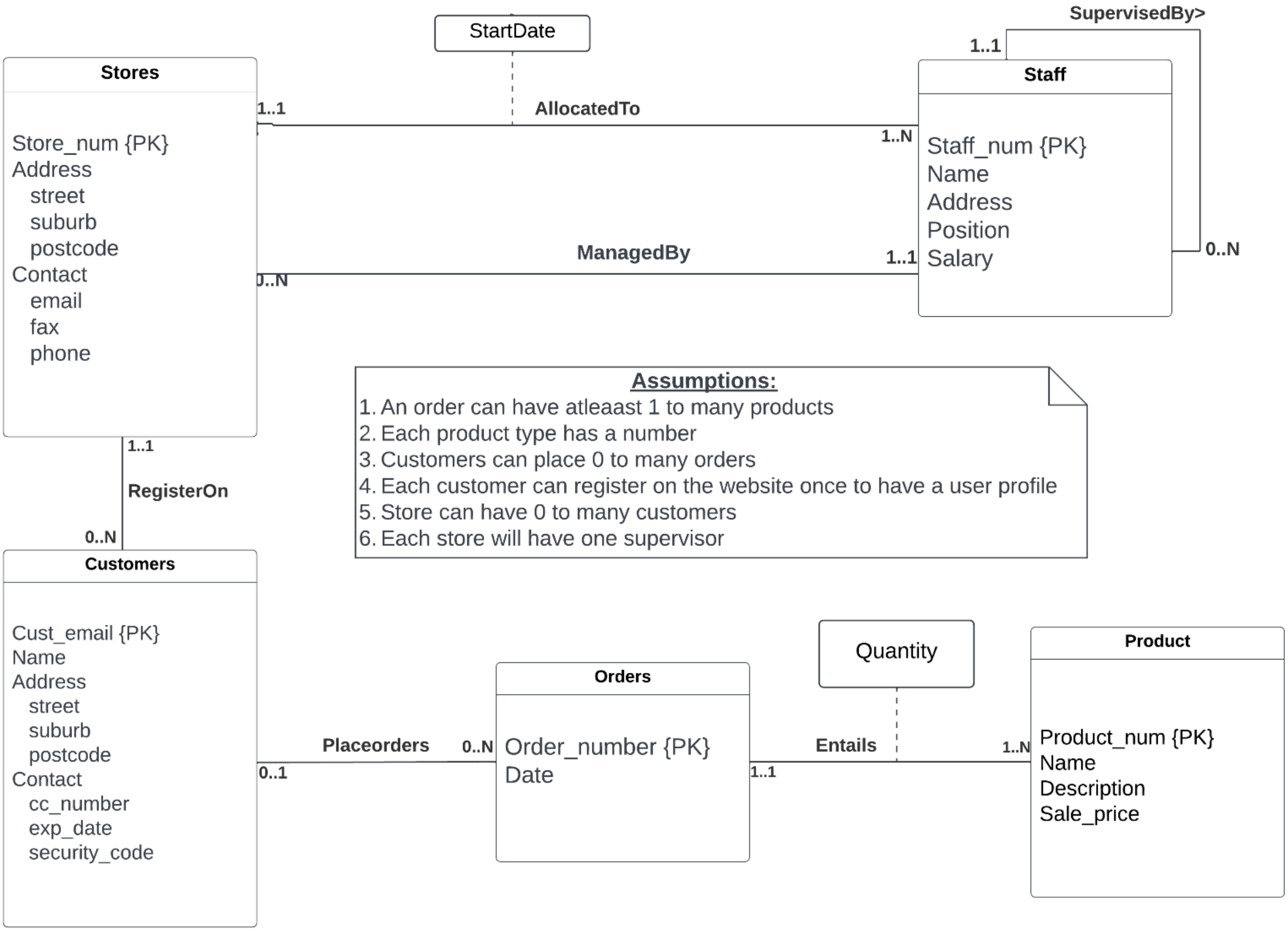
Database Concepts – Assignment 1: S3828461

Part A:

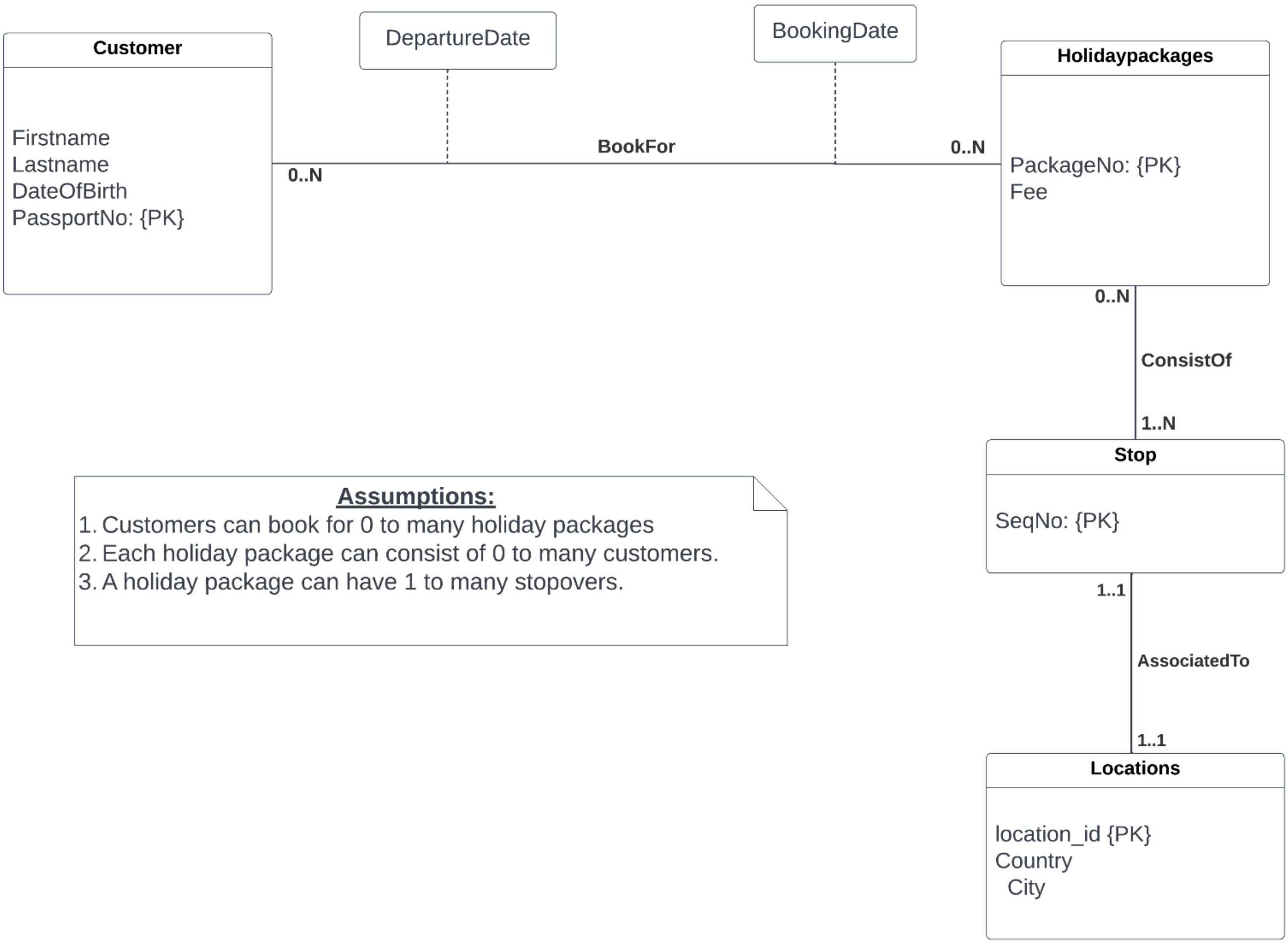
Task1



Assumptions:

1. An order can have atleaast 1 to many products
2. Each product type has a number
3. Customers can place 0 to many orders
4. Each customer can register on the website once to have a user profile
5. Store can have 0 to many customers
6. Each store will have one supervisor

Task 2: Part A:

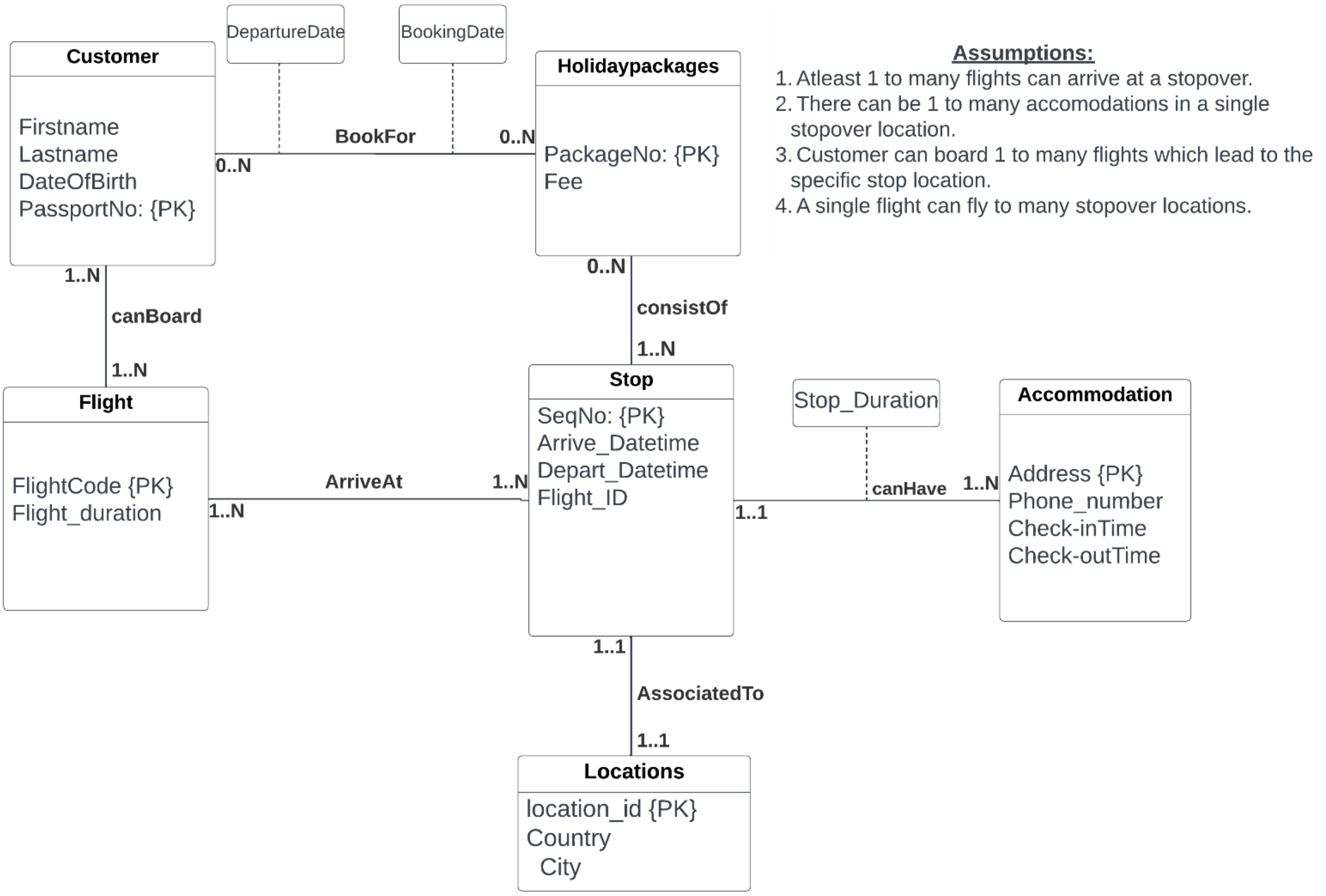


Assumptions:
1. Customers can book for 0 to many holiday packages
2. Each holiday package can consist of 0 to many customers.
3. A holiday package can have 1 to many stopovers.

1. In the table Locations (location_id {PRIMARY KEY}, Country, City) in the system is mapped with the stop table with a 1:1 relationship. They can be combined to a new entity with PK of one entity being the PK of the new entity, during ER schema mapping.

   By this we will know the Stop over locations, the flight arrives at and departs from.

Task2: Part B:



Assumptions:

1. Atleast 1 to many flights can arrive at a stopover.
2. There can be 1 to many accomodations in a single stopover location.
3. Customer can board 1 to many flights which lead to the specific stop location.
4. A single flight can fly to many stopover locations.

1. The customer can arrive at the stop over location by flight at a specific Datetime. The Arrival datetime and the depart datetime will be stored in the Stop table.
2. The duration of the flight and unique code is stored in the Flight table.
3. There can be many accommodations in a single stopover, this depends on the stop over duration. The stop table will have a one-to-many relationship with the accommodation table.

4. The Flight_ID is stored in the Stop table as a foreign key, which is the FlightCode {PRIMARY KEY} in the flight table. The key relation to this is:
   Stop.Flight_ID {Foreign Key} → *Flight.FlightCode {Primary Key}*
   By this, we will be able to track the flight which arrives and departs from the stop.

5. In the table Locations (location_id {PRIMARY KEY}, Country, City) in the system is mapped with the stop table with a 1:1 relationship. They can be combined to a new entity with PK of one entity being the PK of the new entity, during ER schema mapping. *By this we will know the Stop over locations the flight arrives and departs.*

6. Overall, we will be able to track the customer's flight details, stop over location details, arrive and depart datetime details.

Part A : Task 3

1. **Mapping Strong Entities:**

Branch(Branch No, Street, Suburb, Postcode, Telephone)

Customer(Cust No, Street, Suburb, Postcode, Email)

Staff(Staff No, Name, Address, Position, Salary)

2. **Mapping Weak Entities:**

Job(CCust No*, Job No, Fault)

3. **1:1 Relationships:**

No action required

4. **Map 1:N Relationships:**

Staff(Staff No, Name, Address, Position, Salary, BNo*)

Branch(Branch No, Street, Suburb, Postcode, Telephone, SNo*)

Job(CCust No*, Job No, SNo*, Fault)

5. **Map M:N Relationships:**

No action required

6. **Multi-Valued Attributes:**

Telebranchnum(Branch No*, Telephone)

Branch(Branch No, Street, Suburb, Postcode, SNo*)

7. **High degree relationships:**

No Action Required

8. **Special Case: Recursive Relationship:**

Staff(Staff No, Name, Address, Position, Salary, SuperStaffNo*)

Final Schema:

Branch(Branch No, Street, Suburb, Postcode, SNo*)

Customer(Cust No, Street, Suburb, Postcode, Email)

Staff(Staff No, Name, Address, Position, Salary, SuperStaffNo*)

Job(CCust No*, Job No, SNo*, Fault)

Telebranchnum(Branch No*, Telephone)

Part B:

Task4:

4.1:

NO, the database schema does not ensure that there is a department associated with each employee. The connection between keys in the department and the employee table is: Departments.manager_id → Employees.employee_id.

In order to ensure that there is a department associated with each employee the connection between the table should be, Departments.department_id → Employees.employee_id which is not present.

4.2:

The connection between keys in the Jobs and the Job history table is:

Job history.job_id → Jobs.job_id.

To store multiple job histories for the same employee with the same start and end dates but with different titles, we can convert the job_id* in the Job history table to a composite key job_id* .

4.3:

Yes, the following SQL statements will work efficiently.

No, they are NOT sufficient to achieve the requirements specified above.

As mentioned, the managers of all the three Human Resource sub-departments will have to report to a single director. But the data to store the director information has not been mentioned.

There can be a separate director ID in the department table, which stores the ID of the director the manager has to report to. This director ID column can be a foreign key in the department table linked with employee_id of the employee table.

Departments.director_id → Employee.employee_id

4.4:

NO, this request cannot be completed.

The following SQL query will execute perfectly, but to find all contracts Adam Smith used to have cannot be found.

Another SQL statement needs to be executed in order to add the details of Adam Smith such as employee_id, start_date, end_date, job_id, and department_id into the Job history table. This table stores the contracts of the user.

4.5:

The connection between keys in the department and the Locations table is:

Departments.location_id → Locations.location_id.

Location_id in the departments table is the foreign key and it is the primary key in the locations table. The foreign key will not be updated automatically, it will lead to a referential integrity constraint.

Referential Integrity Constraint: this requires that a foreign key must have a matching primary key or the value must be null. This is between the parent and the child table. Basically, the reference from one table to another table must be valid.

4.6:

```
CREATE TABLE Locations (
location_id INT NOT NULL,
country_id INT,
street_address VARCHAR(50) ,
postal_code INT,
city VARCHAR(20),
state_province VARCHAR(20),
PRIMARY KEY (location_id),
FOREIGN KEY (country_id) REFERENCES Countries (country_id)
)
```

4.7:

```
CREATE TABLE JobHistory (
employee_id INT NOT NULL,
start_date DATE NOT NULL,
end_date DATE NOT NULL,
job_id INT,
department_id INT,
PRIMARY KEY (employee_id, start_date, end_date),
FOREIGN KEY (employee_id) REFERENCES Employees (employee_id),
FOREIGN KEY (job_id) REFERENCES Jobs (job_id),
FOREIGN KEY (department_id) REFERENCES Departments (job_id),
)
```

4.8:

```
INSERT INTO Locations VALUES ('40', ‘666 Diablo st’, ‘46001’, ‘El Carmen’, ‘Valencia’,’3’);
```