SQL WORKSHEET 6

1. a, c, d
2. a,c,d
3. b
4. c
5. b
6. b
7. a
8. d
9. d
10. a

11. Denormalization is the process of adding precomputed redundant data to an otherwise normalized relational database to improve read performance of the database. Normalizing a database involves removing redundancy so only a single copy exists of each piece of information. Denormalizing a database requires data has first been normalized.

With denormalization, the database administrator selectively adds back specific instances of redundant data *after* the data structure has been normalized. A denormalized database should not be confused with a database that has never been normalized.

Using normalization in SQL, a database will store different but related types of data in separate logical tables, called relations. When a query combines data from multiple tables into a single result table, it is called a join. The performance of such a join in the face of complex queries is often the occasion for the administrator to explore the denormalization alternative.

Denormalization pros and cons

Performing denormalization on databases has its pros and cons, including the following:

Denormalization pros

- Faster reads for denormalized data
- Simpler queries for application developers
- Less compute on read operations

Denormalization cons

- Slower write operations
- Additional database complexity
- Potential for data inconsistency
- Additional storage required for redundant tables

13. A database cursor is an identifier associated with a group of rows. It is, in a sense, a pointer to the current row in a buffer.

You must use a cursor in the following cases:

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

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML (Data Manipulation Language) operations on Table by User. Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors. These are explained as following below.

1. **Implicit Cursors:**
   Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.
2. **Explicit Cursors :**
   Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

13. A query is a question, regularly communicated formally. A database query can be either a select question or an action query. A select query is an information recovery query, while an activity query requests extra tasks on the information, for example, addition, refreshing or deletion.

Queries are helpful devices with regards to databases and they are regularly called by the client through a structure. They can be utilized to look for and get information from at least one of your tables, play out specific activities on the database and even carry out an assortment of calculations relying upon your necessities.

Types of SQL Queries

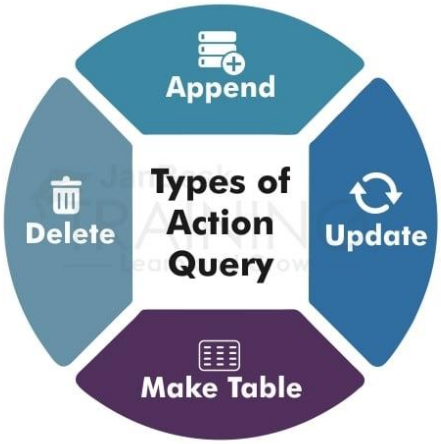
Select Query

The select query is the least difficult kind of inquiry and thus, it is likewise the most ordinarily utilized one in Microsoft Access databases. It very well may be utilized to choose and show information from possibly one table or a progression of them relying upon what is required.

Action Query

At the point when the activity question is called, the database experiences a particular activity relying upon what was indicated in the query itself. This can incorporate such things as making new tables, erasing lines from existing ones and refreshing records or making totally new ones.

Action queries are extremely famous in information the board since they take into account numerous records to be changed at one time rather than just single records like in a select query.



Four types of action queries are:

1. **Append Query** – takes the set consequences of a query and "adds" (or includes) them to a current table.
2. **Delete Query** – erases all records in a hidden table from the set results of a query.
3. **Make Table Query** – as the name proposes, it makes a table dependent on the set consequences of a query
4. **Update Query** – takes into account at least one field in your table to be refreshed.

Parameter Query

In Microsoft Access, a parameter query works with different sorts of queries to get whatever outcomes you are after. This is on the grounds that, when utilizing this kind of query, you can pass a parameter to an alternate query, for example, an activity or a select query. It can either be esteem or a condition and will basically tell the other query explicitly what you need it to do.

Usually picked in light of the fact that it takes into account an exchange box where the end client can enter whatever parameter value, they wish each time the query is being run. The parameter query is only an altered select query.

Aggregate Query

A unique kind of query is known as an aggregate query. It can chip away at different queries, (for example, choice, activity or parameter) simply like the parameter query does, yet as opposed to passing a parameter to another query it aggregates up to the things by chosen by the various groups.

It basically makes a summation of any chosen property in your table. This can be additionally created into measurable sums, for example, midpoints and standard deviation, just to name a couple. The various types of SQL aggregate functions that are available to Microsoft Access are:

14. SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly.

15. Auto-increment **allows a unique number to be generated automatically when a new record is inserted into a table**. Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

The auto increment in SQL is a feature that is applied to a field so that it can automatically generate and provide a unique value to every record that you enter into an SQL table. This field is often used as the PRIMARY KEY column, where you need to provide a unique value for every record you add. However, it can also be used for the UNIQUE constraint columns.

MySQL AUTO_INCREMENT Keyword

MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature.

By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record

The following SQL statement defines the "Personid" column to be an auto-increment primary key field in the "Persons" table:

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

To let the AUTO_INCREMENT sequence start with another value, use the following SQL statement:

```
ALTER TABLE Persons AUTO_INCREMENT=100;
```

When we insert a new record into the "Persons" table, we do NOT have to specify a value for the "Personid" column (a unique value will be added automatically):

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Lars','Monsen');
```

The SQL statement above would insert a new record into the "Persons" table. The "Personid" column would be assigned a unique value automatically. The "FirstName" column would be set to "Lars" and the "LastName" column would be set to "Monsen".