1 (10 points) Terms and Definitions

(2 points each) In *at most* two short sentences each, explain the meaning of the following terms as they relate to database management systems.

1. **Entity Set**

The set of all possible entities of a given type, that all have the same set of attributes.

1 point: set of objects; 1 point: same attributes

2. **Super Key**

A set of fields in a relation, where a subset of the fields uniquely identify a tuple. This means there are more fields than are strictly necessary, which separates this from a candidate key.

1 point: uniquely identifies a tuple; 1 point: has more columns than necessary.

3. **Null**

In SQL, indicates that there is no value or a missing value for an attribute.

1 point: missing / optional / nothing; 1 point: refers to values for attributes.

4. **Integrity Constraint**

Checks on the data contained in the database that must always be true. Ensures that the data is always correct.

1 point: checks or validations; 1 point: correctness for entire database

5. **Natural Join**

A join between where the values for all fields with the same name in the two tables are equal.

1 point: same name / common attributes; 1 point: equality on the values

2 (10 points) EJBank Relational Algebra

Evan's bank stores its data in two relations, with the following SQL schema:

```
CREATE TABLE Customers(
  cid int PRIMARY KEY,
  name text,
  state text
);

CREATE TABLE Accounts(
  aid int PRIMARY KEY,
  cid int NOT NULL REFERENCES cid,
  balance real NOT NULL
);
```

1. **(2 points)** Write a relational algebra expression to compute the account ids that have balances greater than $50,000.

$\pi_{aid}\sigma_{balance>50000}(Accounts)$

2. **(4 points)** Write a relational algebra expression to compute the names of customers with balances greater than $50,000 in the state of "NY".

$\pi_{name}((\sigma_{state=NY}(Customers)) \bowtie_{cid} (\sigma_{balance>50000} Accounts))$

3. **(4 points)** Given the following values for the Account relation:

aid	cid	balance
1	102	1000.00
2	102	2000.00
3	107	2000.00
4	108	1000.00

What is the result of the following relational algebra expression?

$\rho(A, Accounts)$
$\rho(B, Accounts)$
$\pi_{A.aid,B.aid}(A \bowtie_{A.balance>B.balance} B)$

(continued on next page)

Fill in your answer in this table. Do not fill in the names for the fields. *Note*: you may or may not need all the columns and rows.

Note: Order does not matter

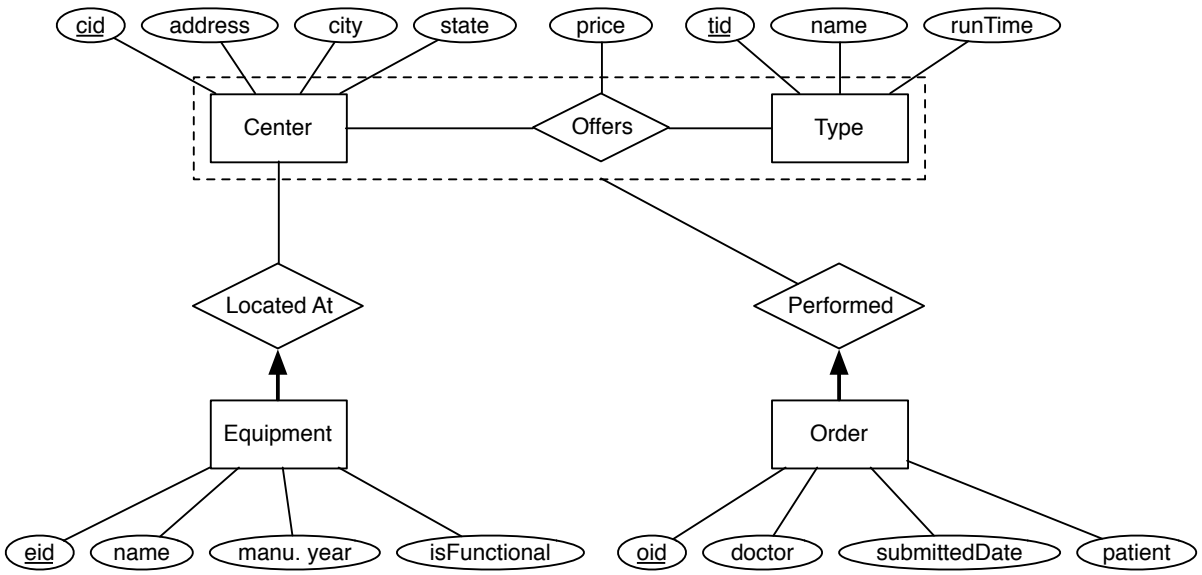
2	1				
3	1				
2	4				
3	4				

3 (20 points) Medical testing Entity-Relationship Modelling

A medical lab testing company has several testing centers all over the country. In this problem, you will design a schema to keep track of the testing centers, tests and order information. Specifically, you will need to keep track of:

1. The address, city, state and manager for each testing center.
2. The equipment at each testing center including the name of the machine, manufacturing year, and status of each machine (either functional or in repair). Note that one center may have multiple machines of the same kind.
3. The types of tests the company can perform. Tests have a name, time to run and price. The price of the test depends on the specific testing center. Some tests are only available at specific centers.
4. When an order for a test is submitted, the lab must record the doctor's name, the patient's name, and the date it was ordered. Each order is performed at a specific testing center. It must be performed at a center that offers that type of test.

Part 1: (10 points) Draw an ER diagram representing your database. Include 1-3 sentences of justification for why you drew it the way you did.



Alternatives and notes:

- We accepted basically any reasonable unique id for the entities, but it was an error to forget to specify ids. The question should have specified that it is okay to add ids if that makes things easier.
- Equipment needs either an exactly one or at most one relationship with Center. Otherwise you can have a machine at multiple testing centers, which isn't physically possible. The fact that there is a "status" attribute means that each Equipment entity represents one physical machine. The note about multiple machines of the same kind is a distraction and can effectively be ignored. Adding a quantity field isn't quite right, because then multiple machines share the same status.
- Order must have exactly one test type. A common mistake was to permit an order to be for multiple tests, or to have zero tests, which doesn't match the description.

- The price from the Test Type must be an attribute of a relationship with Center. Otherwise, you either need to duplicate the test type information for multiple centers, or the price doesn't depend on the specific Center.
- Some people chose to make things weak entities, which is acceptable. (I'd argue there are no weak entities here, but I can see the argument the other way.)

Close but incorrect alternatives:

- Performed is a ternary relationship between Type and Center. This permits tests to be ordered at centers that do not offer them.
- Order has two exactly one relationships: one with Center and one with Type. This is equivalent to the ternary relationship described above.

Part 2: (10 points) Write a SQL schema for your database. Include 1-3 sentences of justification for why you chose the tables you did.

We evaluated if you correctly translated your entity-relationship diagram into SQL. The SQL below is for the solution above. Some common mistakes:

- Equipment status needs a `CHECK` constraint or must be a boolean, to enforce that it can only take a limited set of values. We didn't care about any other constraints (except foreign keys).
- A relationship with an at least one constraint must permit the foreign key to be NULL, or must be in a separate table.
- A relationship with an exactly one constraint must be combined with the entity, and must not permit a NULL foreign key.
- When you have a relationship without constraints, it usually has a composite primary key (e.g. `Offers(cid, tid)`). When it is involved in another relationship, you must reference *that table*, and not the "base" tables (e.g. (Order `REFERENCES` Offers, not Order `REFERENCES` Center and Type). You need the database to enforce the constraint that a record exists in the relationship, not the base tables. This is the difference between a the "correct" aggregate compared to a ternary relationship between Center, Type, and Order.

```
CREATE TABLE Centers(
  cid int PRIMARY KEY,
  address text NOT NULL,
  city text NOT NULL,
  state text NOT NULL,
  manager text NOT NULL
);

CREATE TABLE Types(
  tid int PRIMARY KEY,
  name text NOT NULL,
  runTime int NOT NULL
);

CREATE TABLE Offers(
  cid int REFERENCES Centers,
  tid int REFERENCES Types,
```

```
  price real NOT NULL,
  PRIMARY KEY (cid, tid)
);

CREATE TABLE Equipment(
  eid int PRIMARY KEY,
  locationCid int NOT NULL REFERENCES Centers,
  name text NOT NULL,
  manufactureYear int NOT NULL,
  isFunctional bool NOT NULL
  -- ALTERNATIVE: status text NOT NULL,
  --              CHECK(status = 'functional' OR status = 'in repair')
);

CREATE TABLE Orders(
  oid int PRIMARY KEY,
  performedCid int NOT NULL REFERENCES Centers,
  cid int NOT NULL,
  tid int NOT NULL,
  doctor text NOT NULL,
  submittedDate date NOT NULL,
  patient text NOT NULL,
  FOREIGN KEY (cid, tid) REFERENCES Offers
);
```

4 (20 points) Wikipedia in SQL

For this question, we will use a simplified schema based on Wikipedia, shown below. Each page has a unique id, a human readable title, and the length of the page (in bytes). Links between pages are stored in the Link table. The source is the page that contains the link, and dest is the page that the link points to. As an example, a Link tuple with values (source=50, dest=100) means that page id 50 contains a link to page id 100.

```
CREATE TABLE Page(
  id int PRIMARY KEY,
  title text NOT NULL,
  length int NOT NULL
);

CREATE TABLE Link(
  source int REFERENCES Page,
  dest int REFERENCES Page,
  PRIMARY KEY (source, dest)
);
```

1. **(6 points)** Circle true or false for the following statements:

   (a) **True / False** There can be two pages with the title "Venus".

> **TRUE**: There is no unique constraint on `title`.

   (b) **True / False** Broken links may exist (a link where the source or the destination pages do not exist).

> **FALSE**: The foreign key constraints on Link ensures that both pages must exist.

   (c) **True / False** A page can only be the source of one link.

> **FALSE**: To express that constraint, Page and Link would need to be combined.

   (d) **True / False** If the page "Venus" contains a link to "Mars", deleting "Venus" will not be permitted.

> **TRUE**: The foreign key constraints on Link ensure that you can't delete the page while it is either the source or destination for any links.

   (e) **True / False** If the page "Venus" is not the source of any links, deleting it will be permitted.

> **FALSE**: It could still be the destination for links. Any foreign reference will prevent the entity from being deleted (unless `ON DELETE CASCADE` is specified).

   (f) **True / False** Renaming pages is not permitted.

> **FALSE**: There is no way to specify a constraint that forbids items from being edited. There are also no constraints on title, beyond `NOT NULL`, so you can even rename it to the same name as another page.

Write SQL queries to answer the following questions:

2. **(2 points)** What is the id and title of all pages that have titles that begin with the string "Database"?

```
SELECT id, title
FROM Page
WHERE title LIKE 'Database%';
```

3. **(2 points)** What are the titles of the 10 longest pages (in bytes)?

```
SELECT title
FROM Page
ORDER BY length DESC
LIMIT 10;
```

4. **(2 points)** What are the titles of all pages linked from the page with id 42?

```
SELECT title
FROM Page, Link
WHERE Link.source = 42 AND Page.id = Link.dest;
```

Alternative:

```
SELECT P.title
FROM Page p
WHERE p.id IN (
  SELECT dest
  FROM Link
  WHERE source = 42
);
```

5. **(4 points)** What are the titles of all pages linked from pages with the title "Database"?

```
SELECT destination.title
FROM Page source, Link, Page destination
WHERE source.title = 'Database'
    AND source.id = Link.source
    AND Link.dest = destination.id;
```

Alternative:

```
SELECT title
FROM Page
WHERE id IN (
  SELECT dest
  FROM Page, Link
  WHERE Page.title = 'Database' AND page.id = Link.source
);
```

Yet another version:

```
SELECT p1.title
FROM Page p1, Link l
WHERE p1.id = l.dest AND l.source IN (
  SELECT p2.id
  FROM Page p2
  WHERE p2.title = 'Database'
);
```

6. **(4 points)** Popularity of a page is defined as the number of incoming links (links leading to that page). What are the titles of the 5 most popular pages?

```
SELECT title
FROM Page, (
  SELECT dest, count(*) as count
  FROM Link
  GROUP BY dest
  ORDER BY count DESC
  LIMIT 5) AS Popular
WHERE Page.id = Popular.dest;
```

Note: `ORDER BY` and `LIMIT` can be applied to outer query instead of the inner query. Another version:

```
SELECT Page.title
FROM Link, Page
WHERE Link.dest = Page.id
GROUP BY Link.dest, Page.title
ORDER BY count(*) DESC
LIMIT 5;
```