ASSIGNMENT:

Work on these interview type questions based on what was covered above. They should reword if asked for the definition of a key term instead of copying from the resources.

1. What is the difference between a Database and a Database Management System (DBMS)?

A database is a collection of data organized in a way that's easy for a computer to access that data. A DBMS is an application that helps you create and manage databases.

2. Explain what a Schema is.

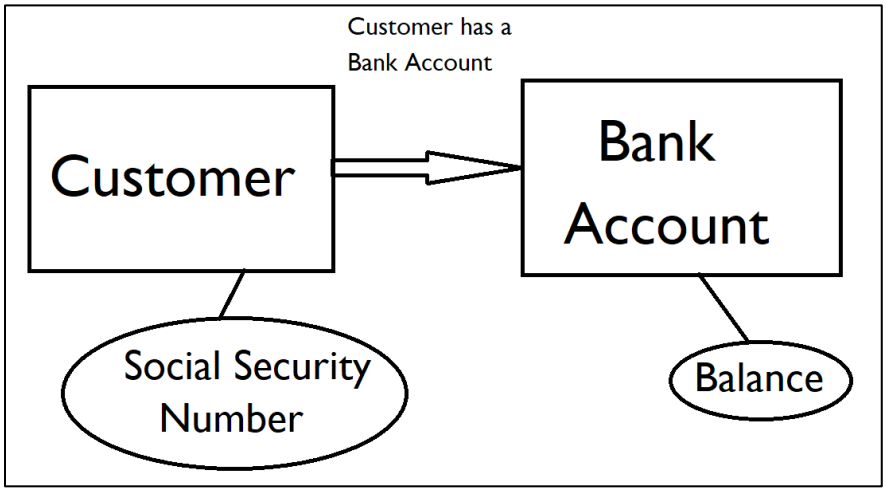
A schema is usually a diagram that describes and outlines a database.

3. What is a Subschema?

The subschema is a subset of the schema. It shows the user's view of the data items and records which they use.

4. What is an Entity Relationship Model? Give a simple example of an ER Model.

An ER Model is a diagram that consists of entities, attributes, and the relationships between the entities. A simple example of an ER Model is shown below.



Customer and Bank Account are the entities. Customer has an attribute social security number and bank account has an attribute of balance. They relationship is that customer has a bank account.

5. Explain what an Entity, Attribute, Relationship, and Instance is in an ER Model.

An entity an object or thing. An attribute describes the entity. A relationship is what links two entities together. An instance is a single record of an entity.

6. Give an example of a One-to-One, One-to-Many, and a Many-to-Many relationship.

One-to-One: A car has one steering wheel and a steering wheel is only apart of one car.

One-to-Many: A school has many students, but a student only has one school.

Many-to-Many: A professor can have many students and students can have many professors.

7. Why are Primary Keys and Foreign Keys used within an ER Model? How do they help organize data?

Primary keys help uniquely identify each record in a table. Foreign keys are keys referencing a primary key in another table. With these keys, an ER Model can create links/relationships between two tables. That way we can split up data into different tables and limit the amount of repeat data.

8. What can happen when data is not Normalized?

When data is not normalized, there is duplication of data. This causes insert, delete, and update anomalies.

9. What are the rules a table must follow to be in First Normal Form?

To be in First Normal Form, the tables should separate any repeating groups, repeating groups should be put in their own table, and the primary key of the repeating group table should be a composite key.

10. What are the rules a table must follow to be in Second Normal Form?

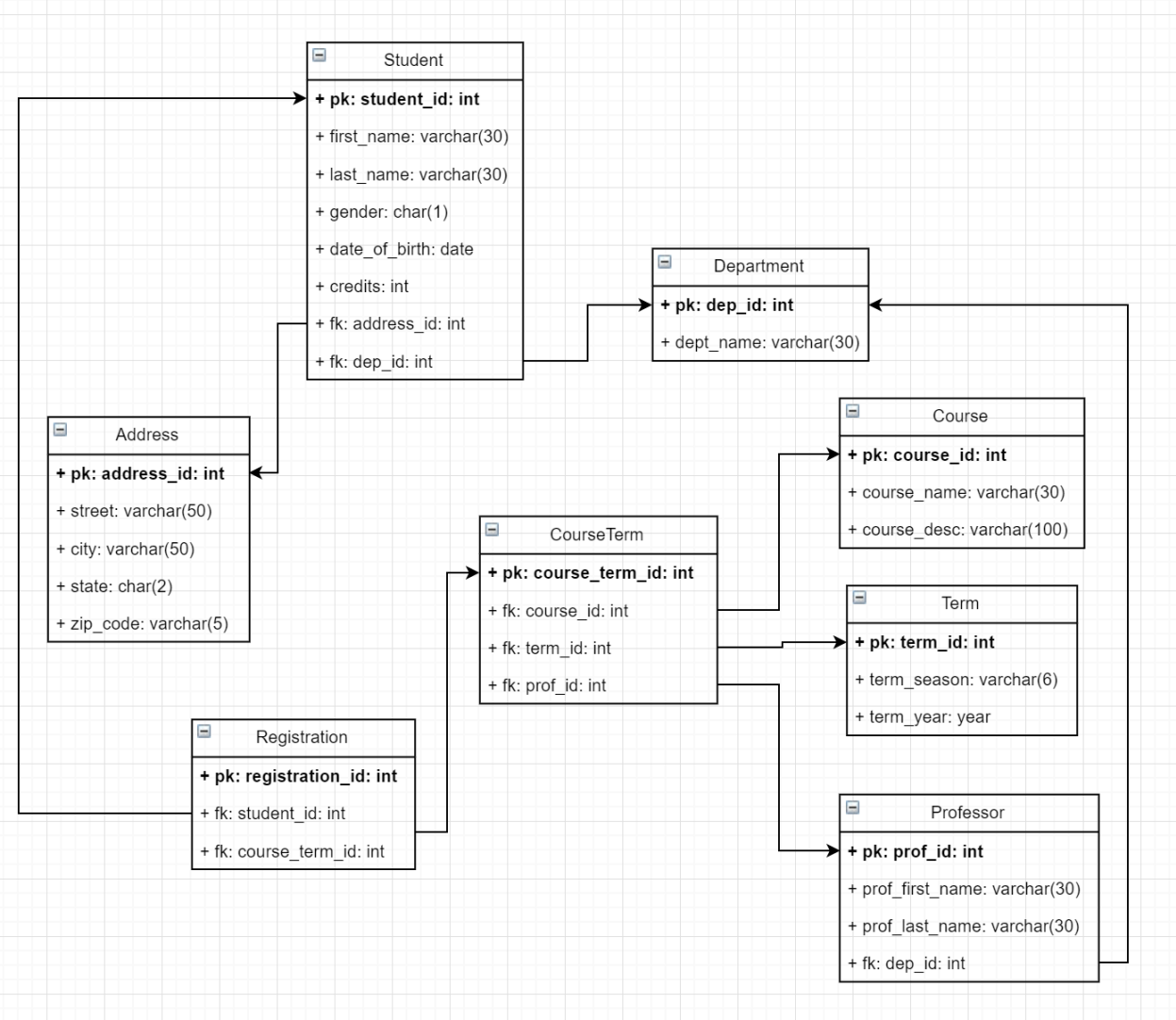
To be in Second Normal Form, you must get rid of any partial dependencies (attribute depends only partially on the primary key), start a new table for the partially dependent data.

11. What are the rules a table must follow to be in Third Normal Form?

To be in Third Normal Form, you must get rid of any transitive dependencies (attribute depends on another attribute other than the primary key).

12. Students at a college are registering for classes online. Each time a student registers for one class, there is certain data that must be saved. Below is the data that is being stored for each registration, it is all currently held in one table. Normalize the data so it is in Third Normal Form. You should display your answer as a UML diagram, it does not need to follow exact UML standards. Just make sure to indicate the data type for each attribute and mark the primary and foreign keys. As well, keep in mind that courses at this college can be taught at different terms and could be taught by different professors. Keep to the attributes/data given in the table below as much as possible unless you are creating new keys. As well give an example for how your final tables avoid duplication, insertion, deletion, and update anomalies.

Table: StudentRegistration	
• Student ID • First Name • Last Name • Address • Gender • Date of Birth • Department Name • Credits	• Course Name • Course Department • Course Description • Course Professor • Professor Department • Term Year • Term Season



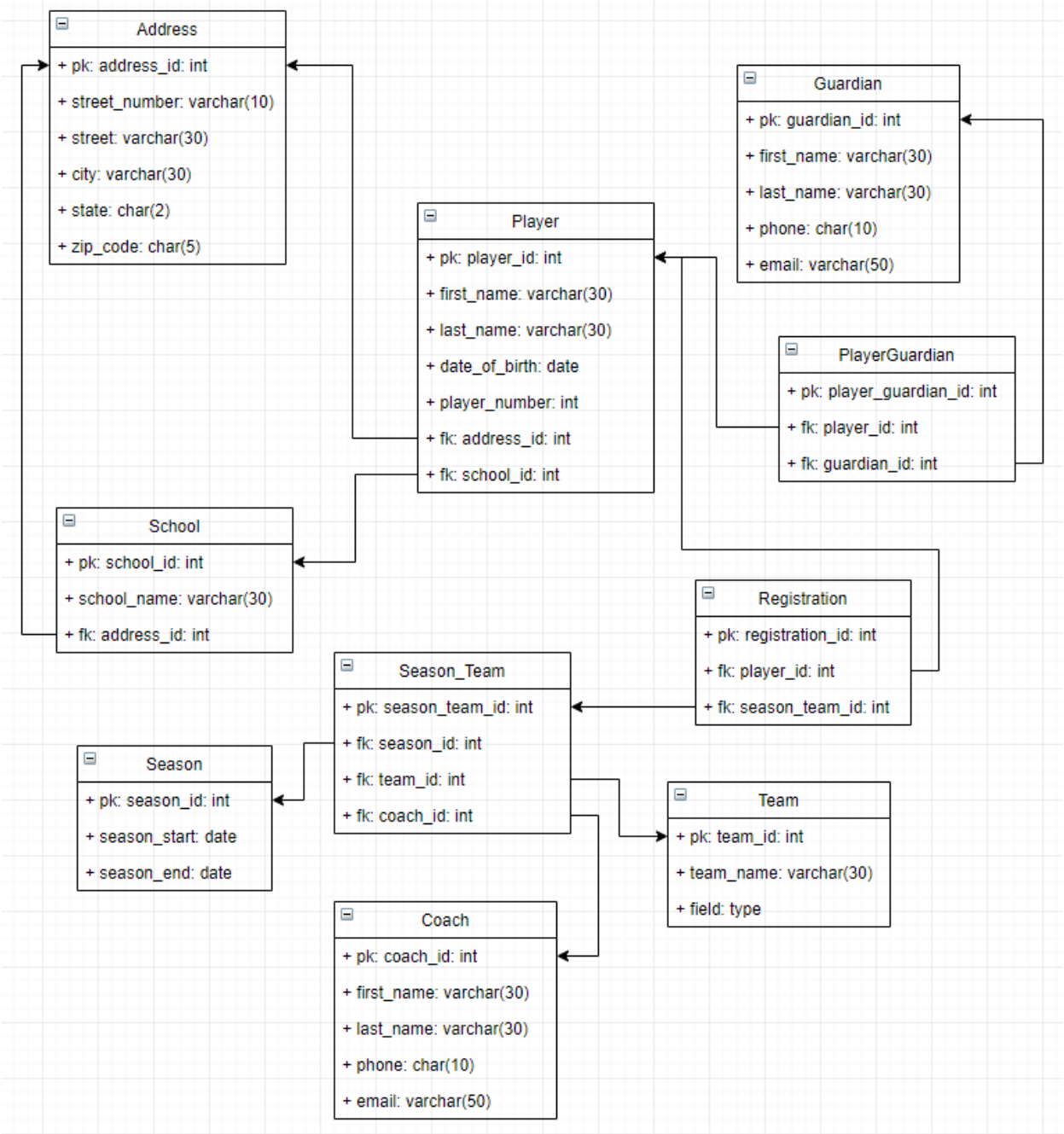
There is a Student table that holds only the necessary data for that student (name, gender, etc.). In this solution, it includes a foreign key to an Address table. The Address table may not be obvious, but it makes the organization of the Student table much nicer and the whole address will not have to be put inside one field. We can add a field for country if you prefer, but in this example solution, the assumption is that the address will be within the US. As well, within the Student table there is a foreign key to the department. Both professors and students need department fields, so it's better to put it in a separate table. Even though it currently only holds its primary key and name, it's possible that the department might hold more information in the future like budget or description. However, that information is not too relevant to a student registration at the moment.

For the actual registration, we created a registration table which has a primary key and foreign keys to the Student and CourseTerm tables. We want a single record for each time a student registers for one course, so this table will keep track of that. Now why does the CourseTerm table exist? Why not put foreign keys to the Course, Term, and Professor tables right in the registration table? A course may be taught at any term, during any year with a different professor. So as not to repeat data and keep tract of what courses were taught during a certain term, we create the CourseTerm table.

Alternate UML question for #12:

Every time a new kid registers for the little league team for the new season, their information is stored in one big table (see below). The players can have more than one guardian, so at the moment, they need to insert another row into this table if they want to add the information of any other guardian. Keep in mind that the coach for a team can change during different seasons. Normalize the data so it's in third normal form and display your answer as a UML diagram. The diagram does not need to be exact, it just need to show the table name, the attributes and their data types, and indicate any primary or foreign keys. As well give an example for how your final tables avoid duplication, insertion, deletion, and update anomalies.

Table: LittleLeagueRegistration	
• Player ID • Player Name • Player Date of Birth • Player's Number • Player Address • Guardian's Name • Guardian's Phone • Guardian's Email	• Team Name • Coach's Name • Coach's Phone • Coach's Email • Player's School Name • Player's School Address • Season Start Date • Season End Date



13. What is noSQL?

A non-relational database like MongoDB. It is document based and has a simple and flexible structure. They are schema-free and based on key-value pairs.

14. What are the differences between noSQL and SQL?

Both are databases, but NoSQL is a non-relational database that has a simple and flexible structure. It can add new attributes to its entities without much trouble. Whereas SQL is a relational database that has table structure. When data is already added, it can be difficult to add new attributes to a table.

15. When would you choose to make a database as noSQL vs. SQL?

NoSQL should be used when you need to store large amounts of data that is constantly growing. If you need flexible data scalability, you should use NoSQL, especially if you are not concerned about data consistency.

SQL should be used when you need to work with complex queries and reports. It's expected that your application will deal with a high transaction rate. SQL is for you if you want to have good data consistency and want to ensure ACID compliance. NoSQL will be overkill for storing your data as well if you don't expect a lot of changes or growth.

ASSIGNMENT

Below are interview questions they should be able to answer as well as some coding exercises.

1. What is the difference between CHAR and VARCHAR?

CHAR is fixed and must be the size you specify, while VARCHAR can hold up to the size you specify. Though both can only have a specified size of up to 255 characters.

2. What is a BLOB and what kind of data can it hold?

A BLOB is a binary large object and can hold any type of media like pictures and videos as well as certain documents like pdfs.

3. Why would you use a TINYINT over an INT when storing data?

If there is a storage limit and you have many records in a table, it is better to use a data type that takes up less space. Especially if the number you may be storing is a small value already and is in the range of -128 to 127 already.

4. Why is the difference between a DOUBLE and a DECIMAL?

Both can take in numbers that have decimal values but DOUBLE stores them as a floating decimal point. While DECIMAL is stored as a string value so it can have a fixed decimal point.

5. A doctor's office wants to store appointments for their patients, which date type should they use?

Either the DATETIME or TIMESTAMP data types so they can store the date and time of the appointment. However, be mindful that TIMESTAMP only supports dates until early in the year 2038, so if they want to keep these data structures for a long time, it is a design constraint they should keep in mind.

6. What is the difference between PRIMARY KEY and UNIQUE?

A PRIMARY KEY is a constraint that is used to uniquely identify a record in a table—it cannot be null. The UNIQUE constraint is not used to identify a table, but it is unique and there cannot be repeat values. As well, the UNIQUE constraint does allow null values. However, since it is unique, only one value in that column can be null.

7. What is DDL?

Data Definition Language, it is used to define the structure of a database or schema.

8. A library wants to save data on all their books. They want to save the book's title, ISBN, author, summary, genre, and published date to one table called book. They want to identify the book by its ISBN (10 or 13 digit number) and if it does not have a summary, they set the summary as "No summary available.". All text data types should be max, 100 characters long. Write the SQL statement to create this book table.

```
CREATE TABLE book
(
    isbn varchar(13) PRIMARY KEY,
    title varchar(50) NOT NULL,
    author varchar(50) NOT NULL,
    summary varchar(100) DEFAULT "No summary available.",
    genre varchar(30),
    publish_date date NOT NULL
);
```

9. Write an SQL query to add a column named breed with a max character length of 30 into a table named dog. The dog table already exists.

```
ALTER TABLE dog ADD breed varchar(30);
```

10. Write an SQL query to change the data type of column weight in table dog to a DOUBLE.

```
ALTER TABLE dog MODIFY COLUMN weight double;
```

11. Write an SQL query to remove the column dog_owner from the table dog.

```
ALTER TABLE dog DROP COLUMN dog_owner;
```

12. Assuming the table dog already exists, write an SQL query to change the column dog_name so it's not null.

```
ALTER TABLE dog ADD UNIQUE(dog_name);
```

13. What does the DESCRIBE command do?

The DESCRIBE command describes a table's structure like what are its columns and the constraints they may have.

14. Write an SQL query to delete the entire table dog. Why should you make sure to comment this delete in your script or avoid writing it in your scripts if you can?

```
DROP TABLE dog;
```

While you made have the need to drop a table, it is good practice to make sure this code is commented out or left out of your scripts on workbench. If you accidently run your entire script or this drop table command, you could lose all your data and table structure without meaning to.

15. Write an SQL query to delete all the data in the table dog.

```
TRUNCATE TABLE dog;
```

16. What is DML?

Data Manipulation Language, used for managing data within schema objects.

17. Below is the create statement for a table called house. Write an SQL query to add a record to house.

```
CREATE TABLE house
(
    house_id int PRIMARY KEY AUTO_INCREMENT,
    address varchar(100) NOT NULL,
    bedrooms int NOT NULL,
    bathrooms float NOT NULL,
    square_feet int
);
```

```
INSERT INTO house VALUES(null, '1 West St', 1, 1, null);
INSERT INTO house(house_id, address, bedrooms, bathrooms, square_feet) values(null, '27 Washington Blvd', 3, 1.5, 1250);
INSERT INTO house(address, bedrooms, bathrooms, square_feet) values('1085 Main St', 4, 2.5, 1400);
INSERT INTO house VALUES(null, '33 Morris Ave', 2, 2, 1225);
INSERT INTO house VALUES(null, '74 Tea St', 2, 1, 1100);
```

There are a few ways to insert into a table, above shows some of the different ways to insert records into the table house.

18. Below are the contents of the house table.

house_id	address	bedrooms	bathrooms	square_feet
1	1 West St	1	1	null

2	27 Washington Blvd	3	1.5	1250
3	1085 Main St	4	2.5	1400
4	33 Morris Ave	2	2	1225
5	74 Tea St	2	1	1100

Write the SQL statements for the following:

a. Change the number of bedrooms to 5 where the house_id is 3.

```
UPDATE house SET bedrooms = 5 WHERE house_id = 3;
```

b. Remove all records that have only 1 bathroom.

```
DELETE FROM house WHERE bathrooms = 1;
```

c. Select all records that have 3 bedrooms.

```
SELECT * FROM house WHERE bedrooms = 3;
```

d. Select only the address and square_feet for all the records.

```
SELECT address, square_feet FROM house;
```

19. What is the Data Control Language (DCL) used for?

DCL is used to control privileges in a database, it dictates which users have read/write access to which databases.

20. Write an SQL statement that creates a user named ‘admin’ with password ‘admin123’.

```
CREATE user 'admin' IDENTIFIED by 'admin123';
```

21. Write an SQL statement that grants ‘admin’ access to all the databases.

```
GRANT all on *.* to 'admin';
```

22. Write an SQL statement that grants ‘user1’ only read access to a database named ‘employee’.

```
GRANT select on employee.* to 'user1';
```

23. Revoke read access to ‘employee’ for ‘user1’.

```
REVOKE all on employee.* from 'user1';
REVOKE select on employee.* from 'user1';
```

24. What is the Transaction Control Language (TCL) used for?

TCL is used to control the transactions made to the database by DML statements. Changes to the database can be managed and be undone if need be.

25. When the ROLLBACK command is run without specifying a SAVEPOINT, when does it rollback to?

As long as the database isn't automatically committing after each statement, it will rollback to the last DDL or DCL statement that was issued.

26. Use the sakila database for the following questions. Make sure to provide the SQL statement you used to find your answer.
    a. Select all the columns from the city table.

```
SELECT * FROM city;
```

    b. Select only the city column from the city table.

```
SELECT city FROM city;
```

    c. The rental_rate in the film table is a daily rate. You want to figure out the rate of each film for every 7 days it's rented, select the title and the weekly rate from the film table.

```
SELECT title, rental_rate*7 FROM film;
```

    d. Edit the select statement from question (c) and set the name of the column that determines the weekly rate as 'weekly rental rate'.

```
SELECT title, rental_rate*7 AS 'weekly rental rate' FROM film;
```

    e. Select the ratings from the film table without any repeat values.

```
SELECT DISTINCT rating FROM film;
```

    f. Select all the columns from the address table where the district is 'California'.

```
SELECT * FROM address WHERE district = 'California';
```

    g. Select the country column from the country table by descending order.

```
SELECT country FROM country ORDER BY country DESC;
```

h. Create a table from the address table that has each district in one column and a count of how many addresses are in that district in another column.

```
SELECT district, count(district) FROM address GROUP BY district;
SELECT district, count(district) AS 'addresses in district' FROM address GROUP BY district;
```

i. In the film table, create a query that will select the rating and the sum of each rating's replacement_cost. Write another query that only selects those sums that are greater than $4,000.

```
SELECT rating, sum(replacement_cost) FROM film GROUP BY rating;
SELECT rating, sum(replacement_cost) FROM film GROUP BY rating HAVING sum(replacement_cost) > 4000;
```

j. From, the film table, select the film_id, title, rental_duration, and rental_rate. The rental_duration should be greater than 5 and the rental_rate should be under $3.00.

```
SELECT film_id, title, rental_duration, rental_rate FROM film WHERE rental_duration > 5 AND rental_rate < 3;
```

k. Select the title, rental_rate, and replacement_cost from the film table, find the films that have a rental_rate less than a dollar or a replacement_cost less than fifteen dollars.

```
SELECT title, rental_rate, replacement_cost FROM film WHERE (replacement_cost < 15) OR (rental_rate < 1);
```

l. Select all the columns from the customer table who do not have the store_id that is 1.

```
SELECT * FROM customer WHERE NOT store_id = 1;
SELECT * FROM customer WHERE store_id != 1;
```

m. Select all the columns from the payment table where the amount is between $2 and $4.

```
SELECT * FROM payment WHERE amount BETWEEN 2 AND 4;
SELECT * FROM payment WHERE (amount > 2) AND (amount < 4);
```

n. Select all the columns from the payment table where the customer_id is 2, 7, or 20.

```
SELECT * FROM payment WHERE customer_id IN (2,7,20);
SELECT * FROM payment WHERE (customer_id = 2) OR (customer_id = 7) OR (customer_id = 20);
```

o. Select the customer_id and first_name from the customer table where the first_name is at least 3 characters long and ends in an 'o'.

```
SELECT customer_id, first_name FROM customer WHERE first_name LIKE '%__o';
```

p. Select all the columns from the address table where address2 is not null.

```
SELECT * FROM address WHERE address2 IS NOT NULL;
```

q. Write an SQL query to find out how many films are in the film table.

```
SELECT count(film_id) FROM film;
```

r. Find what the highest replacement_cost is from the film table. You do not need to give the title, just the amount.

```
SELECT max(replacement_cost) FROM film;
```

s. Find the lowest rental_duration from the film table. You do not need to give the title, just rental duration.

```
SELECT min(rental_duration) FROM film;
```

t. Find the average amount from the payment table.

```
SELECT avg(amount) FROM payment;
```

u. Find the sum of the amount spent by each customer from the payment table. Display the customer_id and the sum total as 'total amount spent'.

```
SELECT customer_id, sum(amount) AS 'total amount spent' FROM payment GROUP BY customer_id;
```

27. What is the DUAL table used for?

The DUAL table is a dummy table that can be used as a place holder when you need to select something that is not in a table. You may need to use it to select and see the result of a specific function like the example below:

```
SELECT curdate() FROM dual;
```

With the dual table in our statement, we can use the function curdate(), that gets todays date. Though in workbench it may not be necessary to have the FROM dual part of the query, when working with Oracle SQL, you need to use the dual table if making these kinds of selects.

28. Use an SQL statement to find the ASCII value of the character '&'.

```
SELECT ascii('&') FROM dual;
```

The ASCII value of '&' is 38.

29. Using the sakila database, write an SQL statement where you select the customer_id and display the customer's full name in another column called 'full name'.

```
SELECT customer_id, concat(first_name, ' ', last_name) AS 'full name' FROM customer;
```

30. What is the longest title length from the film table in the sakila database? Provide the SQL query you wrote to find the answer. This should be the number length, not the actual title.

```
SELECT max(length(title)) FROM film;
```

The longest title has 27 characters.

31. Create an SQL statement to make the word 'sql' uppercase.

```
SELECT upper('sql') FROM dual;
```

32. Use the staff table from the sakila database. The staff are getting new usernames, to create their new usernames, put together their first name, the first letter of their last name, and their staff_id.

```
SELECT concat(first_name, substring(last_name, 1, 1), staff_id) AS 'new username' FROM staff;
```

33. Round up the number 12.34 using an SQL statement.

```
SELECT ceil(12.34) FROM dual;
```

34. Round down the number 3.99 using an SQL statement.

```
SELECT floor(3.99) FROM dual;
```

35. Using an SQL statement find the power of 5^3.

```
SELECT pow(2, 4) FROM dual;
```

36. When would you use the GREATEST vs the MAX function?

The greatest function is used when you want to find the largest/greatest value of the parameters you specify. The max function will find the largest/greatest value in a column of a table. If you try to pass a column of a table to the greatest function, it will give an error, you need to specify each value you pass.

37. If I wanted to add 7 days to the date 10/13/18, how would I do so?

```
SELECT adddate('2018-10-13', 7) FROM dual;
```

38. What does the last_day() function do?

The last_day() function finds the last day of the month of the date passed to it.

39. Get the month from today's current date.

```
SELECT extract(month FROM current_date());
```

40. Explain what a Subquery is.

A Subquery is a query within another query. A subquery will return data that the outer query will use to further restrict the data being retrieved.

41. Use the sakila database for the following questions. Make sure to provide the SQL statement you used to find your answer. You should be using sub queries to find the answers.
    a. Select the film_id and the tile of all the films that the store with store_id = 1 have in their inventory.

```
SELECT film_id, title FROM film WHERE film_id IN (SELECT film_id FROM inventory WHERE store_id = 1);
```

    b. Select the film_id and title of all the films that have an actor that starts with an 'A'.

```
SELECT film_id, title FROM film WHERE film_id IN
        (SELECT film_id FROM film_actor WHERE actor_id IN
            (SELECT actor_id FROM actor WHERE first_name LIKE 'a%'));
```

    c. Select the customer_id, first_name, and last_name of the customers who rented out 'HEAD STRANGER'. Assume you do not know the film_id of 'HEAD STRANGER'.

```
SELECT customer_id, first_name, last_name FROM customer WHERE customer_id IN
    (SELECT customer_id FROM rental WHERE inventory_id IN
        (SELECT inventory_id FROM inventory WHERE film_id IN
            (SELECT film_id FROM film WHERE title = 'head stranger')));
```

42. What is a JOIN used for?

A join is used to combine rows between two or more tables based on a common field between them.

43. What is the difference between an INNER JOIN and an OUTER JOIN?

An inner join will select all rows from both tables if there is a match between the columns in the tables. An outer join will only return the rows of both tables if there is a common value in one of the tables to the other.

44. What is the difference between a LEFT JOIN and a RIGHT JOIN?

The left join will only display the records from the tables if there is a match from the left table while a right join will only display the records if there is a match from the right table.

45. Use the following script to answer the following questions. Provide the query you used to find your answer.

```
create table flight
(
    flight_id int primary key auto_increment,
  location varchar(30) not null,
  destination varchar(30) not null,
  flight_date datetime not null
);

insert into flight values(null, 'Dallas', 'New York', '2019-11-15 12:00:00');
insert into flight values(null, 'New York', 'Dallas', '2019-11-15 11:00:00');
insert into flight values(null, 'New York', 'Los Angeles', '2019-11-15 15:30:00');
insert into flight values(null, 'Las Vegas', 'Miami', '2019-11-15 8:45:00');
insert into flight values(null, 'Miami', 'Chicago', '2019-11-15 12:00:00');

create table passenger
(
    pass_id int primary key auto_increment,
  pass_name varchar(30) not null,
  pass_email varchar(50),
  flight_id int,
  foreign key(flight_id) references flight(flight_id)
);

insert into passenger values(null, 'Sam', 'sam@mail.com', 1);
insert into passenger values(null, 'Jerry', 'jerry@mail.com', 4);
insert into passenger values(null, 'Maria', 'maria@mail.com', 3);
insert into passenger values(null, 'Ashley', 'ashley@mail.com', null);
insert into passenger values(null, 'Kai', 'kai@mail.com', 4);
insert into passenger values(null, 'Layla', 'layla@mail.com', 5);
insert into passenger values(null, 'Alex', 'alex@mail.com', null);
insert into passenger values(null, 'Luis', 'luis@mail.com', 1);
insert into passenger values(null, 'Tom', 'tom@mail.com', 1);
```

a. Do an inner join between the flight and passenger table, display only the passenger, location, destination, and flight date. How many passengers are listed?

```
select pass_name, location, destination, flight_date
from flight join passenger
on flight.flight_id = passenger.flight_id;
```

b. Do a left join between the flight and passenger table. Which flight does not have passengers? How can you tell?

```
select *
from flight left join passenger
on flight.flight_id = passenger.flight_id;
```

c. Do a right join between the flight and passenger table. Which passengers do not have a flight? How can you tell?

```
select *
from flight right join passenger
on flight.flight_id = passenger.flight_id;
```

46. What is a VIEW?

A view displays a virtual table. Its contents are based on a base table, they do not actually contain any data, just display the certain definitions of a table or other view.

47. What are the benefits of Views?

Views are an efficient way to call complicated or long queries that are used often. As well, they provide a good security feature. Certain users can have access to views over whole tables so they do not see any sensitive information they should not have access to.

48. Create a view called kid_friendly_films within the sakila database. It should display the film_id, title, description, rental_rate, and rating of all movies that are rated 'G'.

```
CREATE VIEW kid_friendly_films
AS SELECT film_id, title, description, rental_rate, rating FROM film
WHERE rating = 'G';
```

49. Alter the view for kid_friendly_films and include 'PG' rated films as well.

```
ALTER VIEW kid_friendly_films
AS SELECT film_id, title, description, rental_rate, rating FROM film
WHERE rating = 'G' or rating= 'PG';
```

50. Delete the view kid_friendly_films.

```
DROP VIEW kid_friendly_films;
```

51. What is a stored procedure?

A stored procedure like a function that can save and run multiple SQL statements for later use. It can also take in multiple parameters.

52. What is a delimiter? What is the problem encountered when using delimiters in stored procedures?

A character or string of characters which is used to end an SQL statement. The default delimiter is the semicolon. When writing a stored procedure, you need to write out many statements that have to end in a semicolon. So that you can properly close the procedure, you need to change the delimiter to something other than the semicolon. You can still use semicolons to write out your statements within the procedure and you are able to close out your procedure.

53. For the following questions, run the script below. Hint: you can call a stored procedure within another stored procedure.

```
create database testprocedures;
use testprocedures;

create table bike
(
      bike_id int primary key auto_increment,
   bike_color varchar(30),
   bike_brand varchar(30) not null,
   wheels int not null,
   bike_model varchar(100) not null
);

insert into bike values(null, 'red', 'gt', 2, 'gt-red-2');
insert into bike values(null, 'blue', 'bmc', 3, 'bmc-blue-3');
insert into bike values(null, 'white', 'gt', 2, 'gt-white-2');
insert into bike values(null, 'black', 'pinarello', 2, 'pinarello-black-2');
insert into bike values(null, 'red', 'trek', 3, 'trek-red-3');
insert into bike values(null, 'red', 'bmc', 2, 'bmc-red-2');
insert into bike values(null, 'black', 'pinarello', 2, 'pinarello-black-2');
insert into bike values(null, 'green', 'focus', 4, 'focus-green-4');
insert into bike values(null, 'red', 'giant', 4, 'giant-red-4');
insert into bike values(null, 'white', 'giant', 2, 'giant-white-2');
insert into bike values(null, 'red', 'bmc', 3, 'bmc-red-3');
insert into bike values(null, 'green', 'pinarello', 2, 'pinarello-green-2');
insert into bike values(null, 'red', 'bmc', 2, 'bmc-red-2');
insert into bike values(null, 'black', 'pinarello', 2, 'pinarello-black-2');
insert into bike values(null, 'green', 'focus', 4, 'focus-green-4');
insert into bike values(null, 'red', 'giant', 4, 'giant-red-4');
insert into bike values(null, 'white', 'giant', 2, 'giant-white-2');
insert into bike values(null, 'red', 'bmc', 3, 'bmc-red-3');
insert into bike values(null, 'green', 'pinarello', 2, 'pinarello-green-2');
insert into bike values(null, 'red', 'trek', 3, 'trek-red-3');
insert into bike values(null, 'red', 'bmc', 2, 'bmc-red-2');
insert into bike values(null, 'black', 'pinarello', 2, 'pinarello-black-2');
insert into bike values(null, 'green', 'focus', 4, 'focus-green-4');
insert into bike values(null, null, 'focus', 2, 'focus-none-4');
```

a. Create a procedure that will add a new record to the bike table so that the model for the bike can be generated based on the color, brand, and wheels given. There should be an out parameter called bikeModel. Remember that the color can be null, and if there is a null, to set it as [brand]-none-[wheels].

```
delimiter $$
create procedure spAddBike(in bikeColor varchar(30), in bikeBrand varchar(30), in bikeWheels int,
                           out bikeModel varchar(100))
begin
    declare color varchar(30);

    if (bikeColor is null) then
        select 'none' into color;
    else
        select bikeColor into color;
    end if;

    select concat(bikeBrand, '-', color, '-', bikeWheels) into bikeModel;
    insert into bike values(null, bikeColor, bikeBrand, bikeWheels, bikeModel);
end $$
delimiter ;
```

b. Create a procedure that will add multiple of the same bike to the bike table. For example, add 5 records for a green bike from the giant brand that has 2 wheels.

```
delimiter $$
create procedure spAddMultipleBikes(in bikeColor varchar(30), in bikeBrand varchar(30), in bikeWheels int,
                                    in bikeCount int)
begin
    set @temp = '';
    while(bikeCount > 0)
    do
        call spAddBike(bikeColor, bikeBrand, bikeWheels, @temp);
        select (bikeCount - 1) into bikeCount;
    end while;
end $$
delimiter ;
```

c. Create a procedure that will accept a number corresponding to a bike brand (see below). Based on the number given, it will insert a bike of that brand into the bike table.

Brand	ID
gt	1
bmc	2
pinarello	3

trek	4
focus	5
giant	6

```
delimiter $$
create procedure spAddBike2(in bikeColor varchar(30), in bikeWheels int, in bikeBrandID int)
begin
    set @temp = '';
    set @bikebrand = '';

    case
        when (bikeBrandID = 1) then
            set @bikebrand = 'gt';
        when (bikeBrandID = 2) then
            set @bikebrand = 'bmc';
        when (bikeBrandID = 3) then
            set @bikebrand = 'pinarello';
        when (bikeBrandID = 4) then
            set @bikebrand = 'trek';
        when (bikeBrandID = 5) then
            set @bikebrand = 'focus';
        when (bikeBrandID = 6) then
            set @bikebrand = 'giant';
    end case;

    if (@bikebrand != '') then
        call spAddBike(bikeColor, @bikebrand, bikeWheels, @temp);
    end if;

end $$
delimiter ;
```

54. What is a cursor?

Like an iterator, a cursor is a controlled structure that allows you to traverse through all the records in a database.

55. What are the properties a cursor has?

Asensitive: server may or may not make a copy of its result table

Read only: Not updatable

Nonscrollable: can traverse only in one direction, can't skip rows

56. Write a procedure using the sakila database that uses a cursor to find the average replacement cost for all the films rated PG.

```
delimiter $$
create procedure avgReplacementCost(out ans decimal(5,2))
begin
    declare finished integer default 0;
    declare cost decimal(5,2) default 0.00;

    declare films
    cursor for
    select replacement_cost from film where rating = 'PG';

    declare continue handler for
        not found set finished = 1;

    set @count = 0;
    set @total = 0;

    open films;

    allFilms: loop
        fetch films into cost;

        if (finished =  1) then
            leave allFilms;
        end if;

        set @count = @count + 1;
        set @total = @total + cost;

    end loop allFilms;
    close films;

    select (@total / @count) into ans;
end $$
delimiter ;
```

57. Create a function that returns the bike model from question 50 (a).

```
delimiter $$
create function bikeModel(bikeColor varchar(30), bikeBrand varchar(30), bikeWheels int)
returns varchar(100)
deterministic
begin
    return concat(bikeBrand, '-', bikeColor, '-', bikeWheels);
end $$
delimiter ;
```

58. What does the deterministic keyword used for?

The deterministic keyword is used when a function will always return the same output if given the same input each time. The rand() function is not deterministic since it does not always return the same output. While the pow() function will always return the same output if given the same input each time.

59. What is the difference between a view, stored procedure, and function?

A view is a virtual table that is a definition based on a base table. A stored procedure can display virtual tables, but it is used to run multiple statements in a block. It is a type of function that can take in parameters, but unlike functions, does not return a value. The only way to get a return value, is to use the out or inout parameters.