SQL Worksheet 4

Q1. Which of the following are TCL commands?

A. Commit C. Rollback D. Savepoint

Q2. Which of the following are DDL commands?

A. Create C. Drop D. Alter

Q3. Which of the following is a legal expression in SQL?

C. SELECT * FROM SALES WHEN PRICE = NULL;

Q4. DCL provides commands to perform actions like?

C. Authorizing Access and other control over Database

Q5. Which of the following should be enclosed in double quotes?

B. Column Alias

Q6. Which of the following command makes the updates performed by the transaction permanent in the database?

B. COMMIT

Q7. A subquery in an SQL Select statement is enclosed in:

A. Parenthesis - (...).

Q8. The result of a SQL SELECT statement is a :-

C. TABLE

Q9. Which of the following do you need to consider when you make a table in a SQL?

D. All of the mentioned

10. If you don't specify ASC and DESC after a SQL ORDER BY clause, the following is used by___?

A. ASC

Q11. What is denormalization?

Database denormalization is the process which brings relations down to lower normal forms, reducing the overall integrity of the data. Thus, the data retrieval performances increase. Denormalization is done after normalization for improving the performance of the database. The data from one table is included in another table to reduce the number of joins in the query and hence helps in speeding up the performance.

Example

Suppose after normalization, there are two tables first, Student table and second, Department table. The student has the attributes as STUDENT-ID, STUDENT-NAME, AGE, and DEPT-ID.

STUDENT

STUDENT-ID	STUDENT-NAME	AGE	DEPT-ID
121	Rahul	18	D-01
122	Vijay	19	D-01
123	Raman	18	D-02
124	Rohan	20	D-03

The Department table is related to the Student table with DEPT-ID as the foreign key in the Student table.

DEPARTMENT

DEPT-ID	DEPT-NAME	DEPT-LOC
D-01	B.Tech	Delhi
D-02	MCA	Delhi
D-03	BCA	Noida

Suppose the name of students is needed, along with the name of the DEPT-NAME, then a join operation would be implemented. Issue arises, if the table is vast, a lot of time would be implemented on join operations. So, the data of DEPT-NAME from the DEPARTMENT table to the STUDENT table, can be added and this will help in reducing the time that would have been used in join operation and thus optimize the database.

Q12. What is a database cursor?

- A database cursor is an object used to pinpoint records in a database. Just like a typing cursor is used to alert where the text will appear, a database cursor also shows the specific record in a database that is being worked upon.
- When a database file is opened, the cursor points to the first record in the file, and using various commands the cursor can move to any location within the file.
- When designing a database, too many open cursors uses a (admittedly small) amount of memory. However, if cursors are never closed, that is, discarded after completing their work, they can pile up in memory and cause performance problems.
- A database cursor is an identifier associated with a group of rows. A cursor can be used in the following cases:

    a. Statements that return more than one row of data from the database server:

    - A SELECT statement requires a select cursor.
    - An EXECUTE FUNCTION statement requires a function cursor.

b. An INSERT statement that sends more than one row of data to the database server requires an insert cursor.

Q13. What are the different types of the queries?

Databases often comprise many tables, or collections of related data. A database query is a way to retrieve a specific subset of data from within a database.

<u>Types of database queries:</u>

Select query

Select queries allows to view data from within a table. Two common reasons for performing select queries are to:

a. Show data from specific fields within a table: If database has tables with more information than needed to complete a task. For example, a select query is used to view only the pricing and quantity information from a database full of additional product data.
b. Analyse data from multiple tables at once: Select queries can also allow to pull data from multiple tables. For example, to compile zip code information from one table named Customers and another table named Addresses, create a query that returned information for customers within specific zip codes.

To perform a select query, choose the tables or other queries on the source data. Then, identify the fields wanted to include along with any additional criteria. Then, run the query to see the results in a datasheet view. Saving queries can allow to reuse them if needed to create reports, forms or data sources for future queries.

Append queries

Append queries can help to add data from one or more sources to an existing table. This can be helpful if wanted to update field values or adjust the data in an established set of records. Using queries to copy the data can allow to:

- <u>Perform multiple operations at once</u>: Rather than manually copying and pasting individual data points, use an append query to copy and move all the data at once.
- <u>Refine selections:</u> With append queries, you can use criteria to help you refine your selections and return only the items that match your specifications.
- <u>Move data to destination tables without matching fields:</u> Append queries allow to input data from source tables even if they have different fields. For example, append data from a table with only four of the destination table's seven fields and leave the nonmatching fields blank.
- <u>Review selections before copying:</u> Using the datasheet view, review the selection and make adjustments before copying the data. Because an append query can’t be undo,

this function can help review the criteria and expressions and make sure the query is free from errors.

Make table query

Make table queries are similar to append queries, but they're more useful when an entirely new table need to be created rather than add to an existing one. Make table queries use specific selections of data or data from multiple sources to form a new table. This can be especially helpful when the copy or archive data is needed.

If specific data selections is needed to access often, consider using a make table to create a data source with more convenient access to the data needed regularly. It can produce data archives that reflect existing data that are available on other tables.

Delete query

If a lot of data need to be deleted quickly or regularly, consider using a delete query. This query type allows to establish criteria for the data that need to be deleted, and it can save the time while planning to reuse the function often. If needed, the system to perform repeat deletions, consider saving the query so it can be used later.

Update queries can also help to change or delete information. The primary difference between the two query types is that delete queries can remove entire rows from multiple tables simultaneously. In contrast, update queries only remove individual field values from the tables.

Update query

Update queries can help to change the data in tables quickly and conveniently by allowing to enter the criteria for which rows in a table want to be updated. Update queries allows to review the revised data before submitting the request. While there are some restrictions on the types of fields that need to be updated, like calculated fields, autonumber fields and primary key fields, update queries can save time and energy adjusting the data within the tables.

Q14. Define constraint?

Constraints are the rules enforced on the data columns of a table. This ensures the accuracy and reliability of the data in the database. Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

The different types of constraints are:

a. NOT NULL
b. UNIQUE
c. PRIMARY KEY
d. FOREIGN KEY

e. CHECK
f. DEFAULT
g. CREATE INDEX

1. NOT NULL

- NULL means empty, i.e., the value is not available.
- Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.
- There must exist a value in the column to which the NOT NULL constraint is applied.

2. UNIQUE

- Duplicate values are not allowed in the columns to which the UNIQUE constraint is applied.
- The column with the unique constraint will always contain a unique value.
- This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.
- Using the UNIQUE constraint, you can also modify the already created tables.

3. PRIMARY KEY

- PRIMARY KEY Constraint is a combination of NOT NULL and Unique constraints.
- NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint.
- The column to which we have applied the primary constraint will always contain a unique value and will not allow null values.

4. FOREIGN KEY

- A foreign key is used for referential integrity.
- When we have two tables, and one table takes reference from another table, i.e., the same column is present in both the tables and that column acts as a primary key in one table. That particular column will act as a foreign key in another table.

5. CHECK

- Whenever a check constraint is applied to the table's column, and the user wants to insert the value in it, then the value will first be checked for certain conditions before inserting the value into that column.

6. DEFAULT

- Whenever a default constraint is applied to the table's column, and the user has not specified the value to be inserted in it, then the default value which was specified while applying the default constraint will be inserted into that particular column.

7. CREATE INDEX

- CREATE INDEX constraint is used to create an index on the table. Indexes are not visible to the user, but they help the user to speed up the searching speed or retrieval of data from the database.

Q15. What is auto increment?

Auto Increment is a function that operates on numeric data types. It automatically generates sequential numeric values every time that a record is inserted into a table for a field defined as auto increment.

Auto increment syntax

```
CREATE TABLE `categories` (
  `category_id` int(11) AUTO_INCREMENT,
  `category_name` varchar(150) DEFAULT NULL,
  `remarks` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`category_id`)
);
```

- Auto increment attribute when specified on a column with a numeric data types, generates numbers sequentially whenever a new row is added into the database.
- The Auto increment is commonly used to generate primary keys.
- The defined data type on the Auto increment should be large enough to accommodate many records. Defining TINYINT as the data type for an auto increment field limits the number of records that can be added to the table to 255 only since any values beyond that would not be accepted by the TINYINT data type.
- Specify the unsigned constraint on auto increment primary keys to avoid having negative numbers.
- When a row is deleted from a table, its auto incremented id is not re-used. MySQL continues generating new numbers sequentially.
- By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record
- To let AUTO_INCREMENT sequence start with another value , use AUTO_INCREMENT = 10