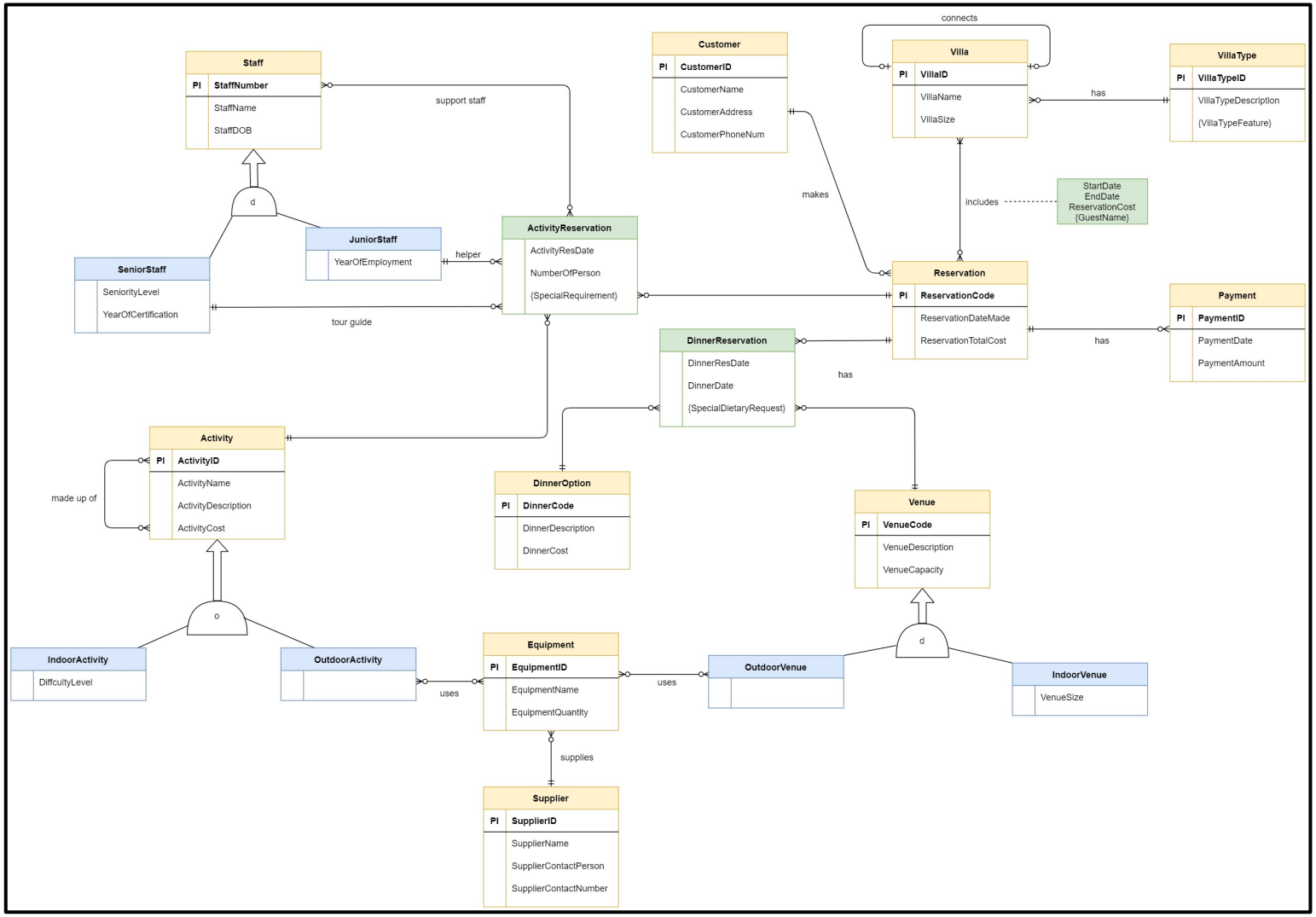
COMP1350 2020 – ASSIGNMENT ONE

Task 1: EER Diagram



Assumptions, if any:

- A villa must have one type and one type only.
- A reservation must include at least one villa.
- A payment must be made for one reservation and one reservation only.
- An outdoor activity can use no equipment.
- An outdoor venue can use no equipment.
- A staff must either be junior or senior, and these are the only two types of staff.
- Both senior and junior staff can be the support staff for an activity reservation.

Task 2: Logical Transformation

Step 1: Strong Entities

- **DinnerOption** (DinnerCode (PK), DinnerDescription, DinnerCost)
- **Venue** (VenueCode (PK), VenueDescription, VenueCapacity)
- **Equipment** (EquipmentID (PK), EquipmentName, EquipmentQuantity)
- **Reservation** (ReservationCode (PK), ReservationDateMade, ReservationTotalCost)

Step 2: Weak Entities - None

Step 3: 1-1 Relationship - None

Step 4: 1-M RelationShip - None

Step 5: M-M RelationShip – None

Step 6: Multivalued Attributes - None

Step 7: Associative Entities

- **DinnerReservation** (ReservationCode (PK, FK), DinnerCode (PK, FK), VenueCode (PK, FK), DinnerResDate, DinnerDate)

Since the associative entity **DinnerReservation** has a multivalued attribute **SpecialDietaryRequest**, we will now proceed with creating a table for this multi-valued attribute.

- **SpecialDietaryRequest** (ReservationCode (PK, FK), DinnerCode (PK, FK), VenueCode (PK, FK), DietaryRequest (PK))

Step 8A:
In this step, the **Venue** supertype will give its primary key to all of its subtypes (**OutdoorVenue, IndoorVenue)**. Only the primary key gets inherited in this case.

- **OutdoorVenue** (VenueCode (PK, FK))
- **IndoorVenue** (VenueCode (PK, FK), VenueSize)

Step 2: Weak Entities - None

Step 3: 1-1 Relationship - None

Step 4: 1-M Relationship – None

Step 5: M-M Relationship

- **Uses** (VenueCode (PK, FK), EquipmentD (PK, FK))

Step 6: Multivalued Attributes – None

Step 7: Associative Entities – None

Final Table List:

- **DinnerOption** (DinnerCode (PK), DinnerDescription, DinnerCost)
- **Venue** (VenueCode (PK), VenueDescription, VenueCapacity)
- **Reservation** (ReservationCode (PK), ReservationDateMade, ReservationTotalCost)
- **Equipment** (EquipmentID (PK), EquipmentName, EquipmentQuantity)
- **DinnerReservation** (ReservationCode (PK, FK), DinnerCode (PK, FK), VenueCode (PK, FK), DinnerResDate, DinnerDate)
- **SpecialDietaryRequest** (ReservationCode (PK, FK), DinnerCode (PK, FK), VenueCode (PK, FK), DietaryRequest (PK))
- **OutdoorVenue** (VenueCode (PK, FK))
- **IndoorVenue** (VenueCode (PK, FK), VenueSize)

- **Uses** (VenueCode (PK, FK), EquipmentD (PK, FK))

Step 8B:
This step applies for total and disjoint/overlap constraints. The supertype **Venue** is disintegrated, and all the attributes will be inherited by the subtypes.

- **OutdoorVenue** (VenueCode (PK), VenueDescription, VenueCapacity)
- **IndoorVenue** (VenueCode (PK), VenueDescription, VenueCapacity, VenueSize)

Step 8C:
This step applies for total/partial and disjoint constraints. All the subtypes (**OutdoorVenue** and **IndoorVenue**) are disintegrated. The supertype **Venue**, will take all the attributes of the subtypes and also create an extra attribute called **VenueType** to differentiate between subtypes.

- **Venue** (VenueCode (PK), VenueDescription, VenueCapacity, VenueSize, VenueType)

Step 8D:
This step cannot be applied in this case as we do not have an overlap constraint.

Task 3: Normalisation

Firstly, we recognize that the given table has a 1st normal form as it does not contain any multivalued attributes and we have a composite primary key which includes **DinnerCode** and **MenuItemID**. Also, every row of the given table is unique. We will name the given table as "DinnerOrder".

In this case, every non-key attribute is functionally dependent on the composite primary key.

Functional Dependency

- **DinnerOrder** (DinnerCode (PK), MenuItemID (PK), MenuItemName, DinnerCost, PortionSize, DressCode, DressCodeDescription

⇨ 1NF

Next, we will proceed with identifying the partial dependency and put them into new tables.

Partial Dependency

In the **DinnerOrder** table, we can see that the **MenuItemName** column is functionally dependent on **MenuItemID (PK)** while **DinnerCost, DressCode, DressCodeDescription** are functionally dependent on **DinnerCode (PK).** As such, we will have the following new tables:

- **Menu** (MenuItemID (PK), MenuItemName) → 2NF
- **Dinner** (DinnerCode (PK), DinnerCost, DressCode, DressCodeDescription) → 2NF

Removing the partial dependency and putting them into new relations as above, both **Menu** and **Dinner** tables are now in the 2NF since each non-key attribute in each table is fully functionally dependent on the primary key.

As we have now constructed two new tables – **Menu** and **Dinner** – the attributes from the **DinnerOrder** table that belong to the new tables will be erased. We will only keep the DinnerCode and MenuItemID, and these columns will act as both the primary key and foreign key of the **DinnerOrder** table.

⇨ **DinnerOrder** (DinnerCode (PK, FK), MenuItemID (PK, FK), PortionSize) → 2NF

The **DinnerOrder** in this case has a 2NF because the PortionSize attribute is fully functionally dependent on the entire primary key.

Looking closely at the above 3 tables, we can see that the **Menu** and **DinnerOrder** table are in fact in a 3rd normalised form (3NF) as there also exists no transitive dependency in this case. On the other hand, the **Dinner** table is still a 2NF relation as there exists a transitive dependency between the DressCode and DressCodeDescription columns. To summarise:

- **Menu** (MenuItemID (PK), MenuItemName) → 3NF
- **Dinner** (DinnerCode (PK), DinnerCost, DressCode, DressCodeDescription) → 2NF
- **DinnerOrder** (DinnerCode (PK, FK), MenuItemID (PK, FK), PortionSize) → 3NF

The last step that we adopt is to remove the transitive dependency in the **Dinner** table and convert it into 3NF. We will use DressCode as the primary key in the new table. For the **Dinner** table, we will only keep DressCode as the foreign key and get rid of the DressCodeDescription attribute.

Transitive Dependency

- **Dress** (DressCode (PK), DressCodeDescription) → 3NF
- **Dinner** (DinnerCode (PK), DinnerCost, DressCode (FK)) → 3NF

In conclusion, our initial table can be normalised into 4 3NF relations as follow:

- **Menu** (MenuItemID (PK), MenuItemName)
- **Dress** (DressCode (PK), DressCodeDescription)
- **Dinner** (DinnerCode (PK), DinnerCost, DressCode (FK))
- **DinnerOrder** (DinnerCode (PK, FK), MenuItemID (PK, FK), PortionSize)