**SECTION A:**

|  |  |
| --- | --- |
| Entities | Space, Entire home/apartment, private room, shared room, Reservation, Transaction, Coupon, Card |
| Attributes | Entire home/apartment, private room, shared room: Number of Guests, Amount of Bedrooms, Amount of Beds, Amount of Bathrooms, Amenities  Reservation: Confirmation Code, Guest Name, Check-In Date, Check-Out Date, Total Nights, Number of Guests, Unit  Coupon: Coupon Code, Coupon Amount, Discount Amount  Card: Card Number, Card Type  Payment: Transaction ID, Payee ID, Rate per Night, Total Amount, Amount Paid, Status |
| Relationships, Cardinalities & Optionalities | Each Home/Apartment belongs to one space only but each space may belong to one home/Apartment(1/1)  Each Private Room belongs to one space only but each space may belong to one Private Room(1/1)  Each Shared Room belongs to one space only but each space may belong to one Shared Room(1/1)  Each Guest may Reserve 0/1 space only and each space may be reserved by 0/M guests(1/M)  Each Coupon may belong to 0/M guest & each Guest may have 0/M Coupons(M/M)  Each Card must belong to 0/1 Guest only,but each Guest may have 0/M Cards(1/M) |
| Data Redundancies | Some of the attributes of Home/Apartment, Private Room & Shared room are the same with each other(e.x:No. of beds, No. of bathrooms, etc) & all three belong to Spaces; |

**FUNCTIONAL DEPENDENCIES(SECTION A):**

1st Normal Form:

The table is already in first normal form due to non existence of multi-valued attributes

2nd Normal Form:

Transaction ID --> Payee ID, Amount Paid, Payment Date, status(transaction table)

3rd Normal Form:

Coupon Code --> Coupon Amount, Discounted Amount(coupon table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Transaction ID** | **Payee ID** | **Amount Paid** | **Payment Date** | **Status** |
| TX281219926 | val033 | RM750 | 28 Dec 2019 | Successful |
| TX6120140 | barbara.s | RM680 | 6 Jan 2020 | Successful |
| TX7120075 | val033 | RM1,000 | 7 Jan 2020 | Void |
| … | … | … | … | … |

Figure: Transaction table

|  |  |  |
| --- | --- | --- |
| **Coupon Code** | **Coupon Amount** | **Discounted Amount** |
| - | - | - |
| NY015 | 0.15 | RM120 |
| NY020 | 0.20 | RM250 |
| … |  | … |

Figure: Coupon table

**SECTION B:**

|  |  |
| --- | --- |
| Entities | Coupon, User[Hosts, Guests](Hosts and Guests are subtypes of User), Space, Assignment(intersection entity between space and host), Review(intersection entity between host and guest), ID(owned by guest and host), Account, Credit card |
| Attributes | Coupon: Coupon Code, Description, Start Date, End Date, Coupon Amount  User: User ID, Password, First Name, Last Name, Email, Date of Birth, Address  Host: User ID, Password, First Name, Last Name, Email, Date of Birth, Address  Guest: User ID, Password, First Name, Last Name, Email, Date of Birth, Address, Credit Card No., Expiry Date  Credit Card: Credit Card No., Expiry Date  ID: ID number, ID type  Account: Beneficiary Account Number, Beneficiary Account Holder, Bank Name, Bank Country  Space: Space ID, Space Type ID, Space Type  Assignment: Host, remark  Review: Communication, cleanliness, public feedback, private feedback  Guest review form: Communication, cleanliness, public feedback, private feedback, accuracy, location, check-in, value  Host review form: Communication, cleanliness, public feedback, private feedback, Observance of house rules |
| Relationships, cardinalities and optionalities | Each space may be hosted by many hosts but each host must host at least one space. (1/M)  Each guest may review zero or many hosts but each host must review one or many hosts. (M/M)  Each guest may use zero or many coupons but each coupon may be used by one or many guests. (M/M)  Each guest may make one or many transactions but each transaction must be made by one guest. (1/M) |
| Data redundancies | Every attribute of the user HOST is also an attribute of the user GUEST except Credit Card No. and Credit Card Expiry Date.  There are some common attributes between host review form and |

**FUNCTIONAL DEPENDENCIES(SECTION B):**

1st table(Coupon Table) is already normalized to third normal form.

Coupon code 🡪 Description, Start date, End date, Coupon Amount

2nd table:

The Space table was in 0NF.

1st Normal Form:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Space ID | Space Type ID | Space Type | Host | Remark |
| P003 | ST001 | Entire Home | Barbara Smith | 1 |
| P003 | ST001 | Entire Home | Mike Smith | 0 |
| P004 | ST002 | Private Room | Rafi Martin | 1 |
| P004 | ST002 | Private Room | Maisha | 0 |
| P004 | ST002 | Private Room | Nurfarah Aziim | 0 |
| P018 | ST003 | Shared Room | Barbara Smith | 1 |
| P018 | ST003 | Shared Room | Mike Smith | 0 |

Figure: Table in 1NF

2nd Normal Form:  
Space\_ID 🡪 Space\_Type\_ID, Space\_Type (partial) (Space table)

|  |  |  |
| --- | --- | --- |
| Space ID | Space Type ID | Space Type |
| P003 | ST001 | Entire Home |
| P003 | ST001 | Entire Home |
| P004 | ST002 | Private Room |
| P004 | ST002 | Private Room |
| P004 | ST002 | Private Room |
| P018 | ST003 | Shared Room |
| P018 | ST003 | Shared Room |

Figure: Space table

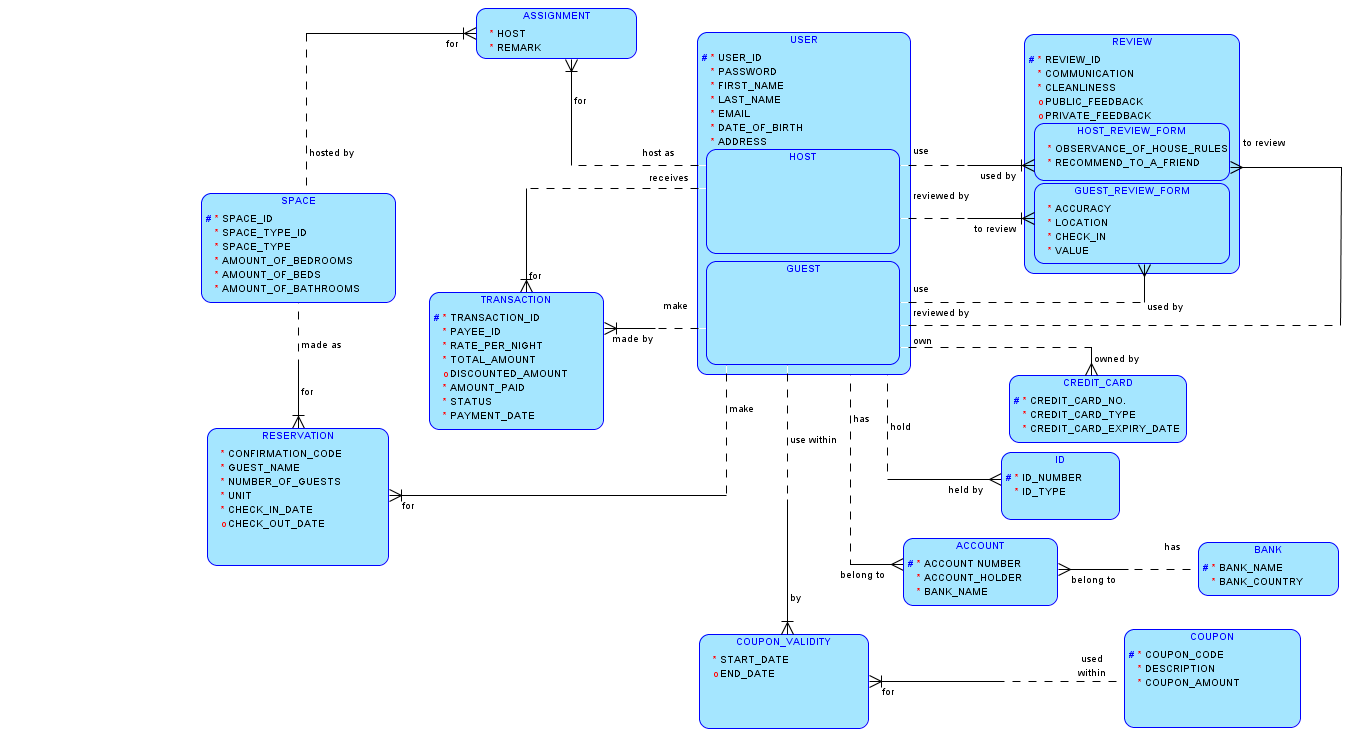
3rd Normal Form:

Host 🡪 Remark (host type table) (transitive dependency)

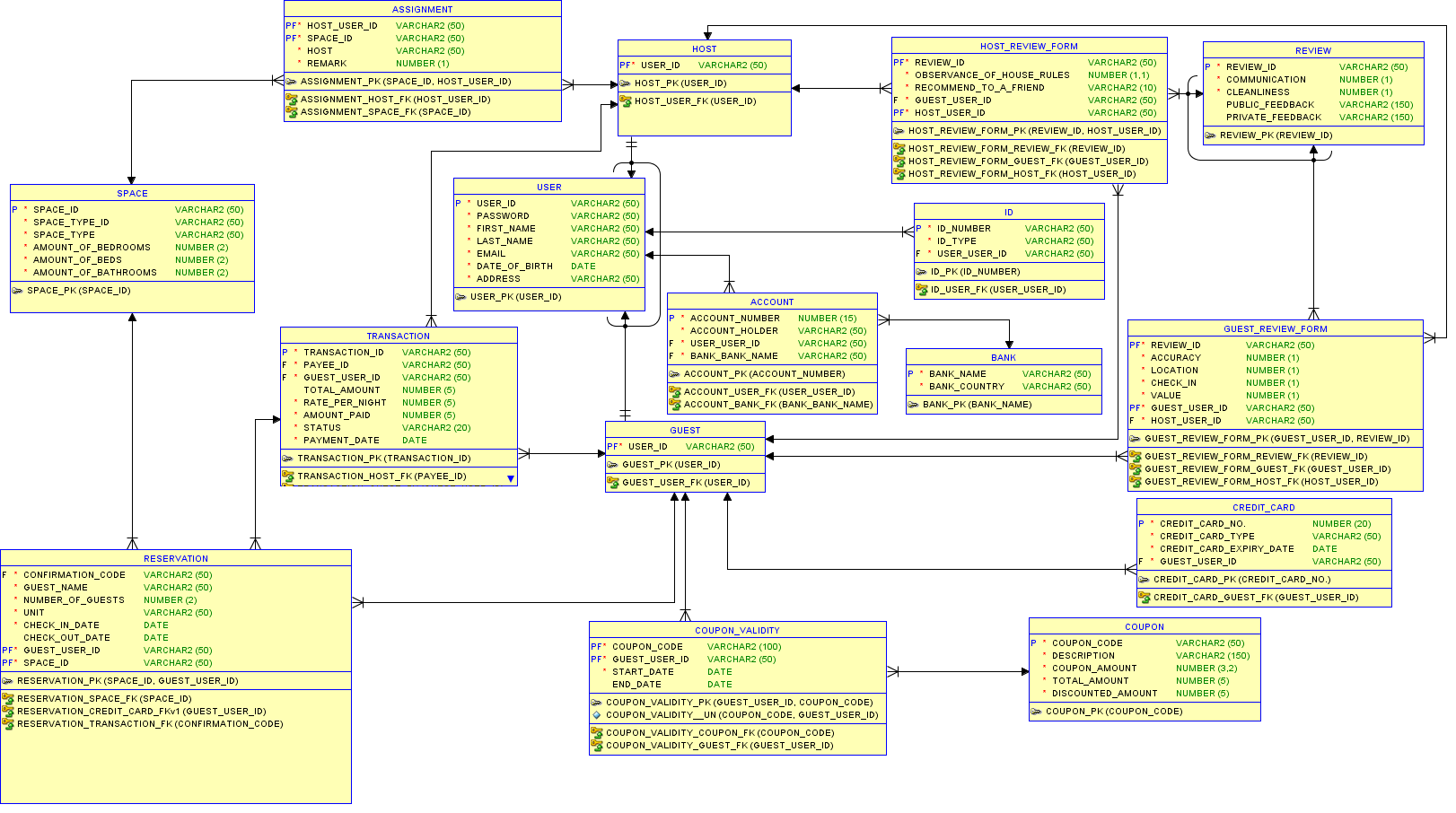
|  |  |
| --- | --- |
| Host | Remark |
| Barbara Smith | 1 |
| Mike Smith | 0 |
| Rafi Martin | 1 |
| Maisha | 0 |
| Nurfarah Aziim | 0 |
| Barbara Smith | 1 |
| Mike Smith | 0 |

Figure: Host type table

**Conceptual Model**

****

**Physical Model**

****

**ASSUMPTIONS MADE:**

A new attribute called review ID was added as an entity cannot be created without a primary UID in the logical model.

Public and private feedback were also put as optional in the review table as it depends on the individual user who’s writing the review.

**IMPLEMENTATION *(Refer to SQL queries)***

**Constraint Testing**

|  |  |
| --- | --- |
| Test Number | 1 |
| Constraint type | Primary Key (guest\_accounts) |
| Test Description | I’ll try to insert the account number that already exist in the table for another guest. |
| SQL Query | INSERT INTO guest\_accounts  VALUES  (4904635385,'jotaro.k','Jotaro Kujo','RHB'); |
| Expected Output | 1 row inserted |
| Actual Output | task 5 amir |
| Action Taken | *I would ask the guest again for another account number.* |
| Discussion | *The data redundancy has been maintained as there cannot be multiple guests having the same account number.* |

|  |  |
| --- | --- |
| Test Number | 2 |
| Constraint type | Foreign key(Transactions table) |
| Test Description | Insert a new record to the transactions table with payee\_ID(host\_user\_ID in parent table) that does NOT exist in the parent table. |
| SQL Query | INSERT INTO transactions  VALUES  ('TX777637472','tracy.n','speedwagon.p','440','220','220','Void','09-Mar-2020'); |
| Expected Output | The query should be rejected as foreign key doesn’t match any primary key |
| Actual Output | task 5 constraint testing |
| Action Taken | *The query needs to be revised so that person would make the transaction to only one of the recognized hosts in the parent table.* |
| Discussion | *We can ensure data consistency and validity as transactions cannot be made to the host who doesn’t even exist in the database.* |

**Data Testing**

Test 1: Basic data testing

|  |  |
| --- | --- |
| Test Scenario | The scenario be that I want to find the list of guests that reserved a room that have apartment in the description. |
| SQL Query | SELECT \* FROM reservations  WHERE lower(unit) LIKE '%apartment%'; |
| Result | *task 6 part 1 amir* |
| Discussion | *This way HELP Space can find all the guests that have rented an entire home/apartment.* |

|  |  |
| --- | --- |
| Test Scenario | List host user ID, first name and last name sorted by first name for hosts who are not hosting any space. |
| SQL Query | SELECT host\_user\_ID, first\_name, last\_name  FROM hosts  WHERE host\_user\_ID NOT IN (SELECT host\_user\_ID FROM assignment)  ORDER BY first\_name; |
| Result | *task 6 part 1* |
| Discussion | *This testing is useful when we want a host to host a new space. If a host who is already hosting other spaces is chosen as a host, it might lead to over workload. So, it is highly advised to not choose a host who is managing more than 2 spaces but rather someone who is hosting only one space or someone who is not hosting any space at all.* |

Test 2: Advanced data testing

|  |  |
| --- | --- |
| Test Scenario | The scenario be that I want to know the number of guests that had successful transactions to the host named ‘Barbara’. |
| SQL Query | SELECT COUNT (\*) AS "Number Of Guests" FROM transactions  WHERE lower(status) = 'successful'  AND lower(payee\_id) = 'barbara.s'; |
| Result | *task 6 part 2 amir* |
| Discussion | *This way HELP Space can easily find the number of successful transactions made for Barbara Smith.* |

|  |  |
| --- | --- |
| Test Scenario | List space\_ID, number of co-hosts for all spaces with the number of co-hosts being at least one or more and sort them by space\_ID. |
| SQL Query | SELECT space\_ID, COUNT(host\_user\_ID) AS "Number of hosts"  FROM assignment JOIN host\_types USING (host)  WHERE remark = 0  GROUP BY space\_ID  HAVING COUNT(host\_user\_ID) >= 1  ORDER BY space\_ID; |
| Result | *task 6 part 2* |
| Discussion | *This testing is useful when we want to see whether there are enough co-hosts for each space to help the main host. It would lead to over work load for the main host if there isn’t enough co-host assigned to one space. So, we can assign new co-hosts to spaces not having enough co-hosts to prevent over workload for the main host.* |

**View Creation**

|  |  |
| --- | --- |
| View Scenario | Create a view table that consist of confirmation code & the status of the transaction. |
| SQL Query | CREATE VIEW Reservation\_Payment  AS  SELECT unit,check\_in\_date, total\_amount, amount\_paid, status, reservations.guest\_user\_ID  FROM reservations,transactions  WHERE reservations.confirmation\_code=transactions.transaction\_ID; |
| Result | task 7 amir |
| Discussion | *This way we can easily access the data from two important table that determines the reservation & payment status without having to spend more time trying to search for the data with a longer query.* |

|  |  |
| --- | --- |
| View Scenario | A view table that list details about spaces reserved more than once by guests and sorted according to space\_ID. |
| SQL Query | CREATE VIEW Popular\_Spaces  AS  SELECT spaces.space\_ID,space\_type\_ID,space\_type, COUNT(Guest\_user\_ID) AS "Number of Guests"  FROM spaces,reservations  WHERE spaces.space\_ID = reservations.space\_ID  GROUP BY spaces.space\_ID,space\_type\_ID,space\_type  HAVING COUNT(Guest\_user\_ID) > 1  ORDER BY spaces.space\_ID; |
| Result | task 7 |
| Discussion | *The manager from the marketing department will no longer need to type in the long queries to look at popular spaces which are reserved by many guests. The marketing manager can continue to promote more effectively for the popular spaces and can make improvements in promoting the unpopular ones. They can also find out what makes the popular spaces popular and apply that to the unpopular ones.* |

**DISCUSSION:**

**Advantages and disadvantages of a relational database for the HELP Space case study**

Just like in every database management system, there are pros and cons in the way the data is stored and retrieved. In this particular case, the advantages are as follows:

1. Relational database systems can easily be understood and can cut down extra workload.
2. The queries used to retrieve, store, or manipulate data from the HELP Space database system are simple enough for users to understand.
3. There will no longer be data duplication in HELP Space database system as the data is stored only once.
4. Only a selective number of users can access the HELP Space database system which makes it reliable and will prevent data leakage.

The disadvantages can be listed as follows as well:

1. If any particular damage is done to the HELP Space database system, the correlated applications which rely on it will be affected and may even lead to data loss.
2. Maintaining the database system can be costly.

In conclusion, the advantages outweigh the disadvantages which lead to the wide usage of database management systems across every business field just like the HELP Space database system.

**How database management system overcome the problems associated with data redundancy**

Data redundancy is the situation where duplicate data is stored in separate places. This situation can lead to a cascade of data inconsistencies in the different places where the data is stored. Duplicate data being stored can also lead to increase in the cost of storing the data. Database management system can solve these cascades of problems by normalizing the data stored in the tables before adding it to the database system itself.

Normalization is initiated with firstly determining the functional dependencies in the tables where the data are stored. Then, we normalize it to first normal form, second normal form and then the third normal form. The dependent columns are put together with the independent columns and separated from the non-associated columns which lead to the formation of a set of tables. The duplicate columns are then deleted to form a set of tables that are consistent with each other. This is when we consider the tables to be normalized as the dependencies are enforced correctly.

In conclusion, the consistency throughout the tables implies that there is no longer any data redundancy present in the tables which contribute to a more accurate and organized database.