AAT (I)

(1) what is normalization ? Explain 1, 2, and 3 normal form.

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and update anomalies.

1NF :-

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multivalued attribute. A relation is in first normal form if every attribute in that relation is single valued attribute.

2NF :-

A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime Attribute (attributes which are not part of any candidate key) is dependend on any proper subset of any candidate key of the table.

3NF

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

(2) With an example explain BCNF. 4

BCNF is an advanced form of the third natural form and hence is quite stricter than it.

If every functional dependency is in the form $X \rightarrow Y$ the table is in BCNF. Here X is the super key to the table. for table to be in BCNF it should be in 3NF. for every FD, LHS is the Super key.

Example:-

Let us consider a company which has employees in more than one department.

Emp_ID	Emp_COUNTRY	Emp_DEPT	Dept_TYPE	Emp_Dept_No.
2	India	V1	D1	5
2	India	QA	D1	6
3	UK	Store	D2	7
3	UK	DEV	D2	8

In above table Functional dependencies are:

(a) Emp_ID → Emp_COUNTRY

(b) Emp_DEPT → {Dept_TYPE, Emp_Dept_No}.

Candidate Key :- {Emp_ID, Emp_DEPT}.

This table is not in BCNF because & Emp_DEPT or Emp_ID are not alone keys.

To convert to BCNF, we break it down into three tables.

(I)

Emp_ID	Emp_COUNTRY
2	India
2	India

(II)

Emp_DEPT	Dept_TYPE	Emp_Dept_No.
UI	D1	5
QA	D1	6
Store	D2	7
DEV	D2	8

(III)

Emp_ID	Emp_DEPT
D1	5
D1	6
D2	7
D2	8

Here the functional Dependencies are

(a) Emp_ID → Emp_COUNTRY

(b) Emp_DEPT → {Dept_TYPE, Emp_Dept_No}

candidate key:-

For I^st Table : Emp_ID

for II^nd Table : Emp_DEPT

For III^rd Table : {Emp_ID, Emp_DEPT}.

This is in BCNF because the left side of the two functional Dependencies is a key.

(3) Write ER to Relational Mapping Algorithm.

Step 1: Mapping of regular Entity types.

Step 2: Mapping of weak entity types

Step 3: Mapping of binary 1:1 Relation types.

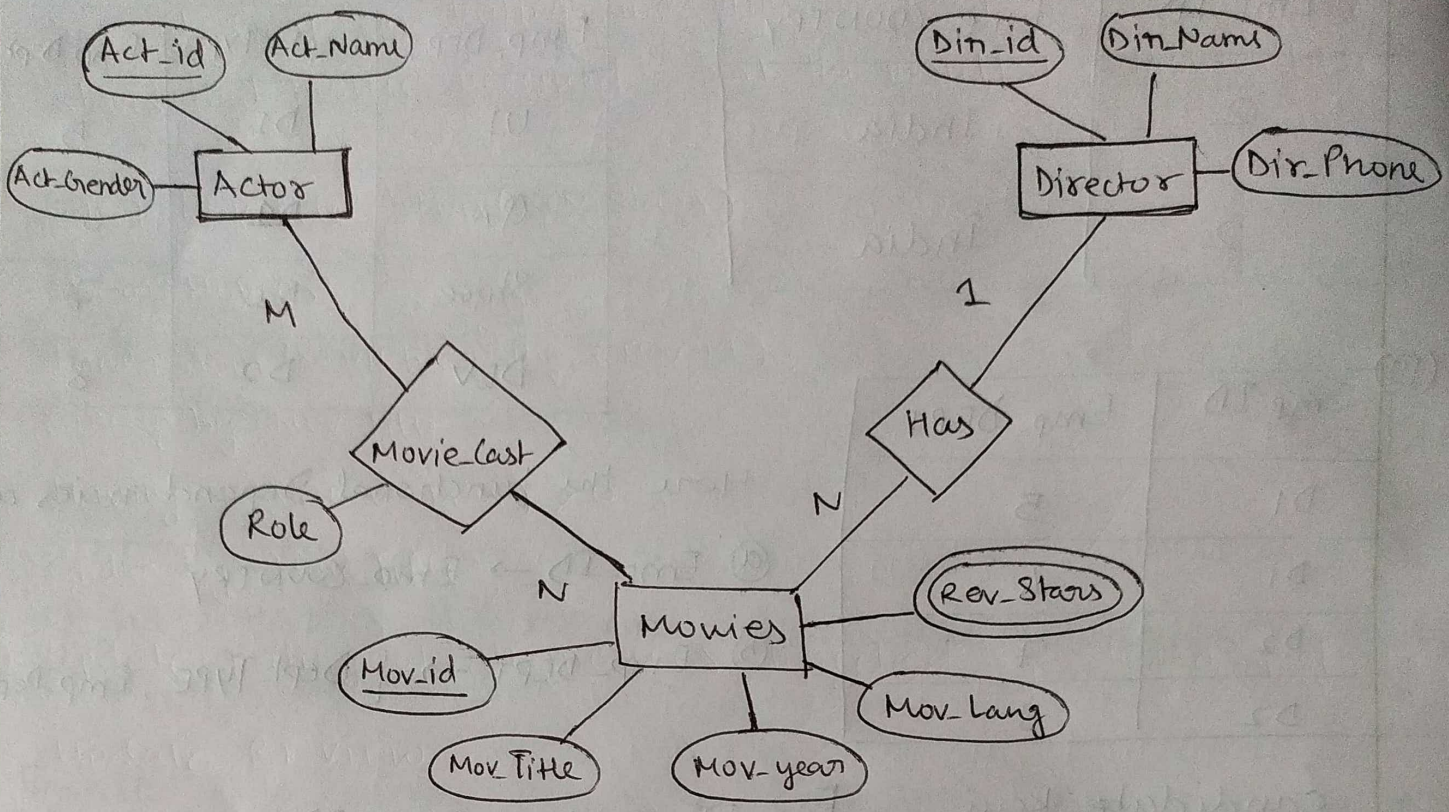
Step 4: Mapping of binary 1:N Relation types.

Step 5: Mapping of Binary M:N Relationship types.

Step 6: Mapping of multivalued attributes.

Step 7: Mapping of N-ary Relationship types.

④ Create a relational database for the given ER diagram



ACTOR

PK		
Act_id	Act_Name	Act_Gender

DIRECTOR

PK		
Dir_id	Dir_Name	Dir_Phone

MOVIES

PK				FK
Mov_id	Mov_Title	Mov_Year	Mov_lang	Dir_id

Movie_Cast

FK	FK	
Act_id	Mov_id	Role

REVIEW

FK	
Mov_id	Rev_Stars

Execute the following queries :-

(a) List the titles of all movies directed by 'Hitchcock'.

```
SELECT Mov_Title FROM MOVIES
where Dir_Id = ( SELECT Dir_id FROM DIRECTOR where Dir_name =
                 'Hitchcock' );
```

(b) Find the movie names where one or more actors acted in two or more movies.

```
SELECT Mov_Title FROM MOVIES M , Movie_Cast MC
where M.Mov_id = MC.Mov_id AND
 Act_id IN ( SELECT Act_id FROM Movie_cast Group by Act_id
 Having Count(Act_id) >1)
Group by Mov_Title Having Count(*) > 1 ;
```

(c) List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
Select Act_Name from ACTOR A
JOIN Movie_cast C
ON A.Act_id = C.Act_id
JOIN Movies M
ON C.Mov_id = M.Mov_id
where [illegible]
      M.Mov_Year NOT Between 2000 AND 2015 ;
```

(d) Find the title of movies and no of stars for each movie that has at least one rating and find the heighest number of stars that movie received. Sort the result by movie_title.

```
Select Mov_Title, Max(Rev_Stars)
From Movies
Inner Join REVIEW using (Mov_id)
Group by Mov_Title
Having Max(Rev_Stars) > 0
Order by Mov_Title;
```

(e) Update rating of all movies directed by 'Steven Spielberg' to 5

```
Update REVIEW
Set Rev_stars = 5
where Mov_id IN (Select Mov_id From MOVIES
where Dir_id IN (select Did_id from DIRECTOR where
Dir_name = 'Steven Spielberg'));
```

(5) Bring out different clauses of SELECT - FROM - WHERE - GROUP - HAVING with an example of each.

(i) FROM clause :-

Specifies the source table and views from which data is to be read.

Eg :- SELECT * FROM Employee.

Above command will display all the details in Employee table.

(ii) SELECT Clause :-

It specifies which values are to be retrieved.

Eg:- SELECT emp_id, emp_name FROM Employee;

(iii) WHERE Clause:-

The "where" clause specifies criteria that restricts the contents of the results table.

Eg:- comparisons:-

```
SELECT emp_name FROM Employee
WHERE emp_id = 'E0018';
```

Pattern Matching :-

```
SELECT Emp-name FROM Employee
WHERE emp_name like 'A %';
```

(iv) GROUP Clauses

This clause groups the selected rows based on identical values in a column or expressions.

```
Eg:- SELECT part_no, count(*) FROM orders
     GROUP By part_no;
```

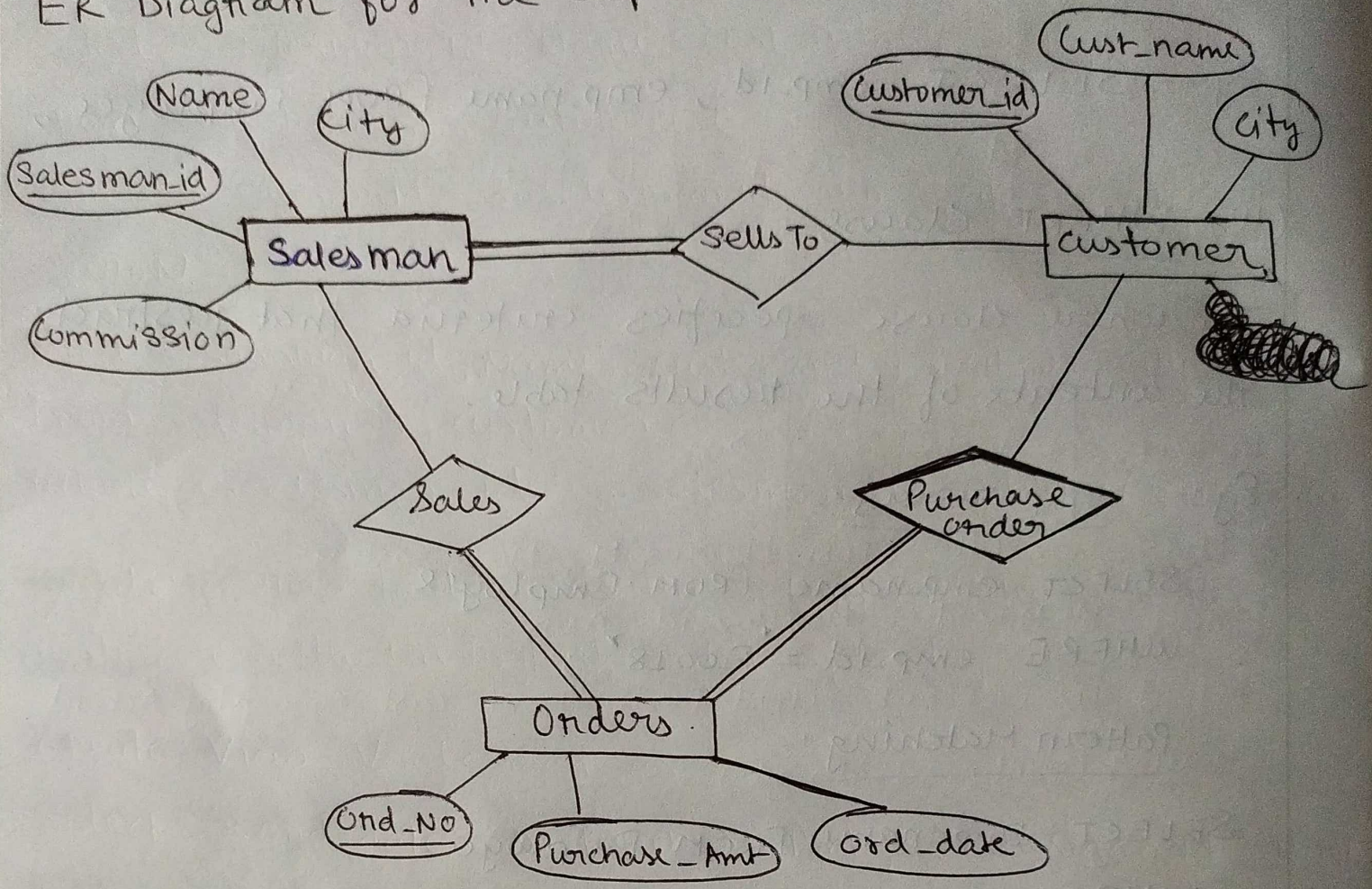
(v) HAVING Clause

It filters the results of the GROUP BY clause by using an aggregate function.

```
Eg:- SELECT count(cid), country FROM Customers
     GROUP BY country
     HAVING COUNT(cid) > 5;
```

AAT (II)

ER Diagram for the supermarket Database.



SCHEMA :-

SALESMAN :

PK			
Salesman_id	Name	City	Commission

CUSTOMER :

PK			FK.
Customer_id	Cust_name	City	Salesman_id

ORDER :

PK		FK	FK
Order_id	Purchase_Amt	Ord_date	Customer_id	Salesman_id

CREATE TABLE

```
Create table Salesman (
  Salesman_id varchar2(5) PRIMARY KEY,
  Name varchar2(10),
  City varchar2(10),
  commission number(3));

Create table customer (
customer_id varchar2(5) PRIMARY KEY,
cust_name varchar2(10),
city varchar2(10),
salesman_id varchar2(5) references Salesmam (salesman_id));

Create table order (
order_id varchar2(5) PRIMARY KEY,
Purchase_Amt number,
ord_date varchar2(10),
customer_id references customer (customer_id),
Salesman_id references Salesman (salesman_id));
```

(1) Find the name and numbers of all salesman who had more than one customer.

```
Select S.salesman_id, Name, count(C.customer_id)
FROM Salesman S, Customer C
WHERE C.salesman_id = S.salesman_id GROUP BY
S.salesman_id, Name HAVING count(C.customer_id) > 1;
```

(2) List all salesman and indicate those who have and don't have customers in their cities (use UNION)

```
Select S.salesman_id, Name, cust_name, S.city, C.city.
from salesman S, customer C
where C.salesmaid = S.salesman_id AND S.city = C.city
UNION
Select S.salesman_id, Name, cust_name, S.city, C.city.
From Salesman S, customer C
where C.salesman_id = S.salesman_id and NOT(S.city = C.city);
```

(3) Create a view that finds the salesman who has the customer with the highest order of a day.

```
Create View VSalesman as
Select O.ord_date, O.Purchase_Amt, S.salesman_id, S.Name
From Salesman S, order O
where S.salesman_id = O.salesman_id
AND O.Purchase_Amt = (Select Max(Purchase_Amt) from order C
where C.ord_date = O.ord_date);
```

(4) Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

```
DELETE From Salesman
WHERE Salesman_id = 1000;
```