ASSIGNMENT - 3

1 The Relational Model

1.1

branch_no → city, street_number, postcode
employee-ID → employee-name
customer-ID → customer-name
account-number → balance

1.2

{customer-ID} + is the list of closures w.r.t customer-ID.
customer-ID += {customer-ID, Customer-name, account-number, balance, branch_no, street_number, city, postcode}
where
Customer-ID → Customer-name (functional dependency from the Customer Data Attributes).
Therefore, customer-ID += {customer-name}

1.3

Relations:
BRANCH:
Branch (branch-no, street-number, city, postcode)
EMPLOYEE:
Employee (employee-ID, Employee-name)
CUSTOMER:
Customer (customer-ID, customer-name)
ACCOUNT:
Account (account-number, balance)

Relationships:

Employee to Branch:

Employee ("Works_In") Branch – Many-to-One relation.
An employee only works for one branch, but a branch can have many employees

Customer to Account:

Customer ("Has") Account – Many-to-Many relation.
A customer can have more than one account and an account may be owned by more than one customer.

New Relations:

BRANCH:

Branch (branch-no, street-number, city, postcode)

EMPLOYEE:

Employee (employee-ID, employee-name, branch-No*)

CUSTOMER:

Customer (customer-ID, customer-name)

ACCOUNT:

Account (account-number, balance)

Works_In (employee-ID*, branch_no*)

employee-ID and branch_no are the foreign keys of Employee and Branch tables, respectively.

Has (customer-ID*, account-number*)

customer-ID and account-number are the foreign keys of Employee and Branch tables, respectively.

Or

A new table can be created instead of Has relationship. Therefore, new relations will be,

BRANCH:

Branch (branch-no, street-number, city, postcode)

EMPLOYEE:

Employee (employee-ID, Employee-name, branch-No*)

CUSTOMER:

Customer (customer-ID, customer-name)

ACCOUNT:

Account (account-number, balance)

CUSTOMER_ACCOUNT:

Cust_Acc (customer-ID*, account-number*)

2. Normalisation

2.1

Consider first two FDs,

- projID→ title, type, manager
- manager→type

the above two relations imply the transitive dependency where the type can be deleted from the projID relation as the manager relation always implies to type.
Therefore, new FD is

- **projID → title, manager**
- **manager → type**

Similarly,

- projID, title, jobID → contractNo

As projID determines title. Therefore, title can be removed
Therefore, it becomes

- projID, jobID → contractNo

Also, jobID determines the projID. Therefore, the projID can be deleted.
New FD,

- jobID → contractNo

And from

- jobID → projID, start-date, end-date, contractor

Since, jobID determines contractNo we can write,

- jobID → projID, contractNo, start-date, end-date, contractor

Now consider,

- **contractNo → jobID, contractor, start-date, end-date**

Since, jobID determines contactNo we can write jobID by removing start-date, end-date, contractor. As they are already determined by contractNo.
Therefore, new FD is

- **jobID → projID, contractNo**

Now consider the final FD

- jobID → contractNo

We can remove this as jobID already determines contractNo

Minimal basis for given PNG functional dependencies:

- **projID → title, manager**
- **manager → type**
- **jobID → projID, contractNo**
- **contractNo → jobID, contractor, start-date, end-date**

2.2

- The relation PNG is not in BCNF or 3NF because it has transitive functional dependencies.
- The relation is not in 3NF because the functional dependencies are transitive dependencies which violates the 3NF rules.
  Consider relations,
  - **projID → title, type, manager**
  - **manager → type**

  projtID implies both manager, type and manager imply type which is transitive functional dependency.
- Since, it is not 3NF it is not BCNF.

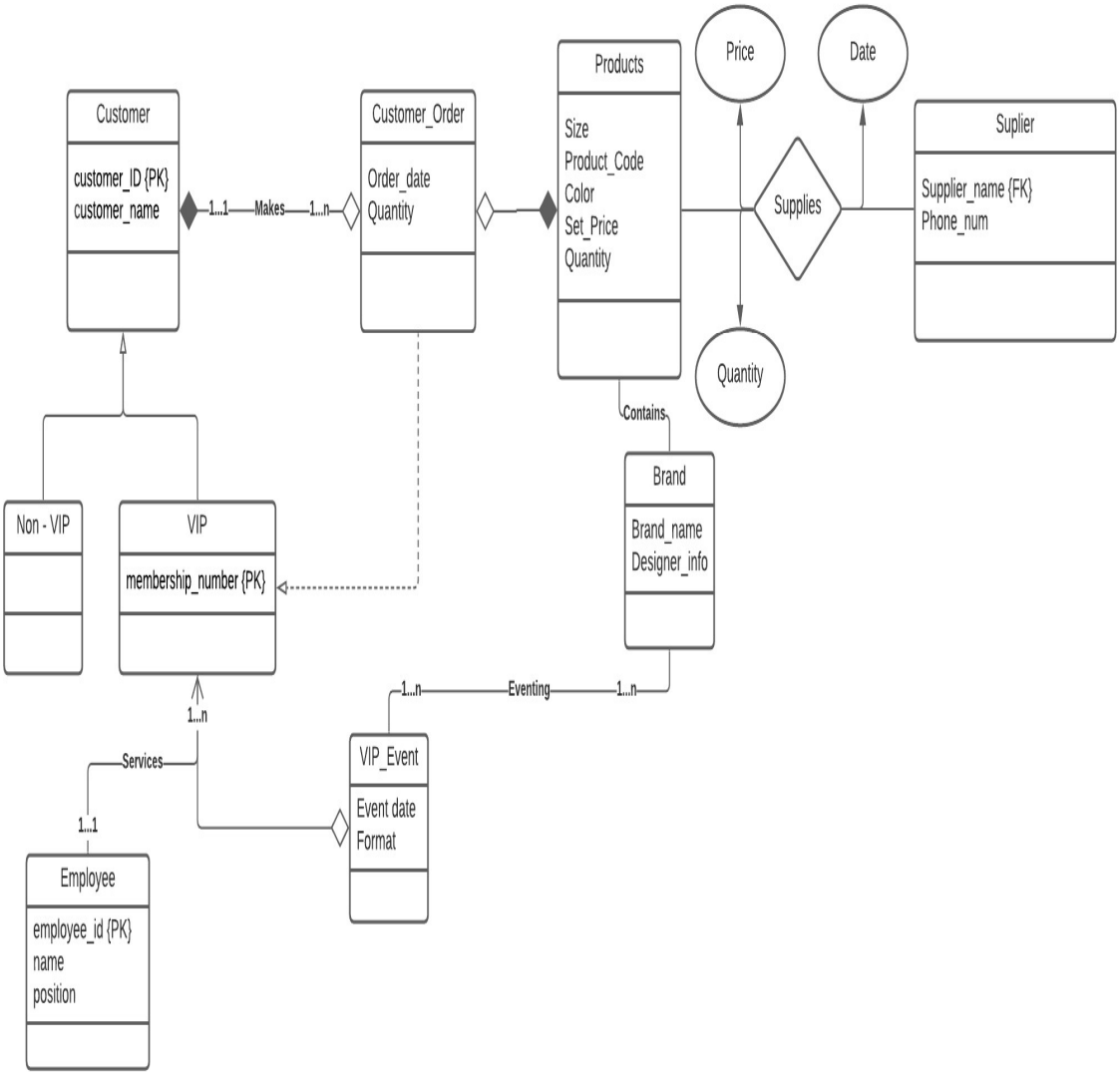
2.3

PMG relation into 3NF:
The relation is said to be in 3NF when it has the full functional dependency and should not have any transitive relations.
Therefore, FDs must be,

- **projID → title, manager**
- **manager → type**
- **jobID → projID, contractNo**
- **contractNo → jobID, contractor, start-date, end-date**

PROJECT:
Project (projID, title, manager*)
MANAGER:
Manager (manager, type)
JOB:
Job (jobID, contractNo*, projID*)
CONTRACT:
Contract (contractNo, contractor, start-date, end-date, jobID*)

3 The ER Model



4 ER to Relational schema Mapping

STAFF:

Staff (empNo, givename, surname, propertyNo*)

PROPERTY:

Property (propertyNo, streetnumber, street, suburb, postcode)

CLIENT:

Client (custNo, givename, surname, phoneno, creditrating)

OPENSESSION:

OpenSession (sessionNo, opendate, opentime, propertyNo*)

LEASE:

Lease (leaseNo, start-date, end-date, propertyNo*, custNo*)

OWNER:

Owner (ownerNo, address, phoneno, propertyNo*)

IndividualOwner (ownerNo*, givename, surname)

EntityOwner (ownerNo*, ABN, entityname)

Or

Owner (ownerNo, address, phoneno, givename, surname, ABN, entityname, propertyNo*)

(Since, inheritance)