Test 1

1. [10pt] Entity/Relationship Modeling

The Sunrise walk-in clinic needs a database system. The clinic has hired you to design a database. At this phase, you need to capture the database requirement in a conceptual design.

Requirements:
Sunrise Walk-in Clinic accepts new patients. We record all patients' information such as name, address, email, and health issuance number. In the first visit, a patient ID is generated for a patient and this ID is unique for each patient. Many doctors work in this clinic. A doctor's information such as doctor ID, name, email, and phone number are recorded for all doctors. A patient can make many appointments with different doctors on the same day or on different days. A doctor can have many appointments with different patients. For each appointment, there could be medications prescribed by the doctor for the patient in that appointment. For every appointment, a bill is issued. The amount of a bill can be paid in different payments. Each payment has a payment method such as credit card, debit card, master card, or cash, and the amount. A doctor may have surgeries done on different patients. A patient may have different surgeries with different doctors. There are different types of surgeries. Each type of surgery has a name and a price.

a) [2pt] would you represent the types of surgeries as an attribute or as an entity set? (Briefly justify your answer.)

The surgery can be an entity if we assume different types of surgeries can be done in a session.

If you assume that in a session or visit only one type of the surgery can be performed then it can be an attribute.

2. [10pt] Redesign the E/R schema

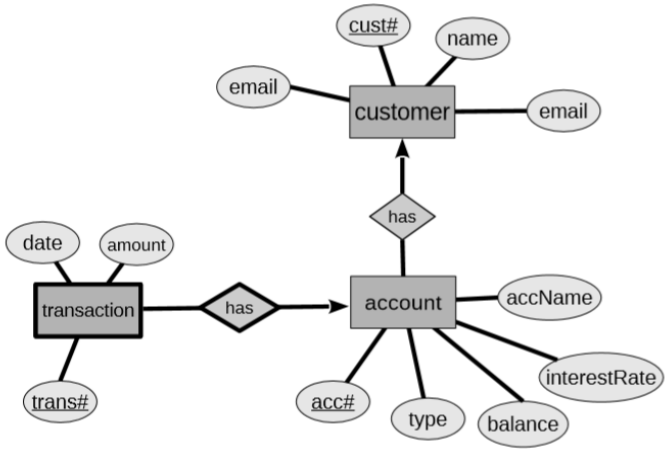
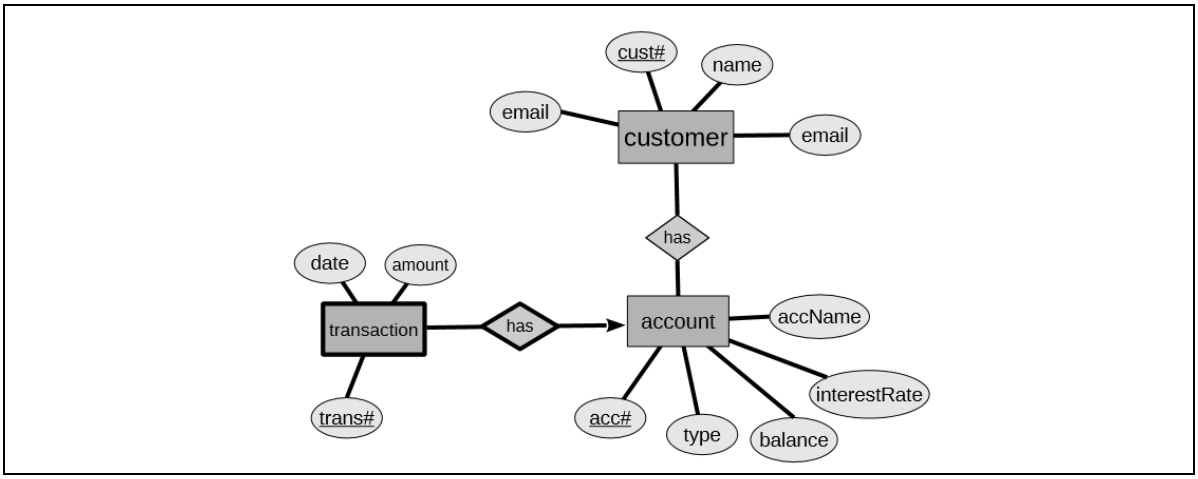
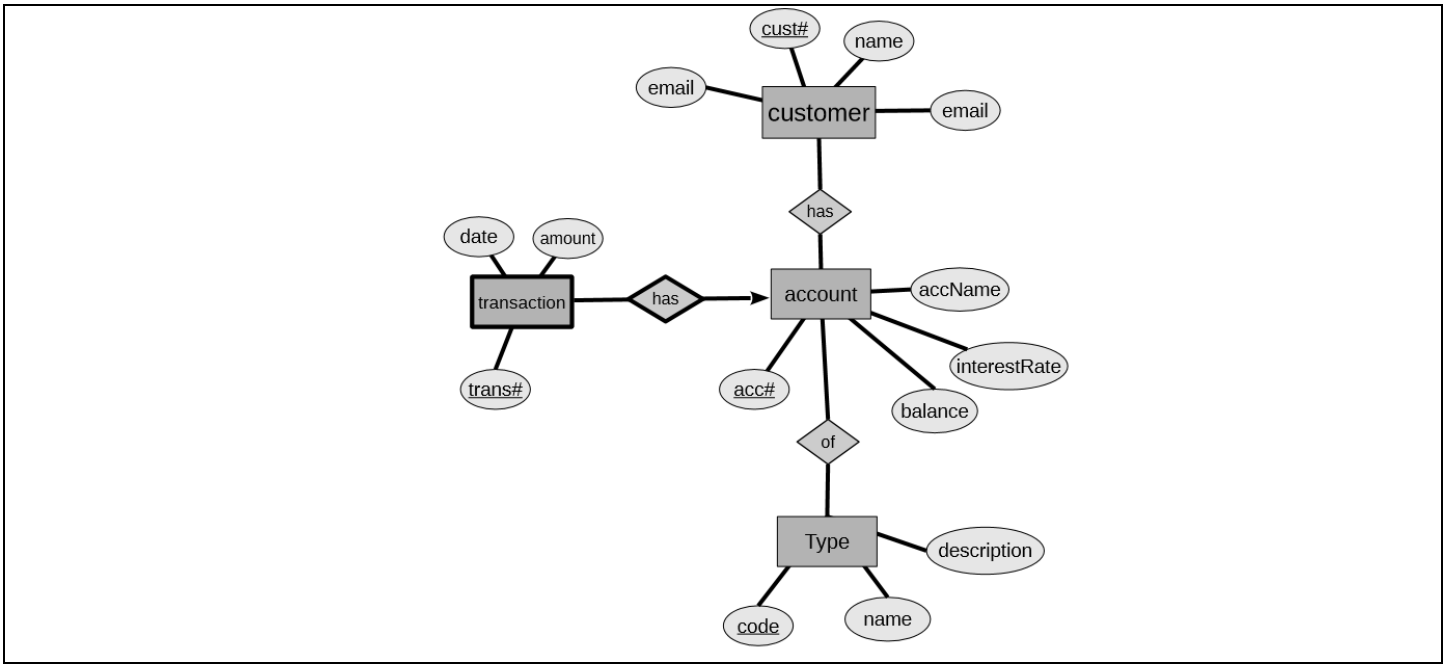


Figure 1

a) [3pt] Consider the E/R schema in Figure 1. This design does not accept joint accounts. Redraw this E/R diagram so joint accounts are allowed?



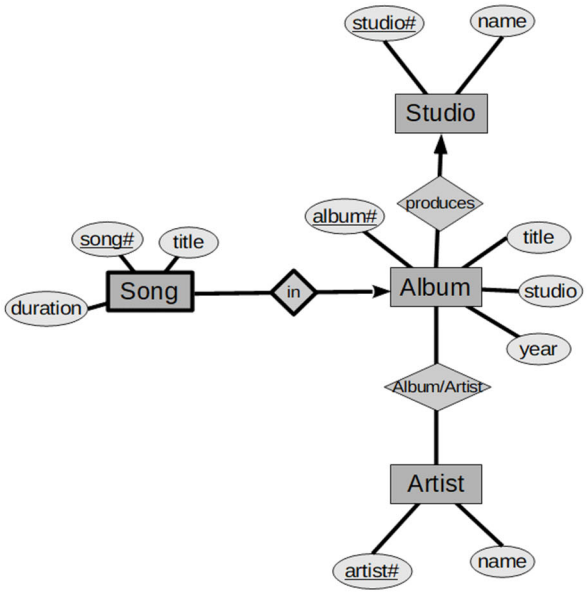
b) [2pt] In the given E/R diagram, an account can have an account type such as chequing, saving, cash back, and student. We want to allow that an account can have multiple account types. For instance, an account can be cash back and chequing. Change the E/R diagram in Question 2a to satisfy this requirement.



c) [4pt] Consider the following relations:

Song(<u>album#, song#</u>, title, duration)
Album(<u>album#</u>, year, title, studio)
FK (studio) refs Studio(studio#)
Artist(<u>artist#</u>,name)
Studio(<u>studio#</u>,name)
AlbumArtist(<u>album</u>,<u>artist</u>)
FK (artist) refs Artist(artist#)
FK (album) refs Album(album#)

Draw an E/R diagram that would result in the above relational schema on translation.



d) [1pt] Change the relational schema in Question 2c so an album has exactly one artist.

AlbumArtist(<u>album</u>,artist)
FK (artist) refs Artist(artist#)
FK (album) refs Album(album#)

3. [10pt] Functional Dependencies

   a) [3pt] Consider relation R(A,B,C,D,E) with functional dependencies:

   AB → C, C → D, BD → E

   Is the following decomposition dependency preserving? Explain your answer.

   R_1(ABC) R_2(DC) R_3(DE)

   AB → C and C → D are preserved in this decomposition but not BD → E

   b) [2pt] Consider the following functional dependencies:

   B→E CD→F A→C C→B

   Is functional dependency AD→F correct? Prove your answer.

We have A→C

By augmentation:
AD→CD

By Transitivity:
AD→CD , CD→F then AD→F

c) [4pt] Consider relation ActiveMamber:

ActiveMember (m#, name, email, a#, aName, t#, tName)

m#: member ID	a#: activity ID	t#: trainer ID
name: member name	aName: activity name	tName: trainer name
email: member email		

Assume the following business rules for Sky Club. Sky Club offers programs that include both indoor and outdoor sport activities.

- To become a member, the name and the email address of the person is required.
- Each member has a unique identifier member# that is generated when a person becomes a member.
- A member can participate in many activities. An activity can have many participating members.
- Each activity has only one trainer. A trainer, however, may be the trainer of many activities.

For relation ActiveMember, write the functional dependencies that apply given the business rules above.

m# → name, email
a# → aName, t#, tName
t# → tName
m#,a# → name, email, aName, t#, tName

d) [1pt] With respect to the functional dependencies you declared in Question 3c, determine a key and a super key in relation ActiveMember. (Prove your answer)

m# → name, email
a# → aName, t#, tName
key: m#,a# → name, email, aName, t#, tName
super key: m#,a#, t# → name, email, aName, t#, tName

4. [10pt] Normalization

e) [3pt] Consider the following relation R and its functional dependencies:

R (A, B, C, D, E)
C→ DE
AC → B

Is the following decomposition of R to R1 and R2 lossy or lossless? Prove your answer.
R1 (A, B, C)
R2 (B, D, E)

$(R1 \cup R2) = (A, B, C, D, E)$
$(R1 \cap R2) \neq \Phi$
$(R1 \cap R2) = \{B\}$

$\{B\}^+ = \Phi$ B is not the key of R1 or R2.

The given decomposition is not the lossless-join decomposition.

f) [4pt] Find the highest normal form for the given relation and functional dependencies. (Prove your answer)

R (A, B, C, D, E, F)
AC→E E→BD D→F

g) [3pt] Is relation R (A, B, C, D) in Boyce-Codd Normal Form (BCNF) for the following set of FDs? (Prove your answer)

B → D C → B B → A AB → D

Key: $\{C\}^+ = \{ABCD\}$
Relation R is in 1NF, the key is defined.
The relation is in 2NF (no partial dependency)
B → A violates the third notmal form (transitive dependency). R is not in BCNF.

Test 2

1. [15pt] **Relational Algebra**
   a) [3pt] Consider the following relations:

Person (p#, name)
Student (p#,pgm)
FK p# refs **Person**
Teacher (p#, dept)
FK p# refs **Person**

Using set operators write a relational algebra that shows the tuples from relation person who are not students or teachers. (You get zero if you do not user set operators).

$\pi_{\text{p\#, name}}$ (Person ⋈ ($\pi_{\text{p\#}}$ (Person) – ($\pi_{\text{p\#}}$ (Student) ∪ $\pi_{\text{p\#}}$ (Teacher)))

To answer the questions 1b through 1d use the Retail schema in Figure 1 on page 13.

b) [3pt] Show product ID, product name, and the supplier name for all products.

$\pi_{\text{productId, name, supName}}$ (product ⋈ $\pi_{\text{id} \rightarrow \text{supplierId, name} \rightarrow \text{supName}}$ (supplier))

c) [3pt] Show customer number, email, city for the customers who have not purchased products whose supplier is from 'Toronto'.

$\pi_{\text{cust\#, email, city}}$ (Customer)
⋈
($\pi_{\text{cust\#}}$ (Customer)
–
$\pi_{\text{cust\#}}$ $\sigma_{\text{city = 'Toronto'}}$ (Order ⋈ OrderItem ⋈ Product ⋈ $\pi_{\text{id} \rightarrow \text{supplierId, name} \rightarrow \text{supName}}$ Supplier))

d) [3pt] Show customer number, last name, and email for all customers who have more that one order.

$\pi_{\text{cust\#, lastName, email}}$ (customer $\bowtie$ $\pi_{\text{cust\#}}$ ($\sigma_{\text{order\#} \neq \text{o\#}}$ (Order $\bowtie$ $\pi_{\text{order\#} \rightarrow \text{o\#}}$ Order))

e) [3pt] Consider the following relational algebra:

$\pi_{\text{productId , productName, supplierId , unitprice}}$ (Product)
$\bowtie$
($\pi_{\text{supplierId , unitprice}}$ (Product)
–
$\pi_{\text{id} \rightarrow \text{supplierId , price1} \rightarrow \text{unitprice}}$(
$\sigma_{\text{price1} > \text{price 2}}$ (($\pi_{\text{supplierId} \rightarrow \text{id, unitPrice} \rightarrow \text{price1}}$ (Product) $\bowtie$ Supplier)
$\bowtie$
($\pi_{\text{supplierId} \rightarrow \text{id, unitPrice} \rightarrow \text{price2}}$ (Product) $\bowtie$ Supplier))
))

Explain what the above query returns without using database terms in your answer. Your answer must show your understanding of the subqueries in the above query. (The answers such as "It returns some products" is not acceptable. Be clear and specific in your answer.

The query for each supplier returns the cheapest price of the product provided by that supplier.

2. [15pt] **Queries in SQL**

To answer the questions of this section use the Movie schema in Figure 2 on page 14.

b) [3pt] Write an SQL query that answers the following.
Show movie name and year of the movies that their title starts with letter ***t*** but does not end with letters ***es***.

```
select mov_title, year
from Movie
where mov_title like 't%' and mov_title not like '%es';
```

c) [3pt] Write an SQL query that answers the following.
Show movie name, year, director first name, and director last name for all movies of the romantic genre.

```
select
from Movie m, Movie_Genre mg, Genre g
where m.mov_id = mg.mov_id and mg.gen_id = g.gen_id
and g.title like 'romantic';
```

d) [3pt] Write an SQL query that answers the following.
Report Id, first name, and last name for the actors that have not played a roll in movies directed by directors with Id 1244 or 4412.

```
select act_id, act_fname, act_lname
from Actor
where act_id in (select act_id
                 from Actor
                 except
                 select ac_id
                 from Movie_Direction md, Movie_Cast mc
                 where md.mov_id = mc.mov_id and md.dir_id in (1244,4412) ) ;
```

e) [3pt] Write an SQL query that answers the following.

Report actor id, actor last name, role, and the genre title for actors who have played rolls in romance, science fiction, and action movies. In your result, you will sort rows with genre science fiction, romance, and action.

```
select distinct a.act_id, a.act_lname, mc.role, g.title
from Actors a, Movie_Cast mc, Movie_Genre mg, Genre g
where a.act_id = mc.act_id and mc.mov_id = mg.mov_id and mg.gen_id = g.gen_id
and (g.title = 'science fiction' or g.title = 'romance' or g.title = 'action')
order by g.title
```

f) [3pt] Write an SQL query that answers the following.

Report movie Id, movie title, and the director's first name and last name for all movies whose rating (stars) is less that the average rating of movies produced from 2010 to 2018 (included).

```
select m.mov_id, m.mov_title, d.dir_fname, d.dir_lname
from Movie mv, Movie_Direction md, Director d, Rating rm
where mv.mov_id = md.mov_id and md.dir_id = d.dir_id and rm.mov_id = mv.mov_id
and  rm.rev_star < select avg(rev_star)
                   from Rating r, Movie m
                   where r.mov_id = m.mov_id and m.mov_year between 2010 and 2018
```

3. **Query Logic.**

a) [2pt] Consider table R (a, *b*, c) with n tuples where n > 0. What are the possible values that can be returned by the following query? Explain your answer.

```
select count(c) as count from R;
```

The value of c can be null. Therefore, if all c values are null, count() returns zero. If the value of c in all rows is a non-null value, it returns n. If some of the c values are null and some of them are not, then count() returns a value between 0 and n.

b) [2pt] Consider following relations.

R (a, b, c)
S (d, e, *a*)
FK (a) refs **R**

Relation **R** has ***n*** tuples and relation **S** has ***m*** tuples. How many rows does the following query return? Justify your answer.

```
select R.a, R.b, S.e
from R
left join S
on R.a = S.a

union

select R.a, R.b, S.e
from S
left join R
on R.a = S.a
```

first join: All rows from R and just matching rows from S

Matching rows: $|R \bowtie S| = m$ since all rows in S, have a matching value in R.
+
All non-matching rows from R

non-matching rows of R, nmr = $(\pi_a R - \pi_a (R \bowtie S))$
where nmr < n
The number of rows returned by the first join:

$0 < (nmr + m) < n+m$

Second join: All rows from S and just the matching rows from R. Since all rows in S have a matching row in R the number of records of the second join is m.

The total is a value < n+m
Since the union remove duplicates.

c) [3pt] Consider the following query on Retail schema:

```
select distinct c.cust#,p.productName
from customer c, order c, orderItem i, product p
where c.cust# = o.cust# and o.order# = i.orderId
      and p.productId = i.productId;
```

Explain what the above query asks in English to a non-technical person. Do not use database terms in your answer.

The query, for each customer, returns the products that the customer has ordered.

d) [3pt] Consider the query from Q3c. Rewrite the query without using distinct so the new query returns the same result as the old one.

```
select c.cust#,p.productName
from customer c, order c, orderItem i, product p
where c.cust# = o.cust# and o.order# = i.orderId
and p.productId = i.productId
group by c.cust#,p.productName;
```