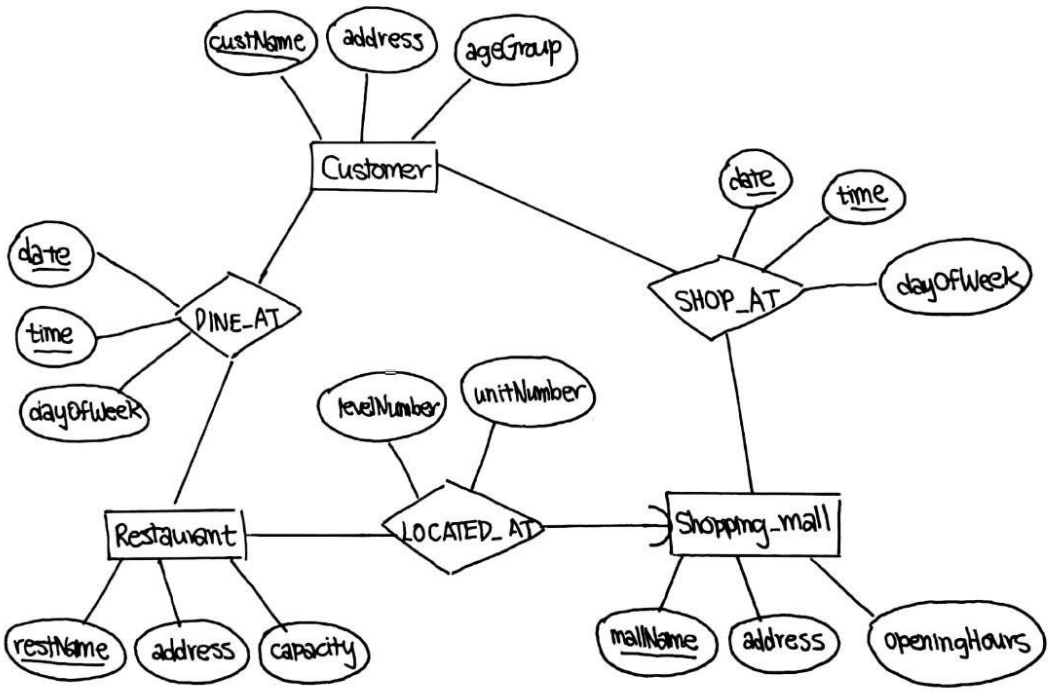
Solver: Tan Zarn Yao

1)

a)



b) CustomerNDineAt ≔ DINE_AT $\bowtie_{L\ DINE\_AT.custName=CUSTOMER.custName}$ CUSTOMER
WithinAge ≔ $\sigma_{ageGroup = \text{“30-40s”}}$ CustomerNDineAt
CustomerCountPerRestaurant ≔ $\gamma_{restName,\ COUNT(custName)\rightarrow noOfCustomer}$ WithinAge
MaxCount ≔ $\gamma_{MAX(noOfCustomer)\rightarrow MaxNoCustomer}$ CustomerCountPerRestaurant
Answer ≔ $\pi_{restName}$ (MaxCount $\bowtie_{L\ MaxNoCustomer\ =\ noOfCustomer}$ CustomerCountPerRestaurant)

c) ShopperWithinAge ≔ $\sigma_{ageGroup = \text{“20-30s”}}$ CUSTOMER
R1 ≔ $\pi_{cusName}$ ShopperWithinAge
ShoppingPeriod ≔ $\sigma_{dayOfWeek = \text{“Friday”}\ AND\ mallName = \text{“Nanyang Shopping Mall”}\ AND\ time \geqslant 19:00\ AND\ time \leqslant 22:00}$ DINE_AT
R2 ≔ $\pi_{cusName}$ ShoppingPeriod
R3 ≔ R1 – R2
Answer ≔ $\pi_{custName,\ address}$ R3 $\bowtie_L$ Customer

2)

a) A schema R is in BCNF if and only if the LHS of every non-trivial FD contains a key of R.
[Editor's note: To determine whether R is in BCNF, first we need to find out the keys of R. This can be done by the smallest set where its closure consists of all of R. Be cautious as there may be more than 1 key. Also check all other sets of the same number of attributes of the key that you found first.]
$\{BE\}^+ = \{ABCDE\}$ ∴ BE is a key.
Since BE must both be in the key and it is already a key itself, any other combination with BE will be a superkey.
Hence, R is not in BCNF since all the 3 FDs do not contain a key in their LHS.

b) B → D violates
$\{B\}^+ = \{BD\}$
R1 (B, D) and R2 (A, B, C, E)

c) A decomposition is lossless if the original table can always be reconstructed from the decomposed tables.
A decomposition is dependency-preserving if all the functional dependencies from the original table are preserved in the decomposed tables.
The common attribute in R1 and R2 is B. Since B is a key of R1, the decomposition is lossless.
B ⟶ D is preserved in R1; AB ⟶ C is preserved in R2. DE ⟶ A is not preserved in both decomposed tables. Hence, the decomposition is not dependency-preserving.

3)

a)

i)
```
SELECT    M.personName, COUNT(P.picture)
FROM      MEMBER M, PICTURE P
WHERE     M.groupName = P.groupName
GROUP BY  M.personName
```

ii)
```
SELECT    M1.personName, M2.groupName
FROM      MEMBER M1, MEMBER M2, FRIEND F1
WHERE     M1.personName = F1.personName1
          AND M2.personName = F1.personName2
          AND M2.groupName IN
          (   SELECT     M3.groupName
              FROM       FRIEND F2, MEMBER M3
              WHERE      F2.personName1 = M1.personName AND
                         F2.personName2 = M3.personName

              EXCEPT

              SELECT     M4.groupName
```

```
            FROM       MEMBER M4
            WHERE      M4.personName = M1.personName )

         AND NOT EXISTS

         (  SELECT     F3.personName2
            FROM       FRIEND F3
            WHERE      M1.personName = F3.personName1

            EXCEPT

            SELECT     M5.personName
            FROM       FRIEND F4, MEMBER M5
            WHERE      M1.personName = F4.personName1 AND
                       F4.personName2 = M5.personName AND
                       M5.groupName = M2.groupName )
```

[Editor's note: The answer above is acceptable but can be less complex, as shown in alternative solution below]

```
SELECT DISTINCT    F1.personName2, M1.groupName
FROM               Friend F1, Member M1
WHERE              F1.personName1 = M1.personName AND
                   M1.groupName NOT IN
                         ( SELECT M2.groupName
                         FROM  Member M2
                         WHERE M2.personName = F1.personName2);
```

iii)
```
CREATE TABLE ACCESS (
   personName VARCHAR(50),
   picture VARCHAR(50),  Assume it saves the url of the picture
   PRIMARY KEY(personName, picture),
   FOREIGN KEY picture REFERENCES PICTURE(picture)
);

INSERT INTO ACCESS (
   SELECT     M1.personName, P1.picture
   FROM       MEMBER M1, PICTURE P1
   WHERE      M1.groupName = P1.groupName

   UNION

   SELECT     M2.personName, P2.picture
   FROM       MEMBER M2, MEMBER M3, PICTURE P2, FRIEND F
   WHERE      M2.personName = F.personName1
                    AND F.personName2 = M3.personName
                    AND M3.groupName = P2.groupName
);
```

b)

i) In SQL, NULL is treated as an unknown value. Hence, the + operation won't work with an unknown value in the equation.

answer
NULL

ii) SUM ignores NULL values.

answer
1000.0

4)

a)
```
CREATE VIEW     HomeWins AS
SELECT          HomeTm, COUNT(*) AS NoOfWins
FROM            GAMES
WHERE           HomePts > AwayPts
GROUP BY        HomeTm;

SELECT     HomeTm
FROM       HomeWins
WHERE      NoOfWins = (    SELECT     MAX(NoOfWins)
                           FROM       HomeWins   );
```

b)

i)
```
CREATE ASSERTION Q4b(i) CHECK (
   NOT EXISTS (     SELECT     CZip
                    FROM       CUSTOMER
                    WHERE      CZip < 10000

                    UNION

                    SELECT     CZip
                    FROM       CUSTOMER
                    WHERE      CZip > 99999    )
);
```

ii)
```
CREATE TRIGGER Q4b(ii)
AFTER DELETE ON CUSTOMER
FOR EACH ROW
BEGIN
   DELETE FROM MAILED
   WHERE Ord IN (   SELECT     Ord
                    FROM       ORDER
                    WHERE      Cust = OLD.Cust );
   DELETE FROM SUBCONTRACTED
```

```
    WHERE Ord IN (   SELECT     Ord
                     FROM       ORDER
                     WHERE      Cust = OLD.Cust  );
    DELETE FROM ORDER
    WHERE Cust = OLD.Cust );
  END;
```

c) 1. The column is queried frequently.
2. A UNIQUE key integrity constraint exists on the column.

d)
```
<!DOCTYPE Webpages [
<!ELEMENT Webpage (word+)>
<!ATTLIST Webpage url ID #REQUIRED>
<!ELEMENT word (language+)>
<!ATTLIST word spelling CDATA #REQUIRED>
<!ELEMENT language (#PCDATA)>
]>
```

--End of Answers--