Desc (Activity_Name, Description) with Activity_Name as primary key

  In BROWSES relation, primary key is (Guest_SSN, Property_Id). Only one non key attribute is Location. It is only dependent on Property_Id but not on Guest_SSN. Hence there is partial dependency and this violates 2NF. To resolve this issue, we need to split the relation Browses to two tables i.e

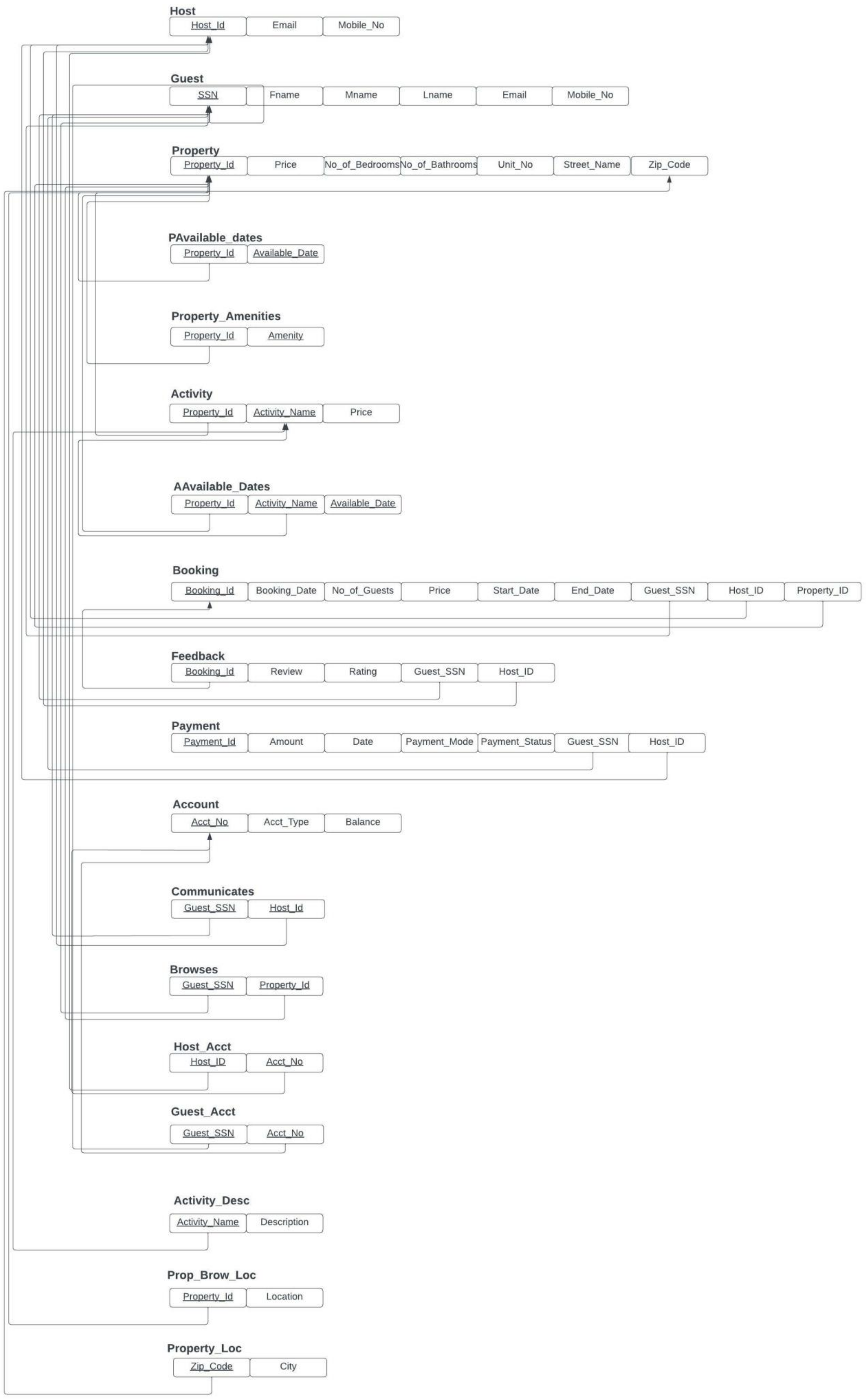
  - Browses (Guest_SSN, Property_Id) with Guest_SSN, Property_id as primary key
  - Prop_Brow_Loc (Property_Id, Location) with Property_Id as primary key

- 3rd Normal Form Violations:

  In PROPERTY relation, there is a transitive dependency between City and Zip_Code, since City can be determined by Zip_Code which is not a primary keys. This violates 3NF rule. In order to resolve this issue, we need to split relation Property into two tables i.e

  - Property (Property_Id, Price, No_of_Bedrooms, No_of_Bathrooms, Unit_No, Street_Name, Zip_Code) with Property_Id as primary key
  - Property_Loc (Zip_Code, City) with Zip_Code as primary key.

Step 5: Final Relational Schema after normalization to 3NF


Host
Host_Id
Email
Mobile_No
Guest
SSN
Fname
Mname
Lname
Email
Mobile_No
Property
Property_Id
Price
No_of_Bedrooms
No_of_Bathrooms
Unit_No
Street_Name
Zip_Code
PAvailable_dates
Property_Id
Available_Date
Property_Amenities
Property_Id
Amenity
Activity
Property_Id
Activity_Name
Price
AAvailable_Dates
Property_Id
Activity_Name
Available_Date
Booking
Booking_Id
Booking_Date
No_of_Guests
Price
Start_Date
End_Date
Guest_SSN
Host_ID
Property_ID
Feedback
Booking_Id
Review
Rating
Guest_SSN
Host_ID
Payment
Payment_Id
Amount
Date
Payment_Mode
Payment_Status
Guest_SSN
Host_ID
Account
Acct_No
Acct_Type
Balance
Communicates
Guest_SSN
Host_Id
Browses
Guest_SSN
Property_Id
Host_Acct
Host_ID
Acct_No
Guest_Acct
Guest_SSN
Acct_No
Activity_Desc
Activity_Name
Description
Prop_Brow_Loc
Property_Id
Location
Property_Loc
Zip_Code
City

Step 6: Creating Tables – With Foreign Keys, Null, Default attributes

```
CREATE TABLE Host (Host_Id INT NOT NULL, Email VARCHAR(20) NOT NULL, Mobile_No
VARCHAR(13), primary key(Host_Id));

CREATE TABLE Guest (SSN INT NOT NULL, Fname VARCHAR(20) NOT NULL, Mname
VARCHAR(20), Lname VARCHAR(20) NOT NULL, Email VARCHAR(20) NOT NULL,
Mobile_No VARCHAR(13), primary key(SSN));

CREATE TABLE Property (Property_Id INT NOT NULL, Price DECIMAL(10,2) NOT NULL,
No_of_bedrooms INT, No_of_bathrooms INT, Unit_No INT, Street_Name VARCHAR(20),
Zip_Code VARCHAR(10) NOT NULL, primary key(Property_Id));

CREATE TABLE PAvailable_dates (Property_Id INT NOT NULL, Available_Date DATE NOT
NULL, primary key(Property_Id, Available_Date));

CREATE TABLE Property_Amenities (Property_Id INT NOT NULL, Amenity VARCHAR(30)
NOT NULL, primary key(Property_Id, Amenity));

CREATE TABLE Activity (Property_Id INT NOT NULL, Activity_Name VARCHAR(20) NOT
NULL, Price DECIMAL(10,2) NOT NULL, primary key(Property_Id, Activity_Name));

CREATE TABLE AAvailable_Dates (Property_Id INT NOT NULL, Activity_Name
VARCHAR(20) NOT NULL, Available_Date DATE NOT NULL, primary key(Property_Id,
Activity_Name, Available_Date));

CREATE TABLE Booking (Booking_Id INT NOT NULL, Booking_Date DATE, No_of_Guests
INT NOT NULL, Price DECIMAL(10,2) NOT NULL, Start_Date DATE NOT NULL, End_Date
DATE, Guest_SSN INT NOT NULL, Host_ID INT NOT NULL, Property_ID INT NOT NULL,
primary key(Booking_Id));

CREATE TABLE Feedback (Booking_Id INT NOT NULL, Review VARCHAR(30), Rating INT,
Guest_SSN INT NOT NULL, Host_ID INT NOT NULL, primary key(Booking_Id));

CREATE TABLE Payment (Payment_Id INT NOT NULL, Amount DECIMAL(10,2), Date DATE,
Payment_Mode VARCHAR(15), Payment_Status VARCHAR(15), Guest_SSN INT NOT NULL,
Host_ID INT NOT NULL, primary key(Payment_Id));

CREATE TABLE Account (Acct_No BIGINT NOT NULL, Acct_Type VARCHAR(10), Balance
DECIMAL(10,2), primary key(Acct_No));

CREATE TABLE Communicates (Guest_SSN INT NOT NULL, Host_Id INT NOT NULL, primary
key(Guest_SSN, Host_Id));

CREATE TABLE Browses (Guest_SSN INT NOT NULL, Property_Id INT NOT NULL, primary
key(Guest_SSN, Property_Id));

CREATE TABLE Host_Acct (Host_ID INT NOT NULL, Acct_No BIGINT NOT NULL, primary
key(Host_Id, Acct_No));
```

CREATE TABLE Guest_Acct (Guest_SSN INT NOT NULL, Acct_No BIGINT NOT NULL, primary key(Guest_SSN, Acct_No));

CREATE TABLE Activity_Desc (Activity_Name VARCHAR(20) NOT NULL, Description VARCHAR(40), primary key(Activity_Name));

CREATE TABLE Prop_Brow_Loc (Property_Id INT NOT NULL, Location VARCHAR(20), primary key(Property_Id));

CREATE TABLE Property_Loc (Zip_Code VARCHAR(10) NOT NULL, City VARCHAR(20), primary key(Zip_Code));

TRIGGERED ACTIONS ON FOREIGN KEYS:

ALTER TABLE PAvailable_Dates ADD CONSTRAINT padpropid FOREIGN KEY(Property_Id) REFERENCES Property(Property_Id) ON DELETE CASCADE;

ALTER TABLE Property_Amenities ADD CONSTRAINT papropid FOREIGN KEY(Property_Id) REFERENCES Property(Property_Id) ON DELETE CASCADE;

ALTER TABLE Activity ADD CONSTRAINT apropid FOREIGN KEY(Property_Id) REFERENCES Property(Property_Id) ON DELETE CASCADE;

ALTER TABLE AAvailable_Dates ADD CONSTRAINT aadpropid FOREIGN KEY(Property_Id) REFERENCES Property(Property_Id) ON DELETE CASCADE;

ALTER TABLE AAvailable_Dates ADD CONSTRAINT aadactnm FOREIGN KEY(Activity_Name) REFERENCES Activity(Activity_Name) ON DELETE CASCADE;

ALTER TABLE Booking ADD CONSTRAINT bkguessn FOREIGN KEY(Guest_SSN) REFERENCES Guest(SSN) ON DELETE CASCADE;

ALTER TABLE Booking ADD CONSTRAINT bkhsid FOREIGN KEY(Host_ID) REFERENCES Host(Host_Id) ON DELETE CASCADE;

ALTER TABLE Booking ADD CONSTRAINT bkpropid FOREIGN KEY(Property_Id) REFERENCES Property(Property_Id) ON DELETE CASCADE;

ALTER TABLE Feedback ADD CONSTRAINT fdbkid FOREIGN KEY(Booking_Id) REFERENCES Booking(Booking_Id) ON DELETE CASCADE;

ALTER TABLE Feedback ADD CONSTRAINT fdguessn FOREIGN KEY(Guest_SSN) REFERENCES Guest(SSN) ON DELETE CASCADE;

ALTER TABLE Feedback ADD CONSTRAINT fdhsid FOREIGN KEY(Host_ID) REFERENCES Host(Host_Id) ON DELETE CASCADE;

ALTER TABLE Payment ADD CONSTRAINT ptguessn FOREIGN KEY(Guest_SSN) REFERENCES Guest(SSN) ON DELETE CASCADE;

```
ALTER TABLE Payment ADD CONSTRAINT pthsid FOREIGN KEY(Host_ID) REFERENCES
Host(Host_Id) ON DELETE CASCADE;

ALTER TABLE Communicates ADD CONSTRAINT cmtguessn FOREIGN KEY(Guest_SSN)
REFERENCES Guest(SSN) ON DELETE CASCADE;

ALTER TABLE Communicates ADD CONSTRAINT cmthsid FOREIGN KEY(Host_ID)
REFERENCES Host(Host_Id) ON DELETE CASCADE;

ALTER TABLE Browses ADD CONSTRAINT brguessn FOREIGN KEY(Guest_SSN)
REFERENCES Guest(SSN) ON DELETE CASCADE;

ALTER TABLE Browses ADD CONSTRAINT brpropid FOREIGN KEY(Property_Id)
REFERENCES Property(Property_Id) ON DELETE CASCADE;

ALTER TABLE Host_Acct ADD CONSTRAINT hahsid FOREIGN KEY(Host_ID) REFERENCES
Host(Host_Id) ON DELETE CASCADE;

ALTER TABLE Host_Acct ADD CONSTRAINT haactno FOREIGN KEY(Acct_No) REFERENCES
Account(Acct_No) ON DELETE CASCADE;

ALTER TABLE Guest_Acct ADD CONSTRAINT gaguessn FOREIGN KEY(Guest_SSN)
REFERENCES Guest(SSN) ON DELETE CASCADE;

ALTER TABLE Guest_Acct ADD CONSTRAINT gaactno FOREIGN KEY(Acct_No) REFERENCES
Account(Acct_No) ON DELETE CASCADE;

ALTER TABLE Activity_Desc ADD CONSTRAINT adactnm FOREIGN KEY(Activity_Name)
REFERENCES Activity(Activity_Name) ON DELETE CASCADE;

ALTER TABLE Property ADD CONSTRAINT propzpcd FOREIGN KEY(Zip_Code)
REFERENCES Property_Loc(Zip_Code) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Prop_Brow_Loc ADD CONSTRAINT pblpropid FOREIGN KEY(Property_Id)
REFERENCES Property(Property_Id) ON DELETE CASCADE;
```

TRIGGERS AND STORED PROCEDURES:

STORED PROCEDURE #1: A procedure to register a guest

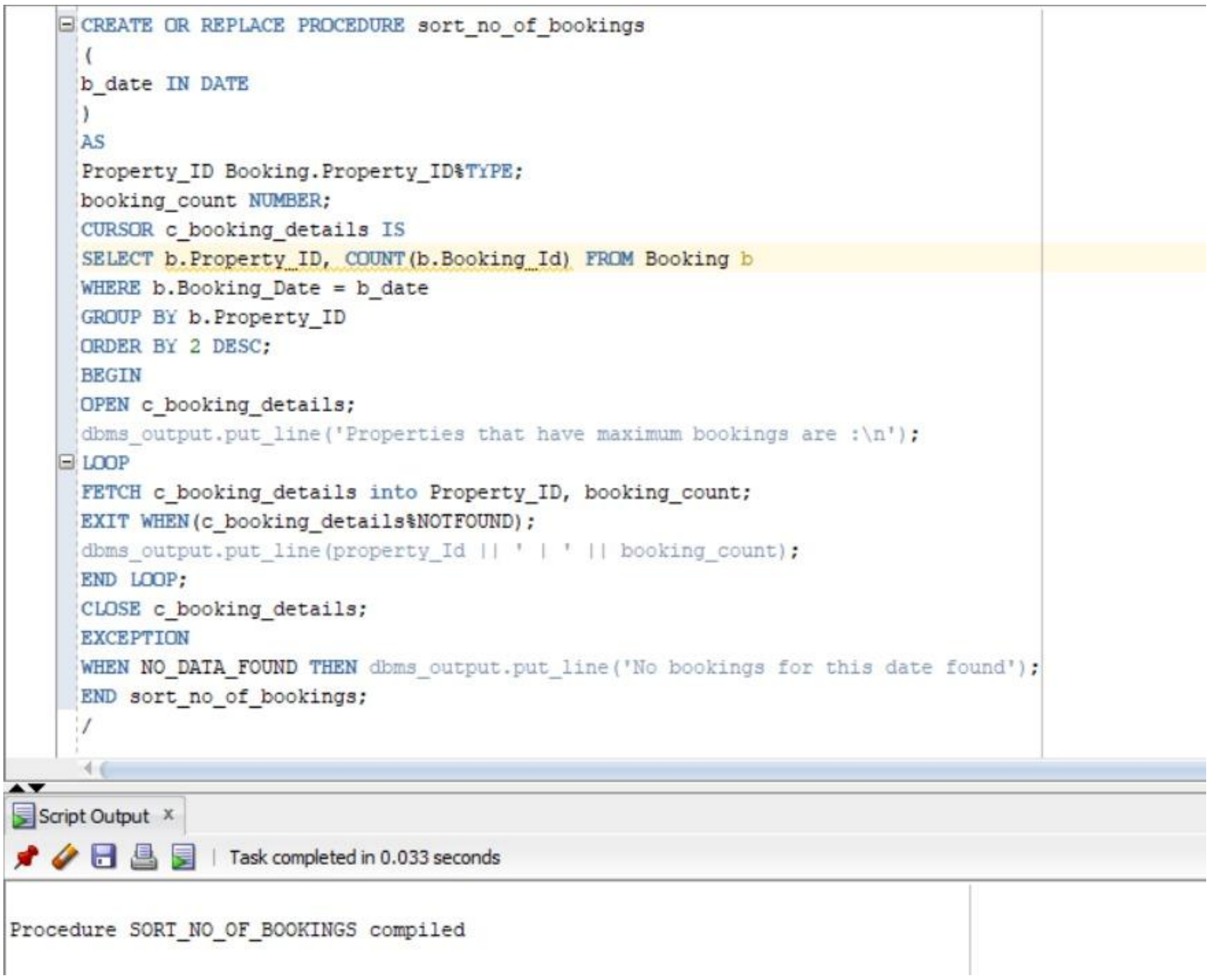
```
CREATE OR REPLACE PROCEDURE register_guest (
SSN IN INTEGER,
Fname IN VARCHAR,
Mname IN VARCHAR,
Lname IN VARCHAR,
Email IN VARCHAR,
Mobile_No IN VARCHAR) AS
BEGIN
INSERT INTO Guest VALUES (SSN,
Fname,
Mname,
Lname,
Email,
Mobile_No
);
END register_guest;
```

STORED PROCEDURE #2: A procedure to register host

```
CREATE OR REPLACE PROCEDURE register_host (
Host_Id IN INTEGER,
Email IN VARCHAR,
Mobile_No IN VARCHAR) AS
BEGIN
INSERT INTO Host VALUES (
Host_Id,
Email,
Mobile_No
);
END register_host;
```

STORED PROCEDURE #3: A procedure to sort Property IDs according to highest number of bookings on given date

```
CREATE OR REPLACE PROCEDURE sort_no_of_bookings
(
b_date IN DATE
)
AS
Property_ID Booking.Property_ID%TYPE;
booking_count NUMBER;
CURSOR c_booking_details IS
SELECT b.Property_ID, COUNT(b.Booking_Id) FROM Booking b
WHERE b.Booking_Date = b_date
GROUP BY b.Property_ID
ORDER BY 2 DESC;
BEGIN
OPEN c_booking_details;
dbms_output.put_line('Properties that have maximum bookings are :\n');
LOOP
FETCH c_booking_details into Property_Id, booking_count;
EXIT WHEN(c_booking_details%NOTFOUND);
dbms_output.put_line(Property_Id || ' | ' || booking_count);
END LOOP;
CLOSE c_booking_details;
EXCEPTION
WHEN NO_DATA_FOUND THEN dbms_output.put_line('No bookings for this date
found');
END sort_no_of_bookings;
```

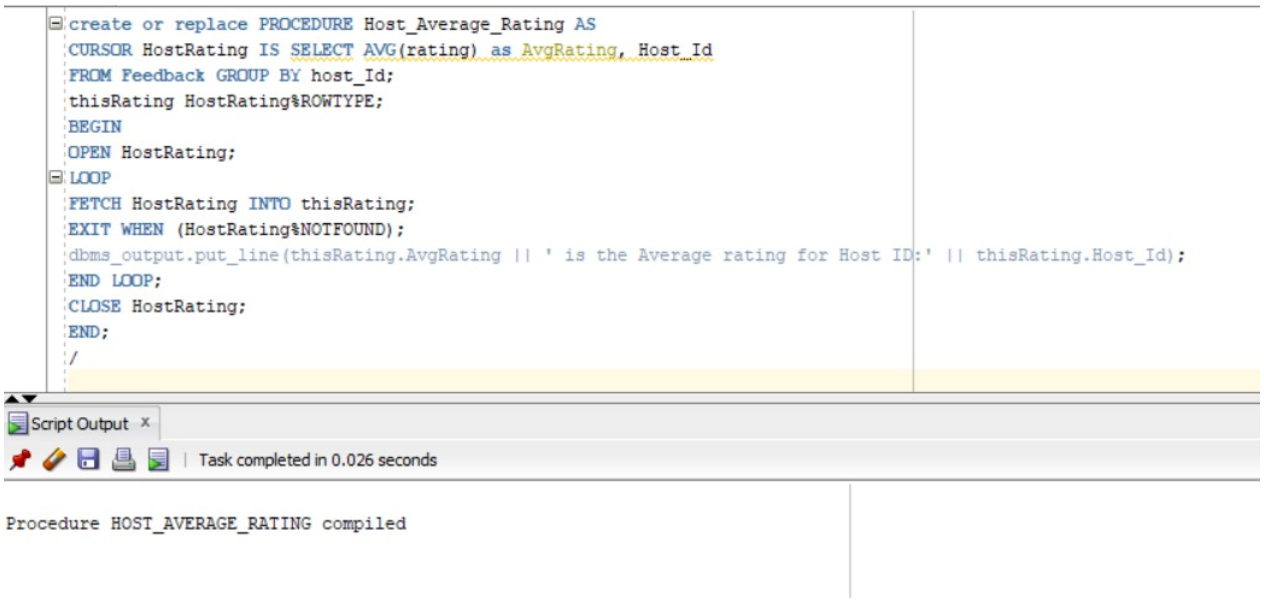


```
CREATE OR REPLACE PROCEDURE sort_no_of_bookings
(
b_date IN DATE
)
AS
Property_ID Booking.Property_ID%TYPE;
booking_count NUMBER;
CURSOR c_booking_details IS
SELECT b.Property_ID, COUNT(b.Booking_Id) FROM Booking b
WHERE b.Booking_Date = b_date
GROUP BY b.Property_ID
ORDER BY 2 DESC;
BEGIN
OPEN c_booking_details;
dbms_output.put_line('Properties that have maximum bookings are :\n');
LOOP
FETCH c_booking_details into Property_ID, booking_count;
EXIT WHEN(c_booking_details%NOTFOUND);
dbms_output.put_line(property_Id || ' | ' || booking_count);
END LOOP;
CLOSE c_booking_details;
EXCEPTION
WHEN NO_DATA_FOUND THEN dbms_output.put_line('No bookings for this date found');
END sort_no_of_bookings;
/
```

```
Procedure SORT_NO_OF_BOOKINGS compiled
```

STORED PROCEDURE #4: A procedure to calculate average rating of host

```
CREATE or REPLACE PROCEDURE Host_Average_Rating AS
CURSOR HostRating IS SELECT AVG(rating) as AvgRating, Host_Id
FROM Feedback GROUP BY host_Id;
thisRating HostRating%ROWTYPE;
BEGIN
OPEN HostRating;
LOOP
FETCH HostRating INTO thisRating;
EXIT WHEN (HostRating%NOTFOUND);
dbms_output.put_line(thisRating.AvgRating || ' is the Average rating for Host ID:' ||
thisRating.Host_Id);
END LOOP;
CLOSE HostRating;
END;
```

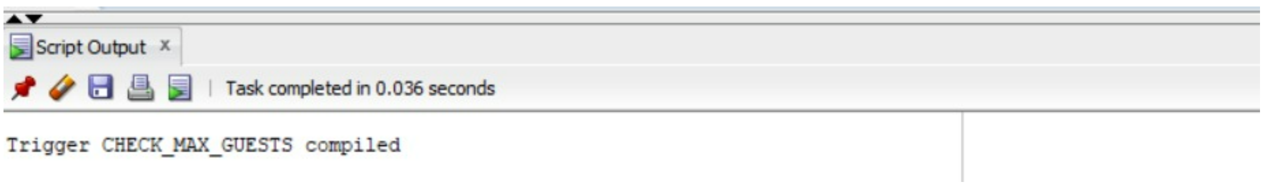


```
create or replace PROCEDURE Host_Average_Rating AS
CURSOR HostRating IS SELECT AVG(rating) as AvgRating, Host_Id
FROM Feedback GROUP BY host_Id;
thisRating HostRating%ROWTYPE;
BEGIN
OPEN HostRating;
LOOP
FETCH HostRating INTO thisRating;
EXIT WHEN (HostRating%NOTFOUND);
dbms_output.put_line(thisRating.AvgRating || ' is the Average rating for Host ID:' || thisRating.Host_Id);
END LOOP;
CLOSE HostRating;
END;
/
```

```
Procedure HOST_AVERAGE_RATING compiled
```

TRIGGER #1: A trigger to check if the number of guests selected for booking exceeds maximum permissible count

```
CREATE or REPLACE TRIGGER check_max_guests
BEFORE INSERT OR UPDATE OF No_of_Guests on Booking
FOR EACH ROW
DECLARE
guests_count Booking.No_of_Guests%TYPE
BEGIN
SELECT No_of_Guests INTO guests_count FROM Booking WHERE
Booking_Id=:NEW.Booking_Id;
IF :NEW.No_of_Guests > 6 THEN
    RAISE_APPLICATION_ERROR(-20000, 'The maximum number of people allowed for
the property is 6')
END IF;
END;
```



```
Trigger CHECK_MAX_GUESTS compiled
```

TRIGGER #2: A Trigger that will calculate average rating of hosts based on guest feedback and will be triggered whenever guest gives feedback to host.

(Trigger will call procedure Host_Average_Rating)

```
CREATE OR REPLACE TRIGGER host_rating_update
AFTER INSERT ON Feedback FOR EACH ROW
DECLARE
BEGIN
END;
PRAGMA AUTONOMOUS_TRANSACTION;
Host_Average_Rating(:NEW.Host_Id);
```

TRIGGER #3: A Trigger that will block dates for booking if there is any overlap between start and end dates of booking

```
CREATE or REPLACE TRIGGER block_dates
BEFORE INSERT OR UPDATE OF Start_Date on BOOKING
FOR EACH ROW
DECLARE
start_dt Booking.Start_Date%TYPE;
end_dt Booking.End_Date%TYPE;
BEGIN
SELECT Start_Date, End_Date into start_dt,end_dt FROM Booking
WHERE Property_Id = :NEW.Property_Id
AND :NEW.Start_Date NOT BETWEEN Start_Date AND End_Date
AND :NEW.End_Date NOT BETWEEN Start_Date AND End_Date
AND Start_Date NOT BETWEEN :NEW.Start_Date AND :NEW.End_Date
AND End_Date NOT BETWEEN :NEW.Start_Date AND :NEW.End_Date;
END;
```