**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **09-July-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **Coursera** | **USN:** | **4AL16EC058** |
| **Topic:** | **SQL data types** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **Github Repository:** |  |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  In my first session today I have studied about - **SQL data types**  The data type of a column defines what value the column can hold: integer, character, money, date and time, binary, and so on. SQL Data Types Each column in a database table is required to have a name and a data type.  An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.  **Note:** Data types might have different names in different database. And even if the name is the same, the size and other details may be different! **Always check the documentation!** MySQL Data Types (Version 8.0) In MySQL there are three main data types: string, numeric, and date and time. String data types:  |  |  | | --- | --- | | **Data type** | **Description** | | CHAR(size) | A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1 | | VARCHAR(size) | A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535 | | BINARY(size) | Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1 | | VARBINARY(size) | Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes. | | TINYBLOB | For BLOBs (Binary Large OBjects). Max length: 255 bytes | | TINYTEXT | Holds a string with a maximum length of 255 characters | | TEXT(size) | Holds a string with a maximum length of 65,535 bytes | | BLOB(size) | For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data | | MEDIUMTEXT | Holds a string with a maximum length of 16,777,215 characters | | MEDIUMBLOB | For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data | | LONGTEXT | Holds a string with a maximum length of 4,294,967,295 characters | | LONGBLOB | For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data | | ENUM(val1, val2, val3, ...) | A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them | | SET(val1, val2, val3, ...) | A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list |   Okay. Now, let's wrap up our discussion about data models by getting  back to talking about the type of data models we'll be working with in this class.  After this video, you should be able to define and  describe both relational and transactional database models,  define entities, attributes and relationship,  describe and explain the difference between one-to-one,  one-to-many and many-to-many relationships,  describe the use of a primary key in a database and explain  how ER diagram is used to document and illustrate relationships.  I want to kick off this video by talking a little bit  about relational versus transactional database models.  A relational model is a database design that shows the relationships between  the different tables and this is really used to optimize querying data,  making it easy and intuitive to access the data.  Transactional, on the other hand,  you can think of as a more operational database.  If you are in healthcare, for example,  you may have a transactional database that is  used to store all the claims information and then  this information may not be stored in a great way for querying and using it for analysis.  In fact, you may need to take and extract  that transactional information from the database and move it into a relational model.  Most of what we will be working with in this class is the relational model.  The building blocks for this relational model are really three simple things.  We have entities which are a person,  place, thing or event.  These are very distinguishable.  They are unique. They are distinct.  For example, I could be an entity, Sadie St. Lawrence,  and then we have attributes,  which are characteristics of this entity.  As an entity, it would be myself and then  an attribute about me would be that I am female.  Then, the third building block of the model is the relationship.  This describes the association among different entities.  There are a few types of relationships in a database and the ones I want to  cover are the three shown here: one-to-many, many-to-many and one-to-one.  If you think of a one-to-many relationship,  this could be one customer that has many invoices.  When you think of a many-to-many relationship,  this could be an example of many students to many classes.  You may have one student who belongs to lots of  different classes or you may have a class who has many different students.  Then, if you think of a one-to-one relationship,  this is a manager to a store.  Let's say you have a sporting goods store and each of those stores has only one manager.  That would be one example of a one-to-one relationship.  To understand these relationships between the tables a lot better,  what's often used to depict this are ER diagrams.  An ER model then is composed of  the entity types and the specific relationships that can exist between these entities.  These are usually displayed in a visual format and  a relate represents a relationship between the tables.  It often helps you to understand and represent  a business process and it will show the links between these tables.  The links are really important because in a later lesson,  we're going to learn how to join these tables together and combine the data.  Being able to look at this diagram and see  how they relate to each other is really important.  What we will use to join these tables together are two things.  We can use the primary keys or foreign keys.  The primary key is a column or set of  columns whose values uniquely identify every row in a table.  Then this allows us to take those unique identities and then join it to another table.  Foreign keys are similar but in regard that  one or more columns can be used together to identify a single row in the table.  When we're looking at ER diagrams,  which again is one of the ways you will start to think before you do,  you'll look at maybe an ER diagram and understand what data elements you  are trying to join together and how do you need to get them.  But one of the things you need to understand is how to read this.  We talked a little bit about relationships and  the different relationships between a table.  And then, there was a different type of notation that explains the relationships.  We have the Chen notation,  and there's the Crow's foot notation,  and then there's the UML class diagram notation.  The Chen notation uses one-to-M for a one-to-many relationship,  and M and N for  a many-to-many relationship and a one to one for a one-to-one relationship.  In Crow's foot notation,  we have the train tracks which represent  1 and then the Crow's foot which represents many.  And finally, with UML we have a 1.1 which represents the concept of one and 1."  which represents the concept of many.  Definitely take notes at these.  You'll be looking at ER diagrams quite frequently and  you'll need to understand these notations when reading ER diagrams  so you can understand how you're going to write your query and join  the table together or even to find out what's listed in the table.  This is really part of that thinking  before you start doing' concept I was talking about earlier.  Having a good understanding of why the data is  structured in a particular way and how to read  the ER diagrams will be very helpful to you in writing  your queries and ensuring you get accurate results.  As we talked about previously,  the majority of what data scientists are doing with SQL is retrieving data.  To be able to do that and get you started,  the first statement we are going to use is called the SELECT statement.  After this lesson, you'll be able to write a basic SELECT statement, tell a database  what table you want your data FROM, SELECT either all or particular columns from  a table in a query, and limit the amount of data that is returned in a query.  With the SELECT statement you're going to specify two pieces of information,  what you want to select and where you want it from.  Let's look at the concept using an example.  So in this example, I'm going to select product name, that's a column from  the table I want and then I'm going to say where I want to get it from.  So I want to get it from Products.  The output of this is then going to look like the column listed below.  Which it has a column product name and then all of the list of products.  We have shampoo, toothpaste, deodorant, and toothbrush.  If you want to retrieve more than a single column from a table,  then what you need to do is add the names of the individual columns together.  But add a comma after you add the column name.  So in this example, we'll still select from the Products table, but  will also select the prod\_name, the prod\_id, and the prod\_price.  You can see below that I've written the same query in two different ways.  I like to usually write it the second way because it helps me make sure I  don't forget any commas after I wrote a column name that I am selecting.  So, each statement is the same.  One to me is just a little bit easier to read,  so that's why I write it the second way.  But both statements will produce the same results.  Now let's say you have a table that has 20 columns and  you want all of the columns in the table.  Instead of having to write out each individual column which would take quite  a while, there is a wildcard that you can use which is the asterisk.  So you can put SELECT \* and then FROM Products and  this is going to go ahead and grab everything from the Products table,  each individual column, and put it into your output.  So that's the fundamentals of using SELECT.  Anytime you're retrieving data you are going to have a SELECT statement.  Because you're retrieving data you need to say something as hey, go get me something.  This is what SELECT is for.  And the FROM that accompanies it will always go hand-in-hand because if you're  selecting something, you need to tell SQL, in the database, where to get it from.  A lot of times, though, we may want to pull the whole table to get a view of it,  to understand what data's in there.  So we may do a SELECT star.  But if there is something like five million records in it and  we may just really want to get a sample of that.  So just to view some of the data in table we may need to limit our results.  To do this we can SELECT the columns we want from the table we want.  Then after the FROM statement we're just going to put a statement that says limit,  and you can put the number.  For this I'm going to limit it to five.  So I just want to see the first five records.  Here though I've also listed the differences and how this syntax of this is  written for different relational database management systems.  I'm not going to spell it out individually for every statement and  how it differs from the different systems.  But this, as I've mentioned before,  I want you to be aware of that there are differences in syntax.  In the meantime, we're using SQLite.  If you understand it's LIMIT 5 and then you switch over to a DB2 system,  that's still something you can easily Google in terms of saying, hey,  I'm using a DB2 system, and I want to be able to limit my results.  What's the syntax to be able to do that?  Just know that here, in this example,  we're going to use a LIMIT 5 because we're working with SQLite.  But you can see that for Oracle,  you can use WHERE the number of row number is greater than or equal to a number.  Then for DB2 you can use FETCH the first five rows.  Okay, so that's it for this one.  So this video's really gone over the backbone of retrieving data.  I showed you a couple of examples of how to SELECT an individual column,  multiple columns, a whole table and then also how you can limit your results.  These are really the key things you need to know in order to understand  writing basic queries. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **06-July-2020** | **Name:** | **Raziya Banu** | |
| **Course:** | **Cisco** | **USN:** | **4AL16EC058** | |
| **Topic:** | [**Everything Needs to be Secured**](https://static-course-assets.s3.amazonaws.com/I2IoT20/en/index.html#5) | **Semester & Section:** | **8th sem & ‘B’ section** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Types of Data** Has data really changed? Well technically no, data generated by computers and digital devices is still groups of 1s and 0s. That has not changed. What has changed is the quantity, volume, variety, and immediacy of the generated data.  Historically companies would have access to our information gathered from forms, spreadsheets, applications, credit card purchases and other types of files. Much of the information was stored and analyzed at a later date. Sensitive data was still collected, stored and analyzed but, historically, hackers were more interested in hacking into systems to obtain corporate or government secrets.  Today, gathered data is taking on new characteristics. The digitized world has opened the floodgates for data gathering. IoT sensor-enabled devices are collecting more and more data of a personal nature. Wearable fitness trackers, home monitoring systems, security cameras, and debit card transactions are all collecting personal data as well as business and environmental data. Data is often combined from different sources and users may be unaware of this. Combining fitness monitoring data with house monitoring data could produce data points to help map the movements or location of a homeowner. This changing type of data collection and aggregation can be used for good purposes to help the environment. It also increases the possibility of invasion of our privacy, identity theft, and corporate espionage.  Personally identifiable information (PII) or sensitive personal information (SPI) is any data relating to a living individual that can be used on its own or with other information to identify, contact, or locate a specific individual. The data gathered by companies and government institutions can also contain sensitive information concerning corporate secrets, new product patents, or national security.  Because we are gathering and storing exponential quantities of both sensitive and informational data, it has increased the need for extra security to protect this information from natural disasters, hackers, and misuse. **Who Wants our Data?** **The Good Guys**  Legitimate companies have an agreement in place that gives them permission to use the collected data about you for purposes of improving their business. Remember those “Terms and Conditions” or “Terms of Service and Agreements” documents that we say yes to but do not usually read? The next time that you are presented with one, take the time to read through it. The contents might surprise you.  Other legitimate users of our data would be companies that use sensors on their own devices or vehicles. Governments that have environmental sensors, and cities who have installed sensors on trains, busses or traffic lights also have a right to the data they generate.  Some hackers, called white hat hackers, are paid by legitimate companies and governments to test the security of a device or system. Their goal is not to steal or modify data but to help to protect it.  **The Bad Guys**  Other hackers, called black hat hackers, want access to collected data for many nefarious reasons:   * To sell the information to a third party. * To modify the data or disable functionality on a device. * To disrupt or to damage the image of a legitimate company. * To access devices, web pages, and data to create political unrest or to make a political statement. * To access user IDs and passwords to steal identities. * To access data to commit a crime. * To hack into systems to prove that they can do it.  **Lab - Internet Fingerprint** The purpose of this lab is to introduce the aspect of “fingerprinting” an individual using the worldwide web. The objective is to introduce various methods to extract as much information as possible using only the Internet browser and various sites effectively. **Security Best Practices** Securing the network involves all of the protocols, technologies, devices, tools, and techniques that secure data and mitigate threats. Network security is largely driven by the effort to stay one step ahead of ill-intentioned hackers. Just as medical doctors attempt to prevent new illnesses while treating existing problems, network security professionals attempt to prevent potential attacks while minimizing the effects of real-time attacks. Networks are routinely under attack. It is common to read in the news about yet another network that has been compromised.  Security policies, procedures, and standards must be followed in the design of all aspects of the entire network. This should include the cables, data in transit, stored data, networking devices, and end devices. **Physical Security** Today’s data centers store vast quantities of sensitive, business-critical information; therefore, physical security is an operational priority. Physical security not only protects access to the premises, but also protects people and equipment. For example, fire alarms, sprinklers, seismically-braced server racks, and redundant heating, ventilation, and air conditioning (HVAC) and UPS systems are in place to protect people and equipment.  Figure one shows a representation of a data center. Select each circle for more information.  Physical security within the data center can be divided into two areas, outside and inside.   * **Outside perimeter security** - This can include on-premise security officers, fences, gates, continuous video surveillance, and security breach alarms. * **Inside perimeter security** - This can include continuous video surveillance, electronic motion detectors, security traps, and biometric access and exit sensors.   Security traps provide access to the data halls where data center data is stored. As shown in Figure 2, security traps are similar to an air lock. A person must first enter the security trap using their badge ID proximity card. After the person is inside the security trap, facial recognition, fingerprints, or other biometric verifications are used to open the second door. The user must repeat the process to exit the data hall. **Challenges of Securing IoT devices** IoT devices are developed with the necessary network connectivity capabilities but often do not implement strong network security. Network security is a critical factor when deploying IoT devices. Methods must be taken to ensure the authenticity, integrity, and security of the data, the path from the sensor to the collector, and the connectivity to the device. | | | |